



**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA
ARTIFICIAL**

UNIVERSIDAD DE GRANADA

INSTITUTO DE ASTROFÍSICA DE ANDALUCÍA

CONSEJO SUPERIOR DE INVESTIGACIONES CIENTÍFICAS

Programa de doctorado: *Tratamiento de la información en inteligencia artificial*

**Un sistema de calidad de datos científicos
para el instrumento GIADA dentro de la
misión espacial Rosetta**



Tesis doctoral presentada por:

Rafael Morales Muñoz

Granada, 30 de octubre del 2015



***Un sistema de calidad de datos científicos para el
instrumento GIADA dentro de la misión espacial
ROSETTA***

Memoria presentada por

Rafael Morales Muñoz

para optar al grado de doctor en informática

Granada, 30 de octubre del 2015

Directores:

Olga Pons Capote⁽¹⁾

Julio Federico Rodríguez Gómez⁽²⁾

(1) Departamento de Ciencias de la Computación e Inteligencia Artificial E.T.S. de Ingeniería Informática. Universidad de Granada.

(2) Instituto de Astrofísica de Andalucía. Consejo Superior de Investigaciones Científicas.

El doctorando Rafael Morales Muñoz y los directores de la tesis D^a Olga Pons Capote y D. Julio Federico Rodríguez Gómez, garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, 30 de octubre del 2015

Los directores:

Fdo. Olga Pons Capote.

Fdo. D. Julio Federico Rodríguez Gómez.

El doctorando:

Fdo. Rafael Morales Muñoz.

Agradecimientos

Después del largo periplo que me ha conducido a la escritura de esta memoria, son innumerables las personas que, de forma directa o indirecta, han hecho posible mi trabajo y a las que agradezco profundamente su labor: personal del IAA, del departamento de Ciencias de la Computación, de SENER (España) y de “Officine Galileo” (Italia).

A Olga y Julio, por sus consejos, orientación y por su más que infinita paciencia y tolerancia con los retrasos.

A Carmina y Chemilla por mimarme y alentarme sin descaso.

A Nacho por su apoyo durante tantos años poniendo luz a mis ideas.

Al equipo de desarrollo en España de GIADA, liderado por el José Juan López-Moreno: Antonio López, Ignacio Olivares, Isabel Bustamante, José María Jerónimo, Julio Federico Rodríguez, Justo Sánchez, Miguel Andrés Sánchez, Miguel Herranz, con los que tantas y tantas horas y viajes he compartido.

A todo el equipo de desarrollo de GIADA en Italia, en especial a Luigi Colangeli y a Pasquale Palumbo que tan sabiamente supieron dirigir a buen puerto el instrumento.

Al Ministerio de Educación y Ciencia, cuyos fondos permitieron el desarrollo de los instrumentos GIADA y OSIRIS.

Y por último, aunque no menos importante, a mi familia y a mis personas queridas que han soportado tan largas esperas sin apenas rechistar.

¡Tarde!

Sí.

Pero no olvido mis compromisos.

Índice de contenidos

1	INTRODUCCIÓN	25
1.1	MOTIVACIÓN Y OBJETIVOS	25
1.2	ESTRUCTURA DE LA MEMORIA	26
2	GIADA, UN INSTRUMENTO EN LA MISIÓN ROSETTA	29
2.1	LA MISIÓN ROSETTA	29
2.2	EL INSTRUMENTO GIADA	33
2.2.1	Objetivos científicos de GIADA	35
2.2.2	Relaciones sinérgicas de GIADA	36
2.2.3	Funcionamiento de GIADA	36
2.2.3.1	El hardware de GIADA	37
2.2.3.1.1	GIADA-1: GDS+IS	39
2.2.3.1.2	GIADA-2: Electrónica principal	49
2.2.3.1.3	GIADA-3: MBS	51
2.2.3.1.4	Cubierta	53
2.2.3.2	Curvas de calibración	54
2.2.4	Prestaciones de GIADA	55
2.2.5	Estado de GIADA tras once años de travesía	57
2.2.6	Primeros resultados científicos	57
2.2.7	Resumen y conclusiones	58
3	EL SOFTWARE DE GIADA	59
3.1	CONFIGURACIÓN DE DESARROLLO DEL SW DE GIADA	59
3.2	ARQUITECTURA DEL SW DE GIADA	60
3.3	INTERFAZ SOFTWARE-HARDWARE	61
3.4	ESTRUCTURA DE LOS BANCOS DE MEMORIA DE GIADA	63
3.5	MODOS OPERATIVOS	64
3.6	PROCESO DE LECTURA DEL GDS	66
3.7	PROCESO DE LECTURA DEL IS	67
3.8	PROCESO DE LECTURA DEL MBS	68
3.9	GESTIÓN DE CONTINGENCIAS	69
3.10	MANTENIMIENTO DEL SOFTWARE	69
3.11	INGENIERÍA DE SOFTWARE	70
3.12	DATOS GENERADOS POR GIADA	73
3.12.1	Flujo de datos de GIADA	73
3.12.2	Telecomandos y telemetrías	74
3.12.3	Formato y estructura de las telemetrías de GIADA	79
3.12.3.1	Precisión y marcas temporales en TM y TC	82
3.13	PRESTACIONES DEL SOFTWARE DE GIADA	82
3.14	RESUMEN Y CONCLUSIONES	84
4	ANTECEDENTES DE MODELOS DE BASES DE DATOS	85
4.1	MODELO RELACIONAL DE BASES DE DATOS	85
4.1.1	Estructura de datos relacional	85
4.1.2	Integridad de datos	86
4.1.3	Definición de datos	87
4.1.4	Manipulación de datos	87
4.2	MODELO RELACIONAL DIFUSO DE BASES DE DATOS	88
4.2.1	Teoría de conjuntos difusos	88
4.2.1.1	Operaciones con conjuntos difusos	90

4.2.1.1.1	Unión de conjuntos difusos	91
4.2.1.1.2	Intersección de conjuntos difusos	91
4.2.1.1.3	Complemento o negación de conjuntos difusos	91
4.2.1.2	Números difusos	92
4.2.1.3	Principio de extensión	93
4.2.1.4	Etiquetas y variables lingüísticas	94
4.2.1.5	Teoría de la posibilidad	95
4.2.2	Modelo relacional difuso	95
4.2.2.1	Modelo de Bucles y Petry	96
4.2.2.2	Modelo de Prade y Testemale	97
4.2.2.3	Modelo de Umano y Fukami	98
4.2.2.4	Modelo de Zemankova y Kandel	98
4.2.2.5	Modelo GEFRED	98
4.2.2.5.1	FIRST una implementación computacional de GEFRED	102
4.3	MODELO LÓGICO DE BASES DE DATOS	108
4.3.1	Lógica de primer orden	108
4.3.1.1	El lenguaje de la Lógica de primer orden	108
4.3.1.2	Inferencia	109
4.3.1.2.1	El método sintáctico o “teoría de pruebas”	109
4.3.1.2.2	El método semántico o “teoría de modelos”	110
4.3.1.3	Algoritmo de resolución	111
4.3.1.3.1	Forma clausular de la Lógica	111
4.3.1.3.2	El algoritmo de resolución	112
4.3.1.4	Lenguajes de programación lógica	113
4.3.1.4.1	Prolog	113
4.3.1.4.2	Datalog	114
4.3.2	Representación mediante Lógica de una base de datos relacional	115
4.3.2.1	La BD como teoría relacional	116
4.3.2.2	La BD como interpretación relacional	117
4.4	MODELO LÓGICO DIFUSO DE BASES DE DATOS	118
4.4.1	BDRD como interpretación relacional difusa	119
4.4.2	BDRD como teoría relacional difusa	120
4.4.3	f-Prolog	120
4.5	MODELO INTEGRADO DE BASE DE DATOS RELACIONAL DIFUSA Y DEDUCTIVA	121
4.5.1	Relación extensiva difusa	121
4.5.2	Relación intensiva difusa	121
4.5.3	Generalización de las reglas de la Lógica de primer orden	122
4.5.4	Flexibilización de reglas generalizadas	122
4.5.5	Regla generalizada difusa	123
4.5.5.1	Tipos de reglas generalizadas difusas	124
4.5.6	Deducción con reglas generalizadas difusas	125
4.5.6.1	Deducción sin grado de acoplamiento	126
4.5.6.2	Deducción con grado de acoplamiento	127
4.5.7	Arquitectura de FREDDI	128
4.6	RESUMEN Y CONCLUSIONES	129
5	MODELO DE BASE DE DATOS DE GDB	131
5.1	REPRESENTACIÓN Y MANIPULACIÓN DE INFORMACIÓN	131
5.1.1	Representación y manejo de información difusa	132
5.1.1.1	Tipos de datos difusos	132
5.1.1.2	Comparadores difusos generalizados	133
5.1.1.2.1	Cálculo del grado de acoplamiento con comparadores difusos generalizados	133
5.1.1.3	Base de meta-conocimiento difuso	137

5.1.1.4	FSQL en el modelo GDB	138
5.1.1.4.1	DDL difuso	138
5.1.1.4.2	DML difuso	139
5.1.2	Representación y manejo de información deductiva	140
5.1.2.1	Regla generalizada con grado de acoplamiento	140
5.1.2.1.1	Cálculo de reglas generalizadas con grados de acoplamiento	142
5.1.2.1.2	Cálculo de una RGGGA con predicados disyuntivos	143
5.1.2.1.3	Cálculo de una RGGGA con predicados negados	145
5.1.2.2	Algoritmo de deducción	149
5.1.2.2.1	Algoritmo de deducción “abajo-arriba”	149
5.1.2.3	Base de meta-conocimiento difuso y base de reglas	157
5.1.2.4	DFSQL en el modelo GDB	159
5.1.2.4.1	DDL difuso deductivo	159
5.1.2.4.2	DML difuso deductivo	160
5.1.2.5	Implementación de intervalos: macros	161
5.1.2.5.1	Modificaciones para el manejo de macros	161
5.1.2.5.2	Modificaciones al diagrama entidad-relación de la base de reglas	161
5.1.2.5.3	Modificaciones a la base de meta-conocimiento difusa y base de reglas	162
5.1.2.5.4	Modificaciones a DDL de DFSQL	163
5.1.2.5.5	Modificaciones a la semántica del DFSQL	163
5.1.2.5.6	Ejemplo de macro	164
5.1.2.6	Implementación de constantes	164
5.2	CONJUNTOS DE REGLAS	165
5.3	SEMÁNTICA DE LA MEDIDA DE CALIDAD	166
5.4	EJEMPLO DE USO DEL MODELO GDB	166
5.4.1	Selección del tipo de dato difuso	167
5.4.2	Definición de relaciones clásicas	168
5.4.3	Definición de la tabla intensiva y regla	169
5.4.4	Semántica de la medida de calidad	171
5.4.5	Datos sintéticos	172
5.4.6	Cálculo de la regla	173
5.4.7	Conclusiones del uso del modelo de datos de GDB	176
5.5	ARQUITECTURA DEL MODELO GDB	176
5.6	RESUMEN Y CONCLUSIONES	178
6	INTERFAZ DE USUARIO GDB-GUI	181
6.1	FLUJO DE DATOS DE GDB	181
6.2	CARACTERÍSTICAS COMUNES A LAS INTERFACES SQL Y DFSQL	183
6.2.1	Diseño de la BDR de GDB	184
6.2.2	Menú principal	185
6.2.3	Creación y ejecución de consultas	187
6.2.3.1	Construcción de una consulta vacía	187
6.2.3.1.1	Gestión de categorías	187
6.2.3.1.2	Gestión de consultas	189
6.2.3.2	Selección de las relaciones y atributos	191
6.2.3.3	Creación de las condiciones sobre atributos	192
6.2.3.4	Aplicación de la consulta y obtención de resultados	194
6.2.4	Almacenamiento y recuperación de consultas	195
6.2.5	Importador de datos	196
6.2.6	Gestión de usuarios	197
6.2.7	Curvas de calibración	197
6.2.8	Ficheros de configuración, directorios de entrada y salida	198
6.2.9	Gestión de la base de datos relacional	199

6.2.10	Gestión del SGBDR	200
6.2.11	Adaptación a otros sistemas	200
6.2.12	Nomenclatura de las tablas	200
6.2.13	Enlace automático de tablas	200
6.2.14	Herramientas de desarrollo de GDB-GUI	202
6.3	INTERFAZ SQL	202
6.3.1	Almacenamiento de sentencias SQL	202
6.4	INTERFAZ DFSQL	203
6.4.1	Atributos clásicos con componente difusa	203
6.4.2	Creación de constantes	204
6.4.3	Limitaciones del servidor DFSQL	204
6.4.4	Almacenamiento de reglas	204
6.5	SIMULADOR DE EVENTOS	205
6.6	RESUMEN Y CONCLUSIONES	211
7	VERIFICACIÓN Y ANÁLISIS DE RESULTADOS	213
7.1	DETECCIÓN DE EVENTOS GDS+IS	213
7.1.1	Rutina de atención a la interrupción	215
7.1.1.1	ISR para el GDS	216
7.1.1.2	ISR para el IS	218
7.1.2	Procesamiento de la cola de eventos	219
7.1.2.1	Procesamiento del evento GDS	219
7.1.2.2	Procesamiento del evento IS	221
7.1.3	Algoritmo de asociación de eventos GDS+IS	223
7.1.3.1	Análisis de la asociación de eventos GDS+IS	224
7.2	REGLAS DE VERIFICACIÓN DE EVENTOS GDS+IS	226
7.2.1	Regla para el cálculo de la coherencia de eventos GDS+IS mediante velocidades	226
7.2.1.1	Selección del tipo de dato difuso	226
7.2.1.2	Definición de relaciones clásicas	227
7.2.1.3	Definición de la tabla intensiva y regla	228
7.2.1.4	Semántica de la medida de calidad	228
7.2.1.5	Datos sintéticos	229
7.2.1.6	Cálculo de la regla	230
7.2.1.7	Conclusiones	231
7.2.2	Regla para el cálculo de poblaciones de eventos GDS e IS	232
7.2.2.1	Selección del tipo de dato difuso	232
7.2.2.2	Definición de relaciones clásicas	233
7.2.2.3	Definición de la tabla intensiva y regla	233
7.2.2.4	Semántica de la medida de calidad	234
7.2.2.5	Datos sintéticos	239
7.2.2.6	Cálculo de la regla	240
7.2.2.7	Conclusiones	241
7.3	APLICACIÓN DE LAS REGLAS DE VERIFICACIÓN DE EVENTOS GDS+IS A LOS DATOS DE CALIBRACIÓN	242
7.4	RESUMEN Y CONCLUSIONES	246
8	CONCLUSIONES Y TRABAJOS FUTUROS	249
9	BIBLIOGRAFÍA	251
10	APÉNDICES	259
10.1	APÉNDICE A. DECLARACIÓN SINTÁCTICA Y SEMÁNTICA DE LA BASE DE DATOS RELACIONAL Y DIFUSA	260
10.2	APÉNDICE B. CONTENIDO DEL DVD	301
10.3	APÉNDICE C. INTRODUCCIÓN A LOS COMETAS	302
10.4	APÉNDICE D. EL COMETA 67P/CHURYUMOV-GERASIMENKO	304

Índice de ilustraciones

ILUSTRACIÓN 1. ROSETTA EN ESTEC. CRÉDITOS: ESA/ESTEC. _____	29
ILUSTRACIÓN 2. LOCALIZACIÓN DE LOS INSTRUMENTOS EN ROSETTA. CRÉDITOS: ESA/ATG MEDIALAB. _____	32
ILUSTRACIÓN 3. VISTA DE 6CG DESDE EL INSTRUMENTO PHILAE DE ROSETTA. CRÉDITOS: ESA/ROSETTA/PHILAE/CIVA. _____	32
ILUSTRACIÓN 4. MODELO DE VUELO DE GIADA. _____	33
ILUSTRACIÓN 5. ESQUEMA DE LA POSICIÓN DE GIADA EN ROSETTA. _____	34
ILUSTRACIÓN 6. GIADA INTEGRADA EN ROSETTA EN LAS INSTALACIONES DE ESA EN ESTEC. _____	34
ILUSTRACIÓN 7. DISPOSICIÓN DE LOS MÓDULOS HARDWARE DE GIADA. _____	37
ILUSTRACIÓN 8. MODELO MECÁNICO DE GIADA. _____	38
ILUSTRACIÓN 9. GDS+IS. _____	39
ILUSTRACIÓN 10. ESQUEMA DEL GDS. _____	40
ILUSTRACIÓN 11. ROBOT DE CALIBRACIÓN DEL GDS. _____	42
ILUSTRACIÓN 12. MAPA BIDIMENSIONAL DE SENSIBILIDAD DEL CANAL IZQUIERDO. _____	43
ILUSTRACIÓN 13. MAPA TRIDIMENSIONAL DE SENSIBILIDAD DEL CANAL IZQUIERDO. _____	43
ILUSTRACIÓN 14. RESPUESTA DEL GDS A UNA PARTÍCULA RÁPIDA. _____	44
ILUSTRACIÓN 15. RESPUESTA DEL GDS A UNA PARTÍCULA LENTA. _____	44
ILUSTRACIÓN 16. ESQUEMA DE POSICIONADO DE LOS SENSORES PZT EN EL IS. _____	45
ILUSTRACIÓN 17. SEÑALES ANALÓGICAS INVOLUCRADAS EN EL LA DETECCIÓN DEL SISTEMA. _____	47
ILUSTRACIÓN 18. MAPA DE SENSIBILIDAD CORRESPONDIENTE AL PZT_A DEL IS. _____	49
ILUSTRACIÓN 19. ESQUEMA DE LA ELECTRÓNICA PRINCIPAL. _____	50
ILUSTRACIÓN 20. MODELO DE VUELO DE LAS TARJETAS (PRINCIPAL Y REDUNDANTE) DE PROCESAMIENTO DIGITAL Y ALIMENTACIONES. _____	51
ILUSTRACIÓN 21. MODELO DE VUELO DE LA TARJETA ANALÓGICA. _____	51
ILUSTRACIÓN 22 . MODELO DE VUELO DE GIADA-2. _____	51
ILUSTRACIÓN 23. ESQUEMA DEL MBS. _____	52
ILUSTRACIÓN 24. DEPOSICIÓN DE POLVO EN UNA MB Y FRECUENCIAS ASOCIADAS. _____	52
ILUSTRACIÓN 25. CURVA DE CALIBRACIÓN PARA EL ADC CANAL PRINCIPAL. _____	55
ILUSTRACIÓN 26. CURVA DE CALIBRACIÓN PARA EL ADC CANAL REDUNDANTE. _____	55
ILUSTRACIÓN 27. CONFIGURACIÓN PARA EL DESARROLLO DEL SOFTWARE DE GIADA. _____	59
ILUSTRACIÓN 28. ARQUITECTURA DEL SOFTWARE DE GIADA. _____	60
ILUSTRACIÓN 29. EVENTOS EN EL SOFTWARE DE GIADA. _____	61
ILUSTRACIÓN 30. ESTRUCTURA DE LA MEMORIA EN GIADA. _____	63
ILUSTRACIÓN 31. PROCESO DE COPIA DE LA MEMORIA ROM Y RAM EN GIADA. _____	64
ILUSTRACIÓN 32. TRANSICIONES ENTRE MODOS DE GIADA. _____	65
ILUSTRACIÓN 33. PROCESO DE LECTURA DEL GDS. _____	67
ILUSTRACIÓN 34. PROCESO DE LECTURA DEL IS. _____	67
ILUSTRACIÓN 35. PROCESO DE LECTURA DEL MBS. _____	68
ILUSTRACIÓN 36. CICLO DE VIDA DEL SOFTWARE DE GIADA. _____	71
ILUSTRACIÓN 37. FLUJO DE DATOS DESDE GIADA HASTA LA TIERRA. _____	73
ILUSTRACIÓN 38. SERVICIOS ESA IMPLEMENTADOS POR GIADA. _____	79
ILUSTRACIÓN 39. ESTRUCTURA DE LAS TELEMETRÍAS TM(20,3) Y TM(3,25). _____	81
ILUSTRACIÓN 40. REPRESENTACIÓN GRÁFICA DE UN ALFA-CORTE. _____	89
ILUSTRACIÓN 41. CONJUNTOS DIFUSOS CONVEXOS Y NO CONVEXOS. _____	90
ILUSTRACIÓN 42. NÚMEROS DIFUSOS. _____	93
ILUSTRACIÓN 43. ETIQUETA LINGÜÍSTICA PARA “JOVEN”. _____	94
ILUSTRACIÓN 44. ETIQUETA LINGÜÍSTICA PARA “JOVEN” EN LA DEFINICIÓN DE ZADEH. _____	95
ILUSTRACIÓN 45. FIRST. ATRIBUTOS DIFUSOS DE TIPO 1. _____	102
ILUSTRACIÓN 46. FIRST. DISTRIBUCIÓN DE POSIBILIDAD TRAPEZOIDAL. _____	103
ILUSTRACIÓN 47. FIRST. DISTRIBUCIÓN DE POSIBILIDAD DE UNA ETIQUETA LINGÜÍSTICA. _____	103
ILUSTRACIÓN 48. FIRST. DISTRIBUCIÓN DE POSIBILIDAD “APROXIMADAMENTE N”. _____	103
ILUSTRACIÓN 49. FIRST. DISTRIBUCIÓN DE POSIBILIDAD DE UN INTERVALO [N,M]. _____	104
ILUSTRACIÓN 50. FIRST. ATRIBUTOS DIFUSOS DE TIPO 3. _____	104

ILUSTRACIÓN 51. FIRST. DISTRIBUCIÓN DE POSIBILIDAD PARA EL VALOR “UNKNOWN”.	105
ILUSTRACIÓN 52. FIRST. DISTRIBUCIÓN DE POSIBILIDAD PARA EL VALOR “UNDEFINED”.	105
ILUSTRACIÓN 53. ESQUEMA DE FIRST.	107
ILUSTRACIÓN 54. RGD CON PROFUNDIDAD 1.	125
ILUSTRACIÓN 55. RGD CON PROFUNDIDAD > 1.	125
ILUSTRACIÓN 56. RGD RECURSIVA CON PROFUNDIDAD > 1.	125
ILUSTRACIÓN 57. ARQUITECTURA DE FREDDI.	129
ILUSTRACIÓN 58. MODELOS TEÓRICOS DE BASE DE DATOS E IMPLEMENTACIONES ANTECEDENTES DE GDB.	130
ILUSTRACIÓN 59. MODIFICACIÓN DE LA DISTRIBUCIÓN DE POSIBILIDAD DE LA PARTE DERECHA DE UNA COMPARACIÓN DIFUSA SEGÚN EL COMPARADOR UTILIZADO.	134
ILUSTRACIÓN 60. EJEMPLO DEL ALGORITMO DE DEDUCCIÓN GDB.	153
ILUSTRACIÓN 61. PASO 1 DEL ALGORITMO DE DEDUCCIÓN CON REGLAS RECURSIVAS.	155
ILUSTRACIÓN 62. PASO 2 DEL ALGORITMO DE DEDUCCIÓN CON REGLAS RECURSIVAS.	156
ILUSTRACIÓN 63. PASO 3 DEL ALGORITMO DE DEDUCCIÓN CON REGLAS RECURSIVAS.	157
ILUSTRACIÓN 64. MODELO ENTIDAD-RELACIÓN DE LA BASE DE REGLAS DE GDB.	158
ILUSTRACIÓN 65. MODELO ENTIDAD-RELACIÓN EXTENDIDO DE LA BASE DE REGLAS DE GDB.	162
ILUSTRACIÓN 66. REPRESENTACIÓN DE TIPO DIFUSO APROXIMADAMENTE 88.	167
ILUSTRACIÓN 67. INTERSECCIONES DE LOS VALORES DIFUSOS APROXIMADAMENTE 88 Y 92.	172
ILUSTRACIÓN 68. INTERSECCIONES DE LOS VALORES DIFUSOS APROXIMADAMENTE 88 Y 96.	172
ILUSTRACIÓN 69. RESULTADO DE LA EJECUCIÓN DEL EJEMPLO DE USO DEL MODELO DE GDB EN GDB-GUI.	175
ILUSTRACIÓN 70. ARQUITECTURA DE LA IMPLEMENTACIÓN DE GDB.	177
ILUSTRACIÓN 71. EVOLUCIÓN DE LOS MODELOS TEÓRICOS E IMPLEMENTACIONES HASTA GDB.	178
ILUSTRACIÓN 72. FLUJO DE DATOS EN GIADA.	182
ILUSTRACIÓN 73. PANTALLA PRINCIPAL DE GDB-GUI.	183
ILUSTRACIÓN 74. DIAGRAMA ENTIDAD-RELACIÓN DE LA BASE DE DATOS RELACIONAL DE GIADA.	185
ILUSTRACIÓN 75. GDB-GUI. MENÚ PRINCIPAL-FILE.	185
ILUSTRACIÓN 76. GDB-GUI. MENÚ PRINCIPAL-DB CONNECTION.	185
ILUSTRACIÓN 77. GDB-GUI. MENÚ PRINCIPAL-CHANGE USER.	186
ILUSTRACIÓN 78. GDB-GUI. MENÚ PRINCIPAL-DB MAINTENANCE.	186
ILUSTRACIÓN 79. GDB-GUI. MENÚ PRINCIPAL-MODE CHANGE.	186
ILUSTRACIÓN 80. GDB-GUI. MENÚ PRINCIPAL-FUZZY TYPE.	186
ILUSTRACIÓN 81. GDB-GUI. MENÚ PRINCIPAL-CALIBRATION CURVE.	186
ILUSTRACIÓN 82. GDB-GUI. HELP.	187
ILUSTRACIÓN 83. GDB-GUI. ACCIONES POSIBLES SOBRE UNA CATEGORÍA.	187
ILUSTRACIÓN 84. GDB-GUI. CREACIÓN DE UNA CATEGORÍA.	188
ILUSTRACIÓN 85. GDB-GUI. BORRADO DE UNA CATEGORÍA.	188
ILUSTRACIÓN 86. GDB-GUI. INFORMACIÓN ACERCA DE UNA CATEGORÍA.	189
ILUSTRACIÓN 87. GDB-GUI. CREACIÓN DE UNA CONSULTA.	189
ILUSTRACIÓN 88. GDB-GUI. INFORMACIÓN Y BORRADO CONSULTA.	190
ILUSTRACIÓN 89. GDB-GUI. INFORMACIÓN DE UNA CONSULTA.	190
ILUSTRACIÓN 90. GDB-GUI. INFORMACIÓN ACERCA DE UN ATRIBUTO.	191
ILUSTRACIÓN 91. GDB-GUI. BORRADO DE UN ATRIBUTO.	192
ILUSTRACIÓN 92. GDB-GUI. MENÚ DE ORDENACIÓN POR ATRIBUTOS EN SENTENCIAS SQL.	192
ILUSTRACIÓN 93. GDB-GUI. CREACIÓN DE UNA COMPARACIÓN CON UN ÚNICO ATRIBUTO.	193
ILUSTRACIÓN 94. GDB-GUI. ALMACENAMIENTO DE UNA COMPARACIÓN CON UN ÚNICO ATRIBUTO.	193
ILUSTRACIÓN 95. GDB-GUI. OPERADOR EXTRA A LA DERECHA DE LA COMPARACIÓN.	193
ILUSTRACIÓN 96. GDB-GUI. CREACIÓN DE UNA COMPARACIÓN CON DOS ATRIBUTOS.	194
ILUSTRACIÓN 97. GDB-GUI. RESULTADO DE UNA CONSULTA.	194
ILUSTRACIÓN 98. GDB-GUI. VENTANA DE RESUMEN TRAS LA EJECUCIÓN DE UNA CONSULTA.	195
ILUSTRACIÓN 99. GDB-GUI. ESQUEMA GENERAL DEL ALMACENAMIENTO Y RECUPERACIÓN DE CONSULTAS.	195
ILUSTRACIÓN 100. GDB-GUI. CALCULADORA DE CURVAS DE CALIBRACIÓN.	198
ILUSTRACIÓN 101. GDB-GUI. TABLA SIN LA CONVERSIÓN DE CURVAS DE CALIBRACIÓN APLICADA.	198
ILUSTRACIÓN 102. GDB-GUI. TABLA CON LA CONVERSIÓN DE CURVAS DE CALIBRACIÓN APLICADA.	198

ILUSTRACIÓN 103. GDB-GUI. ESTRUCTURA JERÁRQUICA DE LAS RELACIONES CLÁSICAS. _____	201
ILUSTRACIÓN 104. GDB-GUI. DIAGRAMA ENTIDAD-RELACIÓN DEL ALMACENAMIENTO DE SENTENCIAS SQL. _____	202
ILUSTRACIÓN 105. GDB-GUI. MENÚ PARA OBTENER LA LISTA DE ATRIBUTOS CLÁSICOS CON COMPONENTE DIFUSA. ____	203
ILUSTRACIÓN 106. GDB-GUI. LISTA DE ATRIBUTOS CLÁSICOS CON COMPONENTE DIFUSA. _____	204
ILUSTRACIÓN 107. GDB-GUI. DIAGRAMA ENTIDAD RELACIÓN PARA EL ALMACENAMIENTO DE REGLAS. _____	204
ILUSTRACIÓN 108. GDB-GUI. GENERACIÓN DE DATOS CON EL SIMULADOR DE EVENTOS. _____	206
ILUSTRACIÓN 109. GDB-GUI. PROCESO DE EJECUCIÓN DE UN SCRIPT EN EL SIMULADOR DE EVENTOS. _____	207
ILUSTRACIÓN 110. GDB-GUI. SELECCIÓN DE LA CADENA DE ADQUISICIÓN ANTES DE LA EJECUCIÓN DE UN SCRIPT EN EL SIMULADOR DE EVENTOS. _____	210
ILUSTRACIÓN 111. GDB-GUI. RESUMEN DE LOS RESULTADOS DE LA EJECUCIÓN DE UN SCRIPT. _____	210
ILUSTRACIÓN 112. DETECCIÓN DE UN EVENTO DE GDS+IS. _____	214
ILUSTRACIÓN 113. DIAGRAMA DE FLUJO DE LA ISR DE GDS. _____	217
ILUSTRACIÓN 114. DIAGRAMA DE FLUJO DE LA ISR DE IS. _____	218
ILUSTRACIÓN 115. DIAGRAMA DE FLUJO DEL PROCESAMIENTO DEL EVENTO DE GDS. _____	220
ILUSTRACIÓN 116. DIAGRAMA DE FLUJO DEL PROCESAMIENTO DEL EVENTO DE IS. _____	222
ILUSTRACIÓN 117. PROCESAMIENTO DE UN EVENTO GDS+IS. _____	223
ILUSTRACIÓN 118. RELACIÓN ENTRE ISR, PROCESAMIENTO DE EVENTOS Y TAREA EN GDS. _____	224
ILUSTRACIÓN 119. RELACIÓN ENTRE ISR, PROCESAMIENTO DE EVENTOS Y TAREA EN IS. _____	224
ILUSTRACIÓN 120. PROCESO DE ASIGNACIÓN DE EVENTOS GDS+IS. _____	225
ILUSTRACIÓN 121. REPRESENTACIÓN DEL TIPO DIFUSO VELOCIDAD APROXIMADAMENTE 95 ms^{-1} . _____	227
ILUSTRACIÓN 122. CÁLCULO DE CALIDAD PARA DISTINTAS DIFERENCIAS DE VELOCIDAD. _____	229
ILUSTRACIÓN 123. EJECUCIÓN DE LA REGLA DE COHERENCIA POR TIEMPO EN GDB-GUI. _____	231
ILUSTRACIÓN 124. REPRESENTACIÓN DE TIPO DIFUSO APROXIMADAMENTE 300 CUENTAS. _____	233
ILUSTRACIÓN 125. INTERVALO DE POBLACIÓN DE EVENTOS GDS DEFINIDO POR UN EVENTO IS. _____	235
ILUSTRACIÓN 126. REPRESENTACIÓN DE LOS TIPOS DIFUSOS APROXIMADAMENTE 295 Y 300. _____	236
ILUSTRACIÓN 127. CÁLCULO DEL GRADO DE COMPATIBILIDAD DE 295 FGEQ 300. _____	236
ILUSTRACIÓN 128. CÁLCULO DE LA CALIDAD ASIGNADA PARA DIFERENTES VALORES DE TIEMPO DE EVENTOS DE GDS RESPECTO A UN TIEMPO DE EVENTO IS. _____	237
ILUSTRACIÓN 129. CÁLCULO DEL GRADO DE COMPATIBILIDAD DE 301 FGEQ 300. _____	238
ILUSTRACIÓN 130. INTERVALO DE POBLACIÓN DE EVENTOS GDS DEFINIDO POR UN EVENTO IS CON TIEMPO DE EVENTO 683 CUENTAS. _____	239
ILUSTRACIÓN 131. EJECUCIÓN DE LA REGLA DE POBLACIONES EN GDB-GUI. _____	241
ILUSTRACIÓN 132. DATOS DE CALIBRACIÓN DE GIADA IMPORTADOS POR GDB-GUI. _____	242
ILUSTRACIÓN 133. COMPONENTES DE UN COMETA. _____	302
ILUSTRACIÓN 134. COMETA 6CG VISTO DESDE EL VLT (2013), ESO, CERRO PARANAL, CHILE. CRÉDITOS: ESO. ____	305
ILUSTRACIÓN 135. COMETA 6CG VISTO CON EL INSTRUMENTO FOR2 (2014). TELESCOPIO ANTU DE 8.2M, VLT, ESO, CERRO PARANAL, CHILE. CRÉDITOS: ESO. _____	305
ILUSTRACIÓN 136. NÚCLEO, COMA Y COLA DE POLVO DE 6GC (2015). CRÉDITOS: ESA, ROSETTA SPACECRAFT, NAVCAM. _____	305
ILUSTRACIÓN 137. COMA Y SUBLIMACIÓN EN 6CG OBSERVADO DESDE ROSETTA (2015). CRÉDITOS: ESA/ROSETTA/MPS FOR OSIRIS TEAM MPS/UPD/LAM/IAA/SSO/INTA/UPM/DASP/IDA. _____	306
ILUSTRACIÓN 138. 6CG OBSERVADO A DECENAS DE KILÓMETROS DESDE ROSETTA (2015). CRÉDITOS: ESA, ROSETTA SPACECRAFT, NAVCAM. _____	306
ILUSTRACIÓN 139. 6CG OBSERVADO A 2 METROS DESDE LA CÁMARA 3 DEL INSTRUMENTO CIVA DE PHILAE EN EL LUGAR DE ATERRIAJE DE PHILAE (JULIO 2015). CRÉDITOS: ESA/ROSETTA/PHILAE/CIVA. _____	307

Índice de tablas

TABLA 1. HITOS DE ROSETTA.	30
TABLA 2. INSTRUMENTOS A BORDO DE ROSETTA	30
TABLA 3. INSTITUCIONES PARTICIPANTES EN EL CONSORCIO DE GIADA	33
TABLA 4. PROPIEDADES FÍSICAS MEDIDAS POR CADA MÓDULO GIADA.	38
TABLA 5. MODOS OPERATIVOS DE GIADA Y SUBSISTEMAS ACTIVOS.	38
TABLA 6. LÍMITES EN EL SENSADO DE GIADA.	56
TABLA 7. TABLA DE INTERRUPCIONES Y PRIORIDADES EN GIADA.	62
TABLA 8. MODOS OPERATIVOS Y VOLUMEN DE DATOS EN GIADA.	65
TABLA 9. TELECOMANDOS Y TELEMETRÍAS IMPLEMENTADOS EN GIADA.	74
TABLA 10. TELECOMANDOS ACEPTADOS Y TELEMETRÍAS GENERADAS RESPECTO AL MODO DE OPERACIÓN	77
TABLA 11. PRESTACIONES GENERALES DEL SOFTWARE DE GIADA VERSIÓN 2.3.	83
TABLA 12. ESTIMACIÓN DE TIEMPOS DE LA INTERRUPCIÓN DE GDS.	83
TABLA 13. ESTIMACIÓN DE TIEMPOS DE LA INTERRUPCIÓN DE IS.	83
TABLA 14. ESTIMACIÓN DE TIEMPOS DE LA INTERRUPCIÓN DE MBS.	83
TABLA 15. ESTIMACIÓN DE TIEMPOS EN EL ENVÍO DE TM.	83
TABLA 16. ESTIMACIÓN DE TIEMPOS DEL PROCESAMIENTO DE LA COLA DE EVENTOS.	84
TABLA 17. TIPOS DE INFORMACIÓN REPRESENTABLE EN MODELOS BASADOS EN LA TEORÍA DE LA POSIBILIDAD.	97
TABLA 18. TIPOS DE INFORMACIÓN REPRESENTABLE EN GEFRED.	99
TABLA 19. COMPARADORES DIFUSOS GENERALIZADOS SOBRE UN REFERENCIAL ORDENADO EN FIRST.	105
TABLA 20. REPRESENTACIÓN DE DIFUSOS TIPO 2 EN UN SGBDR.	132
TABLA 21. REPRESENTACIÓN DE DIFUSOS TIPO 3 EN UN SGBDR.	133
TABLA 22. CÁLCULO DEL GRADO DE COMPATIBILIDAD CON COMPARADORES DIFUSOS GENERALIZADOS DE POSIBILIDAD.	134
TABLA 23. CÁLCULO DEL GRADO DE COMPATIBILIDAD CON COMPARADORES DIFUSOS GENERALIZADOS DE NECESIDAD.	136
TABLA 24. CÁLCULO DEL GRADO DE COMPATIBILIDAD CON OPERADORES LÓGICOS.	137
TABLA 25. RELACIÓN ASOCIADA A UN PREDICADO CONSTANTE.	165
TABLA 26. ATRIBUTOS DE LA TABLA GDB_GDSEVENT UTILIZADOS EN EL EJEMPLO DE USO DEL MODELO GDB.	168
TABLA 27. ATRIBUTOS DE LA TABLA GDB_ISEVENT UTILIZADOS EN EL EJEMPLO DE USO DEL MODELO GDB.	168
TABLA 28. TABLA FUZZY_COL_LIST EN EL EJEMPLO DE USO DEL MODELO GDB.	169
TABLA 29. TABLA FUZZY_APPROX_MUCH EN EL EJEMPLO DE USO DEL MODELO GDB.	169
TABLA 30. TABLA DED_INT_TABLE_DESCRIPTION EN EL EJEMPLO DE USO DEL MODELO GDB.	170
TABLA 31. TABLA DED_INTENSIVA_CATALOG EN EL EJEMPLO DE USO DEL MODELO GDB.	170
TABLA 32. TABLA DED_COMPARISON_DESCRIPTION EN EL EJEMPLO DE USO DEL MODELO GDB.	171
TABLA 33. TABLA DED_PREDICATE_DESCRIPTION EN EL EJEMPLO DE USO DEL MODELO GDB.	171
TABLA 34. TABLA DED_RULE_DESCRIPTION EN EL EJEMPLO DE USO DEL MODELO GDB.	171
TABLA 35. TABLA GDB_GDSEVENT EN EL EJEMPLO DE USO DEL MODELO GDB.	173
TABLA 36. TABLA GDB_ISEVENT EN EL EJEMPLO DE USO DEL MODELO GDB.	173
TABLA 37. TABLA RESULTADO GDB_INT_TESIS_EXAMPLE EN EL EJEMPLO DE USO DEL MODELO GDB.	174
TABLA 38. TABLA RESULTADO GDB_INT_TESIS_EXAMPLE EN EL EJEMPLO DE USO DEL MODELO GDB.	175
TABLA 39. RANGO DE MEDIDAS DE TIEMPO Y VELOCIDAD OBTENIDAS PARTIENDO DEL TIEMPO EN CORTINA LÁSER Y DEL TIEMPO DE VUELO GDS-IS.	214
TABLA 40. TABLA GDB_GDSISEVENT CON DATOS SINTÉTICOS PARA LA VERIFICACIÓN DE LA REGLA DE COHERENCIA.	229
TABLA 41. RESULTADO DE LA APLICACIÓN DE LA REGLA DE COHERENCIA CON CALIDAD MÍNIMA 0.5.	230
TABLA 42. CALIDAD DE LOS EVENTOS GDS A LA IZQUIERDA Y A LA DERECHA DEL INTERVALO DEFINIDO POR UN EVENTO IS CON TIEMPO 683.	235
TABLA 43. CÁLCULO DE LA REGLA DE POBLACIONES PARA UN EVENTO IS CON TIEMPO 683 CUENTAS.	238
TABLA 44. TABLA GDB_GDSEVENT PARA LA REGLA DE POBLACIONES.	239
TABLA 45. TABLA GDB_ISEVENT PARA LA REGLA DE POBLACIONES.	240
TABLA 46. RESULTADO DE LA APLICACIÓN DE LA REGLA DE POBLACIONES CON CALIDAD MÍNIMA 0.5.	240
TABLA 47. TABLA GDB_GDSEVENT CON DATOS DE CALIBRACIÓN.	243
TABLA 48. TABLA GDB_ISEVENT CON DATOS DE CALIBRACIÓN.	243
TABLA 49. TABLA GDB_GDS_ISEVENT CON DATOS DE CALIBRACIÓN.	243

TABLA 50. ANOTACIONES MANUALES EN LAS SESIONES CALIBRACIÓN. _____	244
TABLA 51. RESULTADO DE LA REGLA DE COHERENCIA APLICADA A LOS DATOS DE CALIBRACIÓN CON CALIDAD MÍNIMA 0.5. _____	245
TABLA 52. RESULTADO DE LA REGLA DE POBLACIONES APLICADA A LOS DATOS DE CALIBRACIÓN CON CALIDAD MÍNIMA 0.1. _____	246
TABLA 53. CARACTERÍSTICAS PRINCIPALES DEL COMETA 67P/CHURYUMOV-GERASIMENKO. _____	304

Siglas, acrónimos y definiciones

μ	Micro (10^{-6})
μm	Micro metros
μs	Micro segundos
6CG	67P/Churyumov-Gerasimenko
AC	Atributo de Compatibilidad
ADC	Analog to Digital Converter
Aphelio	Posición del cometa más alejada al sol
B	Byte
BD	Base de Datos
BDHE	Base de Datos de Hechos Extensivos
BDR	Base de Datos Relacional
BDRD	Base de Datos Relacional Difusa
BH	Base de Hechos
BHE	Base de Hechos Extensivos
BR	Base de Reglas
CD	Conjunto Difuso
CDG	Comparador Difuso Generalizado
cm	centímetros
cm²	centímetros cuadrados
Coma	Atmósfera que rodea a un cometa y está compuesta de gases y pequeños trozos de material sólido que aparecen por la acción de la radiación solar
CPU	Central Processing Unit
CSIC	Consejo Superior de Investigaciones Científicas
DDG	Dominio Difuso Generalizado
DDL	Data Definition Language
DFSQL	Deductive FSQL
DML	Data Manipulation Language
DMP	Distribución de la Masa de las Partículas
DP	Distribución de Posibilidad

DTP	D istribución de T amaño de las P artículas
DV	D isco V irtual
DVD	D igital V ersatile D isc
DVP	D istribución de V elocidad de las P artículas
EDAC	E rror D etector A nd C orrector
EGSE	E lectrical G round S upport E quipment
EL	E tiqueta L ingüística
EP	E lectrónica P rincipal
Eq	E cuación
ER	E squema de R elación
ESA	E uropean S pace A gency
ESOC	E uropean S pace O perations C entre
fbf	fórmula b ien f ormada
FC	F ichero de C ontexto
FCP	F light C ontrol P rocedure
FIFO	F irst I n F irst O ut
Fly-by	Maniobra asistida por gravedad de una sonda espacial
FMB	F uzzy M etaknowledge B ase
FPGA	F ield P rogrammable G ate A rray
FSQL	F uzzy S QL
FT	F light T ime
G	GIADA
GA	G rado de A coplamiento
GDB	GIADA D ata B ase
GDB-GUI	GDB G raphical U ser I nterface
GDS	G rain D etection S ystem
GIADA	G rain I mpact A nalyser and D ust A ccumulator
h	hora
HK	H ousekeeping
HW	H ardware

IAA	Instituto de Astrofísica de Andalucía
IdBIS	Intelligent dataBases and Information Systems
IRQ	Interrupt ReQuest
IS	Impact Sensor
ISR	Interrupt Service Routine
KHz	Kilo Herzio
KiB	KibiByte
Kib	Kibibit
Km	Kilómetro
LEP	Llave Externa Primaria
m	metro
MB	Micro Balance
MBS	Micro Balance System
MD	Membership Degree
ME	Main Electronics
min	minuto
mm	milímetro
MR	Modelo Relacional
mV	miliVoltios
N	Newton
NASA	National Aeronautics and Space Administration
ND	Número Difuso
NVRAM	No Volátil RAM
OAC	Osservatorio Astronomico di Capodimonte
OBCP	On-Board Control Procedude
°C	grados Centígrados
OSIRIS	Optical, Spectroscopic and InfraRed Imaging System
p.e.	por ejemplo
PD	Peak Detector
PDF	Portable Document Format

PDS	Planetary Data System
PE	Proximity Electronic
Perihelio	Posición del cometa más cercana al sol
PZT	PieZo-ElecTrico
R	Rosetta
RAM	Random Access Memory
RDG	Relación Difusa Generalizada
RGD	Regla Generalizada Difusa
RGGA	Regla Generalizada con Grado Acoplamiento
RI	Restricción de Integridad
ROM	Read Only Memory
ROSIS	Rosetta Spacecraft Interface Simulator
s	segundo
SC	SpaceCraft Rosetta
SGBD	Sistema Gestor de Base de Datos
SGBDR	Sistema Gestor de Base de Datos Relacional
SQL	Structured Query Language
SW	Software
TC	TeleComando
TCD	Teoría de Conjuntos Difusos
TM	TeleMetría
UA	Unidades Astronómica. Es una unidad de distancia que es aproximadamente igual a la distancia media entre la Tierra y el Sol y cuyo valor, es alrededor de 149.597.870 Km
UD	Universo de Discurso
UML	Unified Modeling Language

Notación y reglas generales

a) La tipografía utilizada en esta memoria es la siguiente:

- 1) Texto normal, sin cursiva ni negrita: esto es un texto normal.
- 2) Referencia a una sección, ilustración, tabla, definición o fórmula de la memoria, en cursiva y subrayada: 2 *GIADA, un instrumento en la misión Rosetta*.
- 3) Referencia a bibliografía, en negrita: **[Pon97]**.
- 4) Palabras o frases con especial interés, en cursiva, negrita, comillas simples o dobles: *atributo de compatibilidad*, “calidad”.
- 5) Código fuente, secuencias hexadecimales y sentencias SQL o DFSQL, en un tipo de fuente mono-espaciada: `create intensiva table.`

b) El texto se encuentra principalmente en español, pero aparecen algunos términos en inglés que no tienen traducción, como es el caso de “frangibolt”, o cuyo uso es ampliamente utilizado; “script” o “buffer”. Además, el Apéndice A y algunas ilustraciones, se encuentran también en inglés, en ambos casos por ser de especial relevancia para los miembros del consorcio internacional de desarrollo de GIADA.

c) Los valores numéricos se expresan por defecto en base diez, excepto cuando se indique explícitamente lo contrario o cuando los preceda el prefijo 0x, en tal caso, indican base 16: 0x53.

1 Introducción

Los cometas son, probablemente, los cuerpos celestes que más han cautivado a la humanidad desde la más remota antigüedad. El gran avance experimentado en el estudio de estos cuerpos durante los últimos treinta 30 años, se debe en gran parte, a la utilización de instrumentación a bordo de sondas espaciales. Una de estas sondas, es la que conforma la misión Rosetta de la Agencia Espacial Europea (ESA).

Rosetta toma su nombre de la piedra basáltica hallada en 1799 por el soldado Pierre-François Bouchard durante la campaña francesa en Egipto. Esta piedra fue el elemento clave para poder descifrar la escritura jeroglífica egipcia, de la misma manera, se espera que Rosetta sea la llave que permita abrir la puerta de los “misterios” que entrañan los cometas. Las actuales hipótesis sobre las primeras etapas de la evolución del Sistema Solar, identifican a los cometas como restos de la formación de nuestro sistema planetario. Los cometas mantendrían un registro de las actividades físicas y químicas que acontecieron en estas etapas iniciales, dado que los únicos procesos que habrían alterado su composición serían los derivados de la radiación solar. En definitiva, el material comenario contendría una información única sobre las fuentes que contribuyeron a la nebulosa protosolar, así como sobre los procesos de condensación que dieron a lugar a la formación de planetesimales y posteriormente a los planetas, incluyendo la Tierra, por lo que son objetivos de estudio de primer orden.

El Instituto de Astrofísica de Andalucía (IAA) del Consejo Superior de Investigaciones Científicas (CSIC) ha participado en el diseño y la construcción de dos de los instrumentos embarcados en el orbital de Rosetta: OSIRIS (Optical, Spectroscopic and InfraRed Imaging System) y GIADA (Grain Impact Analyser and Dust Accumulator). La contribución del IAA al proyecto Rosetta, se conjuga en dos grandes vertientes, una a nivel científico, y otra a nivel de ingeniería, en concreto en la construcción de parte de los sistemas electrónicos (GIADA y OSIRIS) y de la totalidad del software embarcado (GIADA).

Por otro lado, el área de investigación desarrollada en los últimos veinte años en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada, dedicado al tratamiento de la información imprecisa (difusa) y deductiva, ha generado una evolución de modelos de base de datos que han ido heredando y acrecentando sus capacidades de forma progresiva.

Esta memoria une estos campos de estudio, a priori alejados entre sí: el análisis de los datos científicos proporcionados por el cometa y los modelos de bases de datos difusos deductivos. Para ello se ha diseñado y construido el programa de computador GDB-GUI. Este programa permite almacenar, manipular y explotar los datos y el conocimiento adquiridos durante la construcción de GIADA, utilizando para ello una adaptación del modelo de base de datos difuso deductivo.

1.1 Motivación y objetivos

Los datos del cometa 67P/Churyumov-Gerasimenko, obtenidos in situ por la sonda espacial Rosetta y sus instrumentos, son una fuente extremadamente valiosa de información, no sólo por su repercusión científica sino también por el enorme esfuerzo internacional humano y económico que ha sido necesario para su obtención. Cualquier herramienta que permita analizar, explotar y validar estos datos, se convierte en un motivo loable y necesario que pone en valor el trabajo realizado durante la última década por ingenieros, científicos e instituciones.

El objetivo principal de esta memoria, consiste en proporcionar una herramienta que permita representar el conocimiento experto acerca del instrumento GIADA. Esta herramienta será aplicada sobre los datos científicos obtenidos, permitiendo verificar y calificar los resultados, a fin de profundizar en el estudio de la dinámica del polvo comenario. Para realizar esta tarea, se han utilizado tres elementos principales: el software de GIADA, el modelo de base de datos y la interfaz GDB-GUI.

Como primer elemento, se encuentra el trabajo realizado por el autor de esta memoria, centrado en el desarrollo de software del instrumento GIADA. Los detalles precisos de cómo se realiza la detección de partículas de polvo, servirán como fuente de conocimiento experto, que será representado mediante el modelo de base de datos y se utilizará posteriormente en la validación de GDB-GUI.

El segundo elemento lo constituye un modelo de base de datos difuso deductivo. Este modelo teórico y su adaptación a las necesidades para la explotación de datos científicos de GIADA, constituye la herramienta de representación y explotación de todo el conocimiento experto de GIADA.

La pieza que unifica, centraliza y hace accesible a un usuario los dos elementos anteriores es GDB-GUI, una interfaz para el modelo de base de datos difuso deductivo. Permite importar los datos de partículas cometaria (reales, simuladas o de calibración), expresar el conocimiento experto utilizando las reglas definidas por el modelo y crear conjuntos de datos de entrada controlados, que serán posteriormente utilizados en la validación de dicho conocimiento. Una vez almacenada la información, podrá ser consultada, permitiendo obtener nuevos datos. Tanto la consulta como el almacenamiento pueden llevarse a cabo en términos tanto difusos como precisos. Los resultados obtenidos, en el caso de incorporar componentes difusos, serán calificados numéricamente con un determinado grado de validez o ajuste, calculado a partir de las directrices del modelo.

1.2 Estructura de la memoria

Esta memoria se compone de diez capítulos que se describen a continuación.

En el capítulo primero se indica la motivación y objetivos que han guiado el trabajo presentado en ese documento.

La misión espacial Rosetta y uno de sus instrumentos, GIADA, centran el contenido del capítulo segundo. Tras un resumen de los hitos de Rosetta, se presentan los objetivos científicos de GIADA, su interacción con otros instrumentos a bordo de Rosetta, su funcionamiento, las prestaciones conseguidas, su estado tras diez años de travesía y los primeros resultados científicos obtenidos.

En el capítulo tercero se describen los aspectos principales del software de GIADA. Cómo se ha desarrollado, su arquitectura, la interfaz con el hardware, el contenido de los diferentes tipos de memoria, los diferentes modos de operación, el proceso de lectura de los sistemas de detección de partículas, el mantenimiento remoto, la ingeniería del software aplicada, la estructura de los datos generados y las prestaciones alcanzadas.

Los modelos de bases de datos antecesores del modelo utilizado en esta memoria, se describen en el capítulo cuarto: el modelo relacional, el relacional difuso, el modelo lógico, el lógico difuso y el modelo integrado difuso y deductivo.

El modelo de base de datos GDB es el marco teórico de representación y manejo del conocimiento utilizado en esta memoria y es descrito en el capítulo quinto. Se detalla cómo se representa y procesa la información difusa y deductiva, así como los diversos aspectos de la implementación, se presenta un ejemplo de uso de reglas y se describe la arquitectura del modelo.

El capítulo sexto se centra en la descripción de la interfaz de usuario del modelo de base de datos GDB, GDB-GUI: tipo de fuentes de datos aceptadas, interfaz visual con el usuario, proceso de construcción, almacenamiento y recuperación de consultas, importador de datos, herramientas para trabajar con curvas de calibración, descripción de los ficheros de entrada y salida, gestión de la base de datos relacional, adaptación a otros sistemas, nomenclatura de las tablas utilizadas, funcionamiento del enlace automático de tablas y las herramientas software utilizadas en su desarrollo.

La verificación del modelo de base de datos y de la interfaz GDB-GUI es descrito en el capítulo séptimo. Se comienza ampliando el conocimiento acerca de cómo se realiza la detección y construcción de un tipo especial de eventos. A continuación se crearán dos reglas que aprovechan el conocimiento descrito. Se seguirá con la generación a medida de un conjunto de datos de entrada, que permitirán calcular con antelación los resultados, para, finalmente, contrastarlos con los calculados mediante GDB-GUI.

Las conclusiones del desarrollo de esta memoria y los caminos a seguir en el futuro se desganan en el capítulo octavo.

Los artículos, libros y tesis usadas como referencia se listan en el capítulo noveno.

El capítulo décimo cierra esta memoria, con un conjunto de apéndices: sintaxis y semántica de las tablas utilizadas, contenido del DVD anexo a este documento, una introducción a los cometas y una breve descripción de los aspectos más relevantes del objetivo final de esta memoria: el cometa 67P/Churyumov-Gerasimenko.

2 GIADA, un instrumento en la misión Rosetta

En este capítulo se describe la misión espacial Rosetta y uno de sus instrumentos G. Se detallan los objetivos científicos de G., su relación con el resto de instrumentos de Rosetta y se analiza el funcionamiento de los principales elementos del hardware (HW) de G., sus prestaciones, estado tras once años de travesía y los primeros resultados obtenidos.

2.1 La misión Rosetta

Rosetta ([Bar93], [Bie02], [Hec99], [Sche98], [Sch99] y [Vil97]) es una misión de la Agencia Espacial Europea (ESA) que se enmarca dentro del programa científico HORIZON 2000. Fue lanzada el 2 de marzo del 2004 desde Kourou, en la Guayana Francesa, a bordo de un cohete lanzadera Ariane V y uno de sus instrumentos (Philae) se posó sobre la superficie del cometa el 12 de noviembre de 2014. El objetivo principal Rosetta, es el de investigar in situ la composición y la estructura del núcleo de un cometa (ver *Apéndice C. Introducción a los cometas*).

El lanzamiento inicial, con destino al cometa 46P/Wirtanen, estaba previsto para enero del 2003, pero tuvo que ser retrasado, debido a una explosión en el lanzador de la misión inmediatamente anterior. Este hecho, junto a las reducidas ventanas temporales para alcanzar la trayectoria del 46P/Wirtanen, provocó un cambio de objetivo, siendo finalmente elegido el cometa 67P/Churyumov-Gerasimenko (6GC [Alt15], [Kro03], [Sie15] y *Apéndice D*).

Rosetta se aproximó al cometa en 2014, tras realizar una asistencia gravitatoria en Marte, tres en la Tierra y llevando a cabo dos visitas rápidas al cinturón de asteroides del Sistema Solar (situado entre Marte y Júpiter). Tras entrar en órbita alrededor del cometa, a unas 3 unidades astronómicas (UA) y observar su núcleo a sólo unas decenas de kilómetros, Rosetta viajará junto él hasta su paso por el perihelio, en diciembre del 2015. La misión ha sido extendida hasta septiembre de 2016, lo que permitirá estudiarlo en la fase de decrecimiento de su actividad conforme se produzca su alejamiento del Sol.

La siguiente ilustración muestra el modelo de vuelo de Rosetta, y continuación se incluye una tabla con las fechas e hitos de la travesía.



Ilustración 1. Rosetta en ESTEC. Créditos: ESA/ESTEC.

Tabla 1. Hitos de Rosetta.

Hito	Fecha nominal
<i>Lanzamiento</i>	2 marzo 2004
<i>1ª Asistencia gravitatoria en la Tierra</i>	4 marzo 2005
<i>Asistencia gravitatoria en Marte</i>	25 febrero 2007
<i>2ª Asistencia gravitatoria en la Tierra</i>	13 noviembre 2007
<i>Acercamiento a Steins</i>	5 septiembre 2008
<i>3ª Asistencia gravitatoria en la Tierra</i>	13 noviembre 2009
<i>Acercamiento a Lutetia</i>	10 julio 2010
<i>Comienzo de la hibernación en espacio profundo</i>	8 junio 2011
<i>Fin de la hibernación en espacio profundo</i>	20 enero 2014
<i>Maniobras para el encuentro con el cometa</i>	De mayo a agosto del 2014
<i>Llegada al cometa</i>	6 agosto 2014
<i>Aterrizaje del "Lander" Philae</i>	12 noviembre 2014
<i>Acompañamiento del cometa</i>	Desde noviembre 2014 a diciembre 2015
<i>Fin de misión</i>	30 septiembre 2016

Los instrumentos a bordo de Rosetta (**Bie021**), su propósito y su principal responsable, se detallan en la siguiente tabla (se han omitido los 10 instrumentos presentes en el módulo de descenso Philae):

Tabla 2. Instrumentos a bordo de Rosetta

INSTRUMENTO	PROPÓSITO	INVESTIGADOR PRINCIPAL
ALICE	Espectrógrafo de imágenes ultravioleta	Dr. Alan Stern. Southwest Research Institute, Boulder, Colorado ,USA
CONSERT	Medición de ondas de radio y tomografía del núcleo	Prof. Wlodek Kofman. Ecole Nationale Supérieure d'Ingenieurs, CNRS, Grenoble, Francia
COSIMA	Espectrómetro de masa de polvo	Dr. Jochen Kissel. Max-Planck-Institut für Extraterrestrische Physik, Garching, Alemania
GIADA	Medidas del flujo, distribuciones y propiedades físicas de las partículas de polvo	Inicialmente:

		<p>Prof. Luigi Colangeli. Head of Solar System Missions Division / Research and Scientific Support Department (SRE-SM). ESA.</p> <p>En la actualidad:</p> <p>Dr. Alessandra Rotundi. Università degli Studi di Napoli "Parthenope", Nápoles, Italia</p>
MIDAS	Análisis de micro-imágenes de polvo	Prof. Willi Riedler. Space Research Institute, Graz, Austria
MIRO	Espectrómetro de microondas	Dr. Samuel Gulkis. Jet Propulsion Laboratory Pasadena, California, USA
OSIRIS	Captación de imágenes	<p>Inicialmente:</p> <p>Dr. Horst Uwe Keller. Max-Planck-Institut für Aeronomie, Katlenburg Lindau, Alemania</p> <p>En la actualidad:</p> <p>Dr. Holger Sierks Max Planck Institute for Solar System Research, Göttingen, Alemania</p>
ROSINA	Espectrómetro de masas iónicas	Prof. Hans Balsiger. University of Bern, Suiza
RPC	Medida de plasma	<p>Dr. Rolf Boström. Swedish Inst. of Space Physics, Uppsala, Suecia.</p> <p>Dr. James Burch . Southwest Research Institute, San Antonio, Texas, USA.</p> <p>Prof. Karl-Heinz Glassmeier. Technische Universität, Braunschweig, Alemania.</p> <p>Prof. Rickard Lundin. Swedish Institute of Space Physics, Kiruna, Suecia.</p> <p>Dr. Jean-Gabriel Trotignon. LPCE/CNRS, Orleans, Francia</p>
RSI	Comunicaciones con la Tierra	Dr. Martin Pätzold. Universität Köln, Alemania

VIRTIS	Espectrógrafo de imágenes térmicas infrarrojas y visibles	Dr. Angioletta Coradini. Istituto di Astrofisica Spaziale, CNR, Rome, Italia
LANDER	Estudio In-Situ del núcleo	Dr. Stephan Ulamec, DLR, Köln Porz-Wahn, Alemania. Prof. Denis Moura, CNES, Toulouse, Francia. Dr. R. Mugnuolo, Italian Space Agency, Matera, Italia

La posición de cada instrumento en Rosetta y una imagen de Marte tomada desde Philae se detallan en las siguientes dos ilustraciones.

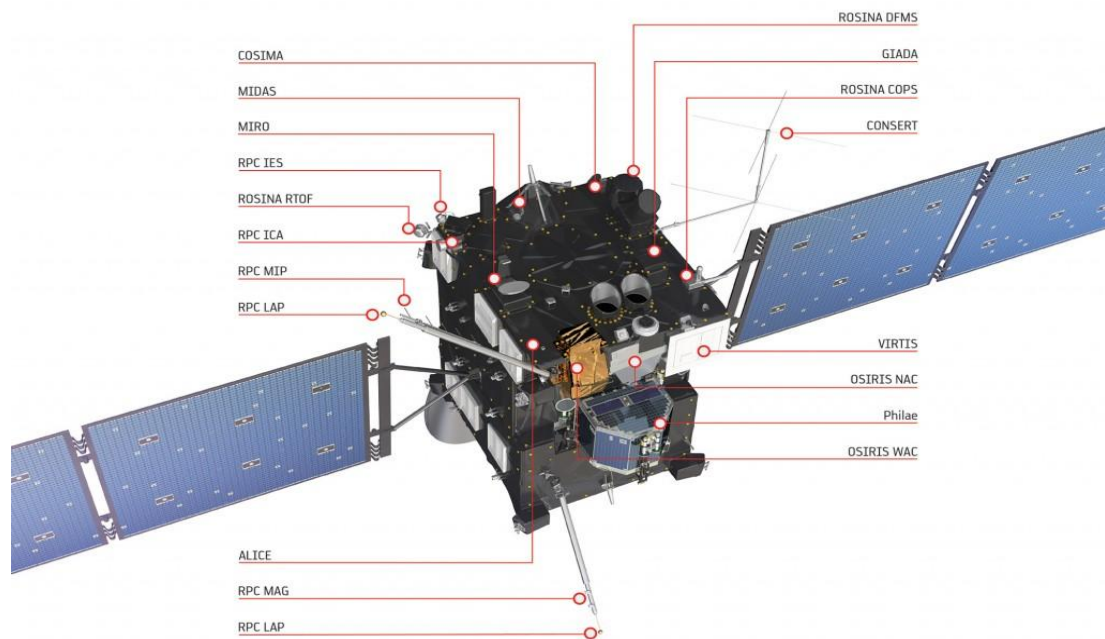


Ilustración 2. Localización de los instrumentos en Rosetta. Créditos: ESA/ATG medialab.

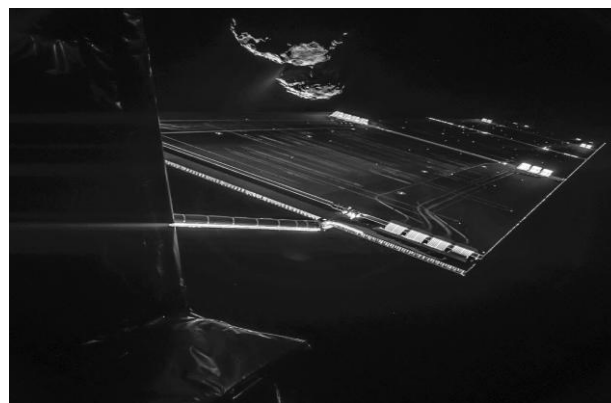


Ilustración 3. Vista de 6CG desde el instrumento Philae de Rosetta. Créditos: ESA/Rosetta/Philae/CIVA.

2.2 El instrumento GIADA

GIADA, acrónimo de “Grain Impact Analyser and Dust Accumulator” ([Bus00], [Bus99], [Col04], [Col06], [Col07], [Ful10] y [Rot15]) es uno de los experimentos a bordo de la misión espacial Rosetta y responde a la necesidad de realizar estudios “in situ” de las propiedades físicas del polvo cometario. En la siguiente ilustración se muestra una fotografía del modelo de vuelo del instrumento.

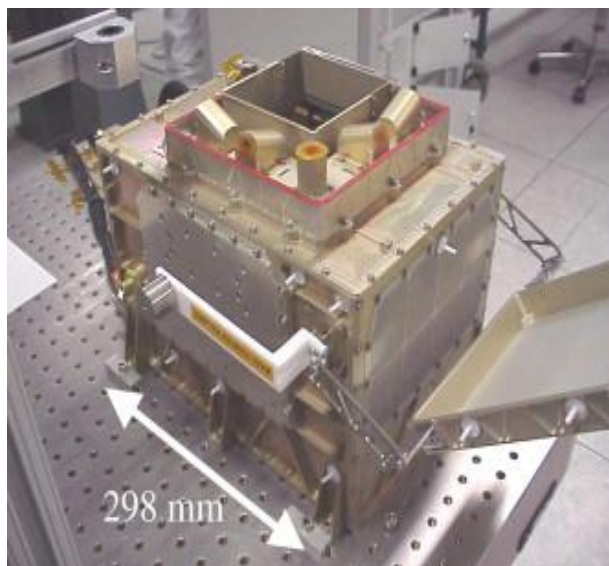


Ilustración 4. Modelo de vuelo de GIADA.

De forma resumida, el objetivo científico de G., consiste estudiar un conjunto de propiedades físicas de las partículas de polvo cometario y su evolución. Para ello, cuenta con diferentes sensores capaces de medir las siguientes magnitudes de las partículas: masa, velocidad, momento lineal, flujo y algunas propiedades ópticas. Todas estas propiedades, y aquellas que se pueden deducir de ellas como la masa de cada partícula, se están estudiando a lo largo de la trayectoria del cometa, por lo que servirán para cuantificarlas a distintas distancias del Sol. Los datos obtenidos por G., proporcionarán la base primordial para el cálculo y la evolución de la relación polvo/gas emitido por 6GC. Esta relación es de capital importancia para el estudio de los objetos celestes que son origen del Sistema Solar. Es importante resaltar, que el tratamiento de los datos obtenidos es estadístico, en lugar de un análisis individual de partículas observadas (ver [2.2.4 Prestaciones de GIADA](#)).

Para el desarrollo y la construcción de G., se creó un consorcio internacional, liderado por el Profesor Luigi Colangeli (y actualmente dirigido por la Dr. Alessandra Rotundi) cuya composición y responsabilidades se detallan en la siguiente tabla.

Tabla 3. Instituciones participantes en el consorcio de GIADA

Institución	Atribuciones
Istituto Universitario Navale (IUN), Nápoles, Italia.	Definición del instrumento y construcción de los subsistemas.
Osservatorio Astronomico di Capodimonte (OAC), Nápoles, Italia	
Università "Parthenope", Nápoles, Italia	

Universidad de Padova (UPD), Pádua, Italia	Simulación termo-mecánica preliminar, análisis y diseño
Instituto de Astrofísica de Andalucía (IAA-CSIC), Granada, España	Desarrollo de la electrónica principal y del software embarcado. Co-investigador principal Prof. José Juan López-Moreno

En las dos siguientes ilustraciones se detalla la posición de G. en Rosetta y un momento de la fase de integración de G. en Rosetta.

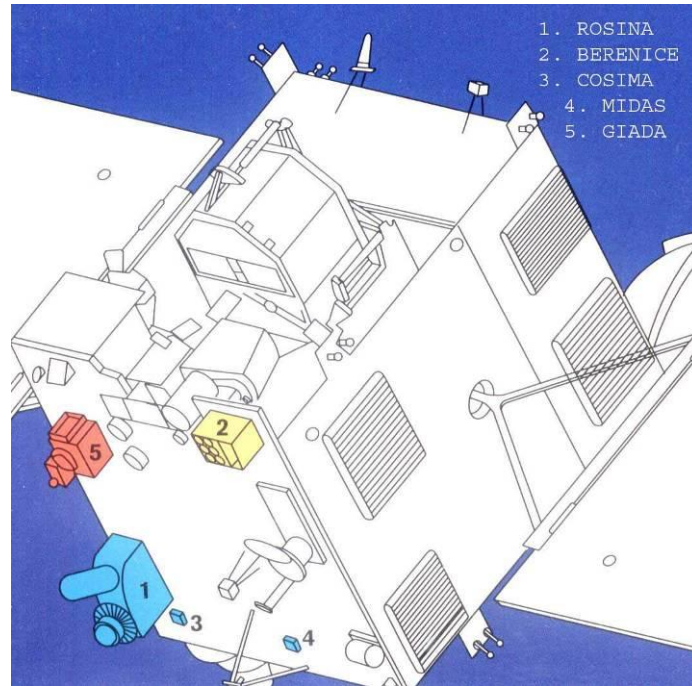


Ilustración 5. Esquema de la posición de GIADA en Rosetta.



Ilustración 6. GIADA integrada en Rosetta en las instalaciones de ESA en ESTEC.

En los siguientes apartados se describen los objetivos científicos de G., su relación con el resto de instrumentos de Rosetta y su funcionamiento a nivel de hardware (HW). El software (SW) de G. se describe en el capítulo 3.

2.2.1 Objetivos científicos de GIADA

Los modelos teóricos de comportamiento del cometa ([Cri97], [Ful99]) han sido, hasta la fecha, desarrollados a partir de observaciones terrestres (limitadas por la distancia al objeto y por la atmósfera terrestre) o mediante los datos procedentes de las misiones Giotto y Stardust.

La misión espacial Giotto con el experimento Didsy ([McD81]) fue la primera en proporcionar datos reales de flujo de partículas de los cometas 1P/Halley y 26P/Grigg-Skjellerup. Stardust, en 2004, fue la siguiente misión en visitar cometas, el 81P/Wild2 ([Gre04]) y el 9P/Tempel ([Eco13]). En ambas misiones, la velocidad relativa de la sonda con respecto al cometa era demasiado alta, por lo que los datos no fueron satisfactorios. Este problema lo resuelve Rosetta ya que su velocidad es menor que la velocidad del polvo.

Se resumen a continuación las medidas científicas proporcionadas por G.:

- a) **Medida de flujo de polvo cometario y de las propiedades dinámicas de las partículas.** El modelo más utilizado para determinar la evolución de las partículas de polvo en el entorno de un cometa, está basado en la suposición de una emisión estacionaria procedente de un único origen y en un medio isotrópico. Esta suposición no se ajusta a la realidad, donde parámetros como radiación solar, anisotropías, zonas activas, zonas muertas, etcétera han de ser tenidos en cuenta. Para cada posición en el espacio con respecto al cometa, se deben considerar dos conjuntos de partículas cometarias (*Apéndice D. El cometa 67P/Churyumov-Gerasimenko*): aquellas que llegan a la zona de detección directamente del núcleo (partículas directas) y las que llegan debido a la acción de la presión de la radiación solar (partículas reflejadas). Estos dos conjuntos de partículas se caracterizan por una evolución dinámica diferente. En el caso de Rosetta, donde la velocidad de la sonda es mucho menor que la velocidad de escape de las partículas de la superficie del núcleo cometario, las partículas directas y reflejadas pueden ser reconocidas y recogidas simultáneamente desde sensores dispuestos en diferentes direcciones. Mediante los datos obtenidos por estos sensores (**2.2.3.1.3 GIADA-3: MBS**), se podrá determinar la distribución del tamaño de las partículas (DTP) que, a su vez, influye fuertemente en la estimación de la pérdida de masa del cometa ([Ful95]).
- b) **Medida de la distribución del momento lineal y la velocidad de las partículas.** Para un tamaño de partículas determinado, se puede esperar una amplia distribución de sus velocidades. Gracias a G., será posible medir el momento lineal (masa por velocidad) de las partículas mediante el subsistema “Impact Sensor” (IS, ver **2.2.3.1.1.2 IS**). Además, el subsistema “Grain Detector System” (GDS, ver **2.2.3.1.1.1 GDS**) permitirá obtener la velocidad de las mismas. Procesando los datos proporcionados por GDS e IS, se podrá calcular la masa de cada partícula. Para un ángulo de visión suficientemente pequeño en la dirección del núcleo y para la mayoría de las posiciones relativas del sol, la sonda y el núcleo, las partículas observadas por G. serán con una muy alta probabilidad, “directas”. Por primera vez, será posible medir “in-situ” la distribución de la velocidad de las partículas (DVP) con respecto a la masa, su dispersión y la relación entre la velocidad más probable y su masa. La DTP influye de forma directa en la DVP, pudiendo encontrarse partículas de igual tamaño con diferentes velocidades, por lo que se espera una DVP de amplia variabilidad.
- c) **Determinación de la relación entre gas y polvo.** La relación entre polvo y gas es uno de los parámetros más relevantes en el estudio de un cometa. La monitorización del flujo de polvo permite una estimación bastante precisa de ésta relación a la largo de la

trayectoria de Rosetta. G. realizará esta labor con el subsistema “Micro Balance System” (MBS, ver [2.2.3.1.3 GIADA-3: MBS](#)).

- d) **Estudio de la evolución de polvo cometario en términos de variación de la distribución de tamaños y su correlación con las áreas activas de emisión.** Los modelos teóricos indican la existencia de áreas activas del núcleo donde se generarán chorros de partículas expulsadas a gran velocidad (“jets”). Estos jets han sido confirmados por el instrumento OSIRIS ([Tabla 2. Instrumentos a bordo de Rosetta](#)). Las variaciones de flujo medidas con G. en conjunción con las imágenes de las cámaras y los datos de los espectrómetros a bordo de Rosetta, podrán identificar estas áreas activas. La información de la dinámica del polvo de estas áreas, proporcionará nuevas restricciones para los actuales modelos teóricos de la evolución hidrodinámica del coma del cometa. Así mismo, los retrasos temporales entre el incremento de flujo y la detección efectiva de un jet, podrán aportar restricciones a los modelos de períodos de rotación del núcleo y a su estado del giro (“spin”).
- e) **Caracterización de la evolución temporal del polvo en relación con la evolución del núcleo.** Rosetta sigue al cometa, a relativamente poca distancia, durante su aproximación al perihelio, lo que permite a G. caracterizar la evolución temporal del entorno de polvo, en relación con la evolución del núcleo.

2.2.2 Relaciones sinérgicas de GIADA

Se produce una doble interacción, entre las medidas obtenidas por G. y el las del resto de los instrumentos a bordo de Rosetta. Por un lado, los datos de flujo de polvo proporcionados por G. ayudan a corregir e interpretar los datos de otros instrumentos ([Tabla 2. Instrumentos a bordo de Rosetta](#)). Por otro lado, los datos de otros experimentos ayudan a restringir y aclarar los resultados de G. Una correcta interpretación de imágenes ópticas e infrarrojas del coma, sólo es posible con la DVP y DTP y es G. el único experimento especialmente diseñado para tal fin. Mediante otros experimentos, como el espectrógrafo de masas ROSINA, se podrán validar y refinar los datos de velocidad y de flujo que caracterizan el gas presente ([\[Rot15\]](#))

G. también desempeña un papel importante en la salud y en la seguridad en el resto de instrumentos y en la propia Rosetta. Las superficies ópticas de los instrumentos, y en general de todo instrumento que apunte al núcleo, se contamina por el flujo de polvo. G. ayuda a predecir los ratios de deposición de polvo sobre los instrumentos que apuntan al núcleo y a tomar las decisiones apropiadas en el plan de la misión. Esta información proporcionada por G., es también aplicable a las superficies de los instrumentos que se orientan en direcciones distintas a la del núcleo. Este es el caso de los radiadores pasivos utilizados en la evacuación de calor.

Un factor crítico a considerar, es la deposición de polvo sobre los paneles solares de Rosetta, puesto que constituyen la única fuente de alimentación de los dispositivos electrónicos. Dichos paneles se contaminan con los flujos de polvo que provienen de la dirección del sol, ya que siempre están orientados hacia él. G. es capaz de proporcionar datos fundamentales para modelar adecuadamente este flujo y predecir sus prestaciones futuras.

2.2.3 Funcionamiento de GIADA

En esta sección se realiza una descripción del funcionamiento del instrumento ([\[Gia06\]](#)) desde el punto de vista del HW. El HW fue desarrollado (ver [Tabla 3. Instituciones participantes en el consorcio de GIADA](#)) en Italia (cubierta, GIADA-1 y GIADA-3) y en España (GIADA-2), más un conjunto de empresas privadas (SENER España y “Officine Galileo” en Italia).

2.2.3.1 El hardware de GIADA

El hardware del instrumento G. se divide en cuatro grandes módulos o subsistemas:

- 1) **Cubierta.** (“Cover”). Protege al resto de subsistemas de la entrada de materiales no deseados, así como de la radiación solar.
- 2) **GIADA-1.** GDS+IS. Detección de partículas de polvo.
- 3) **GIADA-2.** ME. Electrónica principal (EP) y control de todo el instrumento.
- 4) **GIADA-3.** MBS. Medida del flujo de polvo en diferentes direcciones.

La ubicación de cada uno de los módulos se presenta en la siguiente ilustración:

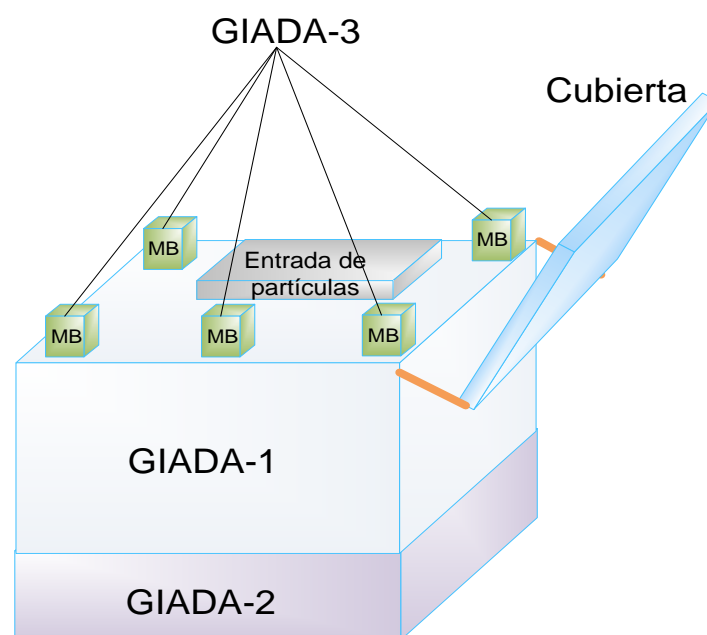


Ilustración 7. Disposición de los módulos hardware de GIADA.

A grandes rasgos, el funcionamiento del instrumento es el siguiente: cuando el dispositivo de seguridad “franfibolt” se encuentre desbloqueado y la cubierta se encuentre abierta, se permite que las partículas de polvo puedan entrar por una apertura. En dicha apertura se sitúa una cortina láser generada por conjunto de diodos láser. El tránsito por dicha cortina será detectado por los fotodetectores del GDS, poniéndose en marcha un contador, que será parado al impactar la partícula sobre la superficie de detección del IS. De esta forma se obtiene el tiempo de vuelo de la partícula. Puesto que la distancia entre la cortina láser y la superficie de impacto del IS es conocida, es posible calcular la velocidad de la partícula. Las ondas mecánicas, generadas por el impacto, serán detectadas y procesadas por cinco sensores piezoeléctricos, obteniéndose el valor absoluto del momento lineal de la partícula entrante. La masa de la partícula se obtiene indirectamente, mediante los valores de momento y velocidad. La dispersión difusa (“scattering”) de luz reflejada por la partícula y sensada por los fotodetectores, proporciona una primera estimación del tamaño y forma. Además de estas medidas, el sistema MBS medirá el flujo de polvo por deposición sobre las superficies colectoras de cinco microbalanzas. Todos estos módulos se encuentran controlados y alimentados por la electrónica principal y el SW del instrumento.

Las propiedades físicas medidas por cada módulo (exceptuando la cubierta de protección) se muestran en la siguiente tabla:

Tabla 4. Propiedades físicas medidas por cada módulo GIADA.

Medida	Subsistema
Flujo de polvo	MBS
Propiedades ópticas y de forma de las partículas individuales	GDS
Momento de partículas individuales	IS
Velocidad escalar de partículas individuales	GDS+IS
Masa de partículas individuales	GDS+IS

En la siguiente ilustración se muestra el modelo mecánico de G. y la localización de los diodos láser del GDS, el IS, las microbalanzas, la cubierta y el dispositivo “frangibolt”.

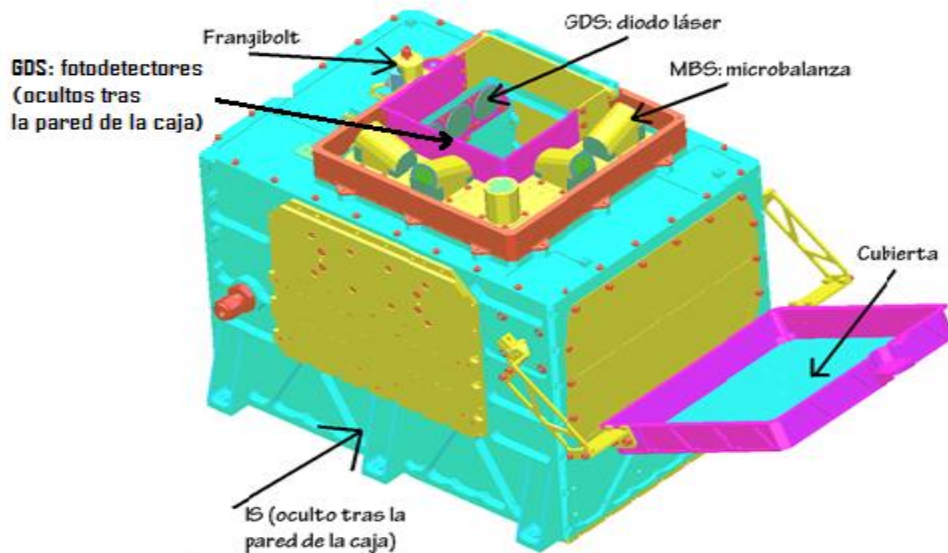


Ilustración 8. Modelo mecánico de GIADA.

G. dispone de varios modos de funcionamiento (ver [3.5 Modos operativos](#)), con el objetivo de medir las propiedades físicas indicadas en la anterior tabla. Los subsistemas activos en cada modo se indican a continuación:

Tabla 5. Modos operativos de GIADA y subsistemas activos.

Modo operativo	Subsistemas
Safe	ME
Normal	ME+GDS+IS+MBS

Flux	ME+MBS
Cover	ME+Cubierta

En el capítulo 2 se volverá a retomar los modos operativos y se detallará el comportamiento del instrumento en cada uno de ellos.

2.2.3.1.1 GIADA-1: GDS+IS

El conjunto de GDS+IS constituye el sistema de detección y medida de impactos de partículas en G. Es capaz de cuantificar el momento lineal y la velocidad escalar de partículas individuales.

El tamaño de la apertura destinada a la entrada de partículas es de 10x10 cm² y el campo de visión es de 40 grados. Cada partícula detectada, debe pasar por una cortina láser generada por el GDS, ser localizada por los fotodetectores (llamados también fotodiodos) e impactar en la superficie colectora del IS. La cortina láser y el plano colector del IS son paralelos entre sí. El diagrama del sistema de detección GDS+IS se muestra en la siguiente ilustración.

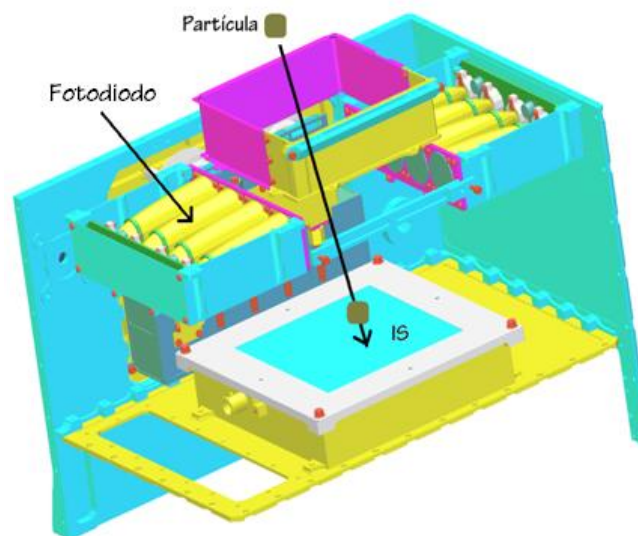


Ilustración 9. GDS+IS.

A continuación se detalla el funcionamiento individual de GDS e IS.

2.2.3.1.1.1 GDS

El GDS (**[Maz02]**) basa la detección de partículas de polvo mediante un sistema óptico. Este sistema tiene dos elementos principales: una cortina láser y un conjunto de fotodiodos destinados a medir la luz dispersada por la partícula entrante a su paso por la cortina. La cortina es generada mediante cuatro fotodiodos láser emisores de 1W de potencia, que generan luz en el rango del infrarrojo cercano y trabajan nominalmente en conmutación, alimentados a una frecuencia de 100KHz. Las dimensiones de la cortina son 85mm x 85mm de superficie y 3mm de espesor. La detección se realiza a través de 8 fotodiodos receptores situados a 90 grados de los diodos emisores. Estos receptores se agrupan, en el procesamiento de las señales, en dos canales denominados “canal derecho” y “canal izquierdo”.

Es de resaltar el hecho de que esta es la primera vez que se utiliza una cortina láser para detectar partículas en un instrumento espacial y esto constituye unas de las innovaciones de G.

El esquema de la zona de sensado del GDS se muestra en la siguiente ilustración:

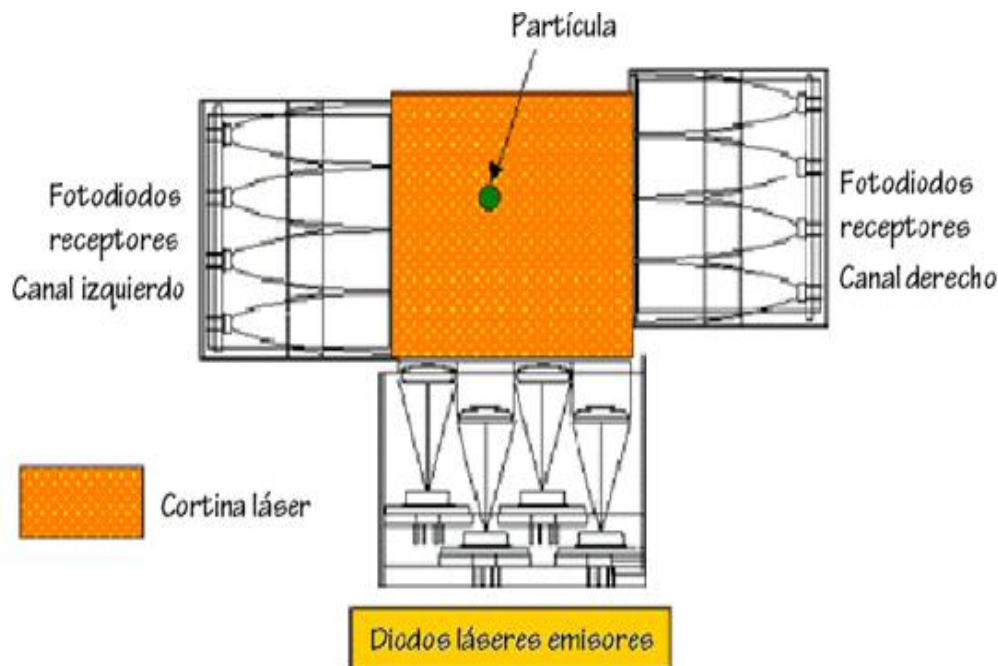


Ilustración 10. Esquema del GDS.

La partícula, al atravesar la cortina, produce luz dispersa de forma difusa (“scattering”), que será detectada por los fotodiodos receptores. Este scattering permite obtener una estimación cualitativa de la forma de la partícula (geometría roma, angulosa, redondeada etcétera) y cuantitativa respecto a su tamaño. La luz dispersada es muy débil, por lo que se ha elegido una geometría de conos (en los que se alojan los fotodiodos) del tipo “Winston”. Este tipo de geometría colectora, permite que luz incidente sobre la superficie de entrada llegue siempre hasta el foco de detección, independientemente de su ángulo de incidencia.

Por motivos de compatibilidad electromagnética y de manejo de energía, la electrónica que componen los preamplificadores de las señales de los fotodiodos y la de potencia (para la alimentación de los diodos láser) se encuentran en las cercanías de los elementos sensores y emisores. Esto es lo que se denomina “electrónica de proximidad”. Esta parte de la electrónica, además, es la encargada de mantener el control de la temperatura de los diodos láser. Este control es necesario dado que la longitud de onda de la luz emitida por los fotodiodos, es fuertemente dependiente de su temperatura de trabajo. El consecuente procesado de las señales, se realizan en la EP (GIADA-2).

Las señales de salida de los 8 fotodiodos son restadas entre sí, de dos en dos, para eliminar el ruido de modo común de fondo de la electrónica de proximidad. Las señales resultantes se dividen en dos canales, y ya en la EP, se acoplan en alterna (“AC”) a sendos amplificadores detectores de pico. El acoplo de la señal de los fotodiodos a la cadena de amplificación se realiza en AC, con el fin de eliminar la componente continua de fondo, evitando así la saturación de los amplificadores. La constante de tiempo de la red de acoplo AC, que es de $1\mu\text{s}$, ha sido establecida experimentalmente para llegar a un compromiso entre la sensibilidad y el tiempo de reacción del sistema de amplificación. Los detectores de pico utilizados, son de onda completa, por lo que podrán ser detectados por valores positivos y negativos. Después de la amplificación y detección, la señal es comparada con una referencia programable y, si esta supera un umbral previamente establecido, es monitorizada y retenida por otro detector de pico que trabaja en modo muestreo-retención, hasta ser finalmente procesada por un convertidor analógico-digital de 12bit.

De manera síncrona con esta adquisición, se realiza la medida de la duración de los impulsos de señal detectada. Es posible que el sistema GDS detecte pulsos espurios de ruido, como por ejemplo los producidos por los rayos cósmicos incidentes en los fotodiodos. En tal caso, se adopta el convenio de que las partículas válidas deben ser detectadas y contadas al menos durante tres pulsos consecutivos. Esta “integración” reduce el ancho de banda del sistema de detección, y por tanto su tiempo de reacción, pero aumenta significativamente la relación señal/ruido del mismo. Para poder abarcar diferentes escenarios de detección, este número de pulsos detectados puede ser programado con otros valores: tres, dos o un pulso.

La cortina láser generada no es completamente uniforme, apareciendo irregularidades a lo largo de sus ejes, como se verá más adelante. Estas irregularidades se pueden cuantificar en los mapas de sensibilidad característicos del instrumento. Este hecho y el de que la cortina laser se produce a un ritmo de 100KHz, implican que una partícula, dependiendo de la zona de incidencia, emitirá más o menos luz y con ello, más o menos pulsos modulados en amplitud. Para compensar la no homogeneidad de la cortina en sentido transversal, el control del proceso de detección de partícula, realizado mediante un dispositivo FPGA, admite que exista la no detección de uno de los pulsos de entre los tres previstos. La no homogeneidad horizontal se mitiga haciendo uso de los mapas de sensibilidad.

Cuando la partícula es detectada de forma satisfactoria, se pone en marcha un contador de tiempo que se detiene cuando esta impacta sobre el IS.

Por motivos de consumo, sólo dos diodos láseres funcionan (son alimentados) a la vez. Esto permite solventar el posible mal funcionamiento de uno de los láseres. Es posible encender dos de ellos alimentados de forma continua (no con la señal pulsante de 100KHz) y mantener los otros dos apagados.

Calibración del GDS

El análisis detallado de la calibración del GDS puede encontrarse en **[Lop06]**. Existen dos tipos de calibración, una en Tierra y otra en vuelo (durante la travesía de Rosetta).

La calibración en vuelo se realiza de forma periódica, además de existir un telecomando (TC) por si se desea realizar calibraciones adicionales. Durante este tipo de calibración, se deshabilitan las interrupciones de GDS, se apagan los láseres, las lecturas de los fotodiodos permanecen habilitadas así como el detector de picos, a fin de obtener el ruido de fondo del sistema. Como resultado de la calibración, se envía a Tierra un informe que incluye la temperatura, luz observada y el ruido en cada canal (izquierdo y derecho).

La calibración desde Tierra se realizó desde dos puntos de vista diferentes:

- a) Relativo. Mediante la introducción de un hilo de cromo guiado por un brazo robotizado hasta cubrir toda la cortina láser.
- b) Absoluto. Lanzando partículas irregulares de diferentes composiciones químicas (andesita, nontronita, y silicato amorfo) y diferentes tamaños conocidos (desde 50µm hasta 500µm de diámetro) y con diferentes velocidades (desde 1m/s hasta 70m/s).

El mínimo tamaño detectable por el GDS depende de la composición química de las partículas, dado que el principio de detección está basado en la difusión de la luz producida por las mismas, y por tanto es dependiente de sus características ópticas.

Se detallan a continuación estas dos calibraciones de GDS realizadas en laboratorio.

Calibración relativa del GDS

El equipo de G. en Italia utilizó para la calibración, un dispositivo de posicionamiento de tres ejes controlado por computador y basado en motores paso a paso. Se consigue así poder introducir perpendicularmente en la cortina un hilo de cromo de 13 μ m. La siguiente ilustración, muestra el aspecto del conjunto empleado.

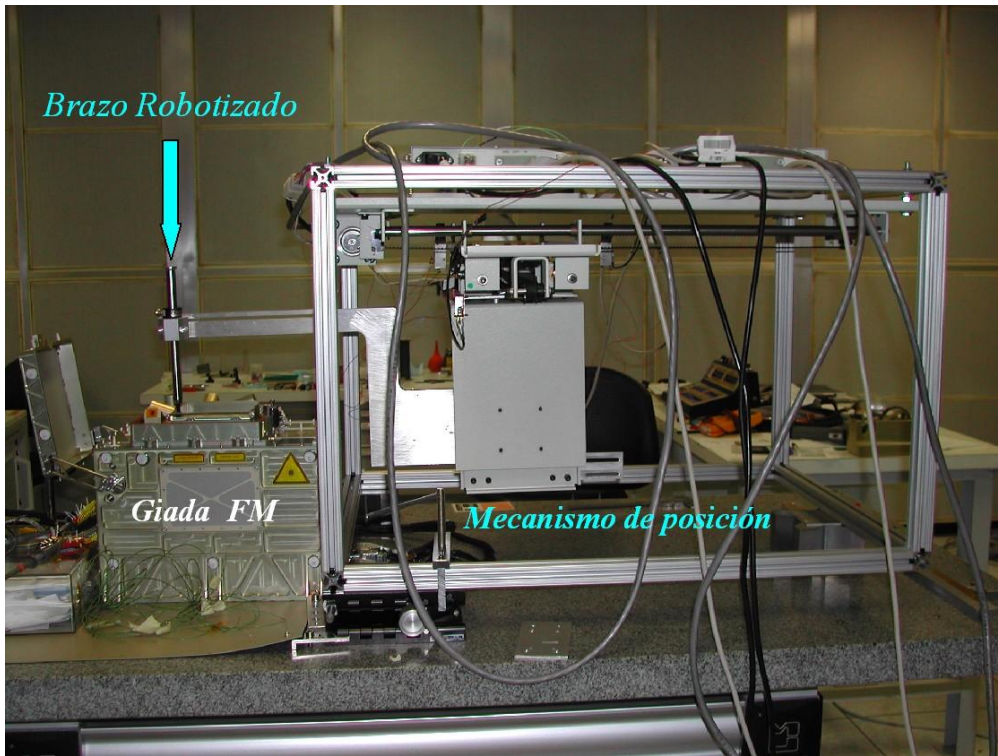


Ilustración 11. Robot de calibración del GDS.

Se realizó un barrido, introduciendo el hilo metálico en las diferentes zonas en que se ha dividido la superficie de la cortina laser. La cortina se dividió en una matriz de 36 filas por 42 columnas. A modo de ejemplo se muestran dos de las gráficas obtenidas (en dos y tres dimensiones) de los mapas de sensibilidad del canal izquierdo del instrumento.

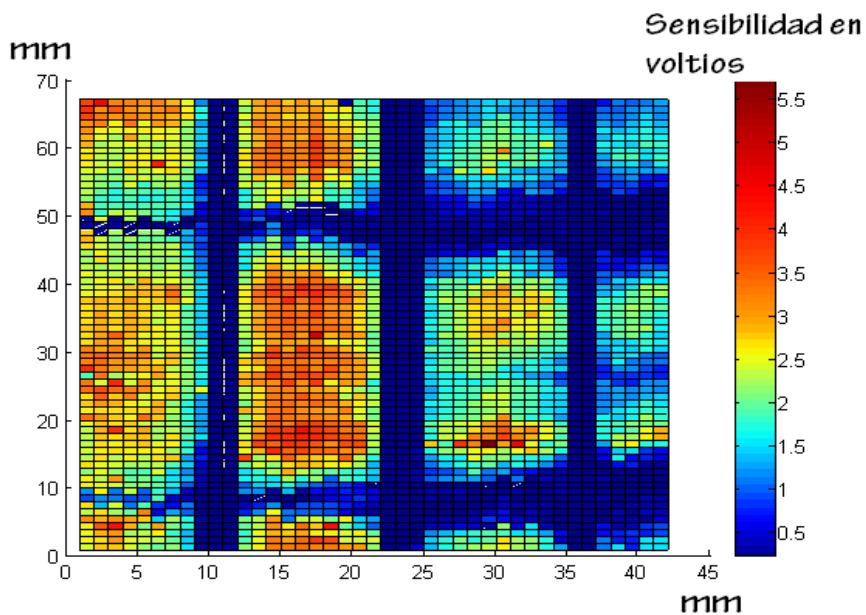
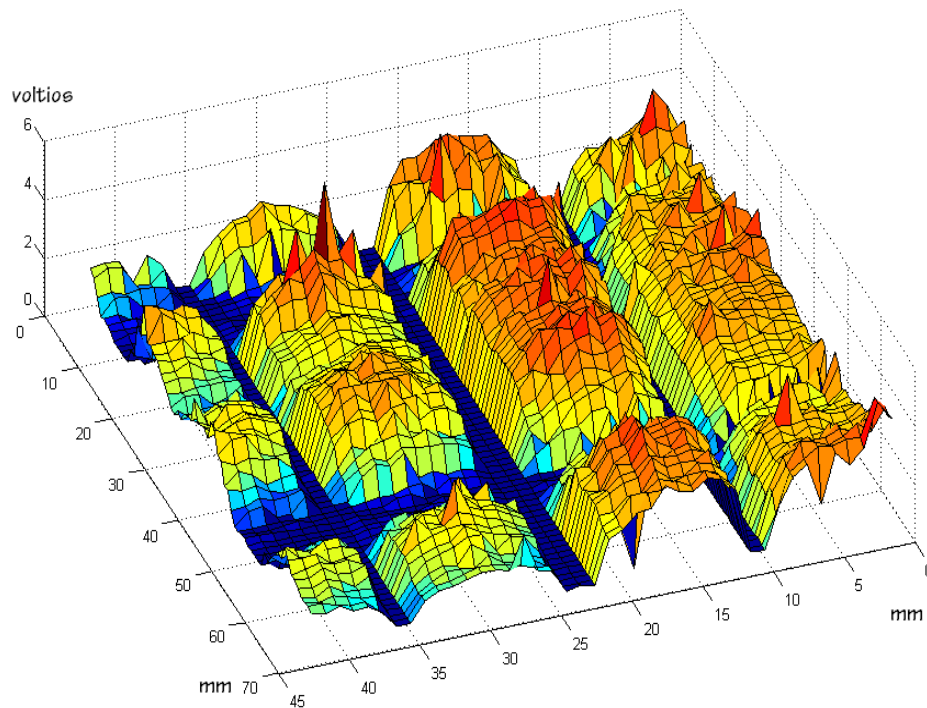


Ilustración 12. Mapa bidimensional de sensibilidad del canal izquierdo.**Ilustración 13. Mapa tridimensional de sensibilidad del canal izquierdo.**

En el mapa de sensibilidad bidimensional, se aprecian zonas de muy baja sensibilidad, las representadas en color azul claro, Esto es debido a la ausencia de solapamiento en el campo de visión de los cuatro fotodiodos receptores de un mismo canal. Este hecho no debe constituir mayor problema ya que justamente estas posiciones serán cubiertas por el otro canal (el canal derecho en el caso del ejemplo). Asimismo, en el mapa de sensibilidad en tres dimensiones, se aprecian acusadas zonas azules oscuras, que son producidas por el no solapamiento de la luz emitida por los diodos láser, estas serán *zonas ciegas* (de no detección) al paso de partículas.

Calibración absoluta del GDS

Para este tipo de calibración, se utilizó alrededor de 400 partículas con diferente composición química (andesita y nontronita), tamaño (50-100, 100-200 y 200-500 μm) y velocidad (1-70 ms^{-1}), simulando los modelos de composición química de las partículas de polvo de los cometas. Las partículas se han lanzado con velocidades representativas de los límites teóricos, según los modelos existentes de comportamiento del polvo cometario. Para ello se ha utilizado una pistola de aire comprimido a la que se puede ajustar la velocidad de salida del lanzamiento. Para la simulación de partículas lentas, se ha dejado caer la partícula libremente, por gravedad, utilizando una plantilla para aproximar el área de impacto en la cortina. Como resultado de la calibración, el comportamiento es bastante similar para ambos canales, no apreciándose diferencias significativas entre las repuestas de ambos.

Las partículas más rápidas están justo al límite de la sensibilidad de detección de la electrónica de GDS, esto es, tres ciclos completos de 100 kHz. En la siguiente ilustración se muestra una partícula rápida detectada (evento doble de GDS) por los dos canales: canal izquierdo en rojo y canal derecho azul. Los pulsos son producidos por el "scattering" de la partícula. Estos pulsos se encuentran separados entre sí los 10 μs y corresponden al ritmo de la luz pulsada de la cortina laser, que es de 100kHz. Así pues, el número de pulsos detectados dependerá de la velocidad con la que la partícula

atraviase la cortina. Las amplitudes de cada uno de los pulsos del tren detectado, varían con la posición de la partícula dentro de la cortina, del tamaño de esta y de las propiedades ópticas de la misma.

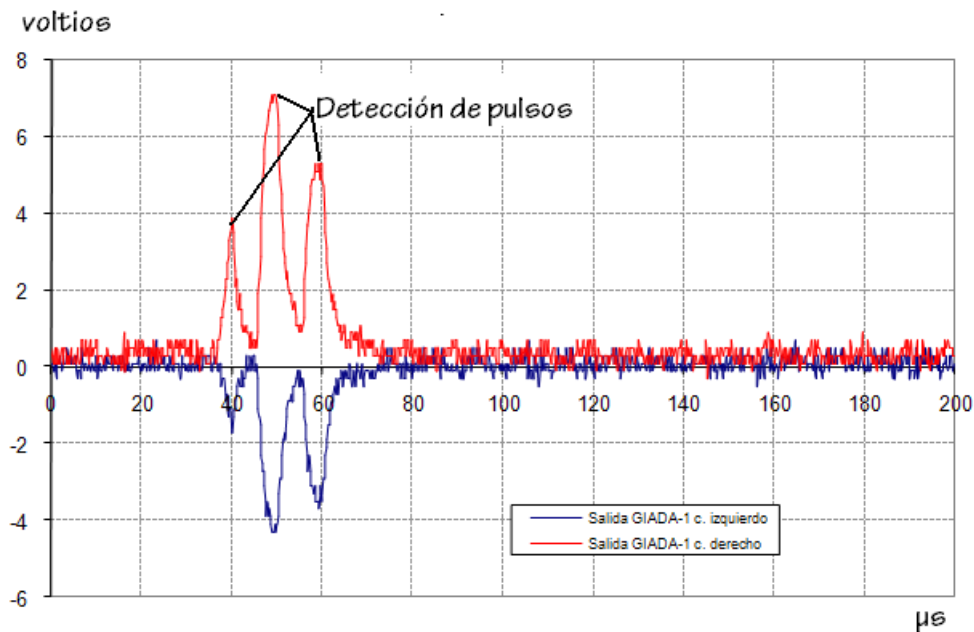


Ilustración 14. Respuesta del GDS a una partícula rápida.

La siguiente ilustración muestra el transitorio de una señal obtenida a la salida del canal de amplificación, producida por una partícula de velocidad lenta que genera múltiples detecciones. Se puede apreciar el efecto del tiempo de establecimiento de la red de acoplo AC en la envolvente de la señal muestreada.

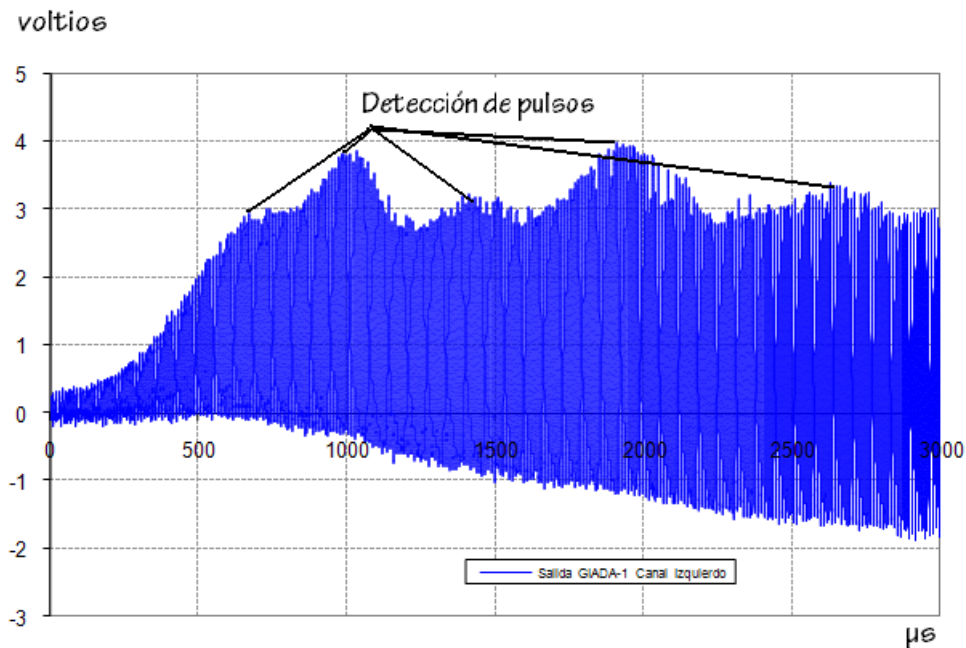


Ilustración 15. Respuesta del GDS a una partícula lenta.

2.2.3.1.1.2 IS

El sistema de detección del IS (**[Esp02]**) se compone de una lámina cuadrada de aluminio de 10cm x 10cm y de 0.5mm de espesor, que actúa a modo de membrana tambor. La distancia entre la cortina láser y la membrana es de 10cm. Sobre los vértices y el centro geométrico de la lámina, y por la cara no expuesta a la entrada de partículas, se adhieren cinco sensores piezoeléctricos (“PZT”) para fines de sensado. Existe además un PZT extra, situado en una de las diagonales del cuadrado, con el fin de utilizarlo como estimulador para fines de calibración, tanto en tierra como en vuelo. Es posicionamiento de los PZT se aprecia en la siguiente ilustración:

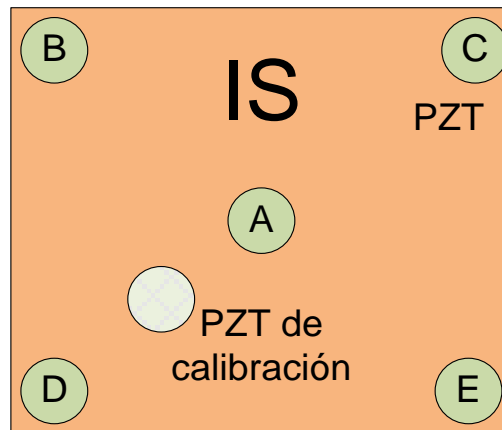


Ilustración 16. Esquema de posicionado de los sensores PZT en el IS.

Los PZT, al ser sometidos a un esfuerzo sobre el eje adecuado de su red cristalina, generan una señal de voltaje, de frecuencia igual a la de resonancia del cristal y de amplitud proporcional a la fuerza aplicada sobre el mismo. Este fenómeno, también sucede en el sentido contrario, por lo que al aplicar una diferencia de potencial eléctrico sobre la red cristalina, se obtendrá un desplazamiento mecánico de la misma. Si una partícula incide sobre la membrana de aluminio, se genera una onda transversal que se propaga a lo largo de toda su superficie. Los desplazamientos propagados harán vibrar a cada uno de los sensores en su frecuencia de resonancia, obteniéndose, en su salida, un voltaje proporcional al momento lineal de la partícula impactada.

Dado que los PZT son sistemas bidireccionales, es decir que cuando son sometidos a un voltaje se produce un desplazamiento proporcional al mismo, se destinará un PZT para la calibración del IS.

La onda generada por la partícula incidente, onda primaria, se propagará por toda la superficie de la membrana el aluminio y será reflejada, en mayor o menor medida, por los contornos de la lámina, generando ondas reflejadas que provocarán señales interferenciales. Estas interferencias pueden ser constructivas o destructivas, dependiendo de la fase de cada una de las ondas que se propagan.

El punto de impacto de la partícula, se obtiene por triangulación a partir de la medida de los tiempos relativos, en los que las señales alcanzan a los diferentes sensores del plato. Mediante estas medidas, el sensado de las distintas amplitudes de las señales generadas por los PZT, y utilizando detectores de pico, se podrá discriminar la señal producida por la onda directa.

Cada uno de los cinco PZT (exceptuando el de calibración) dispone de su propia electrónica de proximidad y de tratamiento de la señal. En concreto, para cada PZT existe:

- 1) Sistema de detección, retención de picos y de puesta cero (RESET) de la señal a muestrear.
- 2) Sistema de ampliación, adaptación y filtrado de señal.

- 3) Sistema generador de umbrales de detección y de comparación.
- 4) Sistema contador de tiempos, disparado por la primera de las señales detectadas.
- 5) Las características individuales de cada uno de los cinco PZT son las siguientes:
- 6) El PZT A, está situado en el centro geométrico de la membrana (ver *Ilustración 16. Esquema de posicionado de los sensores PZT en el IS.*). Su frecuencia de resonancia es de 100KHz. Dispone de dos etapas de amplificación independientes: una de ganancia 600 V/V (“high”) y otra de ganancia 10V/V (“low”). Estas ganancias son seleccionadas mediante una señal “GAIN A” generada en la FPGA de la electrónica de control en GIADA-2.
- 7) Los PZT B y PZT C, están situados en dos de los extremos de uno de los lados de la membrana, su frecuencia de resonancia es de 100KHz y disponen de una etapa de amplificación fija de 10V/V.
- 8) Asimismo los PZT D y PZT E, están situados en las dos esquinas restantes de la membrana, su frecuencia de resonancia es de 100KHz y poseen una etapa de amplificación programable de 1V/V y 10V/V.

Cada canal PZT dispone de su propio *umbral de detección*, programable mediante un convertidor digital-analógico de 8bit. Si la señal detectada supera el nivel del umbral establecido, se produce la inicialización del sistema IS. Esta inicialización, también se puede realizar mediante un telecomando (TC) de cambio de modo de operación.

La señal, procedente de la etapa de amplificación y filtrado, al llegar electrónica de proximidad, es retenida en un detector de picos y comparada con el umbral correspondiente. El detector de picos se controla con dos señales de la EP: “HOLD” y “RESET”. La misión de la señal de HOLD es el fijar el valor del voltaje detectado e ignorar las siguientes señales de entrada, ya que proceden de interferencias producidas por ondas reflejadas. La señal RESET inicializa el detector a 0V después de cada una de las adquisiciones. Esta señal de RESET ha de ser aplicada de forma periódica ya que el detector, debido al ruido, posee una deriva que aumenta su voltaje de salida hasta la saturación del sistema. Este RESET periódico es generado sólo cuando al menos uno de los PZT está habilitado y ninguna de las señales detectadas por los PZT habilitados está por encima del umbral de ruido programado.

La onda primaria generada por la partícula, alcanza a todos los PZT en un tiempo que depende de la velocidad de propagación de las ondas sobre la membrana de aluminio. El valor máximo de este tiempo se ha establecido experimentalmente y es de 35 μ s. El valor teórico es de 22 μ s para un cuadrado de 10cm de lado y una velocidad de propagación del sonido en el aluminio de 6400m/s . Por tanto, tras este tiempo de 35 μ s y una vez que la señal de entrada haya sobrepasado el umbral de ruido programado, se retiene el nivel de señal proporcionada por del detector de picos mediante la señal de HOLD. De esta manera, se consigue obtener el valor máximo generado por la onda primaria y se evita procesar las señales producidas por ondas interferenciales secundarias posteriores. La señal retenida es muestreada por el convertidor analógico digital (ADC), antes de ser aplicada la señal de RESET. Una vez finalizado este proceso, el sistema está preparado para una nueva adquisición.

En la siguiente ilustración se muestran las señales analógicas involucradas en la detección. Se representa la acción del detector de picos (señal azul) sobre las señales de entrada, primaria y secundaria (señal rosa). En amarillo se aprecia la acción del HOLD a los 35 μ s para obtener la señal a muestrear

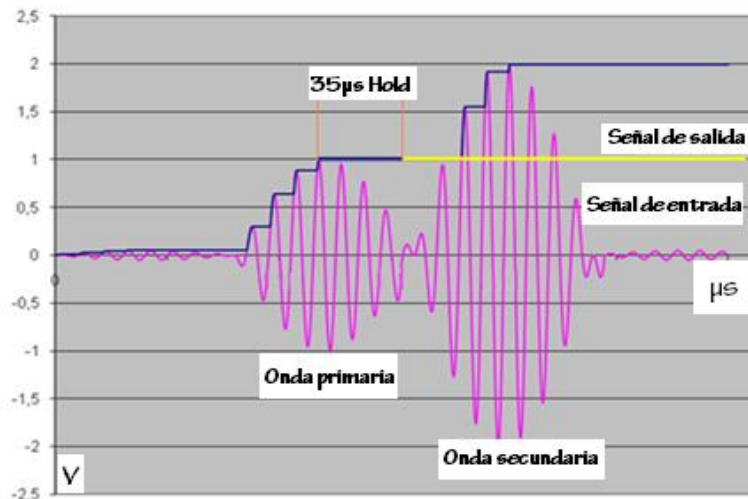


Ilustración 17. Señales analógicas involucradas en el la detección del sistema.

La salida del detector de picos y todas las señales de control de este sistema, están controladas por la EP (GIADA-2). Aquí es donde se realizan la comparación analógica de las señales con los umbrales programados y donde se genera la lógica de HOLD y RESET. Esta electrónica de control es también la encargada de habilitar o deshabilitar cada uno de los PZT.

Cuando la señal de entrada a uno de los PZT sobrepasa su umbral, en la EP se llevan a cabo las siguientes acciones:

- a) Se detiene el contador iniciado por el GDS, obteniendo así el tiempo de vuelo de la partícula.
- b) Se ponen en funcionamiento los contadores del resto de PZT.
- c) Cuando la señal de un PZT supera el umbral de ruido programado, el bit de estado de la detección “detectado” del PZT se establece a uno.

Los contadores puestos en marcha en **b)** serán detenidos cuando las señales que les lleguen sobrepasen sus umbrales. Mediante estos contadores, se obtiene el tiempo de retardo en la llegada de la onda (generada por la partícula al impactar), lo cual servirá para calcular, mediante triangulación, las coordenadas del impacto sobre la membrana. Con este nuevo dato, se podrá recalcular el valor del momento de la partícula incidente y obtener el mapa de sensibilidad del IS (ver **Ilustración 18. Mapa de sensibilidad correspondiente al PZT A del IS.**). Además, conociendo la posición, será posible desechar impactos fantasma, esto es, los impactos localizados fuera de la geometría de la membrana del IS.

Con objeto de eliminar ruidos espurios de la señal, y de paso evitar introducir más componentes en la electrónica de proceso de la señal, se realiza un filtrado digital de los datos en la propia FPGA. Esto conlleva un retraso, que habrá que tener en cuenta a la hora de calcular el tiempo de vuelo. Esta operación se realiza automáticamente.

El sistema de detección, dispone de un bit de desbordamiento (“overflow”) de señal en cada PZT. Este bit se activa cuando el contador de tiempo no es detenido debido a que la señal de entrada no ha superado el umbral predefinido y por tanto la partícula no ha sido detectada correctamente.

En el paquete de datos de la telemetría del sistema, se reservan cinco bits para almacenar el tiempo de propagación de la señal, donde cada bit representa un tiempo de 3 μ s. Esto limita la medida máxima del tiempo de propagación a 96 μ s. Este tiempo es suficiente para que la señal primaria se atraviese completamente la diagonal de la membrana de aluminio.

Calibración del IS

La calibración de este subsistema (**[Lop06]**) se puede realizar durante la travesía de Rosetta o en Tierra.

Calibración en vuelo del IS

Este tipo de calibración, se realiza de forma periódica y existe además un telecomando en caso de que se requieran calibraciones adicionales.

Para esta actividad, se utiliza el PZT de calibrado que, como se ha expuesto anteriormente, se encuentra situado en una de las diagonales de la plataforma de sensado. Este transductor genera una vibración mecánica, al ser sometido a un determinado voltaje, que simula el impacto de una partícula en la posición donde se encuentra fijado el PZT.

Dos parámetros importantes para la simulación son el denominado “nivel de calibración” (que controla la amplitud del voltaje aplicado al transductor) y “número de estímulos” (que indica el número de pulsos de calibración que se generarán y que será un número par). Es importante tener en cuenta que, por cada impulso de señal de calibrado, se producen dos eventos en el sistema de detección, uno correspondiente al flanco de subida del pulso de calibrado y otro al de bajada (de ahí el número par de estímulos). Al finalizar el proceso de calibración se genera una señal de RESET al sistema, para permitir una nueva adquisición.

Calibración en laboratorio del IS

El IS se ha calibrado en el laboratorio desde dos puntos de vista diferentes:

- a) **Relativo.** Mediante la introducción de pequeñas excitaciones, utilizando un PZT similar al de calibración a lo largo de la superficie de la membrana de detección. El PZT es desplazado mediante un brazo robotizado (el mismo utilizado en la calibración del GDS) por la superficie de para obtener el mapa de sensibilidad del IS.
- b) **Absoluto.** Utilizando partículas irregulares de diferentes composiciones químicas (silicatos, andesita, nontronita, y carbón), de diferentes tamaños conocidos (desde 50 μ m a 500 μ m) y con diferentes velocidades (hasta 100m/s).

Las calibraciones realizadas por el equipo de G., demuestran que la señal de salida es, prácticamente independiente de la composición química de las partículas irregulares, de manera que la relación entre el momento lineal y el voltaje de salida de los sensores es una función única.

En la siguiente ilustración se muestra el mapa de sensibilidad del IS obtenido para el “PZT A”. Se observa que la región de mayor sensibilidad (zona roja) se encuentra en torno al propio sensor. Cada uno de los PZT restantes, poseen un mapa de sensibilidad similar.

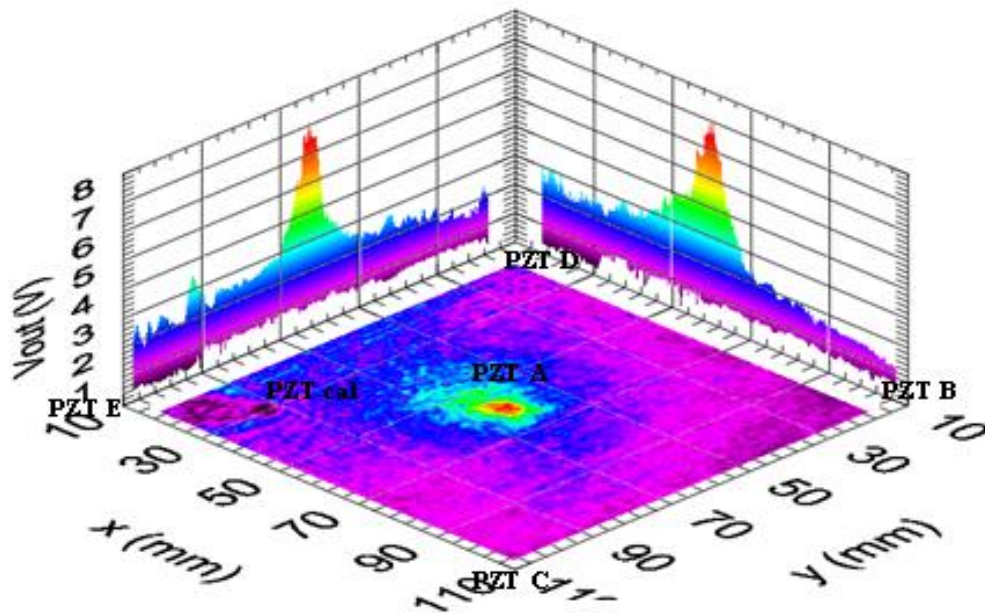


Ilustración 18. Mapa de sensibilidad correspondiente al PZT_A del IS.

Aprovechando la calibración absoluta del GDS, en la que se lanzan partículas contra la cortina láser del instrumento, se realizó una calibración absoluta del IS. Se midió la respuesta obtenida, en cada uno de los canales de detección debida al impacto de estas partículas sobre el IS. En un porcentaje elevado de casos (entorno al 80%) se obtiene señal detectada válida, en cada uno de los cinco canales que componen el sistema de detección. En otro porcentaje menor de casos sólo se obtuvieron señales sólo en alguno de estos cinco canales, debido al ajuste de los umbrales de detección de los mismos y a los rebotes de las partículas causados por la gravedad.

2.2.3.1.2 GIADA-2: Electrónica principal

El objetivo de la Electrónica Principal (EP) (**[Lop06]**) es controlar, adquirir datos, suministrar potencia a todo el instrumento y hacer de interfaz, tanto con la electrónica de proximidad como con Rosetta. GIADA-2 se divide en tres bloques funcionales:

- a) *Electrónica analógica.* Se encarga del procesamiento de señales recibidas de los sistemas de sensado, GDS, IS y MBS, cubierta, y de las de estado del instrumento.
- b) *Unidad de procesamiento digital.* Realiza el control digital, almacenamiento y CPU
- c) *La unidad de suministro de potencia.* Que alimenta a todos los circuitos eléctricos de G.

Por motivos de peso, todo GIADA-2 se implementa en sólo en dos tarjetas electrónicas, una para *procesamiento digital* (bloque funcional **a)** y **b)**) y otro para la *alimentación del sistema* (bloque **c)**). La tarjeta de alimentación se encuentra duplicada, por motivos de fiabilidad, en una *tarjeta principal* y otra *redundante*. En la tarjeta de procesado, por motivos de peso y espacio, solo se duplican los componentes más críticos del sistema de conversión y adquisición. Las tres tarjetas (2 de alimentación redundantes más la de analógica) incorporan dispositivos FPGA RH1280 de la empresa Actel, donde se implementan los controles digitales de los distintos subsistemas. El uso de FPGA redundante en un ahorro del espacio ocupado por componentes y del consumo eléctrico del conjunto.

El esquema de la EP se detalla en la siguiente ilustración.

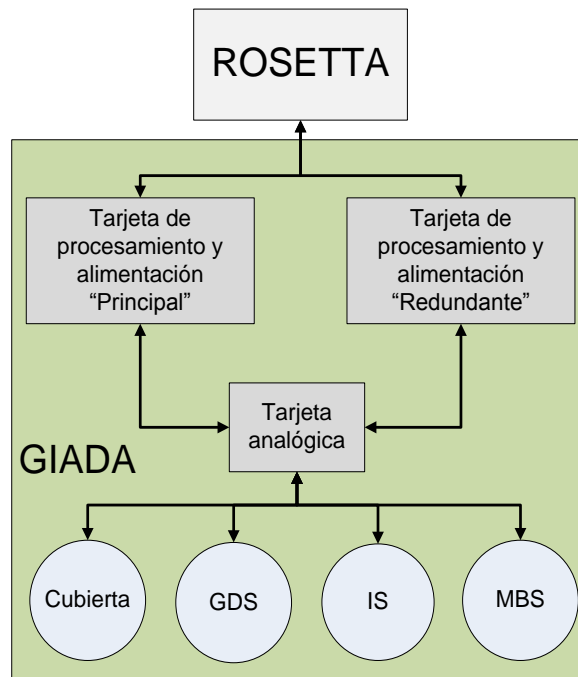


Ilustración 19. Esquema de la electrónica principal.

Cada tarjeta de procesamiento digital, dispone de una unidad central de procesamiento “CPU”, en concreto, un microprocesador 80C86 (versión de bajo consumo del microprocesador 8086) trabajando con un reloj de 4MHz. Además cada una incorpora tres tipos diferentes de bancos de memoria:

- a) *Memoria de RAM* de 64KiB. Esta es una memoria de lectura escritura donde se ejecutarán todos los modos de operación del instrumento (3.5 Modos operativos) excepto el modo safe.
- b) *Memoria de ROM* de 64KiB. Donde se encuentra y se ejecuta el sistema de arranque y el modo de operación safe.
- c) *Memoria de datos no-volátil RAM (NVRAM)* de 64KiB. Es una memoria de lectura y escritura que permite mantener su contenido entre apagados y encendidos del instrumento.

Se muestran a continuación los modelos de vuelo de las dos tarjetas de la EP, la tarjeta analógica, y el conjunto de las tres que conforman GIADA-2.

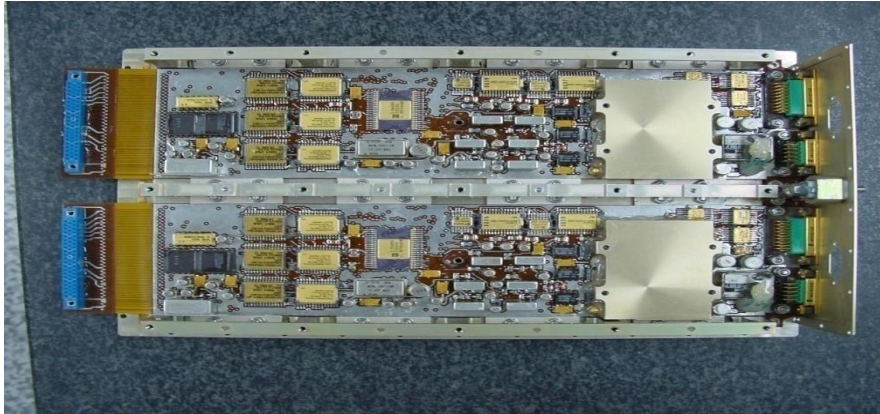


Ilustración 20. Modelo de vuelo de las tarjetas (principal y redundante) de procesamiento digital y alimentaciones.

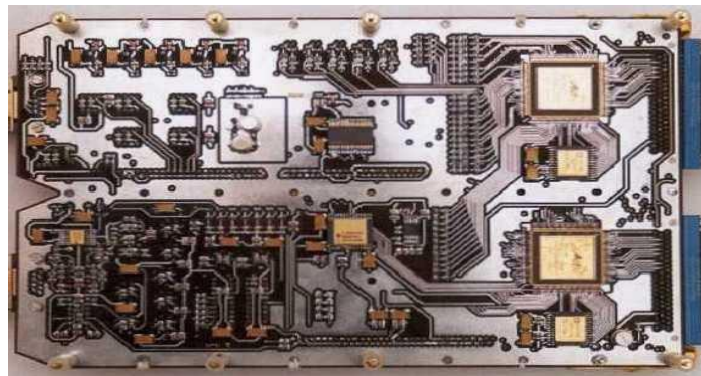


Ilustración 21. Modelo de vuelo de la tarjeta analógica.

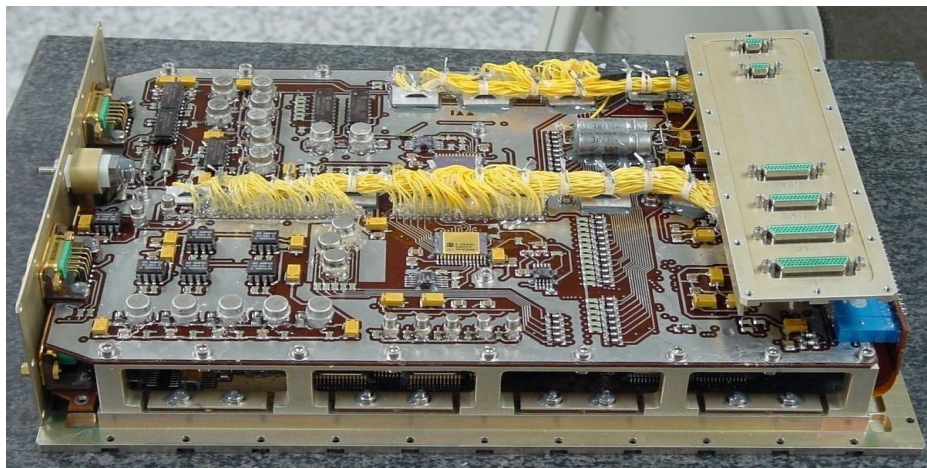


Ilustración 22. Modelo de vuelo de GIADA-2.

2.2.3.1.3 GIADA-3: MBS

Durante la fase de acercamiento de la sonda al núcleo del cometa, el bajo número de partículas esperado, no justifica el encendido de los subsistemas de detección de partículas individuales (GDS+IS). En su lugar, se monitoriza el flujo de masa de polvo depositado por unidad de superficie y tiempo. Cuando esta medida de flujo, supera un umbral preestablecido, el sistema genera una

señal de aviso. Este hecho indica al operador terrestre, el cambio de modo de funcionamiento al de detección de partículas simples. Esta señal, también se utiliza para informar al resto de instrumentos de una posible avalancha de partículas. Existe además la posibilidad de mantener todos los sistemas de detección (GDS, IS y MBS) funcionando a la vez.

El sistema MBS está compuesto por cinco microbalanzas orientadas de tal forma que, una de ellas apunte al núcleo del cometa y las cuatro restantes a cuatro direcciones diferentes, de forma que se cubra el mayor ángulo de detección posible, como puede apreciarse en la siguiente ilustración:

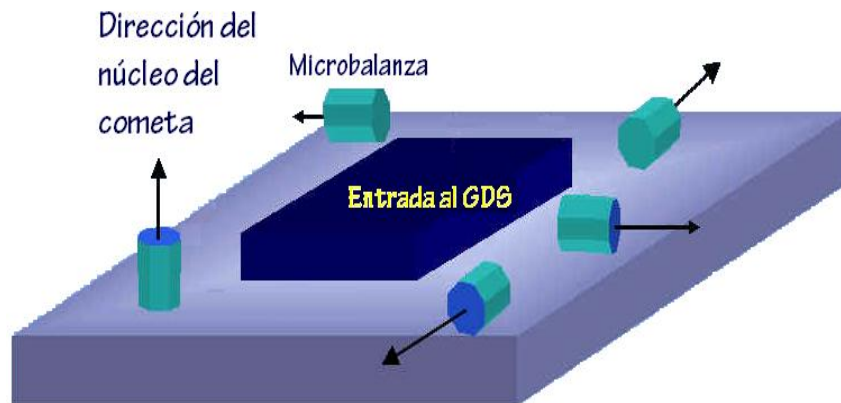


Ilustración 23. Esquema del MBS.

Cada MB dispone de un ángulo de visión de 40° y consta de una pareja de cristales de cuarzo con la misma frecuencia de resonancia. Uno de ellos está aislado del exterior y es utilizado como referencia, y el otro se destina al sensado. El cristal de sensado reacciona con variaciones de la frecuencia de resonancia a la modificación de sus propiedades físicas producidas por la masa de las partículas depositadas sobre su superficie. Las señales generadas por cada uno de los cristales (sensado y referencia), se combinan en un circuito sumador y producen la denominada frecuencia de batido. Esta frecuencia resultante es filtrada hasta obtener sólo las bajas frecuencias presentes, puesto que éstas son debidas a la diferencia, en valor absoluto, de las frecuencias de sensado y referencia. Este proceso lo realiza la electrónica interna de cada microbalanza.

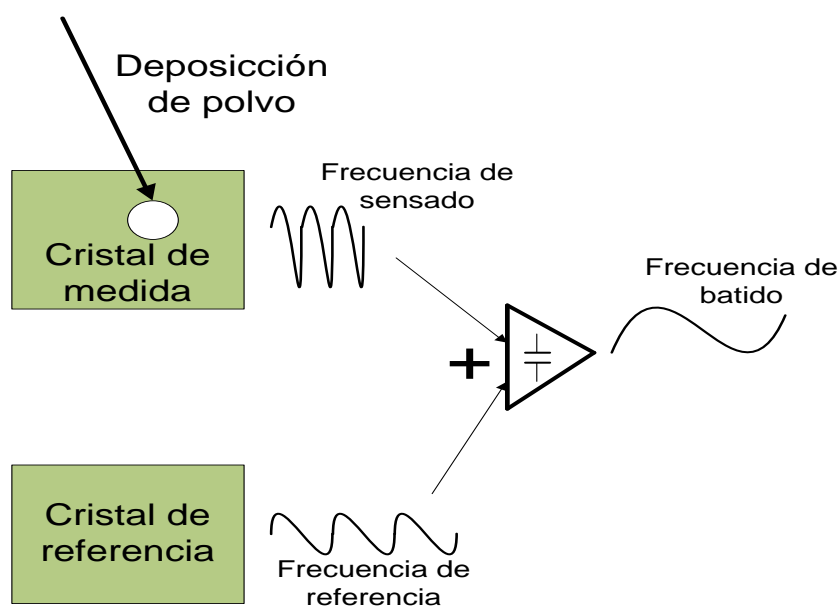


Ilustración 24. Deposición de polvo en una MB y frecuencias asociadas.

La frecuencia de batido es proporcional a la masa depositada, pero también es función de la temperatura, por ello, el dispositivo cuenta con un sensor térmico, para así poder compensar las medidas de frecuencia realizadas.

Los parámetros que caracterizan las prestaciones de sensado de una MB son: su sensibilidad a la deposición de masa, su rango dinámico (rango de medida) y su resolución temporal.

Las MB utilizadas, tienen una sensibilidad tal que son capaces de detectar una variación de masa del orden de 10^{-10} gramos. Un incremento en la frecuencia de resonancia de los cristales origina un incremento en la sensibilidad.

El límite de detección está directamente relacionado con la estabilidad y exactitud de la medida de la frecuencia salida, que a su vez depende del tiempo de muestreo de la señal y de la saturación del sistema.

El rango dinámico de sensado de cada microbalanzas es de seis décadas, rango suficiente para poder detectar las variaciones de flujo de masa esperadas por el instrumento. Para poder cubrir este rango, G. utiliza un tiempo de integración programable de hasta 1024ms, con lo que se obtiene una resolución de 1Hz en todo el rango de medida. Este rango puede variar desde 1Hz hasta 165 KHz.

La saturación de masa de la MB se produce con 10^{-4} gramos depositados. Cuando esto sucede, es posible “limpiarla” mediante calentamiento (“heating”), utilizando una resistencia calefactora que incorpora la MB. Esta limpieza es posible debido a que la masa depositada se compone de partículas de hielo y/o gases helados que subliman al subir la temperatura. Durante la fase de limpieza de la microbalanza y dependiendo de la temperatura para la que la masa se ve reducida (con la correspondiente variación de la frecuencia de batido generada) será posible obtener una indicación del tipo de material depositado.

Calibración del MBS

Un análisis detallado de la calibración del MBS puede encontrarse en [\[Lop06\]](#). Existen dos clases de calibración, una en Tierra y otra durante el vuelo.

La calibración en vuelo del MBS se produce de forma periódica, además de existir un TC por si son requeridas calibraciones adicionales.

En la calibración realizada en laboratorio, se han caracterizado cada una de las microbalanzas, mediante la realización de medidas reiteradas de frecuencia y de temperatura para diferentes ratios de flujo de masa.

Otro factor, a tener en cuenta en la realización de las medidas realizadas, es la eficiencia en la adherencia de las partículas sobre la superficie de sensado, que depende del tamaño y de la velocidad con la que estas impactan. Se ha comprobado que la probabilidad de captura es alta (cercana al 70%), para partículas micrométricas con velocidades de hasta 20m/s, disminuyendo progresivamente esta probabilidad para velocidades de partícula más elevadas.

2.2.3.1.4 Cubierta

El objetivo de la cubierta del instrumento es la de aislar o permitir la entrada de partículas de polvo, así como la de proteger de la radiación solar o de un enfriamiento excesivo. Durante el lanzamiento y en otras fases de la misión, la cubierta se mantiene cerrada y asegurada mediante un

dispositivo rearmable, similar a un trinquete, denominado “frangibolt^B”. El frangibolt es un dispositivo de liberación, no explosivo, que utiliza la propiedad de la memoria de forma que poseen algunas aleaciones que, cuando se calientan por encima de su “temperatura de transformación”, recuperan su forma original. Esta recuperación es utilizada para forzar la ruptura de un perno, que realiza una labor de sellado. En otras palabras, la energía eléctrica utilizada para su calentamiento se transforma en trabajo mecánico que concluye con la rotura de un mecanismo de sellado. Durante la fase inicial de la misión, el frangibolt se cizalla, y a partir de ese momento, la cubierta puede abrirse o cerrarse con libertad.

La rotura del frangibolt fue uno de los hitos de G. y por ello, existen telecomandos especiales para comprobar la buena salud del mismo y realizar la rotura.

La cubierta es movida por un motor paso a paso, a cuyo controlador se le puede especificar la velocidad, la dirección (abrir o cerrar) y el número de pasos a realizar. En el recorrido de la cubierta se han establecido dos puntos de sensado (“reed switches”), que permiten establecer la posición de la misma en caso de pérdida de pasos del motor. El estado de la cubierta, se refleja en el “housekeeping” (HK) del sistema.

En previsión de que el mecanismo de apretura/cierre se congele, debido a las bajas temperaturas de espacio profundo, la cubierta dispone de un conjunto de calentadores que son activados antes de su apertura o de su cierre.

2.2.3.2 Curvas de calibración

Los sensores son capaces de, dada una magnitud física (temperatura, voltaje, etcétera) proporcionar una medida de la misma. Esta medida normalmente se realiza de forma analógica, pero para poder ser procesada por un computador es necesario realizar una conversión a unidades digitales. Para ello se utiliza un conversor analógico digital o “ADC”, cuyas propiedades físicas influirán en la propia conversión, ya que dos “ADC” de la misma remesa de fabricación convertirán de forma ligeramente diferente una misma entrada.

Se realiza por tanto, una conversión de unidades de una magnitud física a unidades digitales. Estas unidades digitales son las que se envían como parámetros de las telemetrías, pero normalmente no son directamente utilizables, ya que se necesita aplicar el proceso inverso de conversión (unidades digitales a físicas) para tener un significado comprensible. Las conversiones se realizan a través de las llamadas curvas de calibración (también llamadas funciones de transferencia) que mapear unidades digitales a unidades físicas y viceversa.

Por motivos de seguridad, G. consta de dos cadenas de adquisición de datos (*principal* y *redundante*), paralelas y duplicadas entre sí. Sólo una de una de ellas está activa a la vez, y en caso de malfuncionamiento se activará la de reserva. Aunque cada cadena se ha construido con los mismos tipos de componentes HW, siempre existen diferencias sutiles. Es por ello que se realizó una caracterización de ambas cadenas, generando dos conjuntos de funciones de transferencia distintas. Conocer qué cadena de datos ha sido utilizada para la medición es el primer paso para realizar cualquier estudio de datos de G.

A modo de ejemplo, se muestran las curvas de calibración del ADC de las cadenas *principal* y *redundante* en la conversión de voltaje. En dichas curvas se observan leves diferencias en los coeficientes.

Cadena *principal*: $y = 340.46 + 287$

Cadena *redundante*: $y = 340.5 + 288$

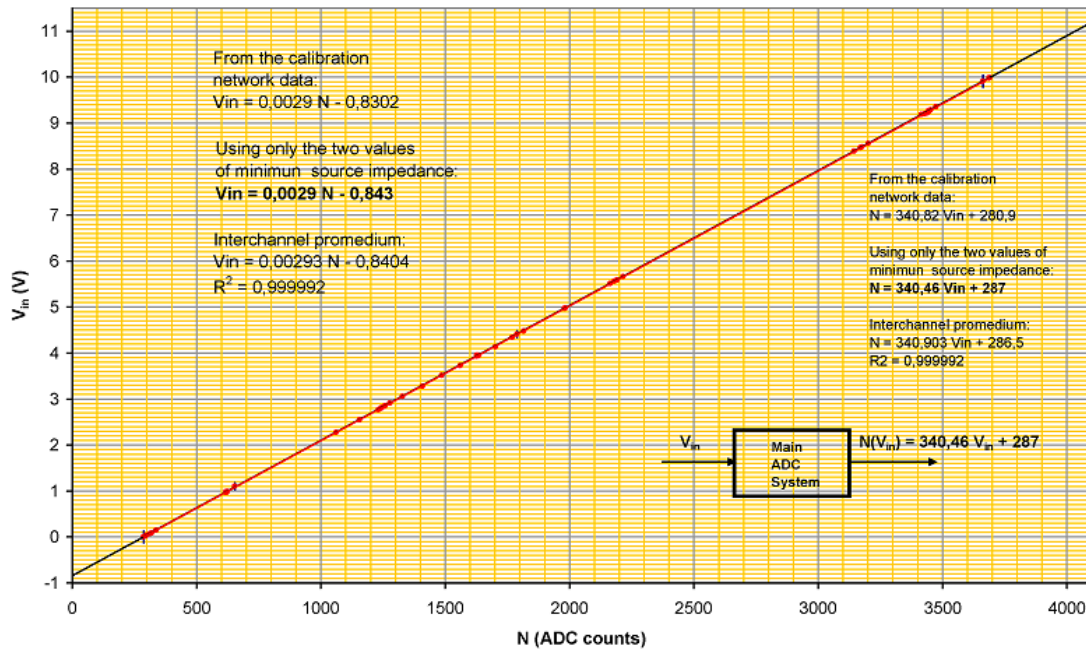


Ilustración 25. Curva de calibración para el ADC canal principal.

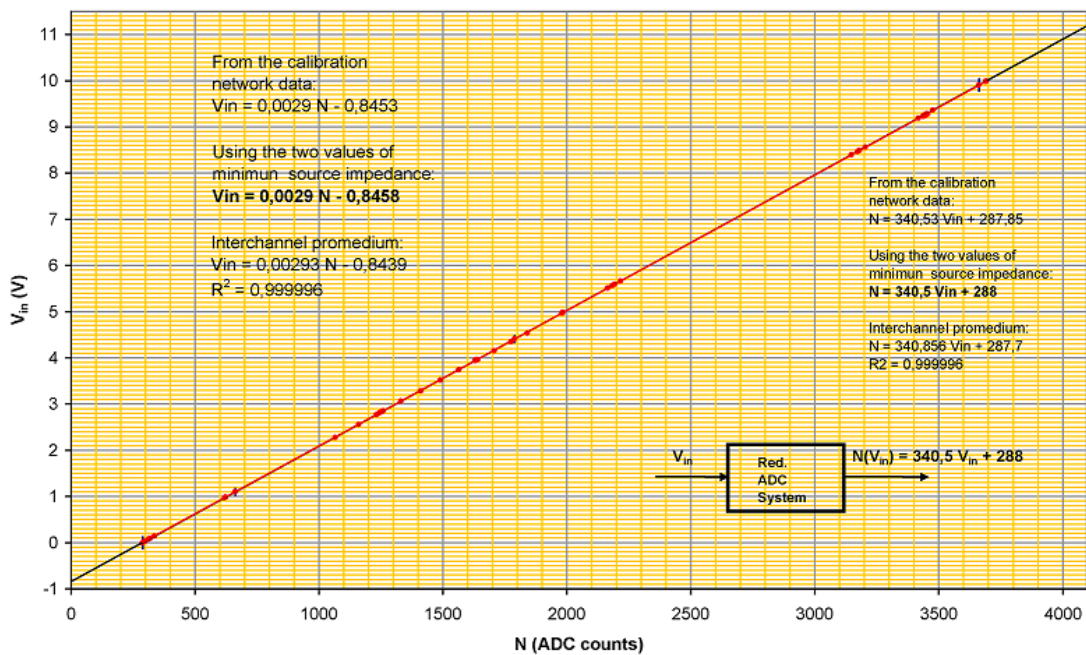


Ilustración 26. Curva de calibración para el ADC canal redundante.

GDB posee un apartado especial para la gestión de las curvas de calibración (6.2.7) que incluye una calculadora para realizar conversiones rápidas. Así mismo, en la descripción de las tablas que almacenan los datos científicos de G. (*Apéndice A*) se indicará la curva de calibración asociada a cada atributo.

2.2.4 Prestaciones de GIADA

Las prestaciones de G. ([Col06], [Rot12] y [Rot15]), en términos de sensibilidad de sus sensores, se calculan teniendo en cuenta los resultados obtenidos de las calibraciones, de un análisis

“end of life” (fin de la vida útil) y de los niveles de ruido medidos durante las calibraciones realizadas en vuelo. Así por ejemplo, el mínimo tamaño de partícula detectable por el GDS y el mínimo momento detectable por el sensor de impacto, se establecen de acuerdo con el nivel de ruido medido por sus respectivos sensores.

Las medidas finales de las prestaciones se han obtenido utilizando el modelo de GIADA PFM (“Proto Flight Model”) que es equivalente en prestaciones al modelo de vuelo de G. Para ello se ha situado el PFM en una sala limpia, especialmente equipada para realizar las calibraciones necesarias (2.2.3.1.1 GIADA-1: GDS+IS).

La siguiente tabla resume las prestaciones (límite de sensado) del instrumento:

Tabla 6. Límites en el sensado de GIADA.

Subsistema	Magnitud física de la partícula	Mínimo detectable	Máximo detectable/ límite de saturación (a)
GDS	Radio (μm) (b)	30	800
GDS	Velocidad (m s^{-1}) (c) (e)	0.3	300
IS	Momento (kg m s^{-1}) (d)	1×10^{-10}	4×10^{-4}
GDS+IS	Velocidad (m s^{-1}) (c) (f)	0.3	300
MBS	Masa acumulada (kg)	2.2×10^{-12}	4×10^{-4}

(a) Los límites mínimo y máximo/saturación de los diferentes parámetros son independientes entre sí.

(b) Para partículas de nontronita. Partículas más grandes que el límite de saturación del radio serán detectadas.

(c) El límite superior está determinado por el ratio de generación del láser (100 KHz por defecto).

(d) Las partículas con un valor de momento mayor que el máximo serán detectadas.

(e) Velocidad medida mediante los pulsos detectados en la cortina láser.

(f) Velocidad medida mediante el tiempo de vuelo desde el GDS a IS.

La tasa de errores cometidos en las medidas obtenidas en la calibración, es la siguiente:

- a) Errores del 10% al 100% en el cálculo de velocidades mediante el tiempo de permanencia de la partícula en la cortina láser. El error aumenta con la velocidad.
- b) Errores menores del 1% al 10% en el cálculo de la velocidad derivada del tiempo de vuelo. El error aumenta con la velocidad.
- c) Errores en torno al 25%, en la medida del momento lineal de una partícula.
- d) Errores del 25% al 35%, en las medidas de masas de partículas.
- e) Errores en torno al 10% en la medida del flujo de masa de polvo.

Una medida directa “*cercana*” al límite de detección, necesita de unos factores de corrección (ver **[Rot15]**). A modo de ejemplo, estas correcciones son necesarias para valores próximos al límite de detección del IS, ya que sólo el 30% del total del área de detección tiene suficiente sensibilidad para detectar valores de momento por debajo de $2.5 \times 10^{-9} \text{ kg m s}^{-1}$.

Hay que indicar que G. es un instrumento diseñado para un análisis estadístico de partículas. Por tanto, la calidad de los datos obtenidos, será mayor cuanto mayor hayan sido el número de partículas estudiadas, dado que la fiabilidad de los estudios estadísticos aumenta con el volumen de datos muestreados. La medida de partículas individuales, aunque dar lugar a información relevante, no constituye el objetivo principal de G.

2.2.5 Estado de GIADA tras once años de travesía

Durante la travesía de Rosetta hasta el encuentro con 6CG (**[Rot12]** y **[Rot15]**), se han realizado 13 verificaciones del estado de G. (“Payload Checkouts”). Estas verificaciones incluyen las calibraciones en vuelo de GDS, IS y MBS.

El GDS ha mostrado una buena estabilidad del nivel de detección y del estado de la óptica con respecto a las calibraciones realizadas en tierra. Se han medido variaciones de 0.10V del canal derecho y 0.30V en el izquierdo.

El IS ha mantenido su nivel de respuesta con diferencias poco relevantes (en torno a 0.10V) respecto a las calibraciones terrestres.

Se ha detectado un pequeño incremento de 500 Hz en la frecuencia del MBS 1, probablemente debido a la contaminación por desgasificación de componentes de Rosetta o a la actividad de los propulsores de la nave.

Por tanto, la travesía no dañó en exceso los sensores de G.

2.2.6 Primeros resultados científicos

Se resumen a continuación los primeros resultados científicos obtenidos a raíz del estudio estadístico de los datos de G.

- 1) G. empezó a funcionar de forma continuada el 18 de julio de 2014. La primera partícula detectada fue el 1 de agosto de 2104 (a 814 km del núcleo). Durante la aproximación de Rosetta al cometa, se detectaron 3 partículas más. Ya en órbita, el IS detectó 19 impactos individuales, mientras que el GDS sólo 7 y se pudieron obtener 9 eventos GDS+IS.
- 2) A partir del 15 de septiembre de 2014 hasta el 4 de febrero de 2015, Rosetta realizó una serie de órbitas en torno al cometa situándolo a 10 km, 20 km y 30 km del núcleo. En esta fase de la misión, se detectaron 119 impactos de IS, 1056 eventos de GDS y 83 eventos GDS+IS.
- 3) Las detecciones de GDS, normalmente, aparecen como lluvia (nubes) de partículas de hasta 30s, compuestas a su vez de lluvias cortas de hasta 1s.
- 4) Las partículas lentas y poco densas, es decir, con velocidades menores a 1 ms^{-1} y masa en torno a 10^{-10} kg , están por debajo del límite de sensibilidad del IS, pero son detectadas por el GDS.
- 5) Las siguientes tipos de partículas no son detectadas por el GDS:
 - a. Oscuras (con propiedades ópticas similares al carbón amorfo).
 - b. Pequeñas (menores de $150 \mu\text{m}$)
 - c. Poco densas (similar a los silicatos poco densos) y pequeñas (menores de $50 \mu\text{m}$)

- 6) Las partículas del tipo 5) que tengan un momento mayor de $1 \times 10^{-10} \text{ kg m s}^{-1}$ serán detectadas sólo por el IS.
- 7) Los eventos GDS-IS están concentrados en el hemisferio norte del cometa.
- 8) Las partículas grandes son más lentas comparadas con las pequeñas (menores de $50 \cdot 10^6 \text{ m}$).
- 9) Las detecciones de eventos GDS no parecen estar conectadas a ninguna área de emisión concreta. Estos eventos corresponden a agregados de partículas de baja densidad y porosos.
- 10) Se detectó una evidencia de masa acumulada en la MB1 (apuntando hacia el Sol) y MB5 (apuntando hacia nadir: núcleo del cometa).
- 11) Se ha confirmado una fuerte anisotropía en el flujo de polvo medido.
- 12) Se ha descubierto una posible correlación entre las emisiones de vapor de agua (medida por el instrumento Rosina) y los eventos de IS y GDS-IS.
- 13) Las velocidades de las partículas medidas a 10 km del núcleo son más lentas que las medidas a 20km, lo que verifica que el polvo se acelera conforme se aleja del cometa. Las velocidades a distancias mayores de 20km, no se ajustan al patrón de aceleramiento esperado y se desconoce la razón.
- 14) Se necesita un análisis más detallado a distintas distancias del cometa para crear un mapa de las áreas de emisión de partículas. Este objetivo será posible con los datos que se esperan obtener hasta el fin de misión.

2.2.7 Resumen y conclusiones

En este capítulo se ha descrito a grandes rasgos la misión espacial Rosetta y en mucho mayor detalle uno de sus instrumentos, G., el objetivo de la herramienta descrita en esta memoria.

Se ha detallado el funcionamiento del HW de G., para poder así conocer en profundidad su comportamiento y explicar minuciosamente los datos generados. Se ha señalado cómo la travesía hasta el cometa, no ha perjudicado en exceso la capacidad de detección de los sensores. Así mismo, se ha indicado el enfoque estadístico del instrumento a la hora de analizar los datos, se han desglosado los límites de detección alcanzables y se han resumido los primeros resultados científicos obtenidos.

El control lógico de G. y la gestión de los datos producidos por el HW es tarea del SW de G. que se tratará en el siguiente capítulo.

3 El software de GIADA

A grandes rasgos, el software de GIADA (**[Mor01a]**, **[Mor01b]**, **[Mor02a]**, **[Mor02b]**, **[Mor02c]** y **[Mor04]**) es el encargado de controlar todo el hardware (*2.2.3.1 El hardware de GIADA*), adquirir y empaquetar los datos obtenidos por los sensores y enviarlos al “spacecraft” (SC). Además, es la única herramienta disponible para realizar modificaciones en vuelo (*3.10 Mantenimiento del software*).

El SW se encuentra almacenado en la memoria ROM de 64KiB en GIADA-2 y se ejecuta sobre un procesador 80c86 a 4 MHz de velocidad. Incorpora su propio sistema operativo, a fin de minimizar espacio y controlar en detalle cada una de las operaciones que se ejecutan.

A continuación se describen los aspectos más importantes del SW de G.: la configuración para su desarrollo, su arquitectura, la interfaz con el HW, la estructura de los bancos de memoria, los modos de funcionamiento, los procesos de lectura del GDS, IS y MBS, el manejo de contingencias, su mantenimiento, las telemetrías y telecomandos implementados, la ingeniería de software aplicada y las medidas de las prestaciones proporcionadas.

El software embarcado en G. fue desarrollado íntegramente por el que suscribe esta memoria, como parte de las responsabilidades del equipo español en el consorcio de desarrollo del instrumento.

3.1 Configuración de desarrollo del SW de GIADA

La configuración para el desarrollo del SW se encuentra en el IAA y se muestra en la siguiente ilustración.



Ilustración 27. Configuración para el desarrollo del software de GIADA.

El “Electrical Ground Support Equipment”(EGSE) es el computador encargado de enviar órdenes (telecomandos) y recibir datos (telemetrías) de G. a través del “Rosetta Spacecraft Interface Simulator“ (ROSIS). Así mismo, almacena, procesa, compara, analiza y representa los datos generados por G., permitiendo crear procedimientos de calibración y test.

ROSIS es el simulador del orbital Rosetta y se encarga de la gestión eléctrica y de suministro de potencia que requiere G., así como de gestionar el interfaz de TC/TM con G.

En GIADA-2 se ubica la EP que se compone la tarjeta analógica y la tarjeta de procesamiento. El resto del de HW de G. (cubierta, GIADA-1, GIADA-3) es reproducido a través de un conjunto de simuladores desarrollados en Italia.

En esta configuración, el EGSE juega el papel de computador de misión en Tierra, ROSIS el del orbital Rosetta y GIADA-2 (con sus simuladores) el papel de G. La propiedad más importante de esta configuración, es que es funcionalmente equivalente a la cadena de datos real que se encuentra órbita. Por lo tanto, no sólo sirvió para el desarrollo del SW, sino que es la encargada de simular los posibles problemas que se encuentren durante la misión, así como de la construcción de actualizaciones de SW para solventarlos. Es por ello, que todos los componentes están protegidos por una mampara y existen elementos redundantes y de repuesto para las partes más sensibles.

3.2 Arquitectura del SW de GIADA

El SW de G., es un SW cuya arquitectura está conducida por eventos HW, como se detalla en la siguiente ilustración.

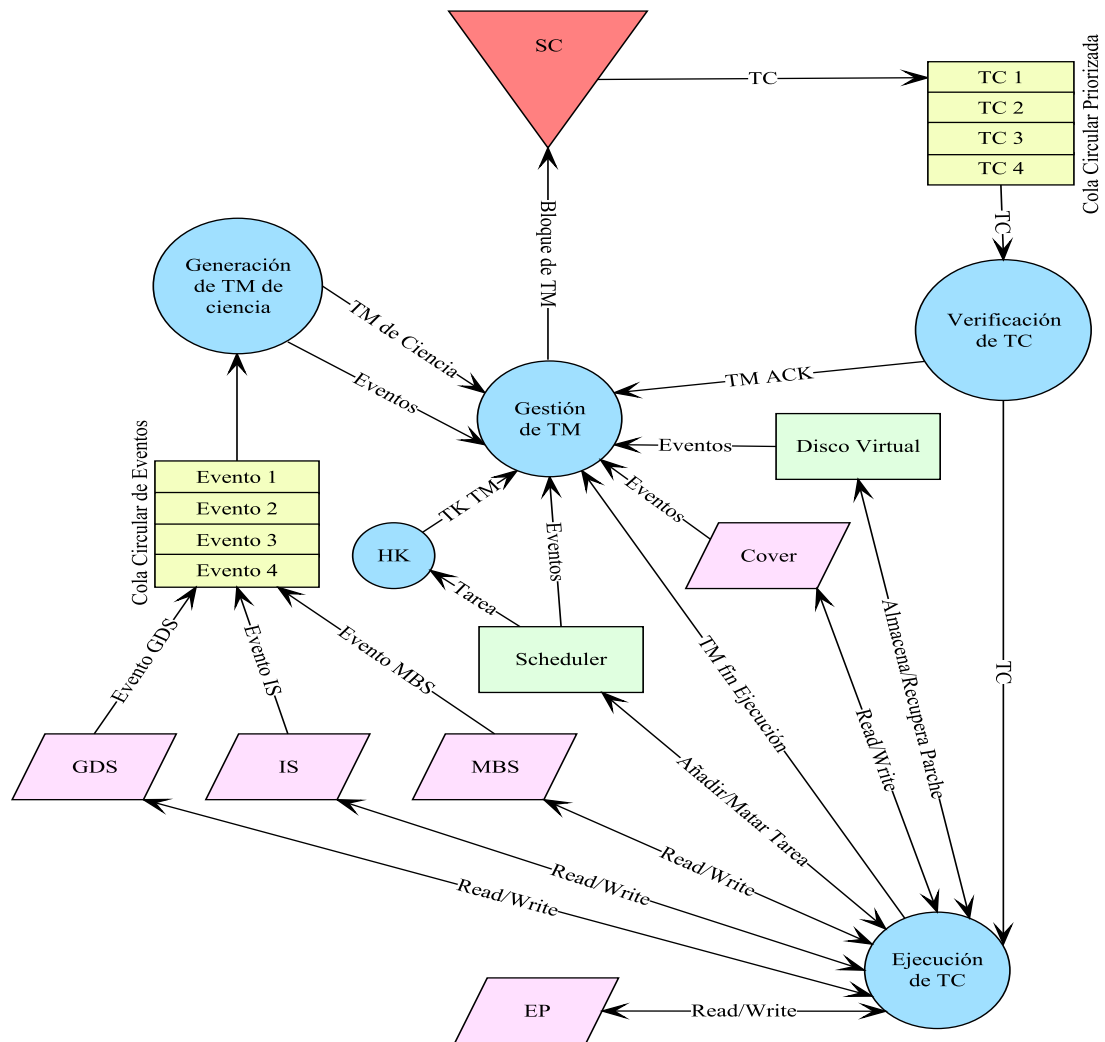


Ilustración 28. Arquitectura del software de GIADA.

Al recibir un TC del SC, éste es almacenado en una cola circular priorizada, para que posteriormente sea extraído de ella, y verificado. El SW, revisa la consistencia de los telecomandos y envía una TM de error o de aceptación. En caso de un TC erróneo, la TM de error que se genera indica

con suficiente claridad la naturaleza del error y se suspenderá la aceptación de TC durante un período de tiempo (15s). G. solo es capaz de ejecutar un solo comando a la vez, y sólo inicia la ejecución del próximo comando cuando ha acabado de ejecutar el anterior. Una excepción a esta norma son los comandos priorizados (TC "Accept Time Update" y TC "Safe Mode"), que interrumpen la ejecución de un posible comando que se estuviera ejecutando, para comenzar su propia ejecución. Los telecomandos pueden afectar a cualquier subsistema, al disco virtual (DV) o al planificador ("scheduler").

El planificador se encarga de ejecutar de forma periódica las tareas programadas, como son la generación del estado de G. o "Housekeeping" (HK) o la medición de los sensores de temperatura para la detección de problemas (contingencias).

El disco virtual (DV) almacena los parches (modificaciones al comportamiento del instrumento) que serán leídos y aplicados según se indique en el fichero de contexto (FC). Este fichero guarda la configuración de los parámetros de ejecución del SW. Los eventos indican error o progreso de una actividad y pueden ser generados por cualquier subsistema o por un módulo de SW. Los datos de ciencia generados por GDS, IS o MBS se guardan en una cola circular para su posterior procesamiento, lo que finalmente conduce a la creación de una TM de ciencia. Finalmente, las telemetrías se aglutinan en bloques (3.12.3 *Formato y estructura de las telemetrías de GIADA*) que son enviados al SC.

3.3 Interfaz software-hardware

Estos eventos generados por el HW son comunicados a través de interrupciones al procesador, como se muestra en la siguiente ilustración:

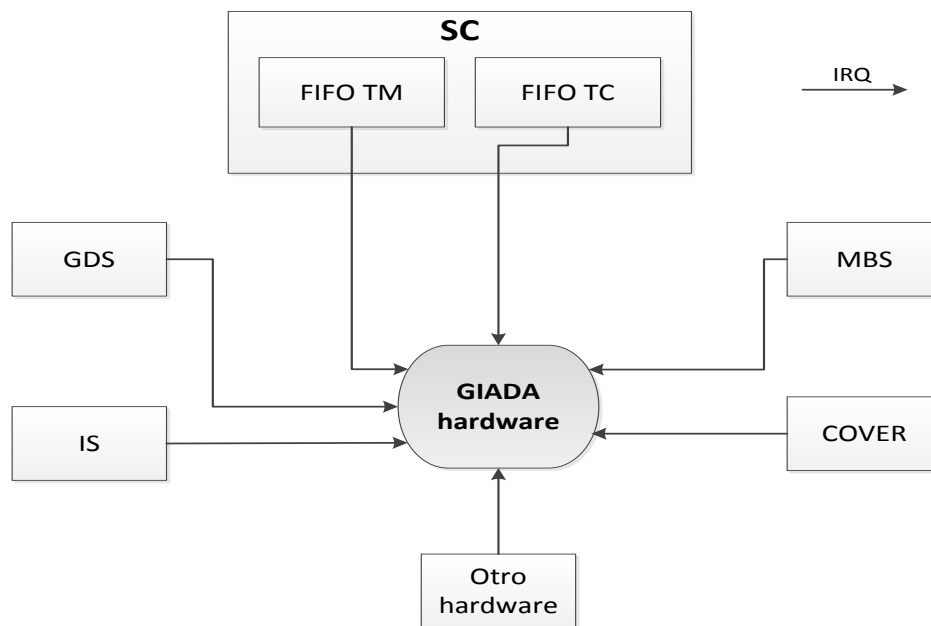


Ilustración 29. Eventos en el software de GIADA.

El SC envía telecomandos a G. a través de una memoria FIFO ("FIFO TC") que generará una interrupción ("interrupt request" IRQ) cuando un TC es completamente recibido. G. envía telemetrías (realmente un bloque o conjunto de telemetrías) al SC a través de la "FIFO TM". La detección de partículas por GDS e IS y el flujo de polvo (número de partículas por segundo) medido por el MBS serán notificados al SW mediante interrupciones al procesador. Después de la notificación y dentro de la rutina de atención a la interrupción, se leerán los registros intermedios presentes en la electrónica, que almacenan los datos obtenidos. Estos datos son empaquetados en forma de tele-

metrías. La cubierta, durante el movimiento de apertura o cierre, genera un conjunto de interrupciones. El SW al procesarlas, recabará datos del estado de la cubierta hasta conformar un informe del movimiento que será enviado al SC en forma de TM. El resto del HW, informará a G. de eventos anómalos (como corrupción de memoria o desbordamiento) o de fin de temporizadores (contadores).

El conjunto de todas las interrupciones implementadas en G. es el siguiente:

Tabla 7. Tabla de interrupciones y prioridades en GIADA.

Tipo de interrupción	Número de interrupción / prioridad de la interrupción	Nombre de la interrupción	Descripción
Reservadas para el procesador	0x00	INT0	Error de división por 0
	0x04	INT1	CPU en modo paso a paso
	0x08	NMI	Interrupción no enmascarable
	0x0c	INT3	IRQ de depuración
	0x10	INT4	Desbordamiento de los contadores de la CPU
Generadas por el HW	0x20	"Serial"	Byte recibido/enviado por el puerto de Test
	0x21	"Cover_motor"	Paso en el motor de cubierta
	0x22	"MBS_event"	Tiempo de integración de una MB finalizado
	0x23	"IS_event"	Partícula detectada en el IS
	0x24	"GDS_event"	Partícula detectada en el GDS
	0x25	"Timer IRQ"	IRQ del contador genérico
	0x26	"TSYNC"	Pulso de sincronización de tiempo
	0x27	"TC timeout"	Tiempo superado en la recepción de un TC
	0x28	TM	TMFIFO IRQ
	0x29	TC	TC FIFO IRQ
	0x30	"RTClock"	Interrupción de tiempo
	0x31	EDAC	Error de EDAC

Hay que indicar que las interrupciones son anidables, esto es, si se está atendiendo a una interrupción A con número de interrupción “n” y ocurre otra interrupción B con un número “m” tal que “m” > “n”, entonces la rutina de atención a la interrupción de A es pausada y comienza la atención de la rutina B. Cuando termine el procesamiento de B, se continuará con el de A.

3.4 Estructura de los bancos de memoria de GIADA

G. posee tres tipos diferentes de bancos de memoria:

- a) Memoria de RAM de 64KiB. Donde se almacenan (datos y código), los modos de operación parcheables (modificables): cubierta, normal y flujo. Así como los datos el FC usado en la ejecución.
- b) Memoria de ROM de 64KiB. Donde se ejecutará el modo de operación seguro.
- c) Memoria no-volátil RAM (NVRAM) de 64KiB. Donde se almacena una copia del FC y un DV que permite almacenar los parches a aplicar a G.

La estructura de la memoria RAM consta de un segmento de vectores de interrupción (“Interruptions”), uno de código de programa (“CODE”), un segmento libre (“FREE”) usado para parches, un segmento de datos inicializados (“DATA”), un segmento de datos no inicializados (“BBS”) y un segmento de pila (“STACK”).

La estructura de la ROM consta de un segmento para el “DUMPER”, un segmento libre (“FREE”) no utilizado, uno de código de programa (“CODE”), un segmento de datos (“DATA”) y el vector de “Reset”.

La NVRAM almacena una copia del FC y el DV. La estructura de las tres memorias se expone en la siguiente ilustración:

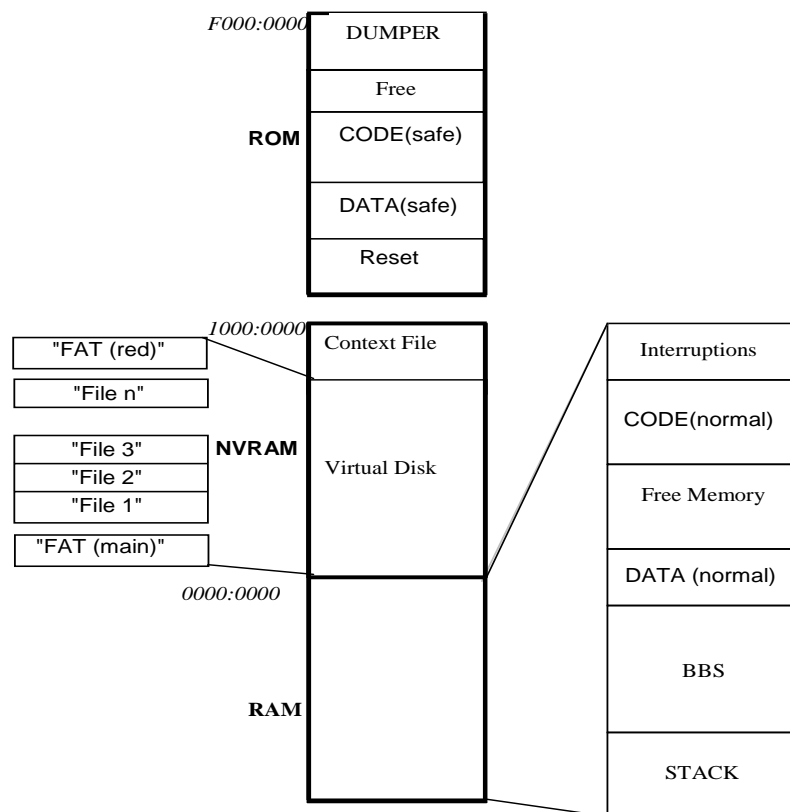


Ilustración 30. Estructura de la memoria en GIADA.

Durante el proceso de encendido, el SC alimentará la unidad de potencia de G. En este momento, la CPU comienza la ejecución del programa cuya dirección de inicio se encuentra en el “vector de reset”. En el caso de G., este vector apunta al programa “dumper” cuya misión es borrar la RAM y copiar los segmentos de CODE y DATA de ROM en RAM. En estos segmentos se encuentran los modos operativos: *cubierta*, *normal* y *flujo*. Una vez terminada la copia, el dumper salta a la primera instrucción del programa en ROM para ejecutar el modo operativo *seguro*. La ejecución en ROM del modo inicial de un instrumento, tras su encendido, es un requerimiento de ESA. En este punto y para ejecutar el resto de modos, bastará con pasar la ejecución a RAM. El proceso de copia de ROM a RAM se muestra en la siguiente ilustración:

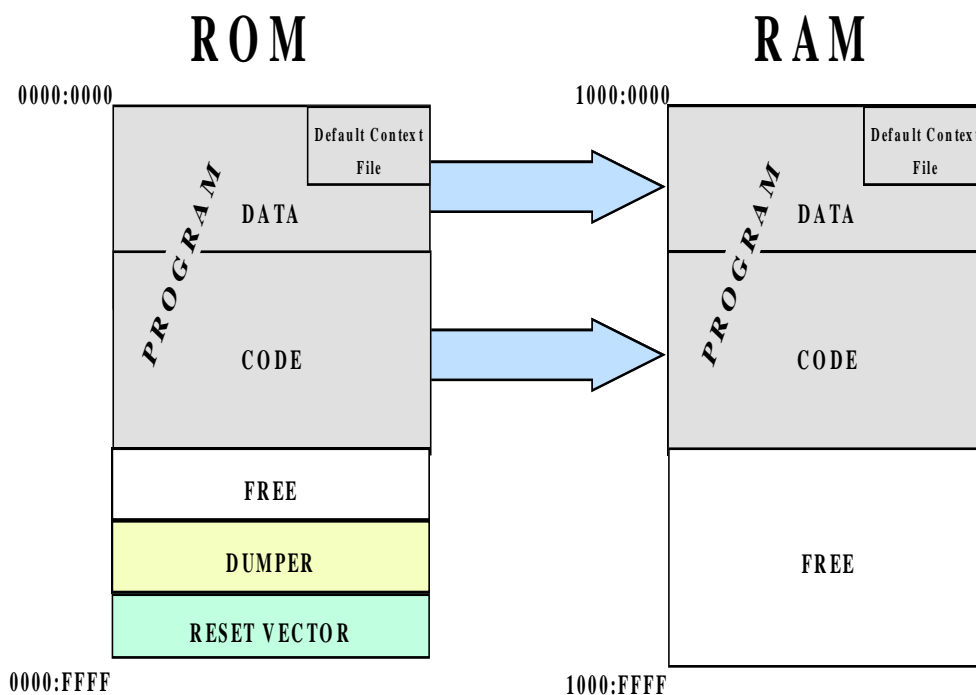


Ilustración 31. Proceso de copia de la memoria ROM y RAM en GIADA.

La memoria de datos NVRAM requiere de una gestión especial. Se encuentra conectada a una línea de alimentación “Keep Alive Line” independiente de la alimentación de G., protegiendo así la información almacenada (FC y DV) de un apagado voluntario o involuntario del instrumento.

El DV implementa un sistema de archivos para el almacenamiento de ficheros. El DV cuenta con un conjunto de mecanismos de seguridad (EDAC) y un conjunto de sumas de seguridad o “checksums” de las zonas críticas. Así mismo cuenta con “file allocation table” (FAT) redundantes que proporcionan un mecanismo extra de seguridad en el manejo de ficheros.

3.5 Modos operativos

G. dispone de cuatro modos operativos (de funcionamiento) que pasan a describirse:

- 1) **Modo seguro (“safe”).** Este es el modo en el que se ejecuta G. tras un encendido, cuando no es necesario la obtención de datos científicos o cuando hay que solventar una contingencia. Permite la aceptación y ejecución de TC, la generación de HK y el manejo de memoria (parches y volcados). Por motivos de seguridad, los parches sólo pueden ser aplicados al resto de modos, no al propio modo *safe*. Esto asegura que después de un encendido, el SW siempre se ejecutará en un modo conocido y no

modificable. Par a reforzar la seguridad, este modo se ejecutará en memoria de sólo lectura (ROM).

- 2) **Modo cubierta (“cover”).** En este modo se permite abrir/cerrar la cubierta, probar/activar el “frangibolt y probar/activar los calentadores de la cubierta.
- 3) **Modo flujo (“flux”).** Este modo está pensado para las fases de la misión por las que por motivos de consumo, y debido a que el volumen de partículas esperado es pequeño, no se justifica el encendido de los subsistemas de detección de partículas (GDS e IS). Se permite el manejo únicamente del MBS.
- 4) **Modo normal.** En este modo todos los sistemas están activos (GDS, IS y MBS), en el que la cubierta debe estar abierta. Este es el modo nominal de funcionamiento del instrumento para la recogida de datos científicos. En caso de problemas con algún subsistema, es posible apagarlo individualmente. Se permite realizar calibraciones, gestión de contingencias y la ejecución desde el SC de ciertos procedimientos “on board control procedure” (“OBCP”) previamente acordados.

La transición entre modos se realiza a través de TC dedicados y ateniéndose a las reglas expuestas en el siguiente gráfico.

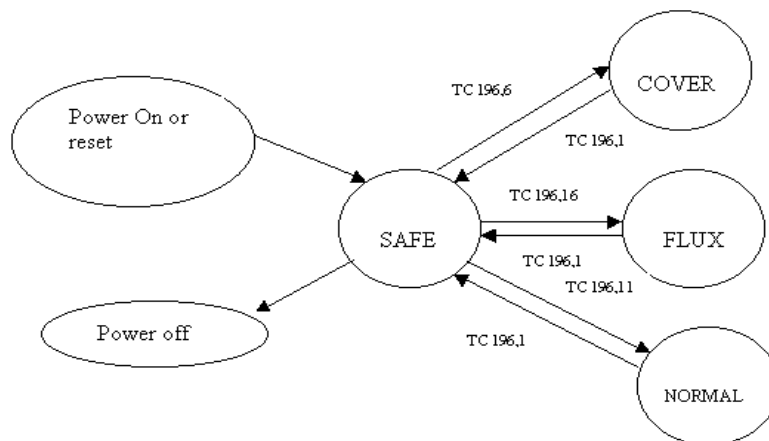


Ilustración 32. Transiciones entre modos de GIADA.

La lista de modos operativos G., así como la memoria en la que se ejecutan y la media y pico del volumen de datos generados, se muestran en la siguiente tabla:

Tabla 8. Modos operativos y volumen de datos en GIADA.

Modo	Subsistema activo	Memoria	Media de volumen de datos [Kib/s]	Pico de volumen de datos [Kib/s]
Safe	EP	ROM	0.1	0.9
Cover	EP + Cover o Frangibolt	RAM	0.1	4
Flux	EP + MBS	RAM	0.1	4
Normal	EP+cualquier combinación de GDS y/o IS y/o MBS	RAM	12	64

Los modos *normal* y *flux*, G. se ejecuta autónomamente conforme a una configuración predefinida. Esta configuración, junto a otros parámetros importantes de G. se almacena en el *fichero de contexto* (FC). Existen dos copias del FC, una en RAM y otra en NVRAM. Este FC, al residir unas de sus copias en el NVRAM, permite a G. iniciar su ejecución con valores de funcionamiento conocidos y seguros (los de RAM pueden haber sido modificados durante la ejecución). En el proceso de arranque la copia en NVRAM es copiada en RAM (*Ilustración 31. Proceso de copia de la memoria ROM y RAM en GIADA.*), y será esta copia en RAM la que se utilizará en la ejecución del programa.

3.6 Proceso de lectura del GDS

La lectura de datos científicos en el GDS, comienza con la inicialización del vector de interrupción (“InitializeGDSIRQ”) al comienzo de la rutina que gestionará dichos eventos. Esta operación se realiza al cambiar a modo de operación normal. Después de entrar en dicho modo, se procede al encendido del GDS (TC “SetGDSOnOff (ON)”), si así lo indica el FC. Después se arma el láser (TC “ArmLaser”). Este comando es necesario, ya que el láser era potencialmente peligroso (durante las operaciones en la Tierra) y necesita de medidas de seguridad adicionales antes de activarlo. Paso seguido, se enciende el láser (TC “SwitchLaserOnOff”), lo que permite recibir partículas. Si la partícula es detectada por el GDS (“particle detected”), se generará una interrupción (“GDSIRQ”), se almacenarán los datos y se inicializará la electrónica del GDS a fin de preparar una nueva detección.

Los valores de la detección de la partícula se guardan en registros intermedios de GIADA-2. Estos registros son leídos por la propia rutina de atención de la interrupción y añadidos a la cola de eventos, para un pareado con un evento posterior de IS (produciendo un evento GDS+IS) o para ser añadidos a una TM de ciencia de forma individual.

Con el láser activo, será posible desde Tierra, calibrar el GDS (TC “CalibrateGDS”), establecer el umbral de los fotodiodos receptores (TC “SetPhotoDiodeThreshold”) o cambiar el modo de operación del láser, para que genere más o menos potencia de salida (TC “SetGDSOperationMode”).

Con el láser encendido, se aplicará un chequeo de las temperaturas (“CheckErrorInLaserTemperature”) que comprobará que se encuentran en el rango nominal de operación, en caso contrario se generará una contingencia (3.8) que apagará el GDS. La gestión del GDS termina con el apagado del subsistema (TC “SetGDSOnOff (OFF)”).

Todos estos procesos se resumen en la siguiente ilustración:

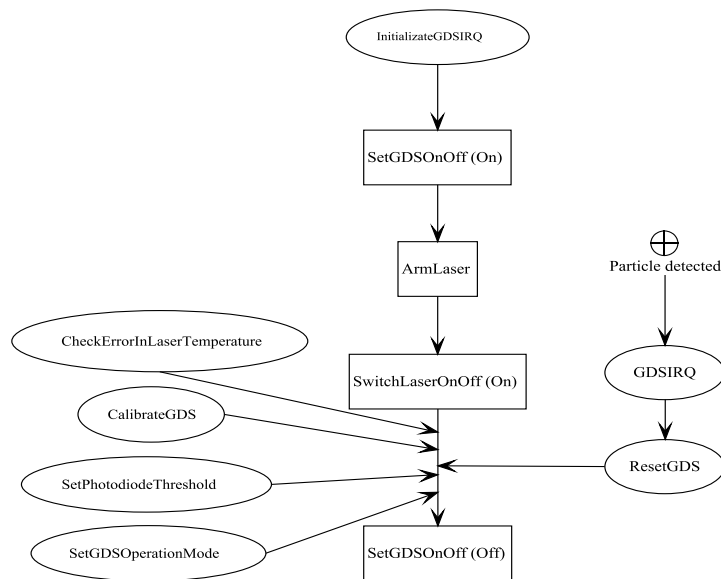


Ilustración 33. Proceso de lectura del GDS.

3.7 Proceso de lectura del IS

El proceso de lectura de datos científicos en el IS, se inicia al entrar en modo *normal*, que establece el vector de interrupción (“InitializeISIRQ”) al comienzo de la rutina de gestión de eventos. A continuación, se procede al encendido del IS (TC “SetISOnOff (ON)”), lo que permite recibir partículas. Si la partícula es detectada por el IS (“particle detected”), se generará una interrupción (“ISIRQ”), se almacenarán los datos y se inicializará la electrónica del IS, a fin de preparar una nueva detección.

Los valores de la detección de la partícula se guardan en registros intermedios de GIADA-2, que serán leídos y añadidos a la cola de eventos, para un posterior pareado con evento previo de GDS (produciendo un evento GDS+IS) o para ser añadidos a una TM de ciencia de forma individual.

Con el IS activo, es posible obtener una medida del flujo (“GetDustFlux”) que se almacenará en el HK y que cuantifica el número de partículas detectadas por el IS en el último minuto. Así mismo se activará una monitorización del sensor de temperatura del IS (“ISTemperatureMonitor”), a fin de cerrar la cubierta si se alcanza una temperatura determinada (posiblemente por una exposición directa al sol). Será posible desde Tierra, calibrar el IS (TC “CalibrateIS”), establecer el umbral de los PZT (TC “SetPZTThreshold”) o cambiar el modo de operación del (“SetISOperationMode”) para activar o desactivar PZT y establecer sus ganancias.

Con el IS encendido, se aplicará un chequeo de las temperaturas (“CheckErrorInISTemperature”) que comprobará que se encuentran en el rango nominal de operación, en caso contrario se generará una contingencia (3.8) que apagará el IS. La gestión del IS termina con el apagado del subsistema (TC “SetISOnOff (OFF)”).

Todo este proceso se resume en la siguiente ilustración:

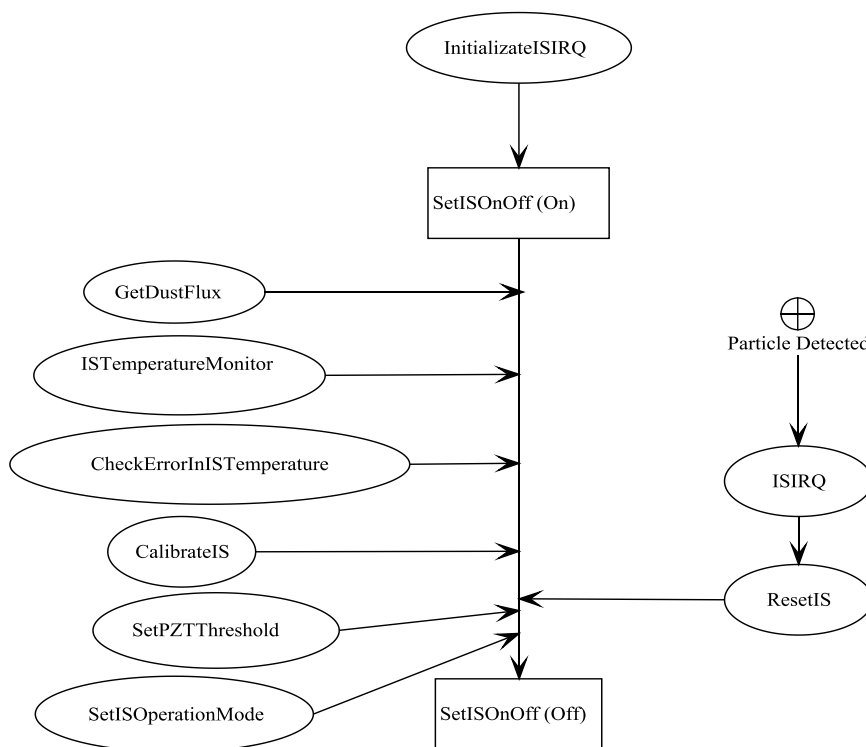


Ilustración 34. Proceso de lectura del IS.

3.8 Proceso de lectura del MBS

La lectura del MBS, comienza con al entrar G. en modo *normal* o *flux*, situando el vector de interrupción (“InitializeGDSIRQ”) al comienzo de la rutina de gestión de eventos de MBS. Posteriormente, se procede al encendido MBS (TC “SetMBSOnOff (ON)”), lo que permite realizar medidas del flujo de polvo. Así mismo, se programa una lectura de frecuencia y temperatura (“MB Reading”) de cada MB. Esta lectura generará una interrupción (“MBSIRQ”) que permitirá almacenar, en registros de GIADA-2, los datos obtenidos. Después de la lectura, se realizará una inicialización de la electrónica del MBS, a fin de preparar la siguiente lectura. Todos los datos leídos se introducen en la cola de eventos, para ser añadidos a una TM de ciencia.

Con el MBS activo, es posible desde Tierra, establecer la frecuencia (por defecto se usará el valor del FC) de lectura periódica de las microbalanzas (TC “SetTimeBetweenMBSMeasures”), indicar que MB están activas (“SetMBSOperationMode”), proceder a una limpieza mediante calentamiento (TC “HeatMB”) o calibrar el MBS (“TC CalibrateMBS”).

Con el MBS encendido, se aplicará un chequeo de temperaturas (“CheckErrorInMBSTemperature”) que comprobará que se encuentran en el rango nominal de operación, en caso contrario se generará una contingencia (3.8) que apagará el MBS. La gestión del MBS termina con el apagado del subsistema (TC “SetMBSOnOff (OFF)”).

Todo este proceso se resume en la siguiente ilustración:

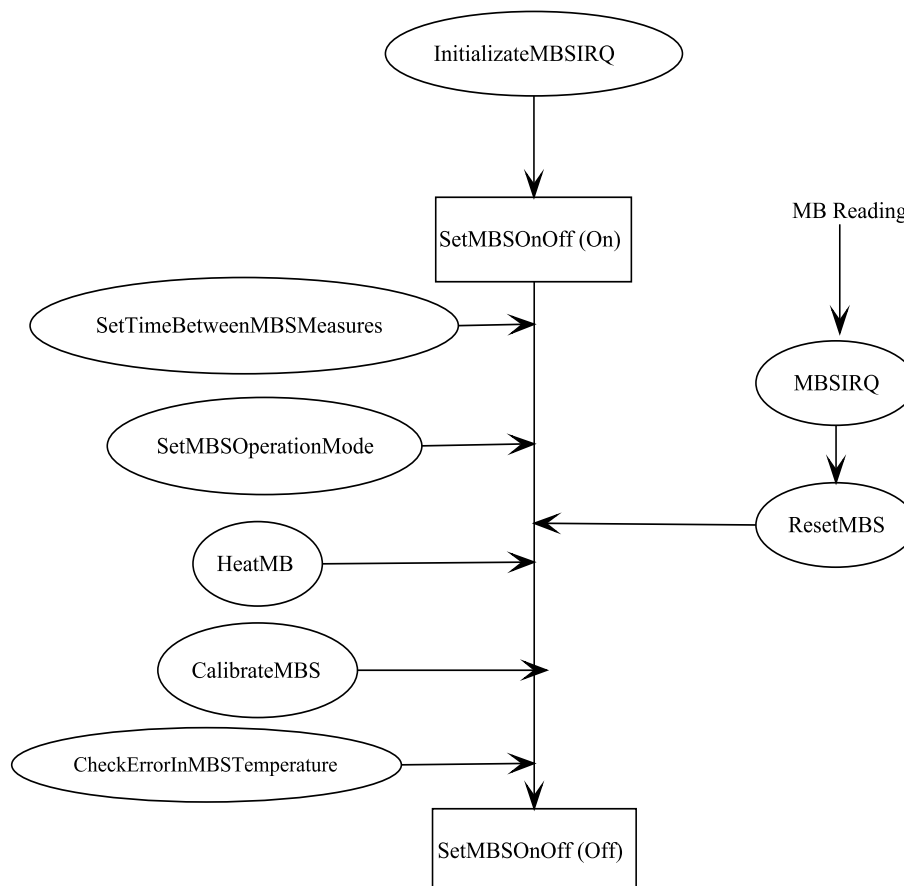


Ilustración 35. Proceso de lectura del MBS.

3.9 Gestión de contingencias

Una contingencia es la constatación de un problema a bordo de un instrumento. En el caso de G., las contingencias son termales, se producen cuando los sensores de temperatura que incorpora el instrumento, al realizar medidas de forma periódica, obtienen valores fuera de un rango (almacenado en el FC).

Asociado a cada tipo de contingencia, existe una acción de recuperación que minimiza o resuelve el problema detectado. La acción de recuperación puede llevarse a cabo en el propio instrumento o fuera de él (por ejemplo en el SC). En G. existen otros tipos de contingencias a parte de las termales, pero están asociadas al HW (EDAC, división por cero, desborde de temporizadores...) y normalmente implican un malfuncionamiento del HW. En estos casos, la acción de recuperación no es automática y requiere de un análisis detallado en Tierra. Es importante tener en cuenta que el retardo de comunicaciones entre Tierra y Rosetta puede suponer días. Por tanto, es crítico, que las contingencias se resuelvan autónomamente siempre que sea posible, ya sea a nivel de instrumento o de SC.

Todos los subsistemas: EP, GDS, IS, MBS y cubierta están equipados con sensores de temperatura, cuya misión es la de velar por los límites térmicos establecidos en el FC. También, es en el FC, donde se encuentra la periodicidad de las medidas, que en su valor nominal es de 40s.

La medición de la temperatura de la EP se realiza siempre que G. esté encendida, independientemente del modo operativo. Para la EP se ha definido un límite máximo de temperatura, por encima del cual, la acción de recuperación asociada consiste en pedir a Rosetta que apague a G.

La medición de la temperatura del IS tiene dos facetas:

- a) *Lectura independientemente del estado (encendido o apagado) del IS en los modos de normal y flux.* Se comprueba que esta lectura se sitúe entre un límite máximo y otro mínimo (presentes ambos en el FC). Con ello se pretende prevenir la incidencia del Sol en el interior de G. y el exceso de enfriamiento por radiación al vacío, dado que en estos dos modos la cubierta se encuentra abierta.
- b) *Lectura sólo cuando está encendido el IS.* Se verificando que dicha lectura no supera el valor de temperatura máxima (almacenado en el FC).

Cuando una temperatura sobrepasa los límites establecidos en GDS, IS (caso b) o MBS, el SW crea una contingencia. La acción de recuperación consiste en enviar un evento al SC que indica el subsistema y la temperatura alcanzada. Además se pide al SC que ejecute un proceso "OBCP Close Cover", cuyo cometido es ir a modo seguro (apagando GDS, IS, MBS) y cerrar la cubierta.

Cuando la medida de la temperatura del IS es sobrepasada (caso a), la acción de recuperación es la siguiente: se genera un evento indicando este hecho, incluyendo la temperatura alcanzada y además se apaga el IS autónomamente para enfriarlo. Si en las siguientes medidas periódicas, la temperatura se encuentra dentro de un límite de seguridad por debajo del máximo, se generará un evento y se encenderá el IS autónomamente para calentarlo.

3.10 Mantenimiento del software

La única forma posible de cambiar la funcionalidad del instrumento en vuelo es mediante modificaciones al SW llamadas "parches". En general, estas correcciones están destinadas a solventar posibles problemas detectados en el SW o HW o bien para adaptar el sistema a los avatares de la misión.

Los parches pueden aplicarse a RAM (afectando al modo de operación que se está ejecutando) o bien a la NVRAM (modificando un modo de operación que tomará efecto cuando se utilice). Sólo

los parches aplicados a RAM se almacenarán en el DV. Los parches a NVRAM se aplicarán pero no se guardarán en el DV. En el FC existe un campo especial en el que se indica a G. qué parches, guardados previamente en el DV, se deben de aplicar automáticamente al arranque de G. Los parches aplicados se especifican en una de las secciones del HK.

El proceso nominal de aplicación (ver **3.11 Ingeniería de software**) de un parche, se inicia detectando qué funcionalidad del SW se desea modificar a nivel de código fuente. Una vez localizado, se procede a actualizarlo hasta obtener el resultado deseado. La verificación de este resultado, ha de realizarse con la cadena de datos en Tierra (**3.1 Configuración de desarrollo del SW de GIADA**) y requiere de la compilación del código fuente.

La compilación es la traducción de las operaciones del código fuente a operaciones de CPU. El resultado de una compilación es un fichero ejecutable, cuyo formato es una secuencia hexadecimal. Por tanto, un parche, se plasma mediante parte de la secuencia hexadecimal generada en la compilación. Cómo localizar la secuencia correcta y la problemática asociada a la reubicación del código es detallada en **[Mor04]**.

Con la secuencia del parche localizada, se ha de iniciar el proceso de modificación del SW según el protocolo ESA. Este proceso incluye rellenar una serie de documentos que definen exactamente qué se ha a cambiar, porqué, cómo se ha de realizarse el cambio y cómo ha realizarse la verificación. Finalmente, la secuencia hexadecimal se pasa a un formato especial reconocido por el centro de operaciones de ESA (ESOC) que los transformará en telecomandos (TC “Load Memory Absolute Addressing”) y los encolará para enviarlos a Rosetta dentro de la siguiente ventana de comunicación disponible.

Con los TC ya en Rosetta, serán enviados a G. que los aplicará. Normalmente, antes de aplicar un parche, se hace un volcado (TC “Dump Memory Absolute Addressing”) de la zona a parchear. Este volcado llegará a Tierra en forma de TM “Dump Memory Absolute Addressing Report”. Se realizará otro volcado después de aplicar la modificación para verificar que la aplicación se ha llevado de forma correcta. El primer o segundo volcado, puede sustituirse por un TC “Check Memory Absolute Addressing” que no es más que una suma de comprobación del área afectada y que minimiza el tamaño de la TM de respuesta.

El mantenimiento del SW es una actividad complicada, laboriosa, peligrosa y en su mayor parte realizada de forma manual. Es por ello que se intenta reducir su uso a situaciones muy controladas.

3.11 Ingeniería de software

Siguiendo las recomendaciones ESA, el ciclo de vida del SW de G. ha sido el siguiente.

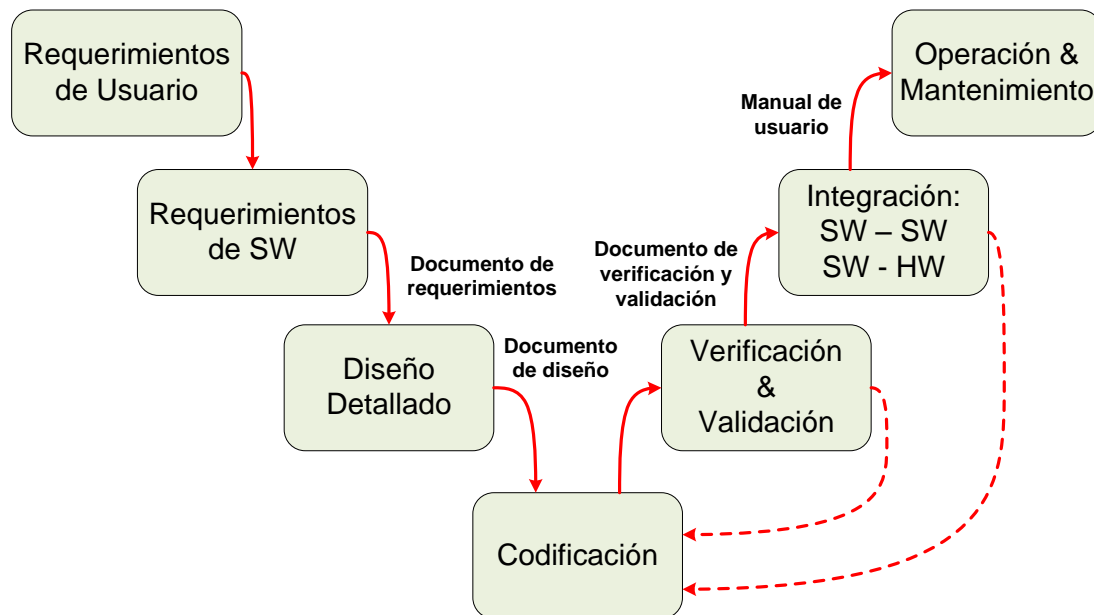


Ilustración 36. Ciclo de vida del software de GIADA.

Partiendo con la descripción de requerimientos al SW ([Mor01a]) y la descripción del HW de DPU y memoria expuesto en [Lop00] se creó un documento de diseño ([Mor02a]), que fundamentó la escritura del código fuente. Cuando el SW estuvo implementado, se le aplicó el proceso de verificación y validación ([Mor02b]). Posteriormente, se pasó a la fase de integración del SW y HW ([Mor02c]) y una vez superada se creó un manual de usuario ([Mor04]) para utilizarlos en la fase de operación y mantenimiento. Todos estos documentos se realizaron cumpliendo los estándares marcados por ESA, fueron revisados por los departamentos de calidad de las empresas privadas: Sener (Madrid) y “Oficine Galileo” (Florencia, Italia) y finalmente aceptados por la ESA.

Las versiones públicas del SW de G. han sido:

- 1) GIADA SW 1.0. Para el modelo de cualificación eléctrica.
- 2) GIADA SW 1.2. Aplicado en el modelo de cualificación eléctrica.
- 3) GIADA SW 2.0. Para el modelo de vuelo.
- 4) GIADA SW 2.3. Para el modelo de vuelo y de repuesto.

El SW de G. ya está “congelado”, esto es, no es posible realizar modificaciones directas. Pero en el caso de descubrir algún tipo de error (SW o HW) todas y cada una de las modificaciones (mediante parches) a aplicar generarán una documentación destinada a mantener la trazabilidad del cambio.

El proceso de modificación del SW es el siguiente: partiendo de una NCR (“Non Conformance Report”) que descubre un comportamiento anómalo/mejora, se analiza el SW y se genera un informe SPR (“Software Problem Report”) que dará lugar a una propuesta de modificación SMR (“Software Modification Report”) y a fichero de parche MPR (“Memory Patch Report”), más los resultados de los tests HW y SW que atestiguan el error y su corrección (“Test procedure” para realizar la verificación y “Test Report” con el resultado de la misma).

La versión del SW 2.3 tiene configurados 3 parches que solventan 5 problemas detectados después de congelar el SW:

- 1) RO-GIA-IAA-SPR-021. Implementa un filtro por tiempo que mejora la diferenciación de los eventos de calentamiento de los de lectura de MBS.

- 2) RO-GIA-IAA-SPR-022. Evita ciertos eventos espurios en GDS e IS al establecer sus umbrales
- 3) RO-GIA-IAA-SPR-023. Este parche (y las dos siguientes) mejoran la adquisición de partículas en el IS cuando la señal se matiné por encima del umbral.
- 4) RO-GIA-IAA-SPR-023.IS Blind
- 5) RO-GIA-IAA-SPR-023.5 IS Reset Pulses

El desarrollo del SW de GIADA ha necesitado un conjunto de herramientas comerciales y otras diseñadas específicamente que se listan a continuación.

- 1) Compilador C. Borland C++.Version 3.1. Copyright (c) 1990, 1992 by Borland International, Inc.
- 2) Linker C. Turbo Linker. Version 5.1. Copyright (c) 1992 Borland International
- 3) Ensamblador MICE.
- 4) Cross Assembler.Version 4.05b. 8086. Copyright (c) 1985 by 2500 A.D. Software, Inc.
- 5) Linker MICE. A.D. Linker.Version 4.05a. 2500. Copyright (c) 1985 by 2500 A.D. Software, Inc. Debugger 8086.
- 6) USD III. Versión 2.2. Copyright (c) 1991 by Microket Internacional Inc.
- 7) Desensamblador. W32Dasm. Version 8.93. Copyright (c) 1998 by URSoftware Co.
- 8) SW de estimación de tiempo de CPU. ASMFLOW Professional. Version 3.08. Copyright (c) 1988 - 1995 Quantasm Corporation,
- 9) SW de depuración de TC y TM a alto nivel interfaz Loader.
- 10) SW de depuración de TC y TM a alto nivel interfaz Dumper.
- 11) SW de conversión de tablas de símbolos del HP16500C a MICE.
- 12) SW de simulación de conversiones de ADC.
- 13) SW de formateo de comentarios de fuente en C.
- 14) SW de generación automática de parches de FC.
- 15) SW de generación automática de código ASM del dumper.
- 16) SW de simulación del algoritmo de cálculo de Dust Flux .
- 17) SW de cálculo de CRC.
- 18) SW de generación de códigos Hamming.
- 19) SW de parcheo de modificación del segmento de código en ejecutable.
- 20) SW de intercambio de bytes altos y bajos.
- 21) SW de cálculo del tamaño final del ejecutable.
- 22) SW de generación de parches automáticos.
- 23) "EGSE Scripts" para "GIADA SW Test Procedures EQM model".
- 24) "EGSE Scripts" para "GIADA SW Test Procedures FM model".
- 25) "EGSE Scripts" para "GIADA SW Test Procedures FS model".
- 26) Cumplimentar la BD de Rosetta ("RSDB").

Junto a las herramientas, se ha generado un gran volumen de información relativa al SW de G., típica de una misión espacial. Toda la documentación generada está a disposición pública en ESA e IAA y ha sido incluida en los diferentes paquetes de documentación de los diversos modelos. Se citan sólo los documentos que han tenido una relación directa con el SW de G., omitiendo la evolución de ediciones y/o revisiones de cada uno de ellos, citando sólo la última versión.

- 1) RO-EST-RS-3009/EID B. "GIADA SW ICD".
- 2) RO-GIA-IAA-RS-001. "GIADA SW Requeriment Document".
- 3) RO-GIA-IAA-DD-002. "GIADA SW Detailed Desing".
- 4) RO-GIA-IAA-TS-001. "GIADA SW SVVP".
- 5) RO-GIA-IAA-MA-009. "GIADA FS SW User Manual".

- 6) RO-GIA-IAA-MA-005. "GIADA FM SW User Manual".
- 7) RO-GIA-IAA-RP-001. "GIADA EQM SW Test Procedures".
- 8) RO-GIA-IAA-RP-002. "GIADA FM SW Test Procedures".
- 9) RO-GIA-IAA-RP-003. "GIADA FS SW Test Procedures".
- 10) RO-GIA-IAA-TN-010. "GIADA Interruption Sources and Services".
- 11) RO-GIA-IAA-TN-012. "DPU Peripheral Description, Memory and I/O Map".
- 12) RO-GIA-IAA-TN-014. "GIADA SW 2.0 Time Measures".
- 13) RO-GIA-IAA-TN-015. "GIADA SW 2.0 New Architecture".
- 14) RO-GIA-IAA-TN-018. "GIADA Software Recovery Actions. RO-GIA-IAA-TN019".
- 15) RO-GIA-IAA-TN-019. "GIADA Software Changes from 2.0 to 2.3".
- 16) RO-GIA-IAA-TN-020. "GIADA SW 2.3 Time Measures".
- 17) "GIADA SW 2.0 Patches: SPR, SMR, MPR and verification report".
- 18) "GIADA SW 2.3 Patches: SPR, SMR, MPR and verification report".

3.12 Datos generados por GIADA

El objetivo de este apartado es describir los datos producidos por G. Inicialmente, se describirá cómo los datos son enviados y recibidos en vuelo por/desde G., se detallarán los TC y TM generado, así como el formato y la composición de telemetrías generadas. Se continúa con algunos aspectos relevantes de los datos: precisión y marcas temporales en TM y TC y las curvas de calibración.

3.12.1 Flujo de datos de GIADA

El flujo de datos generado y recibido por G. desde la Tierra se describe en la siguiente ilustración.

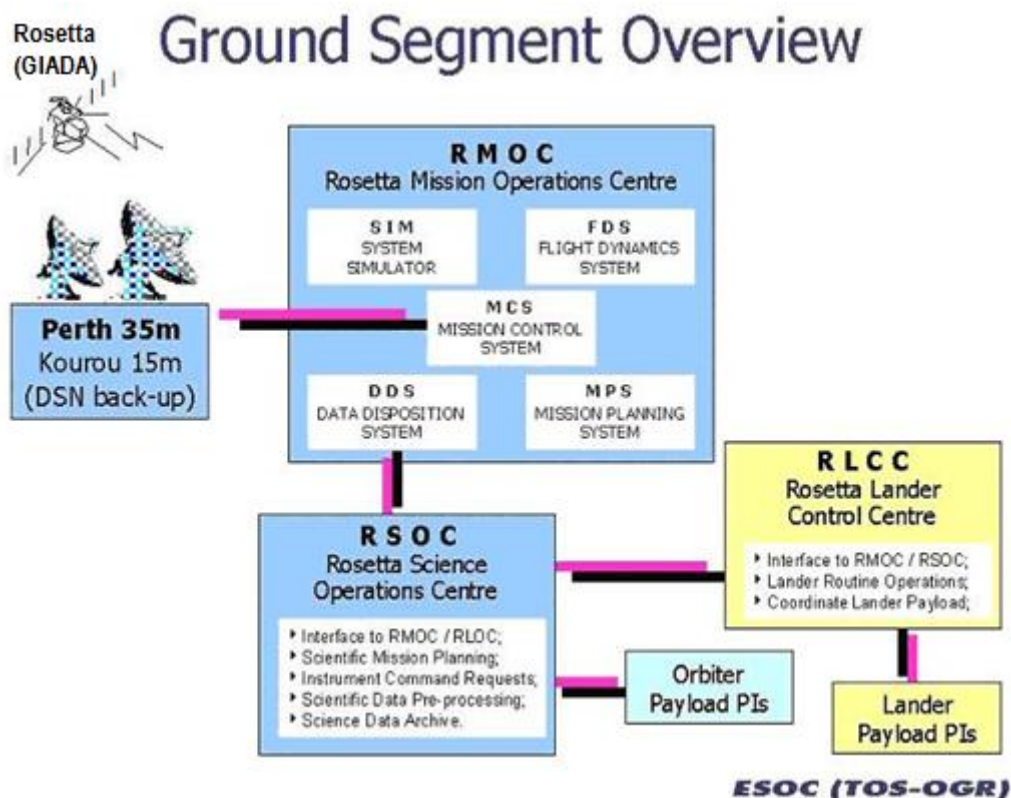


Ilustración 37. Flujo de datos desde GIADA hasta la Tierra.

Las TM son originadas por G. y se agrupan en un bloque (*3.12.3 Formato y estructura de las telemetrías de GIADA*) antes de enviarlas al SC. Ya en el SC, los bloques se almacenan hasta la llegada de la próxima ventana de comunicaciones con la Tierra. En este momento, se utilizará la antena parabólica que incorpora Rosetta para transmitirlos. Los bloques serán recibidos por la estación de seguimiento de espacio profundo que dispone ESA en Perth (Australia). Tras su recepción, los paquetes son enviados al *centro de operaciones de la misión Rosetta* (RMOC) que se encuentra en ESOC (Alemania) y de ahí al *centro de operaciones científicas de Rosetta* (RSOC). En este punto se extraen las telemetrías de los bloques y se enviará a los correspondientes EGSE de cada instrumento. Dado que el “lander” exiten varios sub-instrumentos, se creó un camino especial para sus paquetes a través del *centro de control del lander* (RLLC).

Los telecomandos recorren el camino inverso seguido por las telemetrías. Los telecomandos pueden tener dos orígenes, desde el control en Tierra (EGSE o RSOC) o desde el SC. Del RSOC pasan al RMOC para finalizar en la estación de seguimiento que aprovechará la siguiente ventana de comunicaciones para enviar los telecomandos al SC y de ahí a G. Hay que tener en cuenta que pueden existir un retardo de varios días entre las comunicaciones entre Rosetta y la Tierra, por ello es importante una gestión autónoma de contingencias.

3.12.2 Telecomandos y telemetrías

La siguiente tabla detalla todos los TC y TM soportados por G. Una TM en la misma fila que un TC indica que esta TM será generada por G. en respuesta al TC. Los números indican servicio o sub-servicio (*3.12.3 Formato y estructura de las telemetrías de GIADA*).

Tabla 9. Telecomandos y telemetrías implementados en GIADA.

Telecomandos			Telemetrías
Telecommand Verification Service – 1			
		1	Acceptance Success Report
		2	Acceptance Failure Report
		7	Execution Completion Success Report
		8	Execution Completion Failure Report
Housekeeping Reporting –3			
5	Enable Housekeeping Report Generation		
6	Disable Housekeeping Report Generation		
		25	Housekeeping Report
Event Reporting – 5			
		1	Normal/Progress Report

		2	Error/Anomaly Report – Warning
		3	Error/Anomaly Report – Ground Action
		4	Error/Anomaly Report – On-board Action
Memory Management Service – 6			
2	Load Memory Absolute Addressing		
5	Dump Memory Absolute Addressing	6	Dump Memory Absolute Addressing Report
9	Check Memory Absolute Addressing	10	Check Memory Absolute Addressing Report
Time Management Service – 9			
1	Accept Time Update		
Test Service – 17			
1	Perform Connection Test	22	Connection Test Report
Context Transfer Service – 18			
1	Report Context	2	Context Report
3	Accept Context		
Science Data Transfer Service – 20			
1	Enable Science Packet Generation	3	Science Report
2	Disable Science Packet Generation		
Private Telecommand Service: Cover – 192			
1	Arm Frangibolt		
2	Disarm Frangibolt		
6	Test Frangibolt		
11	Activate Frangibolt		
16	Arm Cover		
17	Disarm Cover		
21	Open Cover		

26	Close Cover		
31	Test Heater		
Private Telecommand Service: GDS – 193			
1	Arm Laser		
2	Disarm Laser		
6	Switch Laser On/Off		
11	Set GDS On/Off		
16	Set GDS Operation Mode		
26	Set Photodiode Threshold		
46	Calibrate GDS		
Private Telecommand Service: IS – 194			
1	Set IS On/Off		
6	Set IS Operation Mode		
11	Set PZT Threshold Level		
26	Calibrate IS		
Private Telecommand Service: - MBS 195			
1	Set MBS On/Off		
6	Set MBS Operation Mode		
11	Set Time Between Measurements		
26	Heat MBS		
36	Calibrate MBS		
Private Telecommand Service: - Mode Transitions 196			
1	Safe Mode		
6	Cover Mode		
11	Normal Mode		
16	Flux Mode		
Private Telecommand Service: Co-ordinated Commands – 255			
1	Reset Telemetry Output		

Se muestran a continuación los TC y TM procesados dependiendo del modo de operación. Los valores numéricos indican servicio y sub-servicio (3.12.3 Formato y estructura de las telemetrías de GIADA).

Tabla 10. Telecomandos aceptados y telemetrías generadas respecto al modo de operación

	Telecommando	Safe	Cover	Normal	Flux
Housekeeping Reporting –3					
5	Enable Housekeeping Report Generation	+	+	+	+
6	Disable Housekeeping Report Generation	+	+	+	+
Memory Management Service – 6					
2	Load Memory Absolute Addressing	+	-	-	-
5	Dump Memory Absolute Addressing	+	-	-	-
9	Check Memory Absolute Addressing	+	-	-	-
Time Management Service – 9					
1	Accept Time Update	+	+	+	+
Test Service – 17					
1	Perform Connection Test	+	+	+	+
Context Transfer Service – 18					
1	Report Context	+	-	-	-
3	Accept Context	+	-	-	-
Science Data Transfer Service – 20					
1	Enable Science Packet Generation	-	-	+	+
3	Disable Science Packet Generation	-	-	+	+
Private Telecommand Service: Cover - 192					
1	Arm Frangibolt	-	+	-	-
2	Disarm Frangibolt	-	+	-	-
6	Test Frangibolt Device	-	+	-	-
11	Activate Frangibolt	-	+	-	-
16	Arm Cover	-	+	-	-
17	Disarm Cover	-	+	-	-
21	Open Cover	-	+	-	-

26	Close Cover	-	+	-	-
31	Test Heater	-	+	-	-
Private Telecommand Service: GDS - 193					
1	Arm Laser	-	-	+	-
2	Disarm Laser	-	-	+	-
6	Switch Laser On/Off	-	-	+	-
11	Set GDS On/Off	-	-	+	-
16	Set GDS Operation Mode	-	-	+	-
26	Set Photodiode Threshold	-	-	+	-
51	Calibrate GDS	-	-	+	-
Private Telecommand Service: IS - 194					
1	Set IS On/Off	-	-	+	-
6	Set IS Operation Mode	-	-	+	-
11	Set PZT Threshold Level	-	-	+	-
26	Calibrate IS	-	-	+	-
Private Telecommand Service: - MBS 195					
1	Set MBS On/Off	-	-	+	+
6	Set MBS Operation Mode	-	-	+	+
21	Set Time Between Measurements	-	-	+	+
26	Heat MBS	-	-	+	+
36	Calibrate MBS	-	-	+	+
Private Telecommand Service: - Mode Transitions 196					
1	Go to Safe Mode	-	+	+	+
6	Go to Cover Mode	+	-	-	-
11	Go to Normal Mode	+	-	-	-
16	Go to Flux Mode	+	-	-	-
Private Telecommand Service: Co-ordinated Commands – 255					
1	Reset Telemetry Output	+	+	+	+

“+” indica TC aceptado o TM generada
 “-” indica TC no aceptado o TM no generada

3.12.3 Formato y estructura de las telemetrías de GIADA

Los datos que recibe y que genera G. implementan el formato y el protocolo definido por ESA: **[ESA01a]**, **[ESA01b]** y **[Mor04]**. Todo el tránsito se realiza a través de conjuntos de información llamados paquetes. Se define dos tipos de paquetes, telemetrías y telecomandos. Los paquetes se agrupan a su vez en servicios, con el fin de dar una funcionalidad concreta. Los servicios ESA implementados por el instrumento G. pueden apreciarse en la siguiente ilustración:

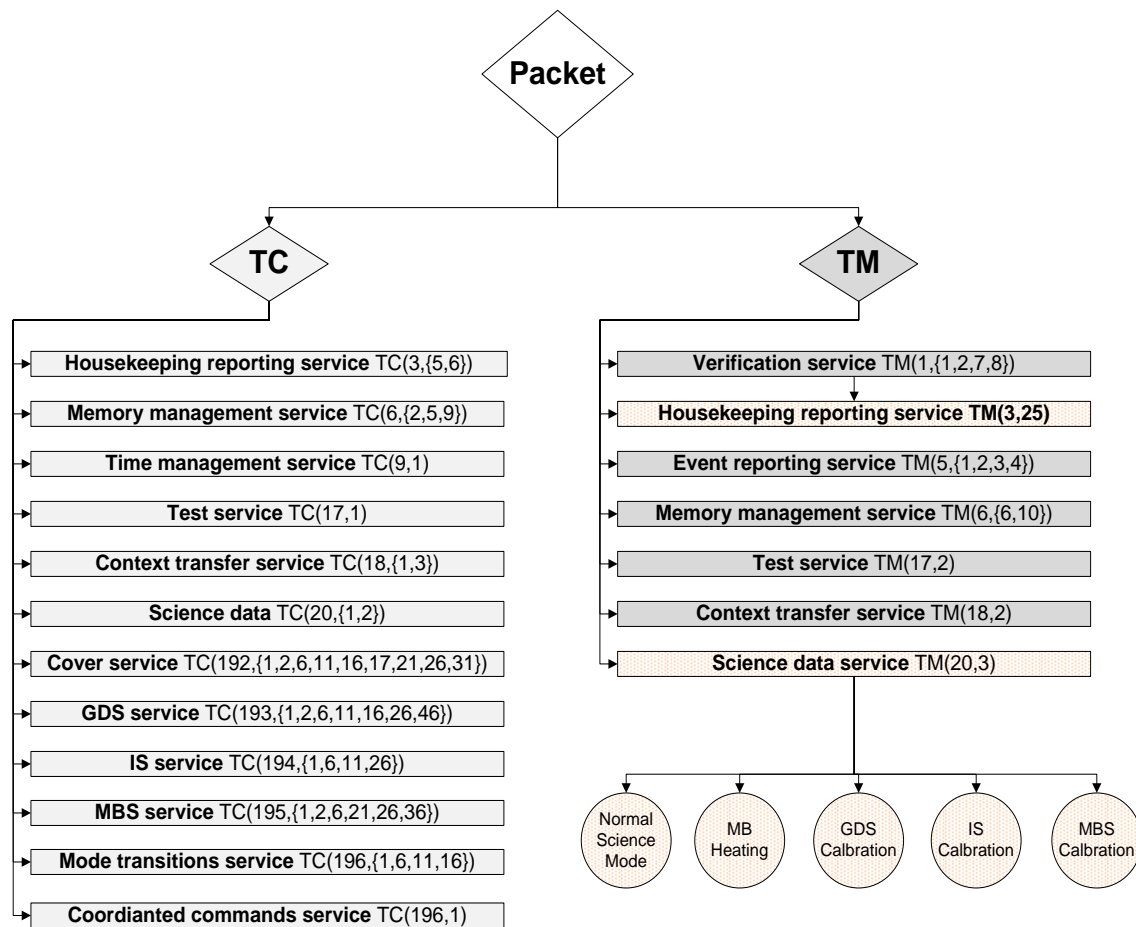


Ilustración 38. Servicios ESA implementados por GIADA.

Donde $T_x(y, \{z_1, \dots, z_n\})$, indica la codificación del paquete dentro del servicio (Tabla 9. Telecomandos y telemetrías implementados en GIADA.). $x = \{C, M\}$ Si $x = C$ indica TC, si $x = M$, TM. y indica el número del servicio, y z_1, \dots, z_n el número del sub-servicio. Esta codificación se encuentra dentro de la cabecera del paquete.

La estructura y la semántica de cada uno de los servicios se detallan en **[Mor01b]**. Los servicios relevantes para GDB son los siguientes: “Housekeeping reporting service”, “Science data service”,

en concreto las telemetrías TM(3,25) y TM(20,3). El primero proporciona la funcionalidad para conocer el estado del instrumento, el segundo nos aporta los datos científicos obtenidos.

A fin de optimizar las transacciones con el SC, las TM que almacenan eventos GDS, IS y MBS (excepto la información de calibración y de calentamiento de MBS), se agrupan en bloques de TM de ciencia. Estos bloques almacenan eventos hasta alcanzar un tamaño máximo (4096 B) o hasta sobrepasar el tiempo predefinido (60s) sin envío de ciencia al SC.

Un análisis de la información almacenada en los campos de TM(3,25) y TM(20,3) permite una reorganización de los campos en grupos o nodos y establecer una estructura jerárquica sobre dichos nodos. Así, campos comunes de nodos, se sitúan en niveles más altos en la estructura, como se muestra en la siguiente ilustración.



Ilustración 39. Estructura de las telemetrías TM(20,3) y TM(3,25).

En la estructura, cada nodo hereda los campos del nodo padre. A modo de ejemplo, la TM “GDB_GDS_ISEVENT” hereda la información de “GDB_SCIENCEHK” que a su vez la hereda de “GDB_Packet”.

Esta estructura jerárquica facilita enormemente la labor del diseño de la base de datos relaciona (BDR) de GDB que almacenará las TM(3,25) y TM(20,3) como se detalla en (6.2.1).

La semántica de cada uno de los campos se define Apéndice A así como en la propia herramienta GDB-GUI (6).

En la ilustración, los campos en negrita de las telemetrías (por ejemplo en “GDB_MBREADIN-GEVENT” los campos “ID” e “Item”) no pertenecen a la estructura original de la TM, sino que han sido añadidos artificialmente para facilitar su almacenamiento en la BD.

3.12.3.1 Precisión y marcas temporales en TM y TC

Por construcción, G. lleva un reloj interno por HW con precisión de $1/256s \cong 4ms$ que será la precisión de todas las marcas temporales de las TM. En el caso nominal, el instrumento será sincronizado cada hora con el tiempo de SC. La sincronización se realiza a nivel de segundos y fracciones de segundo en el registro HW de G. y en cualquier modo de operación.

Cada TM dispone de una marca temporal (Ilustración 39. Estructura de las telemetrías TM(20,3) y TM(3,25).) establecida por G., el “*Packet Time*”. Esta marca indica cuándo fue construida la TM, de acuerdo al reloj interno del instrumento. En casos extremos, como en una generación continua de eventos, es posible que dos TM tengan el mismo *Packet Time*. En tal caso, hay que recurrir al campo “SSC” que es un contador creciente por tipo de servicio. Así pues, “*SCET Time*” + “SSC” constituye una combinación única para cada TM. El contador es puesto a 0 en cada inicialización de G. y por lo tanto debe descartarse como identificador unívoco. Por facilidad, GDB utilizará el tiempo de inserción del paquete en el propio sistema (campo “ID”) como llave primaria de las relaciones.

Los telecomandos no llevan asociado ningún tipo de marca temporal en su estructura, pero cuando son recibidos o rechazados normalmente (hay TC como el de sincronización temporal que no generan telemetrías) se envía una TM del servicio 1, por lo que se puede aproximar el tiempo original en el que se envió dicho TC.

Son las marcas de tiempo la información base que permite establecer el orden cronológico de sucesos que es crítico a la hora de interpretación científica de los datos recogidos. Las partículas expulsadas del cometa han llegado a G. en intervalos menores que la resolución temporal (4ms). Esto ha originado marcas temporales duplicadas en las TM que complica en extremo la interpretación científica. Para resolver esta ambigüedad, es necesario usar otras fuentes de información externas (otros instrumentos de Rosetta) o internas (contexto de la detección). Este es un problema abierto y en el GDB puede ayudar a resolver.

3.13 Prestaciones del software de GIADA

Se detallan a continuación las prestaciones principales del SW de G. Son estas prestaciones las que definen los límites de detección del instrumento, y por tanto son vitales a la hora de una correcta interpretación científica de las medidas realizadas.

Tabla 11. Prestaciones generales del software de GIADA versión 2.3.

Característica	Valor
Tamaño del ejecutable	36142 bytes
Máximo número de eventos procesados por segundo cumpliendo los requerimientos de tiempo	40 (20 GDS+20 IS)
Número de parches almacenables en el VD	64
Máxima área parcheable almacenable en el VD	73728 bytes

Las siguientes estimaciones de tiempos se han realizado con analizador lógico (HP 16500c) sobre GIADA SW 2.3. Hay que indicar que las interrupciones son anidables (*3.3 Interfaz software-hardware*).

Tabla 12. Estimación de tiempos de la interrupción de GDS.

Tiempo desde el levantamiento de la IRQ de GDS hasta el reset . Evento individual (ms)	Tiempo desde el levantamiento de la IRQ de GDS hasta el reset. Evento doble (ms)
1,62	2,92

Tabla 13. Estimación de tiempos de la interrupción de IS.

Tiempo desde el levantamiento de la IRQ de IS hasta el reset "No autocorrección" (ms)	Tiempo desde el levantamiento de la IRQ de IS hasta el reset "Gain autocorrección" (ms)
6.6	7.74

Tabla 14. Estimación de tiempos de la interrupción de MBS.

Tiempo desde la subida de la línea de IRQ MBS hasta el reset de MBS (ms)
1.46

Tabla 15. Estimación de tiempos en el envío de TM.

Máxima TM enviada: 4124 bytes Tiempo (ms)	Mínima TM enviada: 18 bytes Tiempo (ms)
36.56	1.172

Tabla 16. Estimación de tiempos del procesamiento de la cola de eventos.

Evento de GDS (ms)	Evento de IS (ms)	Evento de lectura de MBS (ms)	Evento de calentamiento de MBS (ms)
19.62	19.94	0.98	0.97

3.14 Resumen y conclusiones

Se han descrito en este capítulo los aspectos más relevantes diseño, desarrollo, implementación y mantenimiento del SW de GIADA incluyendo apartados dedicados al proceso de detección de partículas (a nivel de GDS, IS y MBS), estructura y formato de la información generada y las prestaciones obtenidas.

El proceso de detección y la estructura de datos serán piezas fundamentales a la hora de representar el conocimiento, como se verá en el capítulo 7.

4 Antecedentes de modelos de bases de datos

En este capítulo se exponen los modelos de bases de datos precursores del modelo teórico utilizado en GDB que se describirá en el capítulo 5. Por un lado, el “*modelo relacional*” se extiende hasta el “*modelo relacional difuso*”. Por otro lado, desde el “*modelo lógico*” se llegará al “*modelo lógico difuso*”. Para que, finalmente, ambas ramas cristalicen en el “*modelo integrado*”, que constituye el modelo teórico inicial de GDB.

4.1 Modelo relacional de bases de datos

Existen varias formas (modelos) para almacenar los datos y relacionarlos entre sí. El *Modelo Relacional* (MR) [Cod70] es el más utilizado y habitualmente se implementa mediante una *Base de Datos* (BD). De hecho, es el modelo que se utiliza para albergar los datos del modelo en GDB, como se verá más adelante en el capítulo 5.

Una BD es un conjunto de información almacenada que es gestionada por un programa de computador llamado *Sistema Gestor de Base de Datos* (SGBD). En este modelo, los datos y las relaciones existentes entre ellos, se almacenan en forma de tabla (ver 4.1.1). Además, todas las operaciones realizadas sobre tablas generan como resultado nuevas tablas. Un *SGBD Relacional* (polvo/gas) es un SGBD que gestiona una BD que sigue este modelo. Un estudio más profundo del MR de bases de datos, puede encontrarse en [Cod82], [Dat85] y [Ull82]. Uno de los SGBDR comerciales más conocido es Oracle© [Urm98], el cual es utilizado por GDB como herramienta de implementación del modelo.

En el MR, se puede distinguir cuatro partes bien diferenciadas: *estructura de datos*, *integridad de datos*, *definición de datos* y *manipulación de datos*. En los siguientes apartados se expondrán los rasgos más relevantes de cada una de las ellas.

4.1.1 Estructura de datos relacional

La *estructura de los datos relacional* se caracteriza a través del concepto de *tabla* o **relación**. La definición de relación se basa a su vez en los siguientes conceptos:

- ❖ **Definición 4.1. Atributo.** Es cada una de las características de un objeto que serán almacenadas.
- ❖ **Definición 4.2. Dominio.** Es el conjunto de todos los valores posibles de un atributo.
- ❖ **Definición 4.3. Tupla.** Es un conjunto de valores concretos de atributos de un determinado objeto.

Más formalmente, un *dominio* es un conjunto, normalmente finito, de valores. El producto cartesiano $D_1 \times D_2 \times \dots \times D_n$ de un conjunto dominios D_1, D_2, \dots, D_n , es el conjunto de todas las tuplas (x_1, x_2, \dots, x_n) tales que $\forall i, x_i \in D_i \mid i = \{1, 2, \dots, n\}$.

Finalmente, una **relación** es un subconjunto del producto cartesiano de uno o más dominios.

Algunas propiedades de una *relación* que se utilizarán en esta memoria son la *cardinalidad* y la *ariedad*:

- ❖ **Definición 4.4. Cardinalidad.** Aplicado a una relación, indica número de tuplas que contiene.

- ❖ **Definición 4.5. Grado o ariedad.** Referido a una relación, alude al número de dominios de su definición.

4.1.2 Integridad de datos

Para analizar la integridad de los datos, es necesario definir previamente los conceptos de *esquema* e *instancia* de una relación y de una BD.

- ❖ **Definición 4.6. Esquema e instancia de relación.** Una relación R , compuesta por el conjunto de atributos A_1, A_2, \dots, A_n , define un *esquema de relación* (ER) y se nota por $R(A_1, A_2, \dots, A_n)$. Una relación R_i cuyas tuplas almacenan valores concretos de A_1, A_2, \dots, A_n se dice que es una *instancia* de R .

No todas las instancias de un esquema son válidas desde el punto de vista semántico. Es por ello que se introduce un conjunto de condiciones sobre datos, que recibe el nombre de *restricciones de integridad* (RI). Estas restricciones se asocian a cada ER.

- ❖ **Definición 4.7. Esquema e instancia de BD.** Un esquema de BD, se define como el conjunto de *esquemas de relación* junto a sus *restricciones de integridad* asociadas. Se define una *instancia de una BD* como un conjunto de *instancias de relación* (una por cada ER que intervenga en el esquema de la BD).

Un estado de una BD será válido, cuando todas las *instancias de relación* que contiene verifican todas las *restricciones de integridad* aplicables.

Dos de las *restricciones de integridad* más utilizadas en un ER son la *regla de identidad* y la *regla de integridad referencial*. Para definir las, es necesario definir primero los conceptos de *llave primaria* y *llave externa*.

- ❖ **Definición 4.8. Llave primaria.** Es el conjunto de atributos que se usarán para identificar unívocamente cada tupla de una relación. Este conjunto debe ser mínimo, es decir, que no exista un subconjunto de él que sea a su vez *llave primaria*. Puede haber más de una *llave primaria* para una relación, en tal caso se identificará una sola como tal, pasando las otras a llamarse *llaves candidatas*.
- ❖ **Definición 4.9. Llave externa.** Es el conjunto de atributos de una relación que son *llave primaria* o *candidata* en otra relación distinta.

Se definen la *regla de integridad de identidad* y la *regla de integridad referencial* como sigue:

- ❖ **Definición 4.10. Regla de integridad de identidad.** Ningún componente de la *llave primaria* de una relación puede aceptar el valor nulo. Donde nulo significa que la información no se encuentra por alguna razón.
- ❖ **Definición 4.11. Regla de integridad referencial.** La BD no puede contener valores para la llave externa que no hallen correspondencia con los adoptados por la llave primaria de la relación a la que hacen referencia.

4.1.3 Definición de datos

Un lenguaje de definición de datos, permiten la creación, modificación y eliminación de objetos de una BDR. Uno de los lenguajes más conocidos es el SQL [Cha74] y [Cha76] (en su parte de definición de objetos) que ha sido ampliamente descrito en la bibliografía ([Bow97] y [Urm98]).

4.1.4 Manipulación de datos

El lenguaje de manipulación de datos en una BDR toma una o dos relaciones como argumento y devuelven una nueva relación. Desde el punto de vista teórico, el modelo relacional propone dos tipos diferentes de lenguaje: *álgebra relacional* y *cálculo relacional*. El *álgebra relacional* proporciona un conjunto de operadores mediante los cuales se especifica la operación a realizar sobre las tablas, para así obtener la respuesta deseada. Mientras que en el *cálculo relacional*, se proporciona una sintaxis para expresar la relación resultante. En [Ull82] se indica el proceso para convertir una consulta de *álgebra relacional* en *cálculo relacional* y viceversa.

Se muestran a continuación algunos de los operadores del *álgebra relacional* que serán utilizados en esta memoria.

- ❖ **Definición 4.12. Producto cartesiano.** Sean R y S dos relaciones, se define el producto cartesiano $R \times S$ como una nueva relación Q que contiene todas las posibles combinaciones de pares de tuplas, una de cada relación.
- ❖ **Definición 4.13. Selección.** Sea R una relación, F una fórmula o condición que incluye al menos un atributo de R y que además puede incluir constantes de cualquier dominio, comparadores aritméticos ($>$, \geq , $<$, \leq , $=$ y \neq) y operadores lógicos (NOT , AND y OR). Se define el *operador selección* σ_F aplicados sobre la relación R como una nueva relación Q con los mismos atributos que R pero sólo con las tuplas que verifican F . Se notará este operador como $\sigma_F(R)$.
- ❖ **Definición 4.14. Proyección.** Sea R una relación, y sea $P = \{p_1, p_2, \dots, p_n\}$ un conjunto de atributos de R . Se define el *operador proyección* Π aplicado a R como una nueva relación Q que incluye sólo los atributos especificados en P , y de notará por $Q = \Pi_{p_1, p_2, \dots, p_n}(R)$.
- ❖ **Definición 4.15. Reunión.** Sean R y S dos relaciones, sean r y s dos atributos de respectivamente de R y S . Sea un operador aritmético θ , y sea $r\theta s$ una formula o condición. Se define el *operador reunión* $\otimes_{r\theta s}$ aplicado sobre las relaciones R y S , como una nueva relación Q compuesta por la concatenación de las tuplas de R y S que verifican $r\theta s$. Se notará por $Q = R \otimes_{r\theta s} S$. Si el operador θ es el operador de igualdad, se denomina *operador equi-reunión*.

4.2 Modelo relacional difuso de bases de datos

A la hora de abordar un problema, la información disponible no es siempre perfecta o exacta, sino que puede venir dada con cierto grado de imprecisión (sensor muy frío, dato de buena calidad...), y por tanto, se hace necesario incorporar este conocimiento para realizar un razonamiento adecuado. A este tipo de información imprecisa se le denomina, en términos generales, difusa.

Centrándonos en el instrumento G., la información difusa está ligada a su comportamiento y al estudio de los datos generados. Los sensores de medición que incorpora, han degradado sus prestaciones (pero no en exceso, ver 2.2.4) de forma no lineal a lo largo de la misión y esta información es fundamental para interpretar correctamente las medidas que se están realizando. Así mismo, la propia metodología de medida (como en cualquier instrumento científico) origina que no siempre se obtengan datos precisos, como ocurre con los mapas de sensibilidad del GDS e IS (2.2.3.1.1 *GIADA-1: GDS+IS*).

Respecto al estudio de los datos, la información imprecisa también aparece en G.: no siempre los parámetros obtenidos de un evento (p. ej. las marca de tiempo) se encuentran dentro de los límites marcados de forma estricta. Habitualmente los datos que se encuentran “cerca” de estos límites tienen un valor científico significativo.

A modo de ejemplo, se pueden consultar los primeros resultados científicos obtenidos (2.2.6 *Primeros resultados científicos*), donde los términos “poca densidad”, “oscuros” o “pequeños” aplicado a propiedades de las partículas, son habituales.

Se ve por tanto, la necesidad de incluir este tipo de información difusa en el sistema de razonamiento para poder entender el fenómeno bajo estudio, que en el caso de G. es el estudio del polvo cometario.

Existen en la literatura varios modelos de bases de datos capaces de trabajar con información precisa e imprecisa, cuyo eje común es utilizar el MR para la representación y manipulación de los datos. Para realizar este trabajo, es necesario modificar las estructuras de las relaciones y, con esto, las operaciones, además de considerar una multitud de excepciones que no ocurren en un modelo con información precisa ([Gal99]). Estos modelos se agrupan dentro del denominado *modelo relacional difuso*, y utilizan como herramienta fundamental para modelar la imprecisión la *teoría de conjuntos difusos* (TCD).

En esta sección se revisará la TCD, se continúa con un repaso a los distintos modelos considerados dentro del *modelo relacional difuso*, finalizando con el modelo GEFRED y su implementación FIRST.

4.2.1 Teoría de conjuntos difusos

La TCD fue inicialmente propuesta por L.A. Zadeh: [Zad65], [Zad75a], [Zad75b], [Zad76] y [Zad78]. Y más tarde, ampliamente difundida: [Dub80], [Tri80] y [Yag82].

De detallan a continuación los aspectos más relevantes de la TCD: definición y notación, operaciones entre *conjuntos difusos*, *números difusos*, *principio de extensión*, *etiquetas y variables lingüísticas* y *teoría de posibilidad*.

Los *conjuntos difusos* son una generalización de los conjuntos clásicos, puesto que no sólo recogen el sentido tradicional de conjunto, sino que también permiten la descripción de nociones “vagas” o “imprecisas”. Estos conjuntos relajan el concepto clásico de pertenencia de un elemento a un conjunto, asignando un número para medir dicha pertenencia. Así, un concepto cualitativo (difuso) puede ser manejado de una forma cuantitativa.

Se define formalmente un conjunto difuso (CD) como:

- ❖ **Definición 4.16. Conjunto difuso.** Se define el *conjunto difuso* \tilde{A} sobre un universo de discurso (UD) Ω como el conjunto de pares:

$$\tilde{A} = \{(\mu_{\tilde{A}}(x), x) \mid x \in \Omega, \mu_{\tilde{A}}(x) \in [0,1] \subset \mathbb{R}\} \quad \text{Eq. (4.1)}$$

Donde $\mu_{\tilde{A}}(x)$ se denomina *grado de pertenencia* del elemento x al conjunto difuso \tilde{A} , e indicando el símbolo $\tilde{}$ el concepto de difuso.

Un grado de pertenencia cero indica que el elemento no pertenece en absoluto al CD, y un grado uno indica a su vez una pertenencia total.

Habitualmente, para caracterizar un CD no se proporciona una lista exhaustiva de pares, sino la definición de la función $\mu_{\tilde{A}}(x)$, que pasa a llamarse *función característica* o *función de pertenencia* del CD. Un tipo especial de estas funciones, las llamadas “crisp” (traducido como concreto o preciso), que sólo toman valores en el conjunto $\{0,1\}$, representan a los conjuntos en el sentido clásico. Por tanto, dada su definición, los conjuntos difusos engloban a los conjuntos clásicos.

Se define a continuación una serie de definiciones y propiedades útiles para el tratamiento de conjuntos difusos.

- ❖ **Definición 4.17. Igualdad de conjuntos difusos.** Dos conjuntos difusos \tilde{A} y \tilde{B} sobre el mismo UD Ω se dice *iguales* si cumplen:

$$\tilde{A} = \tilde{B} \Leftrightarrow \forall x \in \Omega, \mu_{\tilde{A}}(x) = \mu_{\tilde{B}}(x) \quad \text{Eq. (4.2)}$$

- ❖ **Definición 4.18. Soporte de un conjunto difuso.** El *soporte de un CD* \tilde{A} definido sobre el UD Ω es un subconjunto de dicho universo que satisface:

$$\text{soporte}(\tilde{A}) = \{x \in \Omega, \mu_{\tilde{A}}(x) > 0\} \quad \text{Eq. (4.3)}$$

- ❖ **Definición 4.19. Alfa-corte de un conjunto difuso.** Dado un valor α , se define como *alfa corte de un CD* \tilde{A} sobre el UD Ω y se denotada por A_{α} , como el subconjunto no difuso (clásico) en Ω que satisface:

$$A_{\alpha} = \{x \mid x \in \Omega, \mu_{\tilde{A}}(x) \geq \alpha, \alpha \in [0,1] \subset \mathbb{R}\} \quad \text{Eq. (4.4)}$$

Visualmente:

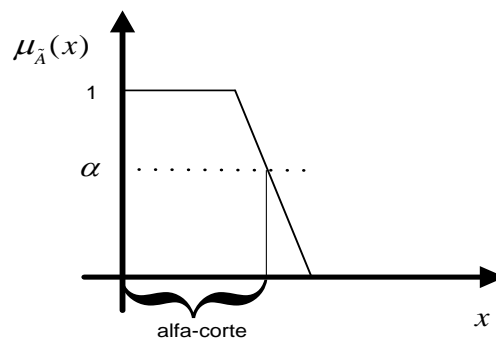


Ilustración 40. Representación gráfica de un alfa-corte.

- ❖ **Definición 4.20. Teorema de representación de un conjunto difuso.** Todo CD \tilde{A} puede ser obtenido a partir de la unión de sus alfa-cortes:

$$\tilde{A} = \bigcup_{\alpha \in [0,1]} A_\alpha \quad \text{Eq. (4.5)}$$

- ❖ **Definición 4.21. Conjunto difuso convexo.** Un CD \tilde{A} es *convexo* si todos sus alfa-cortes son convexos, esto es:

$$\forall x, y \in \Omega, \forall \lambda \in [0,1] \subset \mathbb{R} : \mu_{\tilde{A}}(\lambda \circ x + (1-\lambda) \circ y) \geq \min(\mu_{\tilde{A}}(x), \mu_{\tilde{A}}(y)) \quad \text{Eq. (4.6)}$$

Siendo \circ el operador multiplicación.

En las siguientes figuras se presentan un CD *convexo* y otro *no convexo*.

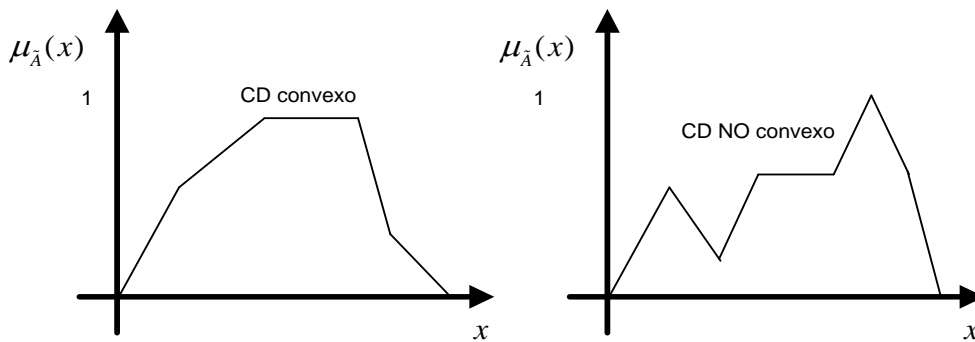


Ilustración 41. Conjuntos difusos convexos y no convexos.

- ❖ **Definición 4.22. Núcleo de un conjunto difuso.** El núcleo de un CD \tilde{A} , definido sobre el UD Ω , es un subconjunto que verifica:

$$\text{núcleo}(A) = \{x \in \Omega, \mu_{\tilde{A}}(x) = 1\} \quad \text{Eq. (4.7)}$$

- ❖ **Definición 4.23. Altura de un conjunto difuso.** Se define la altura de un CD \tilde{A} definido sobre el UD Ω como el siguiente valor:

$$\text{altura}(\tilde{A}) = \max \{ \mu_{\tilde{A}}(x) \mid x \in \Omega \} \quad \text{Eq. (4.8)}$$

- ❖ **Definición 4.24. Conjunto difuso normalizado.** Un CD \tilde{A} se dice normalizado, si cumple:

$$\exists x \in \Omega, \mu_{\tilde{A}}(x) = \text{altura}(\tilde{A}) = 1 \quad \text{Eq. (4.9)}$$

4.2.1.1 Operaciones con conjuntos difusos

Dado que la TCD generaliza la teoría de conjuntos clásica, ésta también debe extender las operaciones clásicas sobre conjuntos: unión, intersección y complemento. Para poder abordarlas, es necesario definir los conceptos de *T-norma* y *T-conorma*. En [Pet96] se puede encontrar una descripción detallada de estas y otras operaciones sobre conjuntos difusos.

- ❖ **Definición 4.25. Norma triangular o T-norma.** Es toda función $T : [0,1] \times [0,1] \rightarrow [0,1]$ que cumple las siguientes propiedades:

- a) Conmutativa: $xTy = yTx$
- b) Asociativa: $xT(yTz) = (xTy)Tz$
- c) Monotonicidad: Si $x \leq y, y w \leq z$, entonces $xTw \leq yTz$
- d) Condiciones frontera: $xT0 = 0, xT1 = x$

❖ **Definición 4.26. Conorma triangular o T-conorma.** Esa toda función $S : [0,1] \times [0,1] \rightarrow [0,1]$ que cumple las siguientes propiedades:

- a) Conmutativa: $xSy = ySx$
- b) Asociativa: $xS(ySz) = (xSy)Sz$
- c) Monotonicidad: Si $x \leq y, y w \leq z$, entonces $xSw \leq ySz$
- d) Condiciones frontera: $xS0 = x, xS1 = 1$

Las *T-normas* se utilizan para modelar las operaciones de intersección de conjuntos difusos y análogamente las T-conorma para las operaciones de unión. Existe una amplia familia de T-normas y T-conorma usada en uniones e intersecciones de conjuntos difusos ([Dub80], [Pet96] y [Yag80]).

4.2.1.1.1 Unión de conjuntos difusos

Se define la operación de unión sobre conjuntos difusos como sigue:

❖ **Definición 4.27. Unión de conjuntos difusos.** El CD $\tilde{A} \cup \tilde{B}$ resultante de la operación unión sobre conjuntos difusos \tilde{A} y \tilde{B} , definidos sobre el mismo UD Ω , queda definido por:

$$\mu_{\tilde{A} \cup \tilde{B}}(x) = S(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)), x \in \Omega \quad \text{Eq. (4.10)}$$

Donde S es una *T-conorma*.

4.2.1.1.2 Intersección de conjuntos difusos

Se define la operación de intersección sobre conjuntos difusos como sigue:

❖ **Definición 4.28. Intersección de conjuntos difusos.** El CD $\tilde{A} \cap \tilde{B}$ resultante de la operación intersección sobre conjuntos difusos \tilde{A} y \tilde{B} , definidos sobre el mismo UD Ω , queda expresado por:

$$\mu_{\tilde{A} \cap \tilde{B}}(x) = T(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)), x \in \Omega \quad \text{Eq. (4.11)}$$

Donde T es una *T-norma*.

4.2.1.1.3 Complemento o negación de conjuntos difusos

El concepto de complemento se puede construir a partir del concepto de *negación fuerte* definido en [Tri79]:

❖ **Definición 4.29. Negación fuerte.** Una función $C : [0,1] \rightarrow [0,1]$ es una *negación fuerte* si satisface las siguientes condiciones:

- a) $C(0) = 1$
- b) $C(C(a)) = a$
- c) C es estrictamente decreciente
- d) C es continua

Una de las funciones más utilizadas como *negación fuerte* es la proporcionada en [Zad65]: $C(x) = 1 - x$, por tanto, dado un CD \tilde{A} definido sobre el UD Ω , el CD definido por su complemento $\neg\tilde{A}$ es:

$$\mu_{\neg\tilde{A}}(x) = 1 - \mu_{\tilde{A}}(x), x \in \Omega \quad \text{Eq. (4.12)}$$

4.2.1.2 Números difusos

El concepto de número difuso (ND) fue definido inicialmente en [Zad75a] a fin de analizar y manipular valores numéricos aproximados: “casi 8”, “próximo a 7”, etcétera. El concepto de ND ha sido extendido en [Dub85].

❖ **Definición 4.30. Número difuso.** Un CD \tilde{A} definido sobre el UD Ω , es un ND si cumple las siguientes condiciones:

- a) \tilde{A} es convexo.
- b) $\mu_{\neg\tilde{A}}(x)$ es semi-continuo superiormente.
- c) El soporte de \tilde{A} es un conjunto acotado.

Se considera que $\mu_{\neg\tilde{A}}(x)$ es semi-continua superiormente en un punto x_0 si los valores de la función en puntos cercanos a x_0 son próximos a $\mu_{\neg\tilde{A}}(x_0)$ o menores que $\mu_{\neg\tilde{A}}(x_0)$.

La forma general de la *función de pertenencia* de un ND es:

$$\mu_{\tilde{A}}(x) = \begin{cases} r_{\tilde{A}}(x) & \text{si } x \in [\alpha, \beta) \\ h & \text{si } x \in [\beta, \gamma] \\ s_{\tilde{A}}(x) & \text{si } x \in (\gamma, \delta] \\ 0 & \text{en otro caso} \end{cases} \quad \text{Eq. (4.13)}$$

Donde $r_{\tilde{A}}, s_{\tilde{A}} : \Omega \rightarrow [0, 1]$, cumpliendo $r_{\tilde{A}}(\beta) = h = s_{\tilde{A}}(\gamma)$ y con $h \in (0, 1]$ y $\alpha, \beta, \delta, \gamma \in \Omega$. Además $r_{\tilde{A}}$ es no decreciente y $s_{\tilde{A}}$ es no creciente.

Al número h se le denomina *altura* del ND, al intervalo $[\beta, \gamma]$ *intervalo modal del ND*, y a los números $\beta - \alpha$ y $\delta - \gamma$ *holguras izquierda y derecha* del ND, respectivamente.

Un ND clásico es el ND trapezoidal.

❖ **Definición 4.31. Número difuso trapezoidal.** Un ND trapezoidal es un ND donde $r_{\tilde{A}}$ y $s_{\tilde{A}}$ son funciones lineales. En tal caso, la *función de pertenencia* toma la forma:

$$\mu_{\tilde{A}}(x) = \begin{cases} h + \frac{(x - \beta)h}{\beta - \alpha} & \text{si } x \in [\alpha, \beta) \\ h & \text{si } x \in [\beta, \gamma] \\ h - \frac{(x - \gamma)h}{\delta - \gamma} & \text{si } x \in (\gamma, \delta] \\ 0 & \text{en otro caso} \end{cases} \quad \text{Eq. (4.14)}$$

Es habitual normalizar el ND, por lo que $h = 1$. En tal caso la función de pertenencia de un ND trapezoidal se puede caracterizar únicamente con α, β, δ y γ .

En la siguiente ilustración se muestran dos ND, uno genérico y otro trapezoidal normalizado.

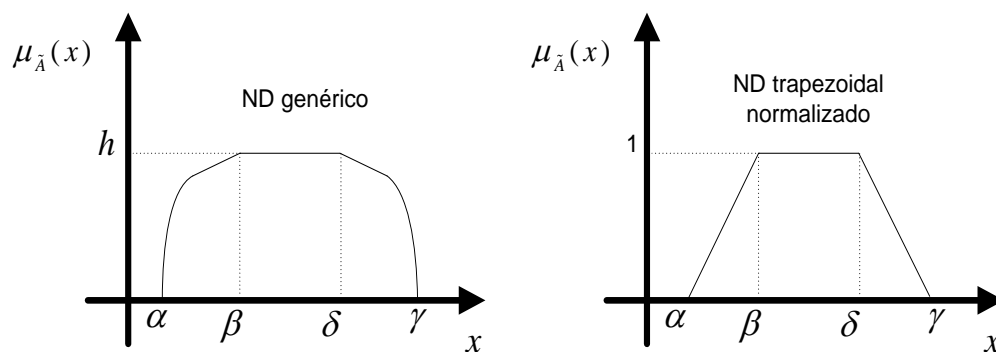


Ilustración 42. Números difusos.

4.2.1.3 Principio de extensión

Este principio fue propuesto inicialmente en [\[Zad75a\]](#) y proporciona una técnica general para ampliar conceptos matemáticos clásicos (no difusos) para que puedan trabajar con números difusos:

❖ **Definición 4.32. Principio de extensión.** Sea $\Omega = \Omega_1 \times \Omega_2 \times \dots \times \Omega_n$ el producto cartesiano de n UD, sean n conjuntos difusos $\tilde{A}_1, \tilde{A}_2, \dots, \tilde{A}_n$ definidos sobre $\Omega_1, \Omega_2, \dots, \Omega_n$ respectivamente y $f : \Omega \rightarrow \Omega^*$ siendo Ω^* un UD. El CD \tilde{B} definido sobre Ω^* y asociado a f se define como:

$$\tilde{B} = \int_{\Omega} \mu_{\tilde{B}}(y) \quad \text{Eq. (4.15)}$$

Donde $y = f(x_1, x_2, \dots, x_n) \mid y \in \Omega^* \wedge \{x_1, x_2, \dots, x_n\} \in \Omega$

Siendo la función de pertenencia:

$$\mu_{\tilde{B}}(y) = \begin{cases} \max \left\{ \min_{(x_1, x_2, \dots, x_n) \in f^{-1}(y)} \left\{ \mu_{\tilde{A}_1}(x_1), \dots, \mu_{\tilde{A}_n}(x_n) \right\} \right\}, & f^{-1}(y) \neq \emptyset \\ 0 & \text{, en otro caso} \end{cases}$$

Se pueden aplicar este principio ([Gal99]) para extender la aritmética sobre número naturales a aritmética difusa.

4.2.1.4 Etiquetas y variables lingüísticas

Una etiqueta lingüística (EL) es aquella palabra, en lenguaje natural, cuyos valores definen un CD. Es decir, un EL se compone de términos lingüísticos definidos como conjuntos difusos sobre un dominio subyacente.

Las etiquetas lingüísticas son algo común en la vida cotidiana: “joven”, “frío”, “alto”, etcétera. La definición intuitiva de estas etiquetas varía del individuo que las utilice, así como el contexto en el que se apliquen. El concepto de “frío” es diferente para un esquimal que para un yanomami y no es lo mismo el “frío” de un helado al “frío” del espacio profundo.

A modo ejemplo, consideremos la etiqueta lingüística “joven” y una posible definición del CD al que identifica:

$$\text{Joven} = \{1/0, \dots, 1/20, 1/25, 0.9/26, 0.8/27, 0.7/28, 0.6/29, 0.5/30, \dots, 0.1/34\}$$

En este caso el universo del discurso es la edad en años enteros. Gráficamente este CD puede representarse como:

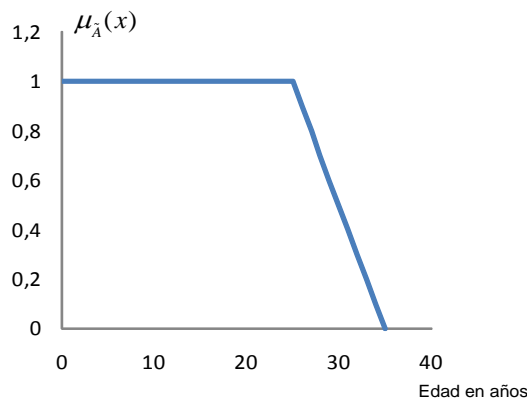


Ilustración 43. Etiqueta lingüística para “joven”.

Zadeh ([Zad75a] y [Zad75b]) define una variable lingüística como sigue:

- ❖ **Definición 4.33. Variable lingüística.** Una variable lingüística es un conjunto de 5 elementos: $\langle N, U, T(N), G, M \rangle$, donde:
- N es el nombre de la variable y U dominio subyacente.
 - $T(N)$ es el conjunto de términos o etiquetas que puede tomar N .
 - G es una gramática para generar las etiquetas de $T(N)$.
 - M es una regla semántica que asocia cada elemento de $T(N)$ con un conjunto difuso en U de entre todos los posibles: $M: T(N) \rightarrow F(U)$.

Retomado el ejemplo de variable lingüística “joven”, la definición de Zadeh para esta variable podría representarse como se muestra en la siguiente ilustración:

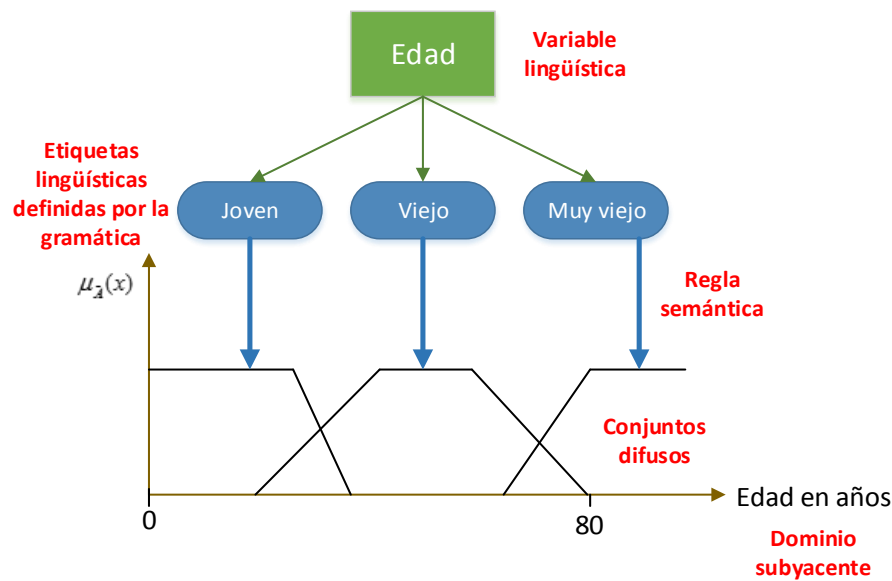


Ilustración 44. Etiqueta lingüística para “joven” en la definición de Zadeh.

La relación entre una variable lingüística y una EL, se encuentra al admitir como valores de la variable una EL.

4.2.1.5 Teoría de la posibilidad

Esta teoría fue definida inicialmente en [Zad78] y establece las relaciones entre conjuntos difusos y variables lingüísticas:

❖ **Definición 4.34. Distribución de posibilidad de un conjunto difuso.** Sea el CD \tilde{A} definido en el UD Ω con función de pertenencia $\mu_{\tilde{A}}(x)$ y una variable X definida en Ω de la cual desconocemos su valor. La proposición “ X es \tilde{A} ” define una *distribución de posibilidad (DP)*, de tal forma que la posibilidad de que $X = u$ es $\mu_{\tilde{A}}(u), \forall u \in \Omega$.

Es decir, se puede evaluar la posibilidad de que una determinada variable X sea o pertenezca a un conjunto difuso \tilde{A} como el grado de pertenencia de los posibles valores de X en \tilde{A} .

4.2.2 Modelo relacional difuso

De forma sencilla, el modelo relacional difuso o modelo de base de datos relacional difusa (BDRD), consiste en añadir un *grado* (entre [0,1]) a cada tupla de una relación. Este *grado* mide hasta qué punto la tupla pertenece a dicha relación. La semántica del *grado* puede variar entre los diversos modelos de BDRD, significando la *dependencia entre atributos* o la *importancia de la tupla en la relación*.

Una de las principales desventajas del uso de los modelos de BDRD consiste en que una tupla asume su *grado* difuso de forma global, sin que se pueda determinar la aportación particular de cada uno de los atributos que la constituyen.

Más formalmente, se define una BDRD como sigue:

❖ **Definición 4.35. Base de datos relacional difusa.** Se define una BDRD Z como el conjunto $Z = \{R_1, R_2, \dots, R_m\}$, donde $R_j, \forall j = 1, \dots, m$ es una relación denominada difusa y caracterizada por la función de pertenencia:

$$\mu_{R_j} : U_1 \times U_2 \times \dots \times U_n \rightarrow [0,1] \quad \text{con } j = 1, \dots, m \quad \text{Eq. (4.16)}$$

Siendo U_i el UD del i -ésimo atributo de la relación y \times el producto cartesiano.

Existen dos aproximaciones principales para implementar una BDRD:

- a) Modelos de unificación por relaciones de similitud. Siendo uno de representantes el modelo de Bucles y Petri (4.2.2.1).
- b) Modelos posibilísticos. Representados por las propuestas de: Prade and Testemale (4.2.2.2), Umano y Fukami (4.2.2.3), y Zemankova y Kandel(4.2.2.4).

A parte de estas dos líneas, se encuentra el modelo GEFRED (4.2.2.5), diseñado con el objetivo de sintetizar las características principales de los modelos a) y b).

A continuación se repasan cada uno de los modelos mencionados.

4.2.2.1 Modelo de Bucles y Petry

Este modelo descrito en [Buc82a], [Buc82b] y [Buc84] se basa a su vez en el modelo basado en *relaciones de similitud* inicialmente descrito en [Zad71].

❖ **Definición 4.36. Relación de similitud.** Se define la función *relación de similitud* como aquella que mide la relación de parecido entre dos valores de atributo definidos en el mismo dominio D :

$$s_r : D \times D \rightarrow [0,1] \quad \text{Eq. (4.17)}$$

Si $s_r(d_i, d_j) = 0 \mid d_i, d_j \in D$ se dice que los valores son totalmente diferentes y cuando $s_r(d_i, d_j) = 1 \mid d_i, d_j \in D$ se dice que son iguales o indistinguibles. Así mismo, dado un valor umbral γ , dos valores d_i, d_j que verifiquen $\forall d_i, d_j \in D \mid s_r(d_i, d_j) \geq \gamma$, se dicen indistinguibles y podrán ser considerados idénticos.

Este modelo define una relación difusa como sigue:

❖ **Definición 4.37. Relación difusa en el modelo de Bucles y Petry.** Se define relación difusa \tilde{R} como:

$$\tilde{R} \subseteq P(D_1) \times \dots \times P(D_n) \quad \text{Eq. (4.18)}$$

Donde $D_i, \forall i = 1, \dots, n$ es un dominio sobre el que hay definida una *relación de similitud* y $P(D_i)$ es el conjunto de partes del dominio (conjunto de todos los posibles subconjuntos).

En este modelo es posible representar los siguientes tipos de valores de atributo (los tipos son disyuntivos entre sí):

- c) Conjunto finito de escalares.
- d) Conjunto finito de números.
- e) Conjunto finito de números difusos.

4.2.2.2 Modelo de Prade y Testemale

Este modelo fue propuesto y desarrollado en [Pra84a], [Pra84b], [Pra87a] y [Pra87b]. El modelo permite incorporar datos *incompletos* o *inciertos* en el ámbito de la *Teoría de la Posibilidad* (4.2.1.5 Teoría de la posibilidad).

❖ **Definición 4.38. Distribución de posibilidad de Prade - Testemale.** Dado un atributo A cuyo dominio es D , el conocimiento disponible acerca del valor que toma A para un objeto x puede ser representado por la siguiente distribución de posibilidad:

$$\Pi_{A(x)} : D \cup \{e\} \rightarrow [0,1] \quad \text{Eq. (4.19)}$$

Donde e es un elemento especial que denota el caso de que x no sea aplicable en A . Además $\Pi_{A(x)}$ debe estar normalizada, es decir $\exists d / \Pi_{A(x)}(d) = 1$.

A partir de esta formulación es posible presentar los siguientes tipos de información (se incluye la comparación con el modelo de Umano y Fukami que se detallará en 4.2.2.3):

Tabla 17. Tipos de información representable en modelos basados en la Teoría de la Posibilidad.

Tipo de información	Modelo Prade-Testemale	Modelo Umano-Fukami
No difusa y conocida : "dato clásico" con valor c	$\begin{cases} \Pi_{A(x)}(e) = 0 \\ \Pi_{A(x)}(c) = 1 \\ \Pi_{A(x)}(d) = 0, \forall d \in D, d \neq c \end{cases}$	$\Pi_{A(x)}(d) = \{1 / c\}_p$
Desconocida pero aplicable	$\begin{cases} \Pi_{A(x)}(e) = 0 \\ \Pi_{A(x)}(d) = 1, \forall d \in D \end{cases}$	Unknown
No aplicable o sin sentido	$\begin{cases} \Pi_{A(x)}(e) = 1 \\ \Pi_{A(x)}(d) = 0, \forall d \in D \end{cases}$	Undefined
Ignorancia total	$\Pi_{A(x)}(d) = 1, \forall d \in D \cup \{e\}$	Null
Rango de valores: $[m, n]$	$\begin{cases} \Pi_{A(x)}(e) = 0 \\ \Pi_{A(x)}(d) = \begin{cases} 1 & \text{si } d \in [m, n] \subseteq D \\ 0 & \text{en otro caso} \end{cases} \end{cases}$	$\Pi_{A(x)}(d) = \begin{cases} 1 & \text{si } d \in [m, n] \subseteq D \\ 0 & \text{en otro caso} \end{cases}$
Distribución de posibilidad: $\mu_{\tilde{A}}$	$\begin{cases} \Pi_{A(x)}(e) = 0 \\ \Pi_{A(x)}(d) = \mu_{\tilde{A}}(d), \forall d \in D \end{cases}$	$\Pi_{A(x)}(d) = \mu_{\tilde{A}}(d), \forall d \in D$

Posibilidad de que no sea aplicable λ y $\mu_{\tilde{A}}$ en otro caso	$\begin{cases} \Pi_{A(x)}(e) = \lambda \\ \Pi_{A(x)}(d) = \mu_{\tilde{A}}(d), \forall d \in D \end{cases}$	No representable
--	--	------------------

4.2.2.3 Modelo de Umano y Fukami

Este modelo de BDRD fue presentado en [Fuk79], [Uma80], [Uma82] y [Uma83]. Los tipos de información representable son muy parecidos a los de modelo de Prade-Testemale, dado que utilizan la misma base teórica (4.2.1.5 Teoría de la posibilidad).

La principal diferencia que aporta este modelo, es la del tratamiento de la información *no aplicable*, y para ello definen los conceptos de “*unknown*”, “*undefined*” y “*NULL*”.

❖ **Definición 4.39. Información “unknown”, “undefined” y “null” en el modelo Umano-Fukami.** Sea un atributo A cuyo dominio es D y $\Pi_{A(x)}(d)$ la distribución de posibilidad de que un atributo A tome el valor d para un objeto x . En tal contexto, se definen los siguientes conceptos:

a) Los valores desconocidos y aplicables llamados “*unknown*”, se representarán mediante:

$$Unknown : \Pi_{A(x)}(d) = 1, \forall d \in D \quad \text{Eq. (4.20)}$$

b) Los valores no aplicables llamados “*undefined*”, verifican:

$$Undefined : \Pi_{A(x)}(d) = 0, \forall d \in D \quad \text{Eq. (4.21)}$$

c) La situación en la que no se conoce si una ausencia de información es aplicable o no, se representa como:

$$NULL : \{1 / Unknown, 1 / Undefined\} \quad \text{Eq. (4.22)}$$

4.2.2.4 Modelo de Zemankova y Kandel

Este modelo ([Zem84] y [Zem85]) representa la información de forma similar a los restantes modelos basados en la *teoría de la posibilidad* y desarrolla un lenguaje de manipulación de datos en el que se analiza las relaciones entre posibilidad y certeza.

4.2.2.5 Modelo GEFRED

El modelo GEFRED fue propuesto en [Med94a], [Med94b] y [Med95a] y surge como síntesis de los *modelos relacionales difusos* anteriores. Plantea un esquema suficientemente general para representar y tratar información difusa muy diversa. Incorpora dos conceptos novedosos: el *dominio difuso generalizado* (DDG) y la *relación difusa generalizada* (RDG).

❖ **Definición 4.40. Dominio difuso generalizado.** Sea, un UD $U, \tilde{P}(U)$ el conjunto de todas las distribuciones de posibilidad definidas sobre U , incluidas las que definen los tipos: “*unknown*”, “*undefined*” y “*NULL*”. Entonces se define el DDG D como:

$$D \subseteq \tilde{P}(U) \cup NULL \quad \text{Eq. (4.23)}$$

Los tipos: “*unknown*”, “*undefined*” y “*NULL*” se definen en el mismo sentido que el declarado en 4.2.2.3 *Modelo de Umano y Fukami*.

Un DDG permite representar los tipos de información imprecisa definidos en la siguiente tabla.

Tabla 18. Tipos de información representable en GEFRED.

Tipo de información imprecisa	Ejemplo
Escalar simple	<i>edad=viejo</i> Representación alternativa como DP {1/viejo}
Número simple	<i>edad=34</i> Representación alternativa como DP {1/34}
Conjunto de posibles asignaciones excluyentes de escalares	<i>aptitud={mala, buena}</i> Representación alternativa como DP {1/mala,1/buena}
Conjunto de posibles asignaciones excluyentes de números	<i>edad={20,21}</i> Representación alternativa como DP {1/20,1/21}
Una DP en el dominio de los escalares	<i>aptitud={0.6/mala,1/regular}</i>
Una DP en el dominio de los números	<i>edad={0.4/20,1/24}</i> , número difuso o etiqueta lingüística
Un número real $r \mid r \in [0,1]$ representando los grados de cumplimiento	<i>calidad=0.5</i>
Un valor unknown	Definido por la DP sobre el UD U como: $unknown = \{1 / u : u \in U\}$
Un valor undefined	Definido por la DP sobre el UD U como: $undefined = \{0 / u : u \in U\}$
Un valor NULL	Definido por la DP sobre el UD U como: $NULL = \{1 / unknown, 1 / undefined\}$

❖ **Definición 4.41. Relación difusa generalizada.** Una RDG R se define como un par de conjuntos $R = (H, B)$. El conjunto cabecera H es un conjunto fijo de ternas *atributo-dominio-compatibilidad*, siendo la *compatibilidad* opcional.

$$H = \{(A_1 : D_1 [, C_1]), (A_2 : D_2 [, C_2]), \dots, (A_n : D_n [, C_n])\} \quad \text{Eq. (4.24)}$$

Donde A_j es un atributo definido sobre el DDG D_j y $C_j \in [0,1]$ es el atributo de compatibilidad, con $j = 1, 2, \dots, n$.

El cuerpo B se compone de un conjunto de tuplas difusas generalizadas distintas, donde cada tupla está compuesta por un conjunto de ternas *atributo-valor-grado de compatibilidad*, siendo el grado opcional.

$$B = \left\{ (A_1 : \tilde{d}_{i1} [, c_{i1}]), (A_1 : \tilde{d}_{i2} [, c_{i2}]), \dots, (A_1 : \tilde{d}_{in} [, c_{in}]) \right\} \quad \text{Eq. (4.25)}$$

Siendo $i = 1, 2, \dots, m$ el número de tuplas de la relación, \tilde{d}_{ij} es el valor del dominio que toma la tupla i sobre el atributo A_j , y c_{ij} es el grado de compatibilidad asociado al valor.

El *atributo de compatibilidad* o *grado de compatibilidad* es usado en las operaciones con el fin de almacenar el grado con el que el valor de un atributo de la relación resultante ha satisfecho dicha operación.

En base a estas definiciones, GEFRED amplía el *álgebra relacional clásica* hasta un *álgebra relacional difusa generalizada*. Para ello se extienden los operadores clásicos del álgebra en [Med94a] y [Med94b], excepto la división relacional que es tratada en detalle en [Gal99]. A continuación, se definirán algunos de los operadores extendidos que se utilizarán en esta memoria: *producto cartesiano*, *proyección* y *selección*. Pero para ello, es necesario definir previamente los *comparadores extendidos* y los *comparadores difusos generalizados*.

❖ **Definición 4.42. Comparador extendido.** Sea un UD U , se denomina *comparador extendido* θ , a cualquier función definida como sigue:

$$\begin{aligned} \theta : U \times U &\rightarrow [0,1] \\ \theta(u_i, u_j) &\rightarrow [0,1] \mid u_i, u_j \in U \end{aligned} \quad \text{Eq. (4.26)}$$

❖ **Definición 4.43. Comparador difuso generalizado.** Sea un UD U , D el DDG asociado y θ un *comparador extendido* definido en U , y sea una función Θ^θ definida como sigue:

$$\begin{aligned} \Theta^\theta : D \times D &\rightarrow [0,1] \\ \Theta^\theta(\tilde{d}_1, \tilde{d}_2) &\in [0,1] \end{aligned} \quad \text{Eq. (4.27)}$$

Se dice que Θ^θ es un *comparador difuso generalizado* (CDG) sobre D inducido por el comparador extendido θ , si cumple

$$\Theta^\theta(\tilde{d}_1, \tilde{d}_2) = \theta(d_1, d_2) \quad \forall d_1, d_2 \in U \quad \text{Eq. (4.28)}$$

Donde \tilde{d}_1, \tilde{d}_2 representan las distribuciones de posibilidad $\{1/\tilde{d}_1\}, \{1/\tilde{d}_2\}$, respectivamente, para los valores d_1, d_2 .

❖ **Definición 4.44. Producto cartesiano difuso generalizado.** Sea R y R^* dos *relaciones difusas generalizadas* definidas por:

$$R = \left\{ \begin{aligned} H &= \left\{ (A_1 : D_1 [, C_1]), \dots, (A_n : D_n [, C_n]) \right\} \\ B &= \left\{ (A_1 : \tilde{d}_{i1} [, c_{i1}]), \dots, (A_1 : \tilde{d}_{in} [, c_{in}]) \right\} \end{aligned} \right.$$

$$R^* = \begin{cases} H^* = \{(A_1^* : D_1^* [, C_1^*]), \dots, (A_n^* : D_n^* [, C_n^*])\} \\ B^* = \{(A_1^* : \tilde{d}_{k1}^* [, c_{k1}^*]), \dots, (A_n^* : \tilde{d}_{kn}^* [, c_{kn}^*])\} \end{cases}$$

Siendo $i = 1, 2, \dots, m$ y $k = 1, 2, \dots, m^*$ las *cardinalidades* y m, m^* los *grados* respectivamente de relaciones de R y R^* . Se define el *producto cartesiano difuso generalizado* $R \times R^*$ como la RDG definida por:

$$R \times R^* = \begin{cases} H_{\times} = H \cup H^* \\ B_{\times} = B \cup B^* \end{cases} \quad \text{Eq. (4.29)}$$

❖ **Definición 4.45. Proyección difusa generalizada.** Sea R una RDG definida como:

$$R = \begin{cases} H = \{(A_1 : D_1 [, C_1]), \dots, (A_n : D_n [, C_n])\} \\ B = \{(A_1 : \tilde{d}_{i1} [, c_{i1}]), \dots, (A_1 : \tilde{d}_{in} [, c_{in}])\} \end{cases}$$

Sea X un conjunto definido como:

$$X \subseteq H, X = \{(A_s : D_s [, C_{s^*}])\} : s \in S, s^* \in S^*; S, S^* \subseteq \{1, 2, \dots, n\}$$

Se define la *proyección difusa generalizada de R sobre X* , $P_X(R)$, a la RDG definida por:

$$P_X(R) = \begin{cases} H_p = X \\ B_p = \{(A_s : \tilde{d}_{is} [, c_{is^*}])\} \end{cases} \quad \text{Eq. (4.30)}$$

Con $s \in S, s^* \in S^*; S, S^* \subseteq \{1, 2, \dots, n\}$

❖ **Definición 4.46. Selección difusa generalizada.** Sea R una RDG:

$$R = \begin{cases} H = \{(A_1 : D_1 [, C_1]), \dots, (A_n : D_n [, C_n])\} \\ B = \{(A_1 : \tilde{d}_{i1} [, c_{i1}]), \dots, (A_1 : \tilde{d}_{in} [, c_{in}])\} \end{cases}$$

Sea $A_k \mid k \in \{1, 2, \dots, n\}$ un atributo, $\tilde{a} \in D$ una constante, \ominus^θ un CDG, y sea $\gamma \in [0, 1]$ un *umbral de cumplimiento*. Se define la *selección difusa generalizada* $S_{\ominus^\theta(A_k, \tilde{a}) \geq \gamma}(R)$ aplicada sobre R con la condición inducida por $\ominus^\theta(A_k, \tilde{a})$ y calificada por γ , como la siguiente RDG:

$$S_{\ominus^\theta(A_k, \tilde{a}) \geq \gamma}(R) = \begin{cases} H_S = \{(A_1 : D_1 [, C_1]), (A_2 : D_2 [, C_2]), \dots, (A_n : D_n [, C_n])\} \\ B_S = \{(A_1 : \tilde{d}_{r1} [, c_{r1}^*]), \dots, (A_k : \tilde{d}_{rk} [, c_{rk}^*]), \dots, (A_n : \tilde{d}_{rn} [, c_{rn}^*])\} \end{cases} \quad \text{Eq. (4.31)}$$

Con $c_{rk}^* = \ominus^\theta(d_{rk}, \tilde{a}) \geq \gamma$ y $r = 1, 2, \dots, m^*$, siendo m^* el número de tuplas de la selección resultante.

4.2.2.5.1 FIRST una implementación computacional de GEFRED

“Fuzzy Interface for Relational Systems” FIRST ([Med94b],[Med95b] y [Gal99]) es una implementación en un SGBDR de una BDRD basada en el modelo teórico GEFRED (4.2.2.5), donde se adapta el lenguaje SQL para obtener un SQL difuso o FSQL ([Gal98a], [Gal98b] y [Gal98c]).

A continuación se describen los componentes principales de FIRST, incluyendo la forma de representar y manipular la información imprecisa.

4.2.2.5.1.1 Representación de datos difusos y/o con tratamiento difuso

FIRST implementa los datos representables en un DDG (Tabla 18. *Tipos de información representable en GEFRED.*) mediante tres tipos de atributo (*atributos difusos de tipo 1, 2 y 3*) más los tipos: *unknown*, *undefined* y *NULL*.

- a) **Atributo difuso de tipo 1** (datos precisos). Representan información clásica (no difusa). Sobre estos atributos se pueden aplicar consultas clásicas (no difusas) o consultas (imprecisas) construidas con etiquetas lingüísticas. Como ejemplo, una consulta sobre un atributo *edad* podría representarse como:

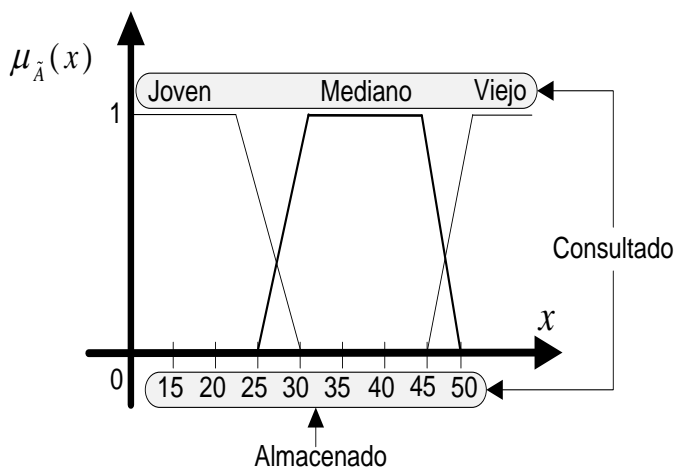


Ilustración 45. FIRST. Atributos difusos de tipo 1.

- b) **Atributo difuso de tipo 2** (datos imprecisos sobre referencial ordenado). Estos tipos de atributos representan información difusa y admiten consultas difusas. Sobre el dominio subyacente del atributo se definen distribuciones de posibilidad y además debe existir una relación de orden entre los elementos del dominio. Se distinguen los siguientes tipos de atributo de tipo 2:
- i. *Distribución de posibilidad trapezoidal.* Para modelar la función de pertenencia asociada a la DP, se utilizará un trapecoide determinado por cuatro parámetros $[\alpha, \beta, \gamma, \delta]$. La función deberá estar normalizada, es decir, deberá existir al menos un elemento en el referencial cuya probabilidad sea 1:

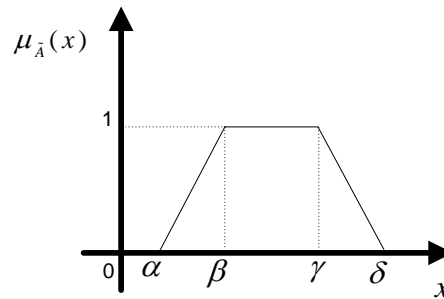


Ilustración 46. FIRST. Distribución de posibilidad trapezoidal.

- ii. *Etiquetas lingüísticas.* Para representar las etiquetas (4.2.1.4) se usará una DP en su representación trapezoidal. Por ejemplo para la etiqueta *alto*, una posible representación sería:

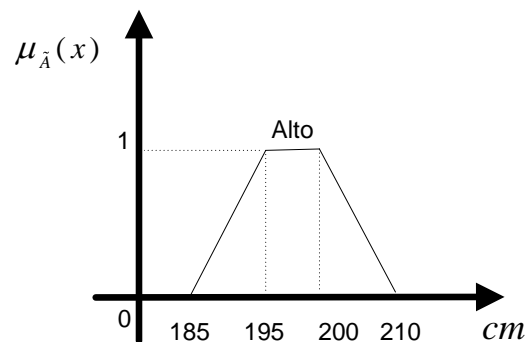


Ilustración 47. FIRST. Distribución de posibilidad de una etiqueta lingüística.

- iii. *Valores aproximados.* Dado un valor del dominio n y un valor llamado *margen* (base del triángulo), se representa el concepto impreciso “aproximadamente n ” como una DP triangular normalizada:

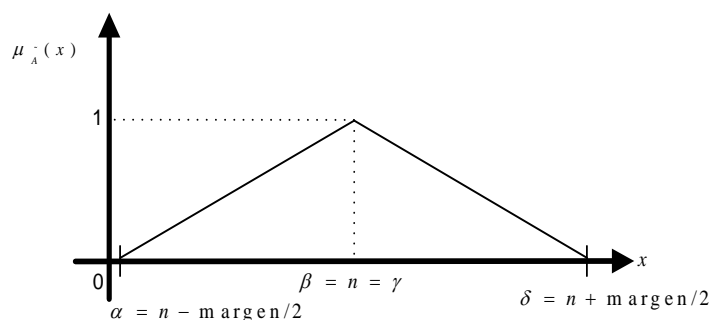


Ilustración 48. FIRST. Distribución de posibilidad “aproximadamente n ”.

- iv. *Intervalos de proximidad.* La representación de un intervalo $[n, m]$ es un caso especial de DP trapezoidal ([Gra80]) donde los extremos tienen posibilidad igual a 1:

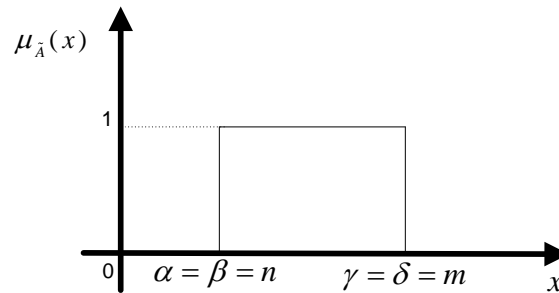


Ilustración 49. FIRST. Distribución de posibilidad de un intervalo [n,m].

c) **Atributo difuso de tipo 3** (datos imprecisos sobre referencial no ordenado). Son atributos que definen una “relaciones de semejanza” entre los valores del dominio subyacente. Un ejemplo de este tipo puede apreciarse en la siguiente ilustración, siendo el dominio del atributo $D = \{\text{rubio, moreno, pelirrojo, castaño}\}$, y $\alpha(i, j)$ el valor de semejanza entre los valores del dominio D .

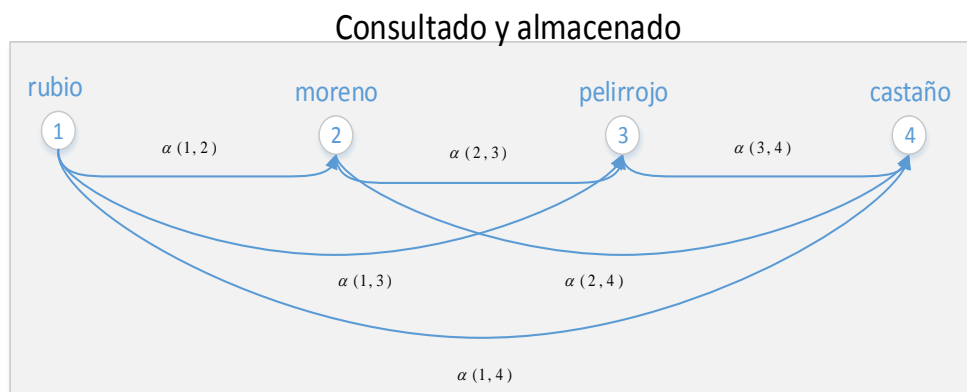


Ilustración 50. FIRST. Atributos difusos de tipo 3.

Se distinguen dos tipos diferentes de este tipo de atributo:

- i. *Escalares simples.* La representación de la DP para un escalar d , es una única pareja de datos $\{1 / d\}$ con máxima posibilidad.
- ii. *DP sobre escalares.* Los datos imprecisos de este tipo, se representan como el conjunto T de pares de valores de dominio $D = p_1, p_2, \dots, p_n$ y sus respectivos valores de posibilidad d_1, d_2, \dots, d_n :

$$T = \{(d_1, p_1), (d_2, p_2), \dots, (d_n, p_n)\}$$
. Donde T ha estar normalizada.

Además de los tres tipos de atributos descritos, FIRST representa los tres valores especiales soportados en GEFRED: *unknown*, *undefined* y *NULL*.

d) **Valor unknown.** Representa el conocimiento de que el valor del atributo es uno de los del dominio de definición U , pero se desconoce exactamente cuál. Para representar

este tipo de información se le asigna la DP $\{1 / u, \forall u \in U\}$ $\{1 / u, \forall u \in U\}$. En la siguiente ilustración se muestra una representación gráfica de la DP.

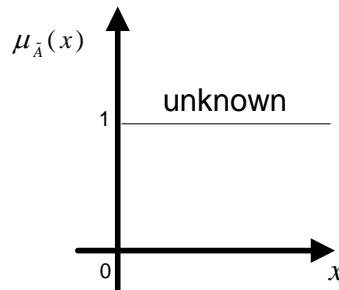


Ilustración 51. FIRST. Distribución de posibilidad para el valor “unknown”.

- e) Valor *undefined*. En este caso, se conoce que el valor del atributo no puede ser ninguno de los de su dominio de definición U . En este caso, su representación es la DP $\{0 / u, \forall u \in U\}$:

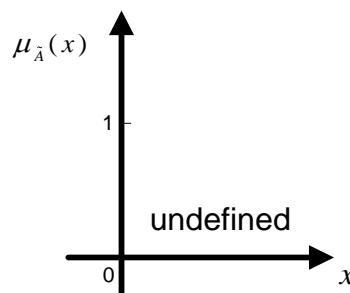


Ilustración 52. FIRST. Distribución de posibilidad para el valor “undefined”.

- f) Valor *NULL*. Este tipo modela una ausencia total de información sobre el valor del atributo. Se representará este tipo de dato por la DP $\{1 / unknown, 1 / undefined\}$.

4.2.2.5.1.2 Comparadores difusos generalizados

Existen varias implementaciones posibles de comparadores de números difusos. FIRST utiliza la aproximación descrita en **[Pra84a]** para implementar 15 comparadores, que se pueden clasificar según su tipo de dominio referencial:

- a) Se crea un CDG sobre *referencial no ordenado*: FEQ (“Fuzzy EQal”). Este comparador se aplica en *atributos difusos de tipo 3*.
- b) Se definen 14 comparadores difusos generalizados sobre *referencial ordenado*. Se distinguen, dos tipos de CDG, de *posibilidad* y de *necesidad*. Los de *posibilidad* son menos restrictivos (recuperan menos tuplas) que los de *necesidad*. Estos 14 comparadores se muestran en la siguiente tabla.

Tabla 19. Comparadores difusos generalizados sobre un referencial ordenado en FIRST.

	Nombre del CDG	Significado en español / traducción en inglés
CDG de posibilidad	FEQ	Igualdad difusa / “Fuzzy EQual”
	FGT	Mayor que difuso / “Fuzzy Greater Than”
	FGEQ	Mayor o igual que difuso / “Fuzzy Greater or EQual than”
	FLT	Menor que difuso / “Fuzzy Less Than”
	FLEQ	Menor o igual que difuso / “Fuzzy Less or EQual than”
	MGT	Mucho mayor que difuso / “Much Greater Than”
	MLT	Mucho menor que difuso / “Much Less Than”
CDG de necesidad (N) indica necesariamente / (N) “necessarily”	(N)FEQ	(N) Igualdad difusa / “(N) Fuzzy EQual”
	(N)FGT	(N) Mayor que difuso / “(N) Fuzzy Greater Than”
	(N)FGEQ	(N) Mayor o igual que difuso / “(N) Fuzzy Greater or EQual than”
	(N)FLT	(N) Menor que difuso / “(N) Fuzzy Less Than”
	(N)FLEQ	(N) Menor o igual que difuso / “(N) Fuzzy Less or EQual than”
	(N)MGT	(N) Mucho mayor que difuso / “(N) Much Greater Than”
	(N)MLT	(N) Mucho menor que difuso / “(N) Much Less Than”

4.2.2.5.1.3 Grado y umbral de cumplimiento

Cuando se calcula el resultado de una consulta difusa, FIRST establece una serie de condiciones que deben de cumplir las tuplas/atributos de la relación final:

- a) “*THOLD*” o *umbral de cumplimiento* $\in [0,1]$. Es el mínimo valor que debe verificarse para que una tupla aparezca como resultado en la relación. Este umbral ejerce un control sobre la precisión del resultado. En FIRST, se permite establecer un número o un *calificador lingüístico* (como “mucho” o “poco”) como umbral.
- b) “*CDEG*”. *Grado de cumplimiento* $\in [0,1]$. Este valor indica como es de compatible una tupla o atributo respecto al criterio de selección. FIRST calcula este valor agregando a la relación resultado un atributo extra. Este atributo almacena del *grado de cumplimiento* (GA) de cada tupla.

4.2.2.5.1.4 Arquitectura de FIRST

En siguiente figura se muestran los módulos principales (arquitectura) de FIRST.

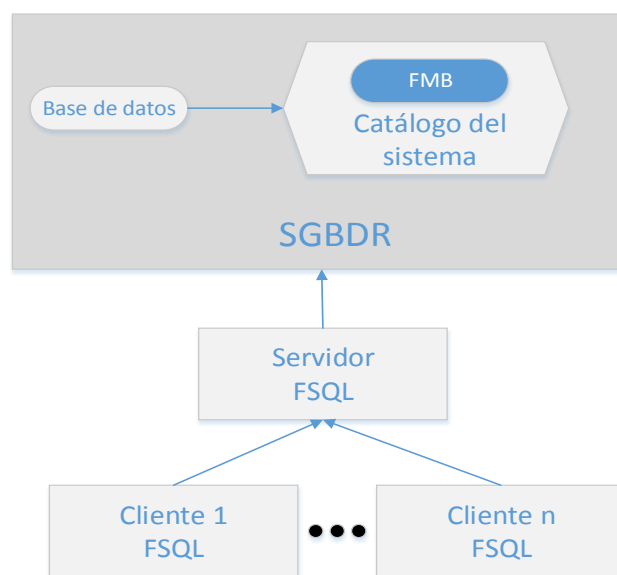


Ilustración 53. Esquema de FIRST.

- a) **SGBDR.** FIRST basa su implementación en una base de datos relacional comercial (en concreto Oracle©). Esto implica que todas las operaciones (difusas o no) que deban realizarse se traducirán en peticiones a dicho SGBDR.
- b) **Base de datos.** Es el conjunto de tablas relacionales que permiten almacenar la información necesaria (imprecisa o no).
- c) **FMB** (“*Fuzzy Metaknowledge Base*”, *Base de meta-conocimiento* difuso). Es el diccionario (o *catálogo*) de un SGBDR el que se encarga de almacenar información acerca de los datos (los llamados meta-datos o meta-conocimiento), así como de usuarios, permisos, etcétera. La FMB extiende este *catálogo* a fin de recoger la meta-datos procedentes de la información difusa.
- d) **Servidor FSQL.** Acepta sentencias en FSQL y las traduce al lenguaje SQL del SGBDR. En el proceso de traducción se hace uso del contenido del FMB.
- e) **Cliente FSQL.** Es la interfaz entre el usuario y el servidor de FSQL. Es posible conectar más de un cliente a la vez al mismo servidor.

4.3 Modelo lógico de bases de datos

Para los seres humanos, expresar el conocimiento mediante reglas (Si sucede X y además sucede Y entonces ocurre Z) es algo habitual. Las reglas permite representar un razonamiento y extraer nuevas conclusiones (ocurre Z) a partir de un conjunto de premisas (X e Y).

Durante la última década, el equipo de expertos a cargo de G. ha tenido la oportunidad de estudiar los datos (reales y de calibración) proporcionados por el instrumento. Esto ha generado un volumen de conocimiento (susceptible de ser expresado con reglas) de incalculable valor para conocer tanto el funcionamiento de G. como el fenómeno bajo estudio: el polvo cometario.

En este contexto, el modelo lógico de base de datos (utilizando la *Lógica formal*), es la herramienta que permite aplicar el conocimiento adquirido a los datos generados por GIADA.

La *Lógica formal* [Dea74] es un excelente marco para el análisis y representación de información mediante *sentencias* (unidades de información). La relación entre sentencias es tema principal de la *Lógica formal* y es la *inferencia* la herramienta que permite deducir nueva información a partir de un conjunto de sentencias iniciales. Las diferentes formas de realizar la *inferencia* distinguen a los dos tipos de Lógica: *Lógica proposicional* y *Lógica de predicados*:

- a) *Lógica proposicional*. Esta *Lógica* atiende sólo los operadores (o conectivos) de las proposiciones para llevar a cabo la inferencia. Entendiendo por proposición, una sentencia que puede ser verdadera o falsa.
- b) *Lógica de predicados*. La inferencia se realiza considerando además de los conectivos, la estructura interna de los predicados, definiendo predicado como una expresión lingüística que puede conectarse con una o varias expresiones.

Es precisamente la *Lógica de predicados* el marco de referencia para definir la *Lógica de primer orden*, que a su vez es la base para formular el *Modelo lógico de base de datos*. La *Lógica de primer orden* junto al modelo de BD basado en Lógica, son los temas tratados en esta sección.

4.3.1 Lógica de primer orden

En esta apartado se presentará la *Lógica de primer orden* [Smu95], la inferencia (herramienta para deducir nueva información), el algoritmo de deducción más conocido en sistemas computacionales lógicos: *el algoritmo de resolución*, y se repasan algunas implantaciones de lenguajes de programación lógica: *Prolog* y *Datalog*.

4.3.1.1 El lenguaje de la Lógica de primer orden

Se define el *lenguaje de primer orden* como el par $L = (A, W)$, donde A es el alfabeto de símbolos y W es un conjunto de fórmulas sintácticamente correctas llamadas fórmulas bien formadas (fbf), que se componen a partir de los elementos de A . El conjunto A consta de los siguientes símbolos:

- a) *Símbolos de puntuación*. () , ; :
- b) *Símbolos de constantes*. Habitualmente se usan las primeras letras minúsculas del alfabeto: a, b, ...
- c) *Símbolos de variables*. Normalmente se usan las últimas letras minúsculas del alfabeto: v, w, etcétera.
- d) *Símbolos de funciones*. Se suelen utilizar se usan las letras del alfabeto: f, g, etcétera.
- e) *Símbolos de predicados*. Habitualmente se usan letras mayúsculas.
- f) *Símbolos de conectivos lógicos*: \rightarrow , \wedge , \vee y \neg .

g) Cuantificadores: \exists y \forall .

A continuación se define una serie de conceptos útiles en un *lenguaje de primer orden*:

- ❖ **Definición 4.47. Término.** Se define *término* de un *lenguaje de primer orden* como:
 - a) Toda *constante* es un *término*.
 - b) Toda *variable* es un *término*.
 - c) Si f es una función n -aria y t_1, t_2, \dots, t_n son términos, entonces $f(t_1, t_2, \dots, t_n)$ es un término.
 - d) Los únicos términos válidos son los obtenidos por las reglas anteriores.

- ❖ **Definición 4.48. Fórmula atómica.** Si P es un predicado n -ario y t_1, t_2, \dots, t_n son términos, entonces $P(t_1, t_2, \dots, t_n)$ es una *fórmula atómica*.

- ❖ **Definición 4.49. Fórmula bien formada.** Se define una *fórmula bien formada* (fbf) como sigue:
 - a) Una *fórmula atómica* es una fbf.
 - b) Si w_1 y w_2 son formulas bien formadas, también lo son: $(w_1 \wedge w_2)$, $(w_1 \vee w_2)$, $(w_1 \rightarrow w_2)$ y $\neg(w_1)$
 - c) Si x es una variable y w es una fbf, entonces $(\exists x, w)$ y $(\forall x, w)$ son formulas bien formadas.

- ❖ **Definición 4.50. Ámbito de un cuantificador, variable ligada y libre.** Sea un cuantificador Q , una variable x , una fbf w y (Qx, w) otra fbf. Se define como el *ámbito del cuantificador* Q en Qxw a la fbf w . Si x ocurre en w se dice que x es una variable ligada, en otro caso se denomina que es libre.

- ❖ **Definición 4.51. fbf cerrada.** Una fbf se dice *cerrada* si no tiene ninguna *variable libre*, es decir sólo tiene *variables ligadas* y *constantes*.

- ❖ **Definición 4.52. Sentencia.** Una *sentencia* es una *fbf* en la que toda variable está ligada.

- ❖ **Definición 4.53. Sustitución elemental de una variable por un término.** Sea una variable x y un término t , se denomina *sustitución elemental* de x por t y se representa $t | x$, a una aplicación que a cada fórmula le hace corresponder el resultado de sustituir las ocurrencias libres x de por t .

4.3.1.2 Inferencia

En la *Lógica de primer orden*, la derivación (obtención) de una conclusión partiendo de una serie de premisas (*inferencia*) se puede conseguir con dos métodos distintos: el sintáctico y el semántico.

4.3.1.2.1 El método sintáctico o “teoría de pruebas”

La inferencia se realiza utilizando la siguiente definición:

- ❖ **Definición 4.54. Relación de derivabilidad.** Sea un lenguaje de primer orden L , un conjunto inicial de sentencias S , conjunto de reglas R , y s el conjunto de nuevas sentencias surgido de aplicar R . Se define la relación de derivabilidad como:

$$\perp = \{ \langle S, s \rangle \mid S \subset L, s \in L \text{ y } s \text{ es derivable de } S \text{ usando } R \} \quad \text{Eq. (4.32)}$$

Habitualmente se denomina a S *axiomas*, a s *teoremas*, y a R reglas de inferencia. Dos de las reglas de inferencia más utilizadas son la *generalización* y el *Modus Ponens*.

En la inferencia por *generalización*, si una variable X toma sus valores en un conjunto A y A está contenido en un conjunto B , se puede afirmar que X toma sus valores del conjunto B .

Utilizando la inferencia por *Modus Ponens*, si un predicado P implica (se puede deducir lógicamente) Q , y se tiene como premisa a P , entonces se puede inferir Q . Más formalmente: $P \rightarrow Q, P \perp Q$

Se presentan a continuación los conceptos de *demostración* y la *aplicación deducción* que serán usados más adelante.

- ❖ **Definición 4.55. Demostración.** Sea un lenguaje de primer orden $L = (A, W)$, $P(L)$ el conjunto de todas las proposiciones de L y $\Gamma \subset P(L)$. Una *demostración* w a partir de Γ y se nota por $\Gamma \perp w$, es una sucesión infinita de proposiciones $\langle B_1, B_2, \dots, B_n \rangle$ que cumple alguna de las siguientes condiciones:

- a) $B_i \in \Gamma$
- b) $B_i \in A$
- c) $\exists j, k < i \mid B_j = B_k \rightarrow B_i$ con $1 \leq i \leq n$

- ❖ **Definición 4.56. Aplicación de deducción.** Sea un lenguaje de primer orden L , $P(L)$ el conjunto de todas las proposiciones de L , $\rho(P(L))$ el conjunto de partes $P(L)$ y $\Gamma \subset P(L)$. Se define la *aplicación deducción* como la función $Ded : \rho(P(L)) \rightarrow \rho(P(L))$ tal que:

$$Ded(\Gamma) = \{ w \in P(L) : \Gamma \perp w \} \quad \text{Eq. (4.33)}$$

4.3.1.2.2 El método semántico o “teoría de modelos”

Con este método, la inferencia se realiza a través de modelos. Antes de llegar a ellos, es necesario presentar varios conceptos.

- ❖ **Definición 4.57. L-estructura.** Una *L-estructura* es un conjunto cuádruple (U, K, F, P) :

- a) U es un conjunto no vacío, llamado universo de interpretación.
- b) $K \subseteq U$ del que toman sus valores las constantes del lenguaje.
- c) F es el conjunto de funciones $f^U : U^u \rightarrow U$, donde f es cada uno de los símbolos funcionales del lenguaje.
- d) P es un conjunto de predicados R^U , uno por cada símbolo de predicado del lenguaje, del que toman sus valores las relaciones n-arias.

❖ **Definición 4.58. Asignación en U .** Sea un universo de interpretación, V el conjunto de todas las variables del lenguaje. Se llama una *asignación en U* , a una función $f : V \rightarrow U$.

❖ **Definición 4.59. Interpretación de lenguaje.** Una *interpretación I de un lenguaje L* , es una pareja formada por una *L -estructura en U* y una *asignación en U* .

La definición de *modelo e inferencia (consecuencia lógica)* deriva de las definiciones anteriores:

❖ **Definición 4.60. Modelo de un conjunto de fórmulas bien formadas.** Un modelo de un conjunto de fórmulas bien formadas, es una *interpretación de lenguaje en la todas las fórmulas bien formadas son verdad*.

❖ **Definición 4.61. Consecuencia lógica de un conjunto de fórmulas bien formadas.** Una *fbf w* es una consecuencia lógica de un conjunto de fórmulas bien formadas W , y se nota por $w \models W$, sí y sólo sí w es verdad en todos los modelos de W .

❖ **Definición 4.62. Conjunto de consecuencias lógicas de un conjunto de conjunto de fórmulas bien formadas.** Sea un *lenguaje de primer orden L* , $P(L)$ el conjunto de todas las proposiciones de L y $\Gamma \subset P(L)$ un conjunto de proposiciones. Se denomina $Con(\Gamma)$ al conjunto de todas las proposiciones que verifican:

$$Con(\Gamma) = \{w \in P(L) : \Gamma \models w\} \quad \text{Eq. (4.34)}$$

Los dos siguientes teoremas garantizan que *el método sintáctico* (o “teoría de pruebas”) y el método semántico (o “teoría de modelos”) son equivalentes.

❖ **Definición 4.63. Teorema de coherencia.** Sea un *lenguaje de primer orden L* , $P(L)$ el conjunto de todas las proposiciones de L , $\Gamma \subset P(L)$, $w \in P(L)$ y $\Gamma \perp w$. Entonces $\Gamma \models w$, esto es: $Ded(\Gamma) \subset Con(\Gamma)$.

❖ **Definición 4.64. Teorema de completitud de Gödel.** Sea un *lenguaje de primer orden L* , $P(L)$ el conjunto de todas las proposiciones de L , $\Gamma \subset P(L)$, $w \in P(L)$ y $\Gamma \perp w$. Entonces $\Gamma \perp w$, esto es: $Con(\Gamma) \subset Ded(\Gamma)$.

4.3.1.3 Algoritmo de resolución

El *algoritmo de resolución* ([Fro86] y [Nil87]) solamente trabaja con reglas de inferencia estructuradas en forma clausular. Antes de ver este algoritmo, es necesario presentar la *forma clausular de la Lógica*.

4.3.1.3.1 Forma clausular de la Lógica

La *forma clausular de la Lógica* ([Cha73] y [Kow79]) es un sub-lenguaje en el que, normalmente, se escriben todos los programas lógicos que se ejecutan en un computador. La forma clausular de una sentencia es muy regular (por tanto fácilmente procesable) sin ello signifique perder expresividad.

A continuación se presentan algunas definiciones útiles hasta llegar el *teorema de la forma normal conjuntiva* que muestra la equivalencia entre una proposición y su forma clausular.

- ❖ **Definición 4.65. Literal.** Se denomina *literal* a una fórmula atómica (*literal positivo*) o a su negación (*literal negativo*).
- ❖ **Definición 4.66. Cláusula.** Se denomina *cláusula* a una disyunción de *literales*.
- ❖ **Definición 4.67. Cláusula de Horn o cláusula de Horn con cabeza.** Se denomina *cláusula de Horn* o *cláusula de Horn con cabeza* a las *cláusulas* que tienen, como máximo, un *literal positivo*.
- ❖ **Definición 4.68. Cláusula sin cabeza.** Se denomina *cláusula sin cabeza* a las *cláusulas* no que tienen *literales positivos*.
- ❖ **Definición 4.69. Cláusula vacía.** Se denomina *cláusula vacía*, y se nota por \square , a las *cláusulas* sin *literales*.
- ❖ **Definición 4.70. Teorema de la forma normal conjuntiva.** Toda proposición es equivalente a una proposición expresada como una disyunción de *cláusulas*.

Existe un algoritmo ([Nil87] y [Win84]) que transforma cualquier sentencia de la *Lógica de primer orden* en un conjunto de *cláusulas*, lo que favorece la expresión de la información de inicial (hechos y reglas) para realizar la inferencia. Este algoritmo introduce las llamadas *funciones de Skolem* para eliminar los cuantificadores existenciales presentes al final del proceso de conversión. Estas *funciones de Skolem* rompen la equivalencia entre sentencia inicial y su forma clausular, pero a cambio se obtiene un conjunto de *cláusulas* que será satisfacible sólo si la sentencia inicial lo era.

4.3.1.3.2 El algoritmo de resolución

Este algoritmo, de forma esquemática, se encarga de buscar *literales complementarios* entre un conjunto de *cláusulas* y aplicar *resolventes*:

- ❖ **Definición 4.71. Literal complementario en un conjunto de cláusulas.** Sea C un conjunto de *cláusulas*, $c_1 \in C$ y $c_2 \in C$ dos *cláusulas*. Se denomina *literal complementario en C* , a un literal p que aparece a la vez como *literal positivo* en c_1 y *negativo* en c_2 .
- ❖ **Definición 4.72. Cláusula resolvente.** Sea C un conjunto de *cláusulas*, $c_1 \in C$ y $c_2 \in C$ dos *cláusulas* (llamadas *padre*) con un *literal complementario* p . Sean c_1^* y c_2^* la *cláusulas* obtenidas tras la eliminación en p de c_1 y c_2 . Se denomina *cláusula resolvente* a la nueva *cláusula* $\{c_1^* \vee c_2^*\}$.

A modo de ejemplo, si $C = \{p \vee w_1, \neg p \vee w_2\}$, siendo p un literal, y w_1, w_2 dos *cláusulas*. El *literal complementario* sería p , ya que p y $\neg p$ ocurren en C . Por tanto la *cláusula resolvente* sería $C^* = \{w_1 \vee w_2\}$.

Formalmente, el proceso de *demostración por resolución* puede expresarse como:

- ❖ **Definición 4.73. Demostración por resolución.** Una *demostración por resolución* de una *cláusula* C , a partir de un conjunto de sentencias Γ (llamadas hipótesis), es una sucesión finita de $\langle B_1, B_2, \dots, B_m \rangle$ de *cláusulas* tales que: $B_m = C$ y $\forall i, 1 \leq i \leq m$, $B_i \in \Gamma$ o bien B_i es una *resolvente* de dos *cláusulas* anteriores.

Del *algoritmo de resolución* se pueden derivar tres importantes propiedades que son aplicables tanto a la *Lógica proposicional* como a la *Lógica de predicados*:

- a) La resolución es consistente, esto es, las *cláusulas* utilizadas como padres implican la *cláusula resultante*.
- b) La *cláusula vacía* se obtiene como *cláusula resolvente*, sí y sólo sí, las dos *cláusulas padre* son *literales complementarios*.
- c) Dado un conjunto de *cláusulas de Horn*, se obtiene la *cláusula vacía* mediante el *algoritmo de resolución*, sólo si el conjunto inicial es insatisfacible.

Por lo general, un conjunto de *cláusulas* insatisfacible, puede llevar a la *cláusula vacía* por múltiples caminos provocando una explosión combinatoria difícilmente tratable en un computador. Por ello se han diseñado diversas técnicas (SLD-Resolución [Hog90]) que optimizan la selección de la pareja de padres se utilizará para obtener la *cláusula resolvente*.

4.3.1.4 Lenguajes de programación lógica

La programación lógica ([Deg86] y [Hog90]) basa su implementación en la *Lógica de predicados de primer orden*, aplicada generalmente sobre un conjunto de *cláusulas de Horn*. Las cualidades más importantes de este tipo de programación, son las siguientes:

- a) Permite que la representación de la información asociada a las restricciones y suposiciones de dominio de un problema, se realiza de una forma muy directa e independientemente de la implementación.
- b) Permite dar una caracterización matemática precisa de las relaciones existentes entre una entrada y los resultados que se derivan de ella.
- c) Ofrece un marco teórico propicio para la representación de conocimiento y para el desarrollo de bases de datos.
- d) Puede ser fácilmente modificado o extendido para representar distintas formas de conocimiento a diferentes niveles (meta-conocimiento).
- e) Implementan eficazmente el algoritmo de resolución.

Se presentan a continuación dos de las implementaciones de *lenguajes lógicos* más conocidas: *Prolog* y *Datalog*.

4.3.1.4.1 Prolog

El lenguaje *Prolog* ([Clo81], [Gra84], [Ste86] y [Zan84]) es un lenguaje de programación lógica para computación simbólica (no numérica) y su modelo teórico es una variante de la *Lógica de predicados de primer orden*. *Prolog* es el lenguaje más utilizado en la representación y programación lógicas, y en concreto en las bases de datos lógicas.

Un programa en *Prolog* se corresponde con un conjunto de cláusulas (también llamada hipótesis), donde el ámbito de cada variable se limita a la cláusula donde aparece. Mientras que una consulta sería un teorema a ser probado (inferido) por un demostrador de teoremas partiendo de las hipótesis iniciales.

Existen tres tipos de cláusulas en *Prolog*:

- a) **Reglas.** Una *regla* expresa una afirmación condicionada y se representa como una cláusula que se compone de una *cabeza* (formado por un literal positivo) y un *cuerpo* (formado por varios literales separados por comas o puntos y comas). *Cabeza* y *cuerpo* se separan a su vez por el símbolo “:-”, significando la palabra “si”. Por ejemplo la información “el abuelo de X es Y, si el padre de X es Z y el padre de Z es Y” puede representarse por la regla:

$$\text{abuelo}(X,Y):- \text{padre}(X,Z),\text{padre}(Z,Y)$$

Por defecto las variables de la *cabeza* se consideran cuantificadas universalmente, mientras que las del *cuerpo* lo están existencialmente. La coma entre los literales indica conjunción, el punto y coma se usa para expresar disyunción.

- b) **Hechos.** Un *hecho* es una *cláusula* formada por un único literal y representa una afirmación axiomática, es decir, que es cierta independientemente de cualquier condición. Por ejemplo:

$$\text{abuelo}(\text{Juan},\text{Pedro})$$

- c) **Objetivos.** Son *cláusulas* sin *cabeza* y representan consultas. Por ejemplo:

$$?\text{abuelo}(X,\text{Pedro})$$

Todo programa en *Prolog* consiste en una secuencia de hechos y reglas que representan una declaración de propiedades verificables por una serie de relaciones. Aunque, desde el punto de vista lógico, el orden en el aparecen los *hechos*, las *reglas* y el orden de los predicados dentro de la *regla*, debería ser irrelevante, esto no sucede en *Prolog*. El orden tiene un impacto directo en el coste computacional a la hora de realizar consultas.

Prolog utiliza el principio de *resolución* de Robinson ([Rob63], [Rob65a] y [Rob65b]) para resolver objetivos. Dado un objetivo a ser probado, el motor de inferencia de *Prolog* realiza una búsqueda en profundidad (de izquierda a derecha y de arriba a abajo) entre los *hechos* y las *reglas* a fin de poder unificar el objetivo actual con alguna o algunas *cláusulas* presentes.

4.3.1.4.2 Datalog

En [Cer90] se presenta a *Datalog* como un lenguaje de programación lógica para bases de datos. *Datalog* es similar a la sintaxis de *Prolog*, pero no así en su semántica.

- a) *Prolog* actúa sobre tuplas de forma individual. Esto plantea un problema de eficiencia con un gran volumen de datos. *Datalog* se orienta a conjuntos, luego este problema se minimiza.
- b) El rendimiento de *Prolog* se ve afectado por el orden de los predicados. Esto no sucede así en *Datalog*, ya que no es sensible a dicho orden, ni al orden de la tuplas de la BD.

- c) *Prolog* utiliza predicados especiales para realizar ciertas actividades (depuración, control de entrada-salida), en *Datalog* estos predicados no son necesarios.
- d) En *Prolog* se dispone de símbolos de función para construir funciones recursivas y estructuras de datos complejas. Esta característica no es útil cuando se trabaja con BD planas (no recursivas y tipos de datos sencillos) y por ello no están presentes en *Datalog*.

El algoritmo de deducción de GDB está basado en *Datalog*, como se detalla en (5.1.2.2 Algoritmo de deducción).

4.3.2 Representación mediante Lógica de una base de datos relacional

La Lógica ofrece una base sólida base como teoría de BD, especialmente para expresar consultas y para definir restricciones de integridad. Una instancia de BD puede caracterizarse de dos enfoques distintos utilizando la *Lógica de primer orden*:

- a) *BD como teoría de pruebas*. Los hechos (tuplas) y las reglas de deducción constituyen en sí mismo una teoría y las consultas son teoremas a ser probados a partir de dicha teoría, utilizando técnicas de demostración (4.3.1.2.1 El método sintáctico o "teoría de pruebas").
- b) *BD como teoría de modelos*. Este es el enfoque más tradicional y consiste en considerar un estado concreto de la BD como una interpretación que debe ser un modelo para la teoría (4.3.1.2.2 El método semántico o "teoría de modelos"). Las consultas se resuelven calculando el valor de verdad del predicado que la constituye.

En [Rei84] se demuestra la equivalencia de ambos enfoques.

Antes de ver en detalle, estas dos aproximaciones, se presentan un conjunto de definiciones previas ([Gal87]): *lenguaje de primer orden relacional*, *interpretación relacional* hasta llegar a la definición de *base de datos lógica relacional*.

❖ **Definición 4.74. Lenguaje de primer orden relacional.** *Un lenguaje de primer orden*

$L = \{ A, W \}$ se dice *relacional*, sí y sólo sí verifica:

- a) $A \neq \emptyset$ y el número de constantes de A es finito.
- b) C no contiene símbolos de función.
- c) Hay un número finito de predicados en A .
- d) Entre los predicados de A debe de estar el predicado binario de igualdad $E_{=}$.
- e) Entre los predicados unarios se distingue un conjunto llamado de *tipos simples*.

Los *tipos simples*, junto con su combinación booleana, permiten a modelizar el concepto de *dominio de relación*.

Esta definición de *lenguaje de primer orden relacional*, también se denomina lenguaje relacional de Reiter.

De entre todas las posibles interpretaciones de un lenguaje de primer orden relacional L , se distingue un grupo denominado *interpretaciones relacionales*.

❖ **Definición 4.75. Interpretación relacional.** Sea $L = \{A, W\}$ un lenguaje relacional. Una *interpretación* $I = (U, K, P)$ se dice *relacional* sí y sólo sí verifica:

- a) K es el conjunto de constantes A de sobre el dominio finito U .
- b) Se extiende el predicado de igualdad $E_ = \{(d, d) \mid d \in U\}$

La condición **a)** equivale al axioma de mundo cerrado (todo lo que no es conocido es falso) y la **b)** al axioma de nombre único: $E_ (d, d^*)$ es falso para $\forall d, d^* \mid d \neq d^*$.

❖ **Definición 4.76. Base de datos lógica relacional.** Una *base de datos lógica relacional* es de una *tripleta* (L, I, RI) donde:

- a) L es un *lenguaje de primer orden relacional*.
- b) I es una *interpretación relacional*.
- c) RI es un conjunto de fórmulas bien formadas de L llamado *restricciones de integridad*.

Una base de datos lógica relacional, necesita una extensión de los predicados. Para cada predicado P n -ario (que no sea el de *igualdad* ni un *tipo simple*), L debe contener, al menos, una fbf de la forma: $\forall X_1, X_2, \dots, X_n, P(X_1, X_2, \dots, X_n) \rightarrow \tau_1(X_1) \wedge \tau_2(X_2) \wedge \dots \wedge \tau_n(X_n)$

Donde los $\tau_i \mid i = 1, 2, \dots, n$ son *tipos simples* y se le llama dominios de P . Para cada predicado P que no sea un *tipo simple*, la extensión se denomina *relación*.

4.3.2.1 La BD como teoría relacional

La formulación de una BD como una teoría se realiza de la siguiente forma.

❖ **Definición 4.77. Teoría relacional.** Sea $L = \{A, W\}$ un *lenguaje relacional*. Una *teoría de primer orden* $T \subseteq W$ es una *teoría relacional*, sí y sólo sí, se verifican las siguientes condiciones:

- a) Si c_1, c_2, \dots, c_n son constantes de A , T contiene las siguientes fórmulas bien formadas:

$$\forall x, E_ (x, c_1) \vee E_ (x, c_2) \vee \dots \vee E_ (x, c_n)$$

$$\forall i, \forall j, i \neq j, \neg E_ (c_i, c_j)$$

Fórmulas que corresponden axioma de mundo (dominio) cerrado y al axioma de nombre único, respectivamente.

- b) T contiene el *predicado de igualdad* $E_ =$ verificando:

- i. Reflexividad: $\forall x, E_ = (x, x)$

- ii. Conmutatividad: $\forall (x, y), E_=(x, y) \rightarrow E_=(y, x)$
- iii. Transitividad: $\forall (x, y, z), E_=(x, y) \wedge E_=(y, z) \rightarrow E_=(x, z)$
- iv. Principio de sustitución de términos iguales: para cada símbolo de predicado P de A , se incluirá las siguientes fbf:

$$\forall (x_1, \dots, x_n, y_1, \dots, y_n),$$

$$P(x_1, \dots, x_n) \wedge E_=(x_1, y_1), \dots, \wedge E_=(x_n, y_n) \rightarrow P(y_1, \dots, y_n)$$

c) Sea $\Delta \subseteq T$ el conjunto de fórmulas atómicas robustas de la teoría T (sin incluir a las que involucran a $E_=($), y sea la $C_p = \{(c_1, c_2, \dots, c_n) \mid P(c_1, c_2, \dots, c_n) \in \Delta\}$ el conjunto denominado extensión de P en T . Para cada predicado P de A (que no sea $E_=($), T contendrá la siguiente fbf: $\forall (x_1, x_2, \dots, x_n), \neg P(x_1, x_2, \dots, x_n)$.

Si P no tiene extensión en T , entonces $C_p = \{ \}$. Pero si no es vacío: $C_p = \{(c_1^1, \dots, c_n^1), \dots, (c_1^\tau, \dots, c_n^\tau)\}$, se deberá incluir las siguientes fórmulas bien formadas:

- i. Axioma de completitud del predicado P :

$$\forall (x_1, \dots, x_n), P(x_1, \dots, x_n) \rightarrow$$

$$(E_=(x_1, c_1^1) \wedge \dots \wedge E_=(x_n, c_n^1)) \vee \dots \vee (E_=(x_1, c_1^\tau) \wedge \dots \wedge E_=(x_n, c_n^\tau))$$
- ii. Extensión del predicado P :

$$\forall i, i \in \{1, 2, \dots, \tau\} P(c_1^i, c_2^i, \dots, c_n^i).$$

d) Las únicas fórmulas bien formadas de T son las que se construyen por puntos a), b). y c).

4.3.2.2 La BD como interpretación relacional

Sea BD_1 una instancia de una BDR, por lo que se compone de un conjunto de relaciones y restricciones de integridad. Sea U la unión de dominios de todos los atributos que aparecen en el esquema de las relaciones de la BD. La *interpretación* es un modelo para las fbf de la forma:

$$\forall X_1, X_2, \dots, X_n, P(X_1, X_2, \dots, X_n) \rightarrow \tau_1(X_1) \wedge \tau_2(X_2) \wedge \dots \wedge \tau_n(X_n)$$

Que establece el concepto de esquema de una relación.

Algunas de las consecuencias de esta forma de interpretación son:

- a) Si R es una relación BD_1 , la fbf $R(a_1, a_2, \dots, a_n)$ es cierta, sí y sólo sí, la tupla $\langle a_1, a_2, \dots, a_n \rangle \in R$.
- b) Una fbf de la forma, $\forall x, W(x)$ es verdad en la interpretación, sí y sólo sí, $\forall d \in U, W(d)$ es verdadera.

- c) Una fbf de la forma, $\exists x, W(x)$ es verdad en la interpretación, sí y sólo sí, $\exists d \in U, W(d)$ es verdadera.
- d) Expresando las restricciones de integridad de la BDR como fórmulas bien formadas, la instancia $_{BD_1}$ estará en un estado válido, sí y sólo sí, cada RI es verdadera, esto es, es un modelo para el conjunto de restricciones de integridad.

4.4 Modelo lógico difuso de bases de datos

Si bien las reglas es una excelente herramienta para representar el conocimiento humano, normalmente, este incorpora de forma natural conceptos difusos. A modo de ejemplo, dentro de los primeros resultados científicos de G. (2.2.6), podemos encontrar: “Partículas *lentas* y *poco densas*, es decir, con velocidades menores a 1 ms^{-1} y masa *en torno* a 10^{-10} kg , están por debajo del límite de sensibilidad del IS, pero son detectadas por el GDS”

Por tanto, disponer un modelo (como el modelo *lógico difuso*) que reúna la representación y manejo de reglas con información difusa, permite capturar y procesar una forma más eficaz el conocimiento.

En esta sección se describe el “modelo lógico difuso” así como una implementación de un lenguaje de programación lógico difuso: f-Prolog

En [Gal87] y [Rei84] se describe cómo utilizar la Lógica para representar y manipular bases de datos. A partir de este punto, en [Vil92], [Vil93], [Pon92], [Vil94] y [Pon94a] se desarrolla una nueva aproximación que incorpora la TCD (4.2.1), formalizándose en [Pon94b], donde se expone el concepto de “base de datos difusa deductiva” o “modelo lógico difuso de base de datos relacional”. Este modelo puede implementarse desde dos puntos de vista (como ya se vio en 4.3.2), como *como una teoría de primer orden* o como una *interpretación de un lenguaje de primer orden*. Antes profundizar en estos dos conceptos, se presentan algunas definiciones destinadas a representar y manejar la imprecisión.

❖ **Definición 4.78. Tipo grado de pertenencia o MD.** Sea $L = \{A, W\}$ un lenguaje de primer orden relacional. El alfabeto A debe de incluir un predicado unario extra MD , llamado *tipo MD* (“Membership Degree” o grado de pertenencia), que sirve para modelizar el dominio de grados de pertenencia para cualquier interpretación de L . Además, A debe de incluir el predicado binario mayor o igual: P_{\geq} .

❖ **Definición 4.79. Símbolo de tipo difuso.** Sea $L = \{A, W\}$ un lenguaje de primer orden relacional y $\tau \in A$ un símbolo. Si τ es un *tipo difuso*, sí y sólo sí:

- Existe un símbolo de tipo simple $NO P_{\tau} \in A$ que representa los nombres de las distribuciones de posibilidad que se definan sobre el dominio τ .
- Existe un símbolo predicado ternario $POS_{\tau} \in A$ que representa las distribuciones de posibilidad que se definan sobre el dominio τ .
- Existe un símbolo predicado ternario $P_{\equiv_{\tau}} \in A$ (aproximadamente igual) que representa la relación de igualdad entre las distribuciones de posibilidad.

❖ **Definición 4.80. Lenguaje relacional difuso.** Sea $L = \{A, W\}$ un lenguaje de primer orden relacional. L es un *lenguaje relacional difuso*, sí y sólo sí:

1. Incluye al *tipo MD* entre sus símbolo.
2. Incluye, al menos, un *símbolo de tipo difuso*.

4.4.1 BDRD como interpretación relacional difusa

Para representar BDRD como una *interpretación relacional difusa*, hay que tener en cuenta que no todas las interpretaciones son válidas. Por tanto se incluyen reglas de integridad en la BDRD de manera que preserven la semántica del *tipo MD* y la de los *tipos difusos*.

❖ **Definición 4.81. Interpretación relacional difusa.** Sea L un lenguaje relacional difuso y sea $I = (U, a)$ una interpretación de dominio U y función de asignación a . I es una *interpretación relacional difusa*, sí y sólo sí:

- a) I es una interpretación relacional
- b) Existe un conjunto finito $T_v \subset [0, 1]$ que verifica:
 - i. $0 \in T_v$
 - ii. $1 \in T_v$
 - iii. $T_v \subset U$
 - iv. $a(MD) = T_v$
- c) I es un modelo para la siguiente conjunto de reglas de integridad:

- i. Reglas para mantener la semántica del tipo MD:

$$\mathbf{R1.} \quad \forall (x, y) (P_{\geq}(x, y) \rightarrow MD(x) \wedge MD(y))$$

$$\mathbf{R2.} \quad \forall (x) (MD(x) \rightarrow P_{\geq}(x, x))$$

$$\mathbf{R3.} \quad \forall (x, y) (P_{\geq}(x, y) \wedge P_{\geq}(y, x) \rightarrow P_{=}(x, y))$$

$$\mathbf{R4.} \quad \forall (x, y, z) (P_{\geq}(x, y) \wedge P_{\geq}(y, z) \rightarrow P_{\geq}(x, z))$$

Estas reglas establecen que P_{\geq} es una relación de orden sobre el *tipo MD*.

- ii. Reglas para mantener la semántica de los *tipos difusos*. Para cada tipo difuso $\tau \in L$, se verifica:

$$\mathbf{R5.} \quad \forall (x, y, z) (POS_{\tau}(x, y, z) \rightarrow NOP_{\tau}(x) \wedge \tau(y) \wedge MD(z) \wedge \neg P_{=}(z, 0))$$

$$\mathbf{R6.} \quad \forall (x) (NOP_{\tau}(x) \rightarrow P_{=}(x, Und_{\tau})) \vee (\exists (y, z) (POS_{\tau}(x, y, z)))$$

$$\mathbf{R7.} \quad \forall (x) (\tau(x) \rightarrow POS_{\tau}(x, x, 1)) \vee (\neg \exists (y) (POS_{\tau}(x, y, 1) \wedge \neg P_{=}(x, y)))$$

- iii. Reglas para mantener la semántica del operador de igualdad difuso (aproximadamente igual) $P_{\approx\tau}$.

$$\mathbf{R8.} \quad \forall (x, y, z) (P_{\approx\tau}(x, y, z) \rightarrow NOP_{\tau}(x) \wedge NOP_{\tau}(y) \wedge MD(z))$$

$$\mathbf{R9.} \quad \forall (x, y, z, w), (P_{\approx\tau}(x, y, z) \wedge P_{\geq}(z, w) \rightarrow P_{\approx\tau}(x, y, w))$$

$$\mathbf{R10.} \quad \forall (x, y), (P_{\equiv\tau}(x, y, z) \wedge P_{\equiv}(x, y) \rightarrow P_{\equiv}(z, 1))$$

$$\mathbf{R11.} \quad \forall (x, y, z), (P_{\equiv\tau}(x, y, z) \rightarrow P_{\equiv\tau}(y, x, z))$$

Con estas definiciones realizadas, es posible formular una *BDRD como interpretación relacional difusa* como sigue:

❖ **Definición 4.82. BDRD como interpretación relacional difusa.** Se define *BDRD como interpretación relacional difusa* como una tripleta (L, R, RI) donde:

- a) L es un lenguaje relacional difuso.
- b) R es una interpretación relacional difusa.
- c) RI es un conjunto de restricciones de integridad, que incluye para cada predicado P n -ario de L la siguiente regla:
- d) $\forall (x_1, x_2, \dots, x_n) (P(x_1, x_2, \dots, x_n) \rightarrow \tau_1(x_1) \wedge \tau_2(x_2) \wedge \dots \wedge \tau_n(x_n))$

Donde τ_i son tipos simples.

4.4.2 BDRD como teoría relacional difusa

Una BDRD vista como una teoría relacional difusa, se obtiene extendiendo la definición sin imprecisión (4.3.2.1 *La BD como teoría relacional*) y aplicando parte de las reglas definidas para la interpretación relacional difusa (Definición 4.81)

❖ **Definición 4.83. Teoría relacional difusa.** Sea $L = (A, W)$ un lenguaje relacional difuso, el conjunto $T \subseteq W$ es una teoría relacional difusa sí, y sólo sí:

- a) T es una teoría relacional según la Definición 4.77.
- b) Existe una interpretación relacional difusa, que es el único modelo para T .
- c) Incluye todas las reglas de integridad de los tipos difusos de L (R1 a R8).

❖ **Definición 4.84. BDRD como teoría relacional difusa.** Una BDRD como teoría relacional difusa es una tripleta (L, T, RI) en la que:

- a) L es un lenguaje relacional difuso.
- b) T es una teoría relacional difusa.
- c) RI es un conjunto específico de reglas de integridad.

4.4.3 f-Prolog

Prolog (4.3.1.4.1) no permite la representación ni el manejo de información precisa. Una de las implementaciones que resuelven este problema es el *f-Prolog* ([Li90]) que sustituye la Lógica binaria clásica por Lógica difusa.

Un programa en *f-Prolog* difiere de un programa en Prolog, en el grado de verdad que algunos hechos llevan asociado y en el grado de implicación que se asocia a las reglas. Un programa en *f-Prolog* consta de tres componentes:

❖ **Definición 4.85. f-hecho.** $P(t_1, t_2, \dots, t_n) : -[f]$ - que establece que el valor de verdad del hecho $P(t_1, t_2, \dots, t_n)$ es f .

- ❖ **Definición 4.86. f-regla.** $P : -[f] - Q_1, Q_2, \dots, Q_n$. Sea $tv(Q_i), i = 1, 2, \dots, n$ el grado de verdad de cada Q_i . Entonces el valor α del conjunto de condiciones Q_i se calcula como:

$$\alpha = \begin{cases} \min \{tv(Q_i) \mid i = 1, 2, \dots, n\} & \text{si } n > 0 \\ 1 & \text{si } n = 0 \end{cases}$$

- ❖ **Definición 4.87. f-objetivo.** Se denomina así a una expresión de la forma $? - [f] - Q_1, Q_2, \dots, Q_n$, siendo f una constante, o bien a $? - [F] - Q_1, Q_2, \dots, Q_n$, siendo F una variable.

4.5 Modelo integrado de base de datos relacional difusa y deductiva

Unir las capacidades deductivas del *modelo lógico difuso* con las de representación y manipulación de la imprecisión del *modelo relacional difuso*, permite plasmar de una forma más eficaz el conocimiento que incluya elementos difusos (como el procedente de expertos) así como obtener nueva información.

El modelo descrito en [Bla00] y [Bla01], desarrolla un método para la deducción de información a partir de datos imprecisos. Para ello, parte del modelo relacional difuso GEFRED (4.2.2.5) y lo extiende con fundamentos de Prolog (4.3.1.4.1) y Datalog (4.3.1.4.2).

En esta sección se revisarán los componentes claves de este modelo: *relaciones extensivas e intensivas difusas*, *generalización* y *flexibilización* de reglas, *regla generalizada difusa* (RGD), algoritmos de *deducción* aplicables, para concluir con una descripción de la una implementación computacional: FREDDI.

4.5.1 Relación extensiva difusa

Una relación extensiva es una enumeración exhaustiva de combinaciones de valores de dominio de un atributo. En este modelo, la representación de la información (difusa o no) se lleva a cabo a través de la *relación difusa generalizada* (RDG *Definición 4.41*). Partiendo del concepto de relación extensiva, una relación extensiva difusa no es más que una RDG donde el *grado de compatibilidad* es uno. Más formalmente.

- ❖ **Definición 4.88. Relación extensiva difusa.** Es una RDG donde el *grado de compatibilidad* tiene el valor uno.

$$\begin{aligned} H &= \{(A_1 : D_1 [C_1]), (A_2 : D_2 [C_2]), \dots, (A_n : D_n [C_n])\} \\ B &= \{(A_1 : \tilde{d}_{i1} [1]), (A_1 : \tilde{d}_{i2} [1]), \dots, (A_1 : \tilde{d}_{in} [1])\} \quad \text{Eq. (4.35)} \end{aligned}$$

4.5.2 Relación intensiva difusa

Se denomina *relación intensiva* (o relación implícita) a la información que *no* se encuentra almacenada de forma explícita en forma de relación. Para transformar una *relación intensiva extensiva*, es necesario calcular (mediante un conjunto de normas o reglas) los valores concretos que instancian la relación.

La extensión difusa de una *relación intensiva*, se basa de nuevo en una RDG.

- ❖ **Definición 4.89. Relación intensiva difusa.** Una relación intensiva difusa asociada a una RDG R , es un par (H, I) . Donde H es el conjunto cabecera de R y el conjunto I es llamado *generador de instancia*. Este conjunto se compone de una serie de reglas orientadas a datos difusos, los cuales permiten el cálculo de la instancia de la relación.

4.5.3 Generalización de las reglas de la Lógica de primer orden

El modelo parte de la definición de regla tipo Prolog (4.3.1.4.1) y de la definición de una regla en forma clausular (4.3.1.3.1) para ampliar la regla de la Lógica de primer orden.

- ❖ **Definición 4.90. Regla generalizada tipo Prolog.** La generalización de las reglas tipo Prolog es la siguiente:

$$P(X_1, X_2, \dots, X_n) \leftarrow Q_1(Y_{1,1}, Y_{1,2}, \dots, Y_{1,n_1}) \wedge Q_2(Y_{2,1}, Y_{2,2}, \dots, Y_{2,n_2}) \wedge \dots \wedge Q_m(Y_{m,1}, Y_{m,2}, \dots, Y_{m,n_m}) \quad \text{Eq. (4.36)}$$

Donde se verifica que:

$$\text{a) } (\forall i \in \{1, 2, \dots, n\}) (\exists j \in \{1, 2, \dots, m\}) \wedge (\exists k \in \{1, 2, \dots, n_m\}) \mid X_i = Y_{j,k}$$

$$\text{b) } (\forall i \in \{1, 2, \dots, m\}) (\forall j \in \{1, 2, \dots, n_i\}) (\exists k \in \{1, 2, \dots, n\} \mid X_i = Y_{j,k}) \vee \vee ((\exists r \in \{1, 2, \dots, m\}), (\exists s \in \{1, 2, \dots, n_r\}) \mid ((r \neq i) \wedge (Y_{i,j} = Y_{r,s})))$$

Estos es, en una regla generalizada, toda variable que aparece en la cabeza de la regla aparece también (está acoplada) en el cuerpo de la regla y que toda variable que aparece en el cuerpo está acoplada a otra variable o del cuerpo o de la cabeza de la regla.

Se dice que existe un acoplamiento entre variables de predicados, si las variables son iguales, o dicho de otro modo, si el predicado de igualdad aplicado sobre estas variables es cierto.

4.5.4 Flexibilización de reglas generalizadas

El relajamiento de las reglas generalizadas se centra en la flexibilización del acoplamiento entre variables. Dado que el dominio de las variables es un DDG, el modelo establece un mecanismo de comparación entre variables con dominio impreciso mediante el uso de un CDG (4.2.2.5.1.2 *Comparadores difusos generalizados*). La flexibilización se realiza en dos frentes: *acoplamiento entre las variables del cuerpo* de la regla y entre las *variables de la cabeza y el cuerpo*.

En la flexibilización del *acoplamiento entre las variables del cuerpo* de la regla, es necesario definir un CDG $=_{\beta}$ (como el utilizado en GEFRED, ver [Med94b] y [Gal99]) que desempeña el papel de operación de igualdad para el acoplamiento entre variables. Así, si X e Y son dos variables del cuerpo de una regla, el acoplamiento se realiza mediante el operador $X =_{\beta} Y$, siendo $\beta \in [0, 1]$ el grado de acoplamiento (GA) entre X e Y .

Para calcular el GA de dos predicados que tienen n variables en común, es necesario calcular la función $\Phi = f(\beta_1, \beta_2, \dots, \beta_n)$, siendo $\beta_k \mid k = 1, 2, \dots, n$ el GA de las dos variables acopladas (unificadas).

En la flexibilización del *acoplamiento entre las variables del cuerpo y la cabeza* de la regla, se parte del mecanismo Prolog para la obtención de hechos de un predicado intensivo (el predicado de cabeza) a partir de los predicados que lo definen (predicados de cuerpo de la regla). En el caso clásico (prueba de consistencia), este mecanismo consiste en comprobar la consistencia de un hecho *determinado* respecto a los hechos almacenados (o base de hechos BH). Para ello se instancian las variables (se obtienen todos las posibles combinaciones de valores) del cuerpo de la regla y a continuación se instancian los predicados de la regla a partir de estas variables instanciadas. Si el resultado del proceso es un único hecho, la consistencia del hecho queda probada, en otro caso (no existe ningún hecho), el hecho no puede ser deducido a partir de la BH.

En el modelo, se flexibiliza la *prueba de consistencia*, por lo que se pasa de verificar la consistencia de un hecho respecto a la BH, a buscar el grado de “cercanía” de un conjunto de hechos al hecho a probar.

4.5.5 Regla generalizada difusa

Formalmente, la flexibilización de la regla generalizada de tipo Prolog se define como:

❖ **Definición 4.91. Regla generalizada difusa.** Sea P, Q_1, Q_2, \dots, Q_m predicados con aridad n, n_1, n_2, \dots, n_m respectivamente, y sea $X_i, i = 1, 2, \dots, n$ y $Y_{j,k}, j = 1, 2, \dots, m$ y $k = 1, 2, \dots, n_m$ variables que toman valores en los DDG D_{X_i} y $D_{Y_{j,k}}$. Se define *regla generalizada difusa* (RGD) a una regla con la siguiente forma:

$$P(X_1, X_2, \dots, X_n) \leftarrow Q_1(Y_{1,1}, Y_{1,2}, \dots, Y_{1,n_1}) \wedge Q_2(Y_{2,1}, Y_{2,2}, \dots, Y_{2,n_2}) \wedge \dots \wedge Q_m(Y_{m,1}, Y_{m,2}, \dots, Y_{m,n_m}) \wedge \Psi$$

Eq. (4.37)

Siendo $P(X_1, X_2, \dots, X_n) \leftarrow Q_1(Y_{1,1}, Y_{1,2}, \dots, Y_{1,n_1}) \wedge Q_2(Y_{2,1}, Y_{2,2}, \dots, Y_{2,n_2}) \wedge \dots \wedge Q_m(Y_{m,1}, Y_{m,2}, \dots, Y_{m,n_m})$

una regla generalizada tipo Prolog, y Ψ la siguiente función:

$$\Psi \equiv \bigwedge (X_i =_{\alpha_{i,j,k}} Y_{j,k}) \wedge (Y_{j,k} =_{\beta_{j,k,l,p}} Y_{l,p}) \wedge \phi_{j,k,l,p}(\Theta_{j,k,l,p}^{\theta}(Y_{j,k}, Y_{l,p}), \gamma_{j,k,l,p})$$

Eq. (4.38)

Donde:

- a) $\bigwedge A_i$ indica la expresión $A_1 \wedge A_2 \wedge \dots \wedge A_n$.
- b) Las expresiones $(X_i =_{\alpha_{i,j,k}} Y_{j,k})$ e $(Y_{j,k} =_{\beta_{j,k,l,p}} Y_{l,p})$ indican el acoplamiento entre variables de cabeza-cuerpo y variables del cuerpo (respectivamente) mediante el CDG $=_{\beta}$ basado en el comparador clásico igualdad. Siendo $\beta = \alpha_{i,j,k}$ y $\beta = \gamma_{j,k,l,p}$ el GA entre variables.
- c) La función $\phi_{j,k,l,p}$ indica si una comparación explícita entre variables en el cuerpo de la regla supera determinado umbral. Está basada en la función umbral clásica ($>$ ó \geq) y tiene dos argumentos. El primero, $\Theta_{j,k,l,p}^{\theta}(Y_{j,k}, Y_{l,p})$, representa la comparación entre variables y se define como un CDG basado en el comparador

clásico θ , que puede representar alguno de estos comparadores: $\{=, \neq, >, \geq, <, \leq\}$.

El segundo, $\gamma_{j,k,l,p}$, es el GA de la comparación entre las variables $Y_{j,k}$ e $Y_{l,p}$.

d) Toda variable de cabeza está ligada con al menos una variable del cuerpo de la regla:

$$(\forall i \in \{1, 2, \dots, n\}), (\exists j \in \{1, 2, \dots, m\}) \wedge (\exists k \in \{1, 2, \dots, n_m\}) | X_i =_{\alpha_{i,j,k}} Y_{j,k}$$

e) Toda variable de cuerpo está ligada con, al menos, otra variable del cuerpo o de la cabeza de la regla. Es decir:

$$\left((\forall i \in \{1, 2, \dots, m\}), (\forall j \in \{1, 2, \dots, n_i\}) \left(\left((\exists k \in \{1, 2, \dots, n\}) | (X_k =_{\alpha_{k,j}} Y_{i,j}) \in \Psi \right) \vee \left((\exists r \in \{1, 2, \dots, m\}), (\exists s \in \{1, 2, \dots, n_r\}) | ((r \neq i) \vee (j \neq s)) \wedge (Y_{i,j} =_{\beta_{i,j,r,s}} Y_{r,s}) \in \Psi \right) \right) \right)$$

El cálculo de la función Ψ se puede interpretar como el cálculo del alfa-corte (*Definición 4.19*) del conjunto de resultados, por lo que se simplifica su definición:

$$\Psi \equiv \bigwedge_{i,j,k} \geq (\Theta_{i,j,k}^= (X_i, Y_{j,k}), \alpha_{i,j,k}) \wedge \geq (\Theta_{j,k,l,p}^= (Y_{j,k}, Y_{l,p}), \beta_{j,k,l,p}) \wedge \bigwedge_{j,k,l,p} \phi_{j,k,l,p}^{\theta} (\Theta_{j,k,l,p}^{\theta} (Y_{j,k}, Y_{l,p}), \gamma_{j,k,l,p}) \quad \text{Eq. (4.39)}$$

Esta representación de regla induce dos tipos diferentes de comparaciones entre variables.

a) Comparaciones de variables implícitas. Son aquellas comparaciones que, mediante el CDG igualdad, proporcionan la semántica del enlace entre predicados de la regla cabeza-cuerpo o cuerpo-cuerpo: $\Theta_{i,j,k}^= (X_i, Y_{j,k})$ y $\Theta_{j,k,l,p}^= (Y_{j,k}, Y_{l,p})$.

b) Comparaciones de variables explícitas. Son las comparaciones difusas generalizadas adicionales que establecen la semántica extra en el cuerpo de la regla: $\Theta_{j,k,l,p}^{\theta} (Y_{j,k}, Y_{l,p})$

4.5.5.1 Tipos de reglas generalizadas difusas

Existen determinadas condiciones entre los predicados de la regla, que permiten identificar diferentes tipos de regla generalizada difusa (RGD):

❖ **Definición 4.92. RGD con profundidad 1.** Sea P un predicado definido mediante una RGD. Si cada uno de los predicados del cuerpo de la regla no se define como una RGD, se dice que P es una RGD de profundidad 1.

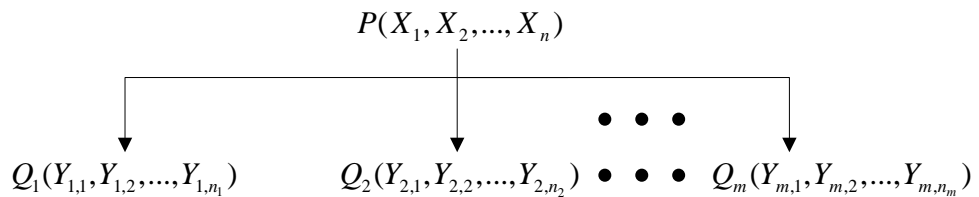


Ilustración 54. RGD con profundidad 1.

Es decir, una RGD con profundidad 1, es una regla en la que todo predicado se corresponde con una relación extensiva.

- ❖ **Definición 4.93. RGD con profundidad > 1.** Sea P un predicado definido mediante una RGD. Si existe algún predicado del cuerpo de la regla definido como un RGD, se dice que P es una RGD de profundidad > 1.

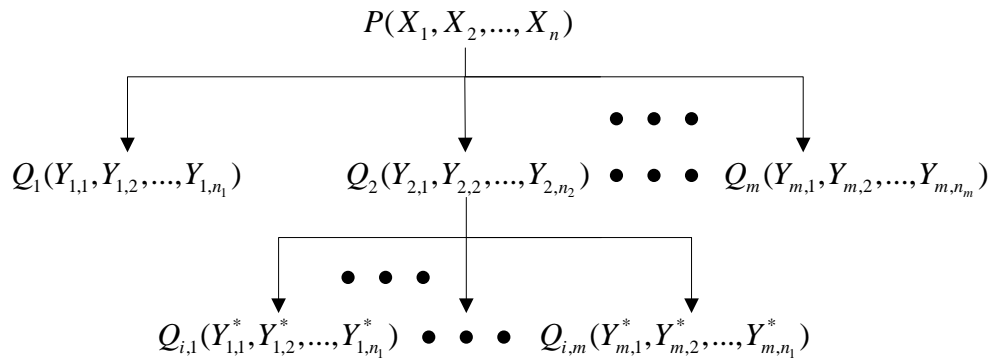


Ilustración 55. RGD con profundidad > 1.

- ❖ **Definición 4.94. RGD no recursiva de profundidad > 1.** Sea P un predicado definido mediante una RGD y profundidad > 1. Si $\nexists i \in [1, m] \mid Q_i \equiv P$, se dice que P es una RGD no recursiva de profundidad > 1.
- ❖ **Definición 4.95. RGD recursiva profundidad > 1.** Sea P un predicado definido mediante una RGD y profundidad > 1. Si $\exists i \in [1, m] \mid Q_i \equiv P$, se dice que P es una RGD recursiva de profundidad > 1.

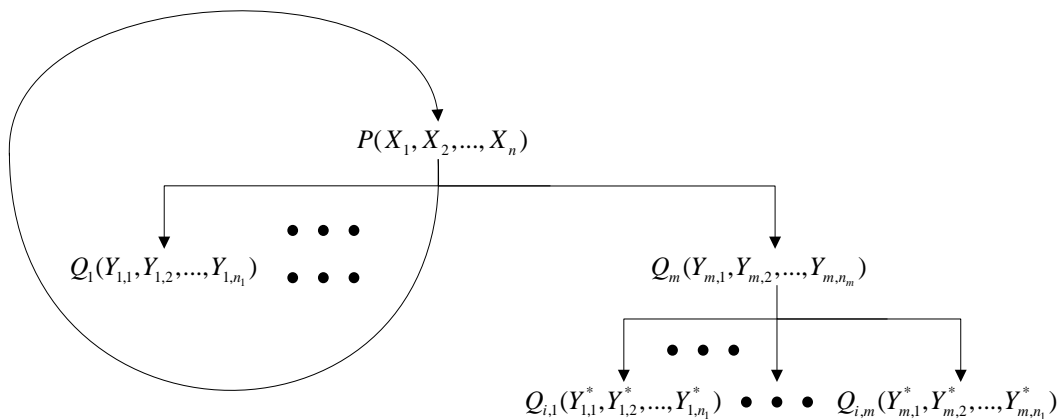


Ilustración 56. RGD recursiva con profundidad > 1.

4.5.6 Deducción con reglas generalizadas difusas

El proceso de utilizar una RGD para obtener el conjunto de hechos de un predicado que son consistentes con una BH, se puede realizar de dos formas diferentes:

- a) Sin considerar el grado de consistencia del hecho con la BH
- b) Considerando el grado de consistencia

En ambos casos, el cálculo de la consistencia se puede conseguir estableciendo un umbral, de modo que si el grado del hecho con la BH supera dicho umbral, el hecho es consistente. Pero cada caso origina un tipo deducción diferente, como se detalla a continuación.

4.5.6.1 Deducción sin grado de acoplamiento

En este caso, la obtención de hechos y la comprobación de su consistencia respecto a la base de hechos se considera un proceso puramente lógico, basado en la verdad del hecho. Así, cada comparación flexible entre variables se evalúa como cierta si los grados de acoplamiento superan un umbral y como falsa en caso contrario.

Se utiliza el *principio de resolución de Robinson* (4.3.1.4.1 Prolog) como método de deducción. En este proceso deductivo, el orden de aplicación de los predicados es fijado su aparición en el cuerpo de la regla, pero en el caso de una RGD esto no es deseable. Además, no se establece un orden de aplicación de los hechos, lo que complica su automatización. Por ello, se han propuestos varios refinamientos: la *resolución semántica* ([Sla67]), la *resolución cerradura* ([Boy71]), y la *resolución lineal* ([Lov70] y [Luc70]).

En la *resolución cerradura*, se establece un orden en los predicados para aplicar el *principio de resolución*. Dicho orden varía en cada iteración del algoritmo para adaptarse a los cambios. Esta reordenación permitirá acelerar el algoritmo con reglas no recursivas y garantizará la convergencia con las recursivas.

La deducción sin grado de acoplamiento de una RGD necesita de una generalización (extensión) de la base de reglas (BR) inicial. En este proceso, se añade a la BR los hechos que describen el comportamiento de los valores de los dominios generalizados difusos. Estos hechos no dependen de la instanciación de los predicados, sino únicamente de la estructura del DDG. Suponiendo que t es un posible valor que puede ser instanciado en un predicado de la RGD, se añade a la BR $\geq (\Theta^=(t, t), \alpha)$, además de incluir hechos que calculan su similitud con otros valores del dominio. Más formalmente.

❖ **Definición 4.96. Base de reglas generalizada para deducción sin grado de acoplamiento.** Sea R una RGD con la simplificación de la Eq. (4.39). La base de reglas generalizada para aplicar resolución sin grado de acoplamiento se define como:

$$\Gamma = \left\{ \begin{array}{l} \neg Q_1(Y_{1,1}, Y_{1,2}, \dots, Y_{1,n_1}) \vee \neg Q_2(Y_{2,1}, Y_{2,2}, \dots, Y_{2,n_2}) \vee \\ \vee \dots \vee \neg Q_m(Y_{m,1}, Y_{m,2}, \dots, Y_{m,n_m}) \vee \bar{\Psi} \vee P(X_1, X_2, \dots, X_n) \end{array} \right\} \cup M \quad \text{Eq. (4.40)}$$

Donde

$$\bar{\Psi} \equiv \bigvee \neg \geq (\Theta^=_{i,j,k}(X_i, Y_{j,k}), \alpha_{i,j,k}) \bigvee \neg \geq (\Theta^=_{j,k,l,p}(Y_{j,k}, Y_{l,p}), \beta_{j,k,l,p}) \bigvee \bigvee \neg \phi_{j,k,l,p}(\Theta^\theta_{j,k,l,p}(Y_{j,k}, Y_{l,p}), \gamma_{j,k,l,p}) \quad \text{Eq. (4.41)}$$

$$M \equiv \left\{ \left\{ \geq (\Theta^= (x, x), \alpha) \mid (w \in Dom(X_i)) \wedge (i \in [1, n]) \right\} \cup \left\{ \geq (\Theta^= (x, y), \beta) \mid (x, y \in Dom(X_i)) \wedge (i \in [1, n]) \wedge (x \neq y) \wedge (\alpha_{x,y} \geq \beta) \right\} \cup \left\{ \neg \geq (\Theta^= (x, y), \beta) \mid (x, y \in Dom(X_i)) \wedge (i \in [1, n]) \wedge (x \neq y) \wedge (\alpha_{x,y} \geq \beta) \right\} \right\}$$

Eq. (4.42)

Siendo

- a) $\Theta_{i,j,k}^= (X_i, Y_{j,k})$ el GA entre de la comparación de variables implícitas (4.5.5) unificadas entre la cabeza y el cuerpo de la regla.
- b) $\Theta_{j,k,l,p}^= (Y_{j,k}, Y_{l,p})$ el GA entre de la comparación de variables implícitas ligadas en cuerpo de la regla.
- c) $\Theta_{j,k,l,p}^\theta (Y_{j,k}, Y_{l,p})$ el GA entre los valores de las variables del cuerpo de la regla que se encuentran ligadas entre sí por *comparaciones de variables explícitas* (4.5.5).

El conjunto M se define como el conjunto de hechos adicionales que completan la descripción del mundo al que representa la base de hechos. Se forma por la unión de tres conjuntos:

- a) El conjunto de pares de constantes del DDG, donde cada elemento del par es la misma constante (propiedad reflexiva).
- b) El conjunto de pares de constantes del DDG cuya similitud basada en el CDG $\Theta^=$ supera un umbral β (se dice que son β -similares).
- c) El conjunto de pares de constantes del DDG cuya similitud basada en el CDG $\Theta^=$ no supera un umbral β (se dice que no son β -similares).

4.5.6.2 Deducción con grado de acoplamiento

La deducción con GA de una RGD, es similar al caso sin GA, pero computando el GA de los diferentes acoplamientos acaecidos en el proceso de deducción: los procedentes del acoplamiento ente los valores de las variables y los originados del acoplamiento de los predicados del cuerpo. Es decir, se pretende calcular, dado un predicado P definido mediante una RGD, hechos de la forma $(P(a_1, a_2, \dots, a_n), \gamma)$ donde γ recoge el GA en el que la combinación de constantes (a_1, a_2, \dots, a_n) satisface el grado P .

Este cálculo, requiere de una transformación a cada predicado y hecho para incorporar el GA.

$$\begin{aligned} P(X_1, X_2, \dots, X_{n_p}) &\Rightarrow \tilde{P}(X_1, X_2, \dots, X_{n_p}, \gamma_p) \\ Q(x_1, x_2, \dots, x_{n_p}) &\Rightarrow \tilde{Q}(x_1, x_2, \dots, x_{n_p}, 1) \end{aligned} \quad \text{Eq. (4.43)}$$

Así, cada predicado P tendrá un GA γ_p y cada hecho Q un GA de 1 (ya que inicialmente se considera cierto).

Por tanto, la nueva BR para este tipo de deducción se define como:

❖ **Definición 4.97. Base de reglas generalizada para deducción con grado de acoplamiento.** Sea R una RGD con la simplificación de la [Eq. \(4.39\)](#). La *base de reglas generalizada* para aplicar resolución *con grado de acoplamiento* se define como:

$$\tilde{\Gamma} = \left\{ \begin{array}{l} \neg \tilde{Q}_1(Y_{1,1}, Y_{1,2}, \dots, Y_{1,n_1}, \gamma_{\tilde{Q}_1}^-) \vee \neg \tilde{Q}_2(Y_{2,1}, Y_{2,2}, \dots, Y_{1,n_2}, \gamma_{\tilde{Q}_2}^-) \vee \\ \vee \dots \vee \neg \tilde{Q}_m(Y_{m,1}, Y_{m,2}, \dots, Y_{m,n_m}, \gamma_{\tilde{Q}_m}^-) \vee \bar{\Psi} \vee \tilde{P}(X_1, X_2, \dots, X_n, \gamma_p^-) \end{array} \right\} \cup M$$

Eq. (4.44)

Donde:

$$\bar{\Psi} \equiv \bigvee \neg \geq (\Theta_{i,j,k}^- (X_i, Y_{j,k}), \alpha_{i,j,k}) \bigvee \neg \geq (\Theta_{j,k,l,p}^- (Y_{j,k}, Y_{l,p}), \beta_{j,k,l,p}) \bigvee \bigvee \neg \phi_{j,k,l,p} (\Theta_{j,k,l,p}^\theta (Y_{j,k}, Y_{l,p}), \gamma_{j,k,l,p})$$

Eq. (4.45)

$$\gamma_p = \Phi \left(\begin{array}{l} \Theta_{i,j,k}^- (X_i, Y_{j,k}), \dots, \Theta_{i,j,k}^- (X_i, Y_{j,k}), \Theta_{j,k,l,p}^- (Y_{j,k}, Y_{l,p}), \dots, \Theta_{j,k,l,p}^- (Y_{j,k}, Y_{l,p}), \\ \Theta_{j,k,l,p}^\theta (Y_{j,k}, Y_{l,p}), \dots, \Theta_{j,k,l,p}^\theta (Y_{j,k}, Y_{l,p}), \gamma_{\tilde{Q}_1}, \gamma_{\tilde{Q}_2}, \dots, \gamma_{\tilde{Q}_m} \end{array} \right)$$

Eq. (4.46)

Siendo:

- a) $\Theta_{i,j,k}^- (X_i, Y_{j,k})$ el GA entre los valores de las variables unificadas entre la cabeza y el cuerpo de la regla.
- b) $\Theta_{j,k,l,p}^- (Y_{j,k}, Y_{l,p})$ el GA entre los valores de las variables ligadas en cuerpo de la regla.
- c) $\Theta_{j,k,l,p}^\theta (Y_{j,k}, Y_{l,p})$ el GA entre los valores de las variables del cuerpo de la regla que se encuentran ligadas entre sí por comparaciones de variables explícitas.
- d) $\gamma_{\tilde{Q}_i}, i = 1, 2, \dots, m$ GA entre de un hecho del predicado \tilde{Q}_i con el propio predicado.
- e) M el conjunto definido en la ecuación [Eq. \(4.42\)](#).

4.5.7 Arquitectura de FREDDI

En [\[Med96\]](#), [\[Med97\]](#), [\[Pon96\]](#) y [\[Pon97\]](#) se presenta una implementación de una arquitectura relacional de una base de datos difusa deductiva. La arquitectura se desarrolla en un SGBDR comercial (Oracle ©) que funciona como base sobre la que se monta un conjunto de módulos que realizan las tareas deductivas y de manejo de información difusa. Los principales componentes de FREDDI se muestran a continuación:

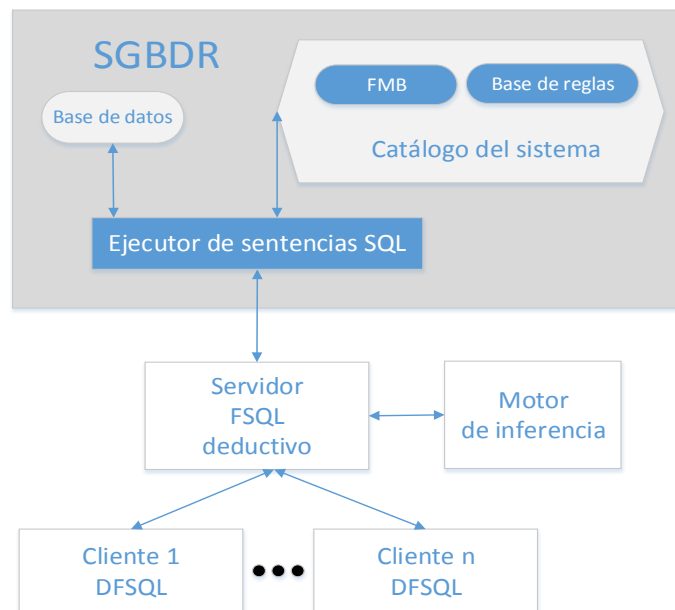


Ilustración 57. Arquitectura de FREDDI.

- a) **SGBDR.** Es el sistema relacional que soporta a FREDDI. Proporciona los recursos de gestión de la BDR y herramientas básicas para programar aplicaciones SQL. Sobre la sintaxis de este SQL, se realizan las extensiones deductivas y difusas de FREDDI.
- b) **Base de datos.** Almacena toda la información extensiva, difusa o no, dentro de un esquema relacional.
- c) **FMB(Fuzzy Meta-knowledge).** La base de meta-conocimiento difusa es una extensión del catálogo del SGBDR que almacena toda la información acerca de las estructuras difusas, datos, reglas y definiciones de FREDDI.
- d) **Base de reglas.** Este es el módulo donde se almacenan las tablas intensivas.
- e) **Ejecutor de sentencias.** Es el encargado de convertir las sentencias en SQL en operaciones sobre las relaciones para obtener una relación resultante.
- f) **Servidor FSQL deductivo.** En este módulo se procesan las consultas deductivas y/o difusas que se realicen desde los clientes, traduciéndolas al SQL del SGBDR.
- g) **Motor de inferencia.** Utiliza las reglas almacenadas en la base de reglas y la información de la base de datos para deducir nuevos elementos de información implícita.
- h) **Cliente DFSQL.** Es la interfaz entre el usuario y el servidor de DFSQL. Es posible conectar más de un cliente a la vez al mismo servidor.

4.6 Resumen y conclusiones

Se ha mostrado en este capítulo diferentes modelos de bases de datos que abordan la representación y el manejo de reglas y de información difusa. En concreto se han revisado dos ramas de modelos teóricos que fundamentan el modelo de datos de GDB: el *modelo relacional* y el *modelo lógico*.

Por un lado, se ha definido el modelo relacional (con su implementación comercial @Oracle), que almacena de forma eficiente la información en forma de tablas (relaciones) dentro de una base de datos. Posteriormente se han presentado los aspectos más relevantes de la teoría de conjuntos difusos y cómo representa y opera con la información. A continuación, se ha detallado como extender el modelo relacional para manejar imprecisión a través de un conjunto de modelos agrupados en el denominado *modelo relacional difuso*, siendo GEFRED uno de estos modelos y *FIRST* su implementación.

Por otro lado, se han revisado los aspectos más relevantes de la *Lógica de primer orden*, y cómo es capaz de representar y deducir nueva información. Se ha presentado un lenguaje de programación lógica (Prolog) que sirve como base para la definición de reglas en GDB. Así mismo, se ha mostrado cómo representar una base de datos relacional mediante *Lógica* (con una implementación *Datalog*). Posteriormente, se detalló cómo extender la *Lógica* para manejar imprecisión (con su implementación *f-Prolog*).

Las dos ramas cristalizan en el *modelo integrado de base de datos relacional difusa y deductiva* (con su implementación FREDDI) que permite representar y manipular información imprecisa o clásica, así como deducir nueva información (a partir de la existente) mediante el uso de reglas.

La evolución de los modelos y sus implementaciones se muestran en la siguiente ilustración:

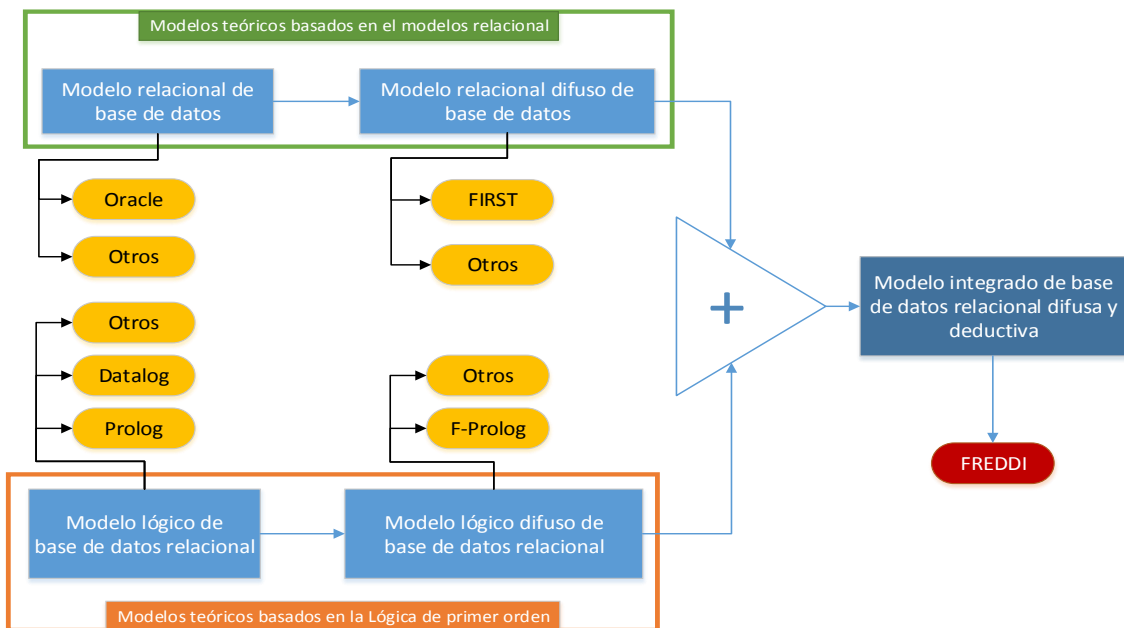


Ilustración 58. Modelos teóricos de base de datos e implementaciones antecedentes de GDB.

Aunque el *modelo integrado* ofrece un marco teórico sólido, no está adaptado para manejar de forma práctica datos reales de GIADA. Por ello, es necesario ampliarlo y dotarlo de los recursos apropiados hasta obtener un modelo teórico *extendido*, proceso que se detalla en el siguiente capítulo.

5 Modelo de base de datos de GDB

El modelo teórico básico empleado en este memoria se encuentra descrito [Bla01], donde se realiza una generalización del *Modelo integrado de base de datos relacional difusa y deductiva* (4.5), construyendo un *Modelo extendido de base de datos relacional difusa y deductiva* y proporcionando una implementación para el SGBDR Oracle©.

Por un lado, este modelo de BD de GDB hereda el planteamiento teórico de los modelos previos para la representación y manipulación de información difusa mediante relaciones (*modelo GEFRED 4.2.2.5*), así como la capacidad de operar con reglas y la posibilidad de deducir nueva información (*modelo integrado 4.5*). Por otro lado, también se beneficia de la implementación de los modelos realizada en FIRST (4.2.2.5.1) y FREDDI (4.5.7).

El objetivo de GDB es construir un sistema que permita a un usuario el manejo y representación de información clásica de los datos procedentes de G., para, posteriormente, enriquecerlos con información difusa y como paso adicional, deducir nuevo conocimiento. Para llevar a cabo esta labor, se retoma el modelo y la implementación de [Bla01], se extiende y adapta a las necesidades del tratamiento de datos requeridos por el instrumento, a fin de construir el *modelo de base de datos de GDB*. Asimismo, se crea una interfaz GDB-GUI (que se describirá en el capítulo 6) que permite a un usuario acceder a la implementación del modelo y a los datos recogidos de G.

En este capítulo se desgana el modelo de base de datos de GDB. En concreto, cómo representa y maneja la información difusa y deductiva. Se analizarán diversos aspectos relacionados con la aplicación práctica del modelo: constantes, conjunto de reglas y semántica de la medida de calidad. Así mismo se detalla un ejemplo de uso del modelo y se describe su arquitectura.

5.1 Representación y manipulación de información

El modelo de GDB se implementa en un SGBDR, dado que es este el que proporciona las herramientas básicas para la representación y manipulación de los datos: el lenguaje SQL y el catálogo.

Típicamente, un SGBDR dispone de dos sub-lenguajes para representar y manipular información:

- a) DDL (Data Manipulation Lenguaje). *Lenguaje de definición de datos* (DDL), que permite crear y modificar la estructura de los objetos de la BD.
- b) DML (Data Manipulation Lenguaje). *Lenguaje de manipulación de datos* (DDL), con el que se pueden recuperar, almacenar, modificar, borrar, insertar y actualizar los datos de la BD.

El lenguaje SQL reúne en su sintaxis estos dos sublenguajes.

La implementación del modelo de GDB, utiliza la extensión de SQL definida en FIRST para manejar imprecisión (“Fuzzy SQL” o FSQL), así como la extensión de FREDDI para trabajar con imprecisión y deducción (“Deductive Fuzzy SQL” DFSQL).

Otro elemento importante de un SGDBR es el catálogo. Inicialmente, almacena datos acerca de datos (esto es, meta-datos): usuarios, permisos, estadísticas, etcétera. FIRST realiza una extensión para soportar la información difusa (“Fuzzy Metaknowledge Base” o FMB) que a su vez es ampliada en FREDDI con la *base de reglas* para manejar la deducción. El modelo de GDB hereda ambas extensiones.

En esta sección se definirán los aspectos concretos de la implementación que dan soporte al manejo y representación de la imprecisión y de la deducción. Es particular, cómo se almacena la

información en un SGBDR, definición de la sintaxis semántica de FSQL y DFSQL y la descripción de la arquitectura del FMB.

5.1.1 Representación y manejo de información difusa

Se presenta a continuación cómo se representarán los *tipos de datos difusos*, así como el cálculo de la imprecisión cuando se involucran comparadores difusos generalizados en las consultas, para concluir con la arquitectura del FMB y la definición del FSQL.

5.1.1.1 Tipos de datos difusos

El modelo de GDB implementa los *tipos de datos difusos* de FIRST, y por tanto también los tipos de dato clásicos. Como ya se indicó en [4.2.2.5.1.1](#), FIRST utiliza tres tipos distintos para representar un DDG ([4.5.2](#)), además de los tipos *unknown*, *undefined* y *null*.

- a) **Difuso tipo 1.** Tipos de datos precisos, consultados de forma clásica o imprecisa.
- b) **Difuso tipo 2.** Datos imprecisos con referencial ordenado, consultados de forma imprecisa.
- c) **Difuso tipo 3.** Datos imprecisos con referencial discreto no ordenado, sobre el que se define una relación de similitud y que pueden ser consultados de forma imprecisa.

En el MR los atributos toman valores en dominios atómicos (enteros, cadenas de caracteres...) que son unidades mínimas de información. El SBDR Oracle ©, utilizado para la implementación del modelo de GDB, también dispone de estos dominios atómicos.

Así pues, el problema es representar estos tres tipos de difusos con dominios atómicos. El *difuso tipo 1* es directamente asumible por el SGBDR, ya que sólo sería difusa la consulta, no la representación. Para *los tipos 2 y 3* se utilizará en GDB la aproximación expuesta en [\[Gal99\]](#), que puede se detalla en las siguientes tablas:

Tabla 20. Representación de difusos tipo 2 en un SGBDR.

Tipo de valor	Atributos difusos tipo 2				
	FT	F1	F2	F3	F4
Unkown	0	Null	Null	Null	Null
Undefined	1	Null	Null	Null	Null
Null	2	Null	Null	Null	Null
Clásico	3	d	Null	Null	Null
Etiqueta	4	Fuzzy_ID	Null	Null	Null
Intervalo [n,m]	5	n	Null	Null	m
Aproximadamente(d)	6	d	d-margen	d+margen	margen
Trapezio [$\alpha, \beta, \gamma, \delta$]	7	α	$\beta - \alpha$	$\gamma - \delta$	δ

Tabla 21. Representación de difusos tipo 3 en un SGBDR.

Tipo de valor	Atributos difusos tipo 3					
	FT	FP1	F1	...	FPn	Fn
Unkown	0	Null	Null	...	Null	Null
Undefined	1	Null	Null	...	Null	Null
Null	2	Null	Null	...	Null	Null
Simple	3	p	d	...	Null	Null
Distribución de posibilidad	4	p ₁	d ₁	...	p _n	d _n

En ambas tablas, el atributo “FT” (“fuzzy type”) es un identificador unívoco del tipo de valor a representar y un valor del atributo “Null” indican el valor nulo en el SGBDR.

En la tabla de representación de *atributos difusos tipo 3*, el par de atributos $\{(FP_x, F_x) \mid x \in \{1, n\}\}$ almacenan la pareja de valores de la distribución de posibilidad: (*posibilidad, etiqueta*).

5.1.1.2 Comparadores difusos generalizados

Partiendo de los operadores clásicos que proporciona un SGBDR para realizar consultas (igual, mayor, menor, etcétera), se añaden los comparadores difusos generalizados que define GEFRED (4.2.2.5). En concreto, GDB utiliza la aproximación de FIRST, definiendo los 14 comparadores de la *Tabla 19*.

Cuando una consulta incorpore un CDG, es necesario calcular el GA resultante, para ello es necesario aplicar las reglas definidas en el siguiente punto.

5.1.1.2.1 Cálculo del grado de acoplamiento con comparadores difusos generalizados

La función GA o grado de compatibilidad $CDEG(\text{condición})$, toma como parámetro una condición (que involucra dos atributos difusos o no) y calcula el GA de dicha condición. Cuando la condición involucra atributos difusos, se utiliza un conjunto de ecuaciones aplicadas sobre las distribuciones de posibilidad trapezoidales (determinadas por cuatro parámetros: $\{\alpha, \beta, \delta, \gamma\}$) asociadas a cada atributo. Estas ecuaciones caracterizan la intersección de ambas distribuciones de posibilidad.

Antes de aplicar la ecuación, y salvo en el caso del comparador de igualdad difusa, la distribución de posibilidad de la parte derecha de la comparación ha de modificarse de acuerdo al tipo de operador a aplicar, como se indica en la siguiente ilustración.

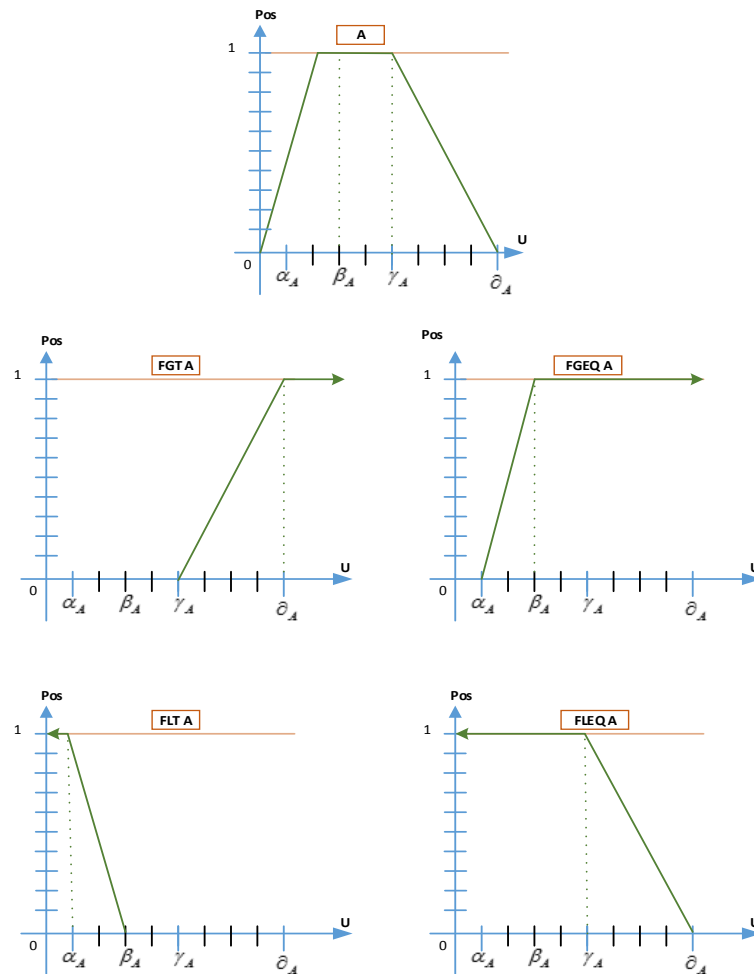


Ilustración 59. Modificación de la distribución de posibilidad de la parte derecha de una comparación difusa según el comparador utilizado.

Existen dos tipos de comparadores difusos: de posibilidad y de necesidad (más restrictivos que los anteriores). Las ecuaciones de ambos tipos se describen en las dos siguientes tablas.

Tabla 22. Cálculo del grado de compatibilidad con comparadores difusos generalizados de posibilidad.

CDG	Significado en español / traducción en inglés	Cálculo CDEG(A CDG B)
FEQ	Igualdad difusa / " Fuzzy Equal"	$= \sup_{d \in U} (\min(A(d), B(d)))$ <p>Donde U es el dominio subyacente de A y B. $A(d)$ es el grado de posibilidad de $d \in U$ en la distribución de posibilidad A y $B(d)$ el grado de posibilidad de en B</p>

FGT	Mayor que difuso / "Fuzzy Greater Than"	$\begin{cases} = 1 & \text{si } \gamma_A \geq \delta_B \\ = \frac{\delta_A - \gamma_B}{(\delta_B - \gamma_B) - (\gamma_A - \delta_A)} & \text{si } (\gamma_A < \delta_B) \wedge \\ & \wedge (\delta_A > \gamma_B) \\ = 0 & \text{en otro caso} \end{cases}$
FGEQ	Mayor o igual que difuso / "Fuzzy Greater or Equal than"	$\begin{cases} = 1 & \text{si } \gamma_A \geq \beta_B \\ = \frac{\delta_A - \alpha_B}{(\beta_B - \alpha_B) - (\gamma_A - \delta_A)} & \text{si } (\gamma_A < \beta_B) \wedge \\ & \wedge (\delta_A > \alpha_B) \\ = 0 & \text{en otro caso} \end{cases}$
FLT	Menor que difuso / "Fuzzy Less Than"	$\begin{cases} = 1 & \text{si } \beta_A \leq \alpha_B \\ = \frac{\alpha_A - \beta_B}{(\alpha_B - \beta_B) - (\beta_A - \alpha_A)} & \text{si } (\beta_A > \alpha_B) \wedge \\ & \wedge (\alpha_A < \beta_B) \\ = 0 & \text{en otro caso} \end{cases}$
FLEQ	Menor o igual que difuso / "Fuzzy Less or Equal than"	$\begin{cases} = 1 & \text{si } \beta_A \leq \gamma_B \\ = \frac{\delta_B - \alpha_A}{(\beta_A - \alpha_A) - (\gamma_B - \delta_B)} & \text{si } (\beta_A > \gamma_B) \wedge \\ & \wedge (\alpha_A < \delta_B) \\ = 0 & \text{en otro caso} \end{cases}$
MGT	Mucho mayor que difuso / "Much Greater Than"	$\begin{cases} = 1 & \text{si } \gamma_A \geq \delta_B + M \\ = \frac{\gamma_B + M - \delta_A}{(\gamma_A - \delta_A) - (\delta_B - \gamma_B)} & \text{si } (\gamma_A < \delta_B + M) \wedge \\ & \wedge (\delta_A > \gamma_B + M) \\ = 0 & \text{en otro caso} \end{cases}$ <p>Donde M define la distancia mínima para considerar dos atributos como muy separados y es definida en la FMB para cada atributo</p>
MLT	Mucho menor que difuso / "Much Less Than"	$\begin{cases} = 1 & \text{si } \beta_A \leq \alpha_B - M \\ = \frac{\beta_B - M - \alpha_A}{(\beta_A - \alpha_A) - (\alpha_B - \beta_B)} & \text{si } (\beta_A > \alpha_B - M) \wedge \\ & \wedge (\alpha_A < \beta_B - M) \\ = 0 & \text{en otro caso} \end{cases}$ <p>Donde M indica la distancia mínima para considerar dos atributos como muy separados y es definida en la FMB para cada atributo</p>

La aplicación de los comparadores de necesidad requiere de la negación de la distribución de posibilidad asociada a la parte izquierda de la comparación antes de aplicar la ecuación oportuna.

Tabla 23. Cálculo del grado de compatibilidad con comparadores difusos generalizados de necesidad.

CDG (N) indica necesariamente / (N) "necessarily"	Significado en español / traducción en inglés	Cálculo CDEG(A CDG B)
(N)FEQ	(N) Igualdad difusa / "(N) Fuzzy Equal"	$= \inf_{d \in U} (\max(1 - A(d), B(d)))$ Donde U es el dominio subyacente de A y B . $A(d)$ es el grado de posibilidad de $d \in U$ en la distribución de posibilidad A
(N)FGT	(N) Mayor que difuso / "(N) Fuzzy Greater Than"	$\begin{cases} = 1 & \text{si } \alpha_A \geq \delta_B \\ = \frac{\beta_A - \gamma_B}{(\delta_B - \gamma_B) - (\alpha_A - \beta_A)} & \text{si } (\alpha_A < \delta_B) \wedge \\ & \wedge (\beta_A > \gamma_B) \\ = 0 & \text{en otro caso} \end{cases}$
(N)FGEQ	(N) Mayor o igual que difuso / "(N) Fuzzy Greater or Equal than"	$\begin{cases} = 1 & \text{si } \alpha_A \geq \beta_B \\ = \frac{\beta_A - \alpha_B}{(\beta_B - \alpha_B) - (\alpha_A - \beta_A)} & \text{si } (\alpha_A < \beta_B) \wedge \\ & \wedge (\beta_A > \alpha_B) \\ = 0 & \text{en otro caso} \end{cases}$
(N)FLT	(N) Menor que difuso / "(N) Fuzzy Less Than"	$\begin{cases} = 1 & \text{si } \delta_A \leq \alpha_B \\ = \frac{\gamma_A - \beta_B}{(\alpha_B - \beta_B) - (\delta_A - \gamma_A)} & \text{si } (\delta_A > \alpha_B) \wedge \\ & \wedge (\gamma_A < \beta_B) \\ = 0 & \text{en otro caso} \end{cases}$
(N)FLEQ	(N) Menor o igual que difuso / "(N) Fuzzy Less or Equal than"	$\begin{cases} = 1 & \text{si } \delta_A \leq \gamma_B \\ = \frac{\gamma_A - \delta_B}{(\gamma_B - \delta_B) - (\delta_A - \gamma_A)} & \text{si } (\delta_A > \gamma_B) \wedge \\ & \wedge (\gamma_A < \delta_B) \\ = 0 & \text{en otro caso} \end{cases}$
(N)MGT	(N) Mucho mayor que difuso / "(N) Much Greater Than"	$\begin{cases} = 1 & \text{si } \alpha_A \geq \delta_B + M \\ = \frac{\gamma_B + M - \beta_A}{(\gamma_A - \beta_A) - (\delta_B - \gamma_B)} & \text{si } (\alpha_A < \delta_B + M) \wedge \\ & \wedge (\beta_A > \gamma_B + M) \\ = 0 & \text{en otro caso} \end{cases}$ Donde M define la distancia mínima para

		considerar dos atributos como muy separados y es definida en la FMB para cada atributo
(N)MLT	(N) Mucho menor que difuso / “(N) Much Less Than”	$\begin{cases} = 1 & \text{si } \delta_A \leq \alpha_B - M \\ = \frac{\beta_B - M - \gamma_A}{(\delta_A - \gamma_A) - (\alpha_B - \beta_B)} & \text{si } (\delta_A > \alpha_B - M) \wedge \\ & \wedge (\gamma_A < \beta_B - M) \\ = 0 & \text{en otro caso} \end{cases}$ <p>Donde M indica la distancia mínima para considerar dos atributos como muy separados y es definida en la FMB para cada atributo</p>

En caso de que se enlacen condiciones mediante *operadores lógicos*, se aplicará la siguiente tabla para el cálculo del GA resultante:

Tabla 24. Cálculo del grado de compatibilidad con operadores lógicos.

Condición	Cálculo CDEG(Condición)
Condición_1 AND Condición_2	$\min \left(\begin{matrix} CDEG(\text{condición}_1), \\ CDEG(\text{condición}_2) \end{matrix} \right)$
Condición_1 OR Condición_2	$\max \left(\begin{matrix} CDEG(\text{condición}_1), \\ CDEG(\text{condición}_2) \end{matrix} \right)$
NOT Condición	$1 - CDEG(\text{condición})$

5.1.1.3 Base de meta-conocimiento difuso

El módulo FMB (base de metaconocimiento difuso) es uno de los componentes de la arquitectura de GDB (5.5) y almacena toda la información adicional requerida para manejar la imprecisión de los atributos. En concreto, se compone de los elementos:

- a) Lista de atributos que reciben un tratamiento difuso.
- b) Tipo de dato difuso del atributo
- c) Cuantificadores difusos
- d) Objetos difusos definidos sobre cada atributo:
 - 1) Etiquetas lingüísticas (para *atributos difusos de tipo 1, 2 y 3*).
 - 2) Margen de los valores aproximados y distancia mínima para que dos valores se consideren muy separados (sólo para *atributos difusos de tipo 1 y 2*).
 - 3) Relaciones de semejanza (sólo para *atributos difusos de tipo 3*).
- e) Descripción de los objetos difusos.

El FMB implementa los anteriores elementos en el SGBDR como el siguiente conjunto de relaciones (los detalles de los atributos de cada tabla pueden consultarse en [Gal99]):

- a) **FUZZY_COL_LIST.** Contiene la lista de atributos que reciben tratamiento difuso. Cada atributo queda descrito por una referencia a la tabla a la que pertenecen, columna en la que se almacenan, el *tipo difuso* (1,2 o 3), el número de valores de la distribución de posibilidad (sólo aplicable para los difusos tipo 3), y un comentario.
- b) **FUZZY_OBJECT_LIST.** Almacena los objetos difusos que pertenecen al dominio de un atributo difuso. Cada objeto se define mediante la tabla y columna del atributo referenciado, un identificador, un nombre y un tipo de objeto.
- c) **FUZZY_LABEL_DEF.** Contiene la definición de las distribuciones de posibilidad trapezoidales que se asocian a etiquetas lingüísticas. Cada una de ellas está descrita por la tabla y columna a cuyo dominio pertenecen, la etiqueta que asocia y los parámetros que la definen.
- d) **FUZZY_APPROX_MUCH.** Almacena los parámetros “*margen*” (área entorno al valor que será usada en las comparaciones difusas) y “*much*” (valor que indica cuando dos valores están suficientemente alejados para no aplicar una comparación difusa) para cada atributo *de tipo difuso 1 y 2*.
- e) **FUZZY_NEARNESS_DEF.** Contiene los valores de similitud entre cada par de valores posible del dominio discreto de valores escalares que se asocia a un atributo de *tipo difuso 3*. A cada par de valores hay asociado un GA.
- f) **FUZZY_COMPATIBLE_COL.** Almacena información sobre aquellos atributos de *tipo difuso 3* que comparten dominio (son compatibles) con otro atributo difuso del mismo tipo.
- g) **FUZZY_QUALIFIERS_DEF.** Contiene el umbral mínimo de cumplimiento para cada cualificador definido sobre una etiqueta lingüística que pertenece al dominio de un atributo difuso.

5.1.1.4 FSQL en el modelo GDB

En [Gal99] y [Bla0b] se describe el FSQL usado en GDB y se indican las modificaciones realizadas al lenguaje SQL para soportar las extensiones difusas: a la hora de la definición (DDL difuso) y a la hora de la manipulación (DML difuso).

De todas las posibilidades del lenguaje FSQL, GDB-GUI sólo hará uso de ciertas sentencias y determinadas capacidades de ellas. Se describen a continuación las sentencias de DDL y DML utilizadas.

5.1.1.4.1 DDL difuso

El DDL es el encargado de dar soporte a la definición de elementos difusos. Se define la sentencia “create table” de este lenguaje.

5.1.1.4.1.1 Sentencia “create table”

Esta sentencia permite definir la lista de atributos (difusos o no) que compone una relación.

create_table: CREATE TABLE nombre_tabla ((' lista_columnas ') clausulas_create_table_clásico
';

lista_columnas: lista_columnas ',' lista_columnas | descripción_columna | restricción_tabla

descripción_columna: id_columnas tipo_dato

tipo_dato: tipo_clásico [lista_restricciones_clásicas] | difuso_tipo [lista_restricciones_difusas]

difuso_tipo: difuso_tipo1 | difuso_tipo2 | difuso_tipo3

difuso_tipo1: (FTYPE1 | CRISP) ('margen ',' much ') [tipo_base]

difuso_tipo2: (FTYPE2 | POSSIBILISTIC) ('margen ',' much ') [tipo_base]

difuso_tipo3: (FTYPE3 | SCALAR) [('nescalares ')] [DOMAIN referencia_columna]

lista_restricciones_difusas: NULL | ONLY LABEL | NOT no_permitidos

no_permitidos: NULL | UNDEFINED | UNKNOWN | LABEL | CRISP | TRAPEZOID | INTERVAL | APPROX

tipo_base: tipo_clásico

El símbolo “clausulas_create_table_clásico” se refiere a las cláusulas que aparecen comúnmente al final de la sentencia “select” clásica en SQL.

En la definición de **difuso_tipo2**, ‘margen’ indica la base del triángulo (en valor absoluto) de la etiqueta *aproximadamente n*, donde *n* es el valor del atributo. ‘much’ es la distancia mínima para que dos valores del atributo sean considerados demasiado alejados para involucrarlos en la consulta.

La cláusula “DOMAIN” en **difuso_tipo3**, permite definir el dominio de un nuevo *atributo difuso de tipo 3* como una copia del dominio del atributo especificado por “referencia_columna”. El símbolo “tipo_clásico” se refiere a los tipos de datos propios del SGBDR.

5.1.1.4.2 DML difuso

El DML permite la manipulación de información difusa. En GDB, se utilizan las sentencias “select” e “insert” de este lenguaje.

5.1.1.4.2.1 Sentencia “select”

Con esta sentencia, el usuario puede obtener una relación que incluya un conjunto de atributos (difusos o no), donde las tuplas cumplen una determinada condición.

select: SELECT lista_columnas FROM lista_Tablas WHERE condición

condición: condición_simple [operador_lógico condición_simple]*

condición_simple: condición_simple THOLD x

Donde el símbolo *condición* puede aparecer, además de las expresiones clásicas de un SGBDR, en expresiones que involucren comparadores difusos generalizados ([5.1.1.2](#)), y *operador_lógico* es cualquier operador lógico clásico del SGBDR.

El umbral de cumplimiento THOLD x , con $x \in \{0,1\}$, indica qué *condición* debe ser satisfecha por una tupla con un GA mayor o igual a x para que dicha tupla aparezca en la relación resultante.

Dentro de *lista_columnas* puede aparecer la función *CDEG(condición)* ([5.1.1.2.1](#)), añadiendo un atributo adicional a la relación resultante. El parámetro de la función puede indicar el nombre atributo, en tal caso, sólo se evalúan las condiciones que incluyan a dicho atributo. También

puede aparecer como parámetro un asterisco $CDEG(*)$, por lo que se calculará el GA para cada tupla de la relación resultante, pero utilizando todos los atributos involucrados en la condición.

5.1.1.4.2.2 Sentencia “insert”

La sentencia insert es utilizada para añadir información (difusa o clásica) a GDB.

insert: INSERT INTO nombre_relación VALUES (' lista_valores ')

lista_valores: valor, lista_valores | valor

valor: valor_clásico | valor_difuso

valor_difuso: valores_nulos | distribución | etiqueta | intervalo | aprox

valores_nulos: UNDEFINED | UNKNOWN | NULL

distribución: \$ [' NÚMERO ',' NÚMERO ',' NÚMERO ',' NÚMERO ']

etiqueta: \$ID

aprox: #NÚMERO

Donde *valor_difuso* indica un valor atómico del SBDR.

5.1.2 Representación y manejo de información deductiva

En el modelo de GDB, la representación y manejo de la información deductiva, se inicia con la propuesta de FREDDI, que posteriormente fue extendida en **[Bla01]**. FREDDI presenta la Regla generalizada difusa (4.5.5) que es ampliada hasta obtener la *regla generalizada con grado de acoplamiento* (RGGA).

Se analizará en esta sección la RGGA y cómo calcularla. Se continuará con el estudio del algoritmo de deducción implementado, las modificaciones al FMB, la definición del DFSQL y la aplicación del modelo en un ejemplo.

5.1.2.1 Regla generalizada con grado de acoplamiento

En este apartado, se planteará inicialmente el concepto de RGGA, para después analizar su cálculo en diferentes contextos: con *predicados disyuntivos* y con *predicados negados*.

En el análisis de la deducción de una RGGA (4.5.6), se vio la necesidad de incorporar el GA tanto en la definición de los predicados, en la BH y en la BR. La formalización de estos conceptos es la siguiente.

❖ **Definición 5.1. Regla generalizada con grado de acoplamiento.** Se denomina *regla generalizada con grado acoplamiento* (RGGA) a una regla de la siguiente forma:

$$P(X_1, X_2, \dots, X_n, \gamma_p^-) \leftarrow \tilde{Q}_1(Y_{1,1}, Y_{1,2}, \dots, Y_{1,n_1}, \gamma_{Q_1}^-) \wedge \tilde{Q}_2(Y_{2,1}, Y_{2,2}, \dots, Y_{2,n_2}, \gamma_{Q_2}^-) \wedge \dots \wedge \tilde{Q}_m(Y_{m,1}, Y_{m,2}, \dots, Y_{m,n_m}, \gamma_{Q_m}^-) \wedge \Psi \quad \text{Eq. (5.1)}$$

Siendo:

$$\Psi \equiv \bigwedge \geq \left(\Theta_{i,j,k}^{\bar{}} (X_i, Y_{j,k}), \alpha_{i,j,k} \right) \bigwedge \geq \left(\Theta_{j,k,l,p}^{\bar{}} (Y_{j,k}, Y_{l,p}), \beta_{j,k,l,p} \right) \bigwedge \phi_{j,k,l,p}^{\theta} \left(\Theta_{j,k,l,p}^{\theta} (Y_{j,k}, Y_{l,p}), \gamma_{j,k,l,p} \right) \quad \text{Eq. (5.2)}$$

Siendo γ_p una *función grado de acoplamiento* de un hecho construida en base a los grados de acoplamiento de la variables y a los grados de acoplamiento de los hechos obtenidos.

La RGGa constituye la regla base que se utilizará en GDB, ya que permite aplicar un algoritmo de deducción (como se verá en 5.1.2.2), y proporciona soporte al cálculo del GA, que servirá a su vez como medida de calidad de los hechos calculados (5.3).

La extensión de la BH con el GA, conduce a la definición de la *base de hechos extendida* (BHE).

❖ **Definición 5.2. Base de hechos extendida.** Sea una BH de forma clausular

$$\Lambda = \left\{ \mathcal{Q}_i(a_1, \dots, a_j, \dots, a_{n_i}) \mid (i \in [1, m]) \wedge (j \in [1, n_i]) \wedge (a_j \in \text{Dom}(Y_{i,j})) \right\} \quad \text{Eq. (5.3)}$$

Donde m es el número de predicados del cuerpo de la regla a la que se asocia la base de hechos y n_i el número de variables del predicado \mathcal{Q}_i . Se denomina *base de hechos extendida* a una base de hechos en forma clausular $\tilde{\Lambda}$ con la siguiente estructura:

$$\tilde{\Lambda} = \left\{ \tilde{\mathcal{Q}}_i(a_1, \dots, a_j, \dots, a_{n_i}, 1) \mid \mathcal{Q}_i(a_1, \dots, a_j, \dots, a_{n_i}, 1) \in \Lambda \right\} \quad \text{Eq. (5.4)}$$

El cálculo el GA de un hecho respecto a un predicado, queda definido como sigue:

❖ **Definición 5.3. Función grado de acoplamiento de un hecho.** Sea P un predicado definido mediante una RGGa. Se denomina *función grado de acoplamiento* de un hecho del predicado P , al propio predicado, como la función Φ :

$$\Phi \left(\begin{array}{l} \left(\Theta_{i,j,k}^{\bar{}} (X_i, Y_{j,k}), \dots, \Theta_{i,j,k}^{\bar{}} (X_i, Y_{j,k}), \Theta_{j,k,l,p}^{\bar{}} (Y_{j,k}, Y_{l,p}), \dots, \Theta_{j,k,l,p}^{\bar{}} (Y_{j,k}, Y_{l,p}), \right) \\ \left(\Theta_{j,k,l,p}^{\theta} (Y_{j,k}, Y_{l,p}), \dots, \Theta_{j,k,l,p}^{\theta} (Y_{j,k}, Y_{l,p}), \gamma_{\mathcal{Q}_1}, \gamma_{\mathcal{Q}_2}, \dots, \gamma_{\mathcal{Q}_m} \right) \end{array} \right) \quad \text{Eq. (5.5)}$$

Siendo:

1. $\Theta_{i,j,k}^{\bar{}} (X_i, Y_{j,k})$ el GA entre los valores de las variables unificadas entre la cabeza y el cuerpo de la regla.
2. $\Theta_{j,k,l,p}^{\bar{}} (Y_{j,k}, Y_{l,p})$ el GA entre los valores de las variables ligadas en cuerpo de la regla.
3. $\Theta_{j,k,l,p}^{\theta} (Y_{j,k}, Y_{l,p})$ el GA entre los valores de las variables del cuerpo de la regla que se encuentran ligadas entre sí por comparaciones de variables explícitas.
4. $\gamma_{\mathcal{Q}_i}, i = 1, 2, \dots, m$ el GA entre de un hecho del predicado \mathcal{Q}_i con el propio predicado.

La función Φ calcula el GA del hecho en base a los grados de acoplamiento de los valores de las variables instanciadas en el predicado y de los grados de acoplamiento de los hechos calculados en el cuerpo de la regla. Es decir, Φ debe modelizar un GA a partir de una conjunción de grados de acoplamiento. Para ello utiliza una T-Norma (4.2.1.1 Operaciones con conjuntos difusos) que permite calcular la intersección difusa de los grados de acoplamiento. Típicamente se utiliza la T-norma $\min(a, b)$, dado que el GA de un hecho con la regla nunca puede superar el menor de los grados de acoplamiento alcanzados en el seno de la regla. Por tanto se puede redefinir la función Φ con r comparaciones y m predicados como sigue:

$$\Phi(c_1, \dots, c_r, \gamma_1, \dots, \gamma_m) = \min \left(\begin{array}{l} \min(c_1, \min(\dots, \min(c_{r-1}, c_r))), \\ \min(\gamma_1, \min(\dots, \min(\gamma_{m-1}, \gamma_m))) \end{array} \right) \quad \text{Eq. (5.6)}$$

Al igual que en la BH, es necesario extender la BR con el GA, creando una base de reglas extendida (BRE).

❖ **Definición 5.4. Base de reglas extendida.** Sea \tilde{P} un predicado definido mediante una RGGA. Se denomina *base de reglas extendida* a un conjunto de cláusulas obtenidas de \tilde{P} que posee la siguiente estructura:

$$\tilde{\Gamma} = \left\{ \begin{array}{l} \neg \tilde{Q}_1(Y_{1,1}, Y_{1,2}, \dots, Y_{1,n_1}, \gamma_{\tilde{Q}_1}^-) \vee \neg \tilde{Q}_2(Y_{2,1}, Y_{2,2}, \dots, Y_{2,n_2}, \gamma_{\tilde{Q}_2}^-) \vee \\ \vee \dots \vee \neg \tilde{Q}_m(Y_{m,1}, Y_{m,2}, \dots, Y_{m,n_m}, \gamma_{\tilde{Q}_m}^-) \vee \bar{\Psi} \vee \tilde{P}(X_1, X_2, \dots, X_n, \gamma_{\tilde{P}}^-) \end{array} \right\} \quad \text{Eq. (5.47)}$$

Siendo:

$$\begin{aligned} \bar{\Psi} \equiv \bigvee \neg \geq (\Theta_{i,j,k}^= (X_i, Y_{j,k}), \alpha_{i,j,k}) \bigvee \neg \geq (\Theta_{j,k,l,p}^= (Y_{j,k}, Y_{l,p}), \beta_{j,k,l,p}) \bigvee \\ \bigvee \neg \phi_{j,k,l,p} (\Theta_{j,k,l,p}^\theta (Y_{j,k}, Y_{l,p}), \gamma_{j,k,l,p}) \end{aligned} \quad \text{Eq. (5.48)}$$

Es decir, para cada predicado se crea un nuevo predicado con la misma estructura, a la que se añade una variable adicional que representa el GA de un hecho con el propio predicado.

5.1.2.1.1 Cálculo de reglas generalizadas con grados de acoplamiento

El conjunto M (4.5.6 Deducción con reglas generalizadas difusas) asociado a una RGGA, posee la misma semántica que en una RGD: es el conjunto de hechos adicionales que completan la descripción del mundo al que representa la BH. Llamaremos a los elementos de M *predicados calculables* y *conjunto de hechos calculados* de un predicado a todas las posibles instancias del predicado respecto a una BH. Es decir:

❖ **Definición 5.5. Predicado calculable.** Se denomina *predicado calculable* a los predicados del conjunto M de las ecuaciones Eq. (4.42) y Eq. (4.44) asociadas a los dos tipos de deducción de una RGD.

❖ **Definición 5.6. Conjuntos de hechos calculados de un predicado.** Sea $P(X_1, \dots, X_n)$ un *predicado calculable* y Λ una BH. Se denomina *conjunto de hechos calculados* de este predicado a partir de Λ , al conjunto de hechos siguiente:

$$Cálculo(P(X_1, \dots, X_n), \Lambda) = \left\{ \begin{array}{l} P(a_1, \dots, a_i, \dots, a_n, \gamma_p) \mid (i \in [1, n]) \wedge \\ (a_i \in Dom(X_i)) \wedge \delta \end{array} \right\} \text{ Eq. (5.49)}$$

Donde los dominios $Dom(X_i) \mid (i \in [1, n])$ se construyen a partir de la BH Λ y δ es una condición basada en las funciones definidas sobre esos dominios.

Para cada *predicado calculable*, será necesario proporcionar la expresión que construye su conjunto de hechos calculados. De esta forma, un predicado definido por una regla (RGGa o no) es un *predicado calculable*, cuya expresión para el conjunto de hechos calculados es la propia regla.

En el proceso de deducción, cuando se busque una resolvente para un *predicado calculable*, bastará con calcular los hechos de dicho predicado en base a la expresión del *conjunto de hechos calculados*.

5.1.2.1.2 Cálculo de una RGGa con predicados disyuntivos

Un caso especial en el cálculo de hechos de un predicado \tilde{P} definido por una RGGa, es cuando aparecen el cuerpo predicados extendidos \tilde{P}_i unidos por el operado disyuntivo \vee . Estos predicados se definen a su vez a través de una RGGa. Los hechos que satisfacen el predicado \tilde{P} son aquellos que satisfacen en algún grado alguno de los $\tilde{P}_i \mid i \in \{1, 2, \dots, r\}$, por tanto se puede construir el conjunto de hechos de los predicados de \tilde{P} a partir de los de \tilde{P}_i , pero distinguiendo dos casos:

- a) Cuando un hecho sólo resulta de un único predicado \tilde{P}_i , por lo que se deduce de la BHE con un único GA (predicados disyuntivos excluyentes).
- b) Cuando un hecho resulta del cálculo de más de un predicado \tilde{P}_i con grados de acoplamiento no necesariamente iguales (predicados disyuntivos no excluyentes).

Para distinguir en cálculo de ambos casos, hay que definir el concepto de *componente clásica de un predicado*, de un *hecho* y de un *conjunto de hechos*.

❖ **Definición 5.7. Componente clásica de un predicado.** Sea $\tilde{P}(X_1, \dots, X_n, \Phi(\dots))$ un predicado definido mediante una RGGa, sea $\Phi(\dots)$ la función que calcula su GA. Se denomina *componente clásica del predicado \tilde{P}* , y se nota por $(\tilde{P}(X_1, \dots, X_n, \Phi(\dots)))^c$, al conjunto de variables $\{X_1, \dots, X_n\}$.

❖ **Definición 5.8. Componente clásica de un hecho extendido.** Sea $\tilde{P}(X_1, \dots, X_n, \Phi(\dots))$ un predicado definido mediante una RGGa, y sea $\tilde{P}(a_1, \dots, a_n, b)$ un hecho que verifica \tilde{P} . Se denomina *componente clásica de un*

hecho extendido del predicado \tilde{P} y se nota por $(\tilde{P}(a_1, \dots, a_n, b))^c$, al conjunto de valores representado por la tupla (a_1, \dots, a_n) .

- ❖ **Definición 5.9. Componente clásica de un conjunto de hechos extendidos.** Sea $\tilde{P}(X_1, \dots, X_n, \Phi(\dots))$ un predicado definido mediante una RGGa, y $\tilde{P}(a_1, \dots, a_n, b)$ un hecho que verifica \tilde{P} . Entonces se denomina *componente clásica de un conjunto de hechos extendidos* del predicado \tilde{P} , y se nota por $\{\tilde{P}(a_1, \dots, a_n, b)\}^c$, al conjunto $\{(\tilde{P}(a_1, \dots, a_n, b))^c\}$ formado por las *componentes clásicas de los hechos extendidos*.

Se analiza a continuación como realizar el cálculo de una RGGa cuando aparecen predicados disyuntivos excluyentes y no excluyentes.

5.1.2.1.2.1 Cálculo con predicados disyuntivos excluyentes

Antes de proceder al cálculo de hechos, se define previamente *el conjunto de reglas excluyentes*:

- ❖ **Definición 5.10. Conjunto de reglas excluyente.** Sea $\tilde{P}(X_1, \dots, X_n, \Phi(\dots))$ un predicado definido por una disyunción de un conjunto reglas $\{\tilde{P}_1(X_1, \dots, X_n, \Phi_1(\dots)) \vee \dots \vee \tilde{P}_r(X_1, \dots, X_n, \Phi_r(\dots))\}$. Este *conjunto de reglas* es *excluyente* (un hecho no resulta del cálculo de más de una regla), si verifica:

$$\bigcap_r \left(\text{Cálculo}(\tilde{P}_r(X_1, \dots, X_n, \Phi_r(\dots)), \tilde{\Lambda}) \right)^c = 0 \quad \text{Eq. (5.7)}$$

Dónde la función *Cálculo* corresponde al *conjuntos de hechos calculados* de \tilde{P} (ver [Definición 5.6](#)), o lo que es lo mismo, el conjunto de hechos que verificados por \tilde{P} .

El cálculo del predicado \tilde{P} se realiza mediante la siguiente operación.

$$\text{Cálculo}(\tilde{P}(X_1, \dots, X_n, \gamma_{\tilde{P}}), \tilde{\Lambda}) = \bigcup_r \text{Cálculo}(\tilde{P}_r(X_1, \dots, X_n, \Phi_r(\dots)), \tilde{\Lambda}) \quad \text{Eq. (5.8)}$$

Donde \tilde{P}_r son las reglas o sub-predicados que intervienen en la disyunción del cuerpo de \tilde{P} , Φ_r es la expresión para el cálculo del GA de cada hecho del predicado \tilde{P} y $\tilde{\Lambda}$ es la BHE obtenida de \tilde{P} .

5.1.2.1.2.2 Cálculo con predicados disyuntivos no excluyentes

En este caso, para el cálculo de hechos, es necesario definir previamente el concepto de *conjunto de reglas no excluyentes*.

- ❖ **Definición 5.11. Conjunto de reglas no excluyente.** Sea $\tilde{P}(X_1, \dots, X_n, \Phi(\dots))$ un predicado definido por una disyunción de un conjunto reglas

$\{\tilde{P}_1(X_1, \dots, X_n, \Phi_1(\dots)) \vee \dots \vee \tilde{P}_r(X_1, \dots, X_n, \Phi_r(\dots))\}$. Este *conjunto de reglas* es no *excluyente*, si un hecho resulta del cálculo de más de una regla con GA no necesariamente iguales:

$$\bigcap_r \left(\text{Cálculo} \left(\tilde{P}_r(X_1, \dots, X_n, \Phi_r(\dots)), \tilde{\Lambda} \right) \right)^c \neq 0 \quad \text{Eq. (5.9)}$$

Por lo tanto, el cálculo del predicado \tilde{P} se puede realizar como:

$$\begin{aligned} \text{Cálculo} \left(\tilde{P}(X_1, \dots, X_n, \gamma_{\tilde{P}}), \tilde{\Lambda} \right) = \\ \left\{ \tilde{P}(a_1, \dots, a_n, b) \mid (a_1, \dots, a_n, b) \notin \bigcap_r \left(\text{Cálculo} \left(\tilde{P}_r(X_1, \dots, X_n, \Phi_r(\dots)), \tilde{\Lambda} \right) \right)^c \right\} \cup \\ \left\{ \tilde{P}(a_1, \dots, a_n, \Omega(w_1, \dots, w_r)) \mid (a_1, \dots, a_n) \in \bigcap_r \left(\text{Cálculo} \left(\tilde{P}_r(X_1, \dots, X_n, \Phi_r(\dots)), \tilde{\Lambda} \right) \right)^c \right\} \end{aligned} \quad \text{Eq. (5.10)}$$

Donde cada $w_i \mid i \in \{1, 2, \dots, r\}$ es el GA del hecho $\tilde{P}_i(a_1, \dots, a_n)$ con el predicado a través de la regla i -ésima \tilde{P}_i y puede calcularse como:

$$w_i = \begin{cases} b & \text{si } \left(\tilde{P}(a_1, \dots, a_n, b) \right)^c \in \left(\text{Cálculo} \left(\tilde{P}(X_1, \dots, X_n, \gamma_{\tilde{P}}), \tilde{\Lambda} \right) \right)^c \\ 0 & \text{en otro caso} \end{cases} \quad \text{Eq. (5.11)}$$

Es decir, w_i asigna GA con valor cero a todos los hechos que no resultan del cálculo de la regla \tilde{P}_i .

Se define $\tilde{\Omega}$ como la función que mide el GA en base al conjunto de grados de acoplamiento de la reglas \tilde{P}_i . Esto es, $\tilde{\Omega}$ modeliza el GA de una disyunción de grados de acoplamiento. Para realizar esta labor, se utiliza una T-conorma ([4.2.1.1 Operaciones con conjuntos difusos](#)) que calcula la unión difusa de los grados de acoplamiento. Semánticamente, dado que el conjunto de variables resulta de hechos calculados en más de una regla, se puede desechar el hecho que tuviera menos GA y mantener el de mayor GA sin limitar la validez de resultado. Normalmente se utiliza la T-conorma $\max(a, b)$. Por tanto la función $\tilde{\Omega}$ puede definirse como:

$$\Phi(w_1, w_2, \dots, w_{r-1}, w_r) = \max(w_1, \max(w_2, \max(\dots, \max(w_{r-1}, w_r)) \dots)) \quad \text{Eq. (5.12)}$$

5.1.2.1.3 Cálculo de una RGGa con predicados negados

Los predicados negados pueden aparecer dentro del cuerpo de una regla, por lo que su cálculo necesita de un tratamiento adicional. Existen dos aproximaciones posibles a la implementación de la negación, una *semántica* (implementada en GDB como se verá más adelante) y otra *sintáctica*.

5.1.2.1.3.1 Enfoque semántico

Esta aproximación basa la implementación del predicado negado en la *hipótesis del mundo cerrado* y en la *ley del tercero excluido*.

La *hipótesis del mundo cerrado*, indica que sólo se conoce (es cierto) aquello que está almacenado en la BD. Por tanto, la negación de un hecho, es cierta, únicamente cuando el hecho afirmado (no negado) no puede probarse a partir de la BD. Su aplicación fue estudiada en **[Rei78]**, **[Rei80]**, **[Min82]**, y **[Rei84]**.

La *ley del tercero excluido* indica que dado un conjunto A , cualquier elemento x , o bien x pertenece a A , o bien $\neg x$ pertenece a A . Esto es, si U es un universo del discurso y A es un conjunto de U , entonces: $U = A \cup \neg A \Rightarrow \neg A = U - A$

Se estudian a continuación el cálculo de los dos casos posibles de predicados negados en una regla con el enfoque semántico: *negación de predicados extensivos difusos* y *negación de predicados intensivos difusos*.

5.1.2.1.3.1.1 Negación de predicados extensivos difusos

Dada una relación extensiva difusa (H, B) , y un predicado \tilde{P} , es posible conocer la tuplas que satisfacen \tilde{P} , pero el problema es conocer las tuplas que verifican $\neg\tilde{P}$. Aplicando la *ley del tercero excluido*, la unión de la tuplas que cumplen \tilde{P} y las que no lo cumplen $\neg\tilde{P}$ constituyen todas las posibilidades del universo de hechos U , es decir $U = \tilde{P} \cup \neg\tilde{P}$. Así mismo, se considera que el GA de las tuplas de $\neg\tilde{P}$ es cero, y las de \tilde{P} es uno.

Con este enfoque, es necesario conocer todos los hechos posibles del universo del predicado. Si cada una de las n variables X_i del predicado \tilde{P} , toma valores en el dominio $Dom(X_i)$, el universo U puede calcularse como:

$$U = (Dom(X_1) \times Dom(X_2) \times \dots \times Dom(X_n)) \times \{[0,1]\} \quad \text{Eq. (5.13)}$$

Siendo $\{[0,1]\}$ el dominio del GA, que para los hechos que verifican el predicado \tilde{P} tiene valor 1.

Aplicando la *hipótesis del mundo cerrado* a los $Dom(X_i)$, los valores del dominio pueden obtenerse de la instancia de la BD, proyectando la relación (H, B) sobre el atributo X_i , por lo que se obtiene que $Dom(X_i) = \{a \mid a \in \Pi_{x_i}(B) \wedge X_i \in H\}$.

Dado que $Cálculo(\tilde{P}(X_1, \dots, X_n, \Phi(\dots))) \subseteq U$ y que el conjunto de tuplas inferido $\{(a_1, a_2, \dots, a_n) \mid a_i \in Dom(X_i), i = \{1, 2, \dots, n\}\}$ satisface el predicado \tilde{P} con un GA γ , se puede concluir que se satisface el predicado $\neg\tilde{P}$ con el GA $\xi = n(\gamma)$.

Una posible definición de ξ , es aportada en **[Est80]** y **[Tri79]**, aunque la más utilizada es la *negación fuerte difusa*:

$$n(a) = 1 - a, a \in [0,1] \quad \text{Eq. (5.14)}$$

Esta función verifica que $n(0) = 1, n(1) = 0, a \leq b \Rightarrow n(a) \geq n(b)$ y $n(n(a)) = a$.

Aplicando estas consideraciones, el cálculo de $\neg \tilde{P}$ sobre una BH extendida $\tilde{\Lambda}$ puede realizarse de la siguiente forma:

$$\begin{aligned} & \text{Cálculo}(\neg \tilde{P}(X_1, \dots, X_n, \gamma), \tilde{\Lambda}) = \\ & \{ \neg \tilde{P}(x_1, \dots, x_n, n(x_1, \dots, x_n, \tilde{p})) \in (\Pi_{x_1}(\tilde{p}) \times \dots \times \Pi_{x_n}(\tilde{p}) \times \{[0,1]\}) \} \end{aligned} \quad \text{Eq. (5.15)}$$

Donde \tilde{p} es la instancia de la relación que representa al predicado \tilde{P} . La función n se define como:

$$n(a_1, \dots, a_n, \tilde{p}) = \begin{cases} 1 - b & \text{si } \exists b \in [0,1] | (a_1, \dots, a_n, b) \in \tilde{p} \\ 1 & \text{en otro caso} \end{cases} \quad \text{Eq. (5.16)}$$

5.1.2.1.3.1.2 Negación de predicados intensivos difusos

Aunque la expresión que define el universo de hechos del *predicado intensivo* es la misma que la para los *predicados extensivos*, el problema se centra en el cálculo los dominios de las variables.

El dominio de una variable en un *predicado intensivo difuso* se compone de los dominios de esta variable cuando aparece en los diferentes predicados o sub-predicados, además de los dominios de todas las variables con las que se enlaza. Estos conceptos, se expresan formalmente con las siguientes definiciones: *clausura transitiva de una variable*, *dominio de una variable en un predicado* y *dominio de una variable en un sub-predicado*.

- ❖ **Definición 5.12. Clausura transitiva de una variable.** Sea \tilde{P} un predicado (o sub-predicado) definido por una RGGA y $X_i | i \in [1, n]$ una variable que ocurre en \tilde{P} . Se denomina *clausura transitiva de la variable X_i en \tilde{P}* y se nota por $Cl_a(X_i, \tilde{P}(X_1, \dots, X_n))$ al conjunto de variables que están enlazadas con ella mediante un comparador clásico o con un CDG.

Para el cálculo de la clausura transitiva de una variable en una regla, se puede aplicar cualquier algoritmo de clausura transitiva de grafos, considerando las comparaciones (clásicas o CDG) como arcos de un grafo cuyos nodos representan las variables ([Aho88] y [Ioa88]).

- ❖ **Definición 5.13. Dominio de una variable en un predicado.** Sea predicado \tilde{P} definido mediante una o varias reglas con grado de acoplamiento. El dominio de la variable $X_i | i \in [1, n]$ en el predicado \tilde{P} se define como:

$$\text{Dom}(X_i, \tilde{P}(X_1, \dots, X_n, \gamma_{\tilde{P}})) = \begin{cases} \emptyset & \text{si } \nexists j \in [1, n] | X_j = X_i \\ \bigcap_{\forall j \in [1, n] | X_j = X_i} \Pi_{X_j}(\tilde{p}) & \text{si } \tilde{P} \text{ extensivo} \\ \bigcup_{\forall j} \text{Dom} R(X_i, \tilde{P}_j(X_1, \dots, X_n, \gamma_{\tilde{P}_j})) & \text{si } \tilde{P} \text{ intensivo} \end{cases} \quad \text{Eq. (5.17)}$$

Donde cada \tilde{P}_j es el sub-predicado definido por una RGGa del predicado \tilde{P} .

❖ **Definición 5.14. Dominio de una variable en un sub-predicado.** Sea sub-predicado \tilde{P}_j definido mediante una RGGa. La función *dominio de sub-predicado* $DomR$, que calcula el dominio de una variable $X_i \mid i \in [1, n]$ en \tilde{P}_j , se expresa como:

$$DomR \left(X_i, \tilde{P}_i \left(X_1, \dots, X_n, \gamma_{\tilde{P}_i} \right) \right) = \bigcup_{\forall i \mid X_i \in Cla(X, \tilde{P}_i(X_1, \dots, X_n, \gamma_{\tilde{P}_i}))} \bigcup_{\forall j \mid X_j \text{ no es recursivo en } \tilde{Q}_j} \bigcap_{\forall k \in [1, n_j] \mid X_i = X_j} Dom_{j,k}$$

Donde; $Dom_{j,k} = Dom \left(X_k, \tilde{Q}_j(Y_{j,1}, Y_{j,2}, \dots, Y_{j,n_j}, \gamma_{\tilde{Q}_j}) \right)$

La definición “ X_i no es recursivo en \tilde{Q}_j ” puede expresarse como $(\tilde{P} \neq \tilde{Q}_j) \vee (\tilde{P} = \tilde{Q}_j \wedge \exists k \mid (X_i = X_k \wedge X_i = Y_{j,k}))$, es decir, o bien \tilde{P} y \tilde{Q}_j no son el mismo predicado, o bien, si lo son, la variable X_i no ocupa la misma posición en ambos. Esta condición es necesaria para que las reglas recursivas no produzcan un desbordamiento al calcular $DomR$.

Estas dos últimas definiciones (5.13 y 5.14), son válidas incluso si la variable en cuestión aparece más de una vez en el predicado. En este caso, $Dom(\dots)$ calcula el dominio de cada ocurrencia de la variable por separado, pudiendo obtener dominios distintos. La única posibilidad de que el conjunto de valores satisfaga la regla, es que el valor que toma la variable esté en todos los dominios, esto es, en la intersección.

5.1.2.1.3.2 Enfoque sintáctico

En este caso, la implementación del predicado negado, se basa en las *leyes de la negación de la Lógica*:

$$\begin{aligned} \neg (A \vee B) &\rightarrow \neg A \wedge \neg B \\ \neg (A \wedge B) &\rightarrow \neg A \vee \neg B \end{aligned}$$

A modo de ejemplo, sea $P(X, Y)$ un predicado intensivo definido mediante una regla definida a partir de dos predicados extensivos:

$$P(X, Y) \leftarrow Q(X, Z) \wedge R(Z, Y)$$

Para calcular el predicado negado $\neg P(X, Y)$, se aplica las *leyes de negación de la Lógica*, obteniendo:

$$\neg P(X, Y) \leftarrow \neg Q(X, Z) \vee \neg R(Z, Y)$$

Sin embargo, para aplicar deducción es necesario normalizar la regla en forma clausular:

$$\begin{aligned} \neg P(X, Y) &\leftarrow \neg P_1(X, Y) \vee \neg P_2(X, Y) \vee \neg P_3(X, Y) \\ \neg P_1(X, Y) &\leftarrow \neg Q(X, Z) \vee R(Z, Y) \\ \neg P_2(X, Y) &\leftarrow Q(X, Z) \vee \neg R(Z, Y) \\ \neg P_3(X, Y) &\leftarrow \neg Q(X, Z) \vee \neg R(Z, Y) \end{aligned}$$

Este desglose de reglas es debido a que la negación $\neg P(X, Y)$ se calcula mediante la negación de $Q(X, Z)$ o la $R(Z, Y)$ o con la de ambas. Generalizando, si un predicado P se define a través de m predicados, el número de nuevas reglas a generar para normalizar de forma clausular es $2^m - 1$. Si m tiene un valor grande, el número de predicados se dispara y el cálculo del predicado negado es ineficiente para el algoritmo de deducción “abajo-arriba” usado en el modelo de GDB (5.1.2.2). Esta es la razón por lo que esta implementación de la negación de predicados no es considerada en el modelo.

5.1.2.2 Algoritmo de deducción

La RGGa definida en 5.1.2.1, es el soporte que permite almacenar y manipular la información difusa en GDB. Más aún, habilita la extracción de nueva información, no almacenada de forma explícita, utilizando algoritmos de deducción, como ya se vio en el modelo de BD previo al utilizado en GDB (4.5.6).

Básicamente, existen dos algoritmos deducción:

- a) Algoritmo de deducción “arriba-abajo”
- b) Algoritmo deducción “abajo-arriba”

El algoritmo **a)** se basa en el clásico algoritmo deducción de Prolog (4.3.1.4.1), mientras que el **b)** parte del algoritmo de Datalog (4.3.1.4.2). Es posible implementar ambos algoritmos, pero por cuestiones de eficiencia, GDB utilizará la opción **b)**, ya que se orienta al cálculo con relaciones mientras que **a)** se orienta al cálculo con tuplas. Esto es un inconveniente cuando se trabaja con grandes en bases de datos, como las que existen en G. Además, la opción **a)** requiere de estructuras adicionales para el almacenamiento de resultados parciales (no necesarias en el caso **b)**) y ante un mismo conjunto de datos de entrada y la misma consulta, **a)** realiza un porcentaje mayor de cambios y consultas a la BD ([Bla01]).

El algoritmo **a)** se denomina de “arriba-abajo” porque se parte de la cabeza de la regla (que define el predicado) y se expande siguiendo el cuerpo de la regla, instanciando las diferentes variables. En **b)**, el predicado es calculado desde abajo hacia arriba: desde el cuerpo hasta obtener la cabeza de la regla.

A continuación se detalla el algoritmo de deducción “abajo-arriba” utilizado en GDB.

5.1.2.2.1 Algoritmo de deducción “abajo-arriba”

Se analizó en 4.3.2, la equivalencia en entre una BDR y un conjunto de hechos y predicados. Esto da pie a construir un algoritmo de deducción utilizado el modelo relacional, cómo se describe en este apartado.

Antes de analizar, este algoritmo, es preciso mostrar dos definiciones. La primera define como se crean relaciones a partir de una BHE, y la segunda permite establece como se crea una BD a partir de estas relaciones.

❖ **Definición 5.15. Relación extensiva difusa generada partir de una base de hechos.**

Sea $\tilde{\Lambda}$ una BHE generada a partir de una BH Λ y $Q_i(Y_{i,1}, \dots, Y_{i,n_i})$ un predicado para el cual se verifica al menos un hecho de Λ . Entonces, Q_i da a lugar a una *relación extensiva difusa* (H, B) definida como:

$$H = \left\{ \left(Y_{i,1} : Dom(Y_{i,1}), C_{Y_{i,1}} \right), \dots, \left(Y_{i,n_i} : Dom(Y_{i,n_i}), C_{Y_{i,n_i}} \right) \right\}$$

$$B = \left\{ \left((Y_{i,1} : y_1, 1), \dots, (Y_{i,n_i} : y_{n_i}, 1) \right) \mid Q_i(y_1, \dots, y_{n_i}, 1) \in \tilde{\Lambda} \right\} \quad \text{Eq. (5.50)}$$

Donde $Dom(Y_{i,j}), \forall j \in [1, n_i]$ es un DDG.

Es decir, el predicado Q_i genera una relación extensiva, compuesta por las instanciaciones de sus variables que aparecen en la BH (por lo que tiene un GA igual a 1).

Partiendo de la definición de *relación extensiva difusa*, se puede dividir BH extendida conforme a los predicados que la han generado, creando una *base de datos de hechos extensivos* (BDHE).

❖ **Definición 5.16. Base de datos de hechos extensivos.** Una BH extendida $\tilde{\Lambda}$ puede expresarse como la unión del siguiente conjunto de particiones:

$$\tilde{\Lambda} = \cup_i P_{Q_i}(\tilde{\Lambda}) \mid \forall i, P_{Q_i} = \{ Q_i(y_1, \dots, y_{n_i}, 1) \in \tilde{\Lambda} \}$$

Si se considera que cada P_{Q_i} genera una *relación extensiva difusa*, se denomina *base de hechos extensivos* a la siguiente estructura $\{(\tilde{q}_i, \tilde{Q}_i)\}$:

$$\tilde{q}_i = \{ X_1, \dots, X_n, \gamma_{Q_i} \}$$

$$\tilde{Q}_i = \{ (a_1, \dots, a_{n_i}, 1) \mid Q_i(a_1, \dots, a_{n_i}, 1) \in \tilde{\Lambda} \}$$

Con $\gamma_{Q_i} = 1$, y siendo \tilde{q}_i el esquema de la relación y \tilde{Q}_i su instancia.

El algoritmo de deducción calcula los hechos verificados en un predicado, o lo que es equivalente, calcula la relación que genera dicho predicado. Esta relación se puede expresar (calcular) de la siguiente forma.

❖ **Definición 5.17. Relación asociada a una regla generalizada con grado de acoplamiento.** Sea $\tilde{P}(X_1, \dots, X_n, \Phi)$ un predicado definido por una RGGa, la relación asociada a este predicado es:

$$\Pi_H \left(\sigma_\Psi \left(\times_{\forall i | (\tilde{q}_i, \tilde{Q}_i) \in BDHE} (\tilde{q}_i) \right) \right) \quad \text{Eq. (5.51)}$$

Siendo Π , σ y \times los operadores relacionales (4.1.4) de proyección (filtrado de columnas), selección (filtrado de filas) y producto cartesiano (combinación de filas) respectivamente.

Cada \tilde{q}_i representa a las relaciones de la BDHE que intervienen en la regla y puede identificar a una *relación extensiva difusa* o una *relación intensiva difusa*, En este último caso, \tilde{q}_i se sustituirá por su consulta expandida, lo que generará una sub-consulta anidada.

Se define Ψ como:

$$\Psi \equiv \bigwedge_u \geq (\Theta_u^{\bar{}}(X_i, Y_{i,k}), \alpha_u) \wedge \bigwedge_v \geq (\Theta_v^{\bar{}}(Y_{j,k}, Y_{l,p}), \alpha_v) \wedge \bigwedge_w \Phi_w^{\theta}(\Theta_w^{\theta}(X_{j,k}, Y_{l,p}), \alpha_w)$$

Donde H el esquema de la *relación intensiva difusa*, definido como sigue:

$$H = \{ \tilde{q}_j.Y_{j,k} \mid \exists i \in [1, n], \geq (\Theta_i^{\bar{}}(X_i, Y_{i,k}), \alpha) \text{ aparece en } \Psi \} \cup \{ \gamma \in [0, 1] \}$$

Donde $\gamma = \Phi(\dots)$ es el GA de hecho con el predicado \tilde{P} calculado en base a los grados de acoplamiento de las variables $\Theta_u^{\bar{}}(X_i, Y_{j,k})$, $\Theta_v^{\bar{}}(X_{j,k}, Y_{l,p})$ y

$\Theta_w^{\theta}(X_{j,k}, Y_{l,p})$ y a los grados $\gamma_{\tilde{Q}_i}$ de los hechos de los predicados del cuerpo de la regla.

Dado que los hechos de la BDHE tiene GA igual a 1 con sus respectivos predicados, $\Phi(\dots)$ sólo dependerá de los grados de acoplamiento entre variables.

Como se ha visto, el cálculo hechos verificados por un predicado \tilde{P} definido por una RGGa, puede realizarse mediante un conjunto de operaciones relacionales. Cada tupla de la relación resultante, representa un hecho verificado por \tilde{P} . De hecho, estas operaciones se traducen como una única consulta a la BDR, a menos que exista en el cuerpo de la regla otras RGGa. En tal caso, serán necesario pasos adicionales para resolver las sub-consultas (una por cada RGGa en el cuerpo) antes de obtener la relación final.

Para mostrar el funcionamiento del algoritmo de deducción de un predicado, se considerará el siguiente ejemplo.

Sea un predicado P definido mediante la regla:

$$P(X, Y) \leftarrow Q(X_Q, Z_Q) \wedge R(Z_R, Y_R) \wedge (X, X_Q) \wedge (Y, Y_R) \wedge (Z_Q, Z_R)$$

Sea una BH $\Lambda = \{Q(a, b), Q(c, d), Q(e, f), R(g, h), R(i, j)\}$

Asumiendo el axioma del mundo cerrado (todo lo que no es conocido es falso), se cumple que $\text{Dominio}(X_Q) = \{a, c, e\}$, $\text{Dominio}(Z_Q) = \{b, d, f\}$, $\text{Dominio}(Z_R) = \{g, i\}$ y $\text{Dominio}(Y_R) = \{h, j\}$. Siendo todos los dominios del tipo clásico.

La RGGa asociada a P es la siguiente:

$$\tilde{P}(X, Y, \gamma_P) \leftarrow \tilde{Q}(X_Q, Z_Q, \gamma_Q) \wedge \tilde{R}(Z_R, Y_R, \gamma_R) \wedge \Psi$$

$$\Psi \equiv \geq (\Theta^=(X, X_Q), \alpha_Q) \wedge \geq (\Theta^=(Y, Y_R), \alpha_R) \wedge \geq (\Theta^=(Z_Q, Z_R), \alpha_{QR})$$

La BH extendida queda como:

$$\tilde{\Lambda} = \{ \tilde{Q}(a, b, 1), \tilde{Q}(c, d, 1), \tilde{Q}(e, f, 1), \tilde{R}(g, h, 1), \tilde{R}(i, j, 1) \}$$

Aplicando las definiciones anteriores, es posible realizar las siguientes transformaciones:

a) La BDHE queda representada mediante las dos relaciones (q, Q) y (r, R) :

$$BDHE = \{ (q, Q), (r, R) \}$$

Siendo:

$$(q, Q) = (\{ X, Y \}, \{ (a, b), (c, d), (e, f) \})$$

$$(r, R) = (\{ X, Y \}, \{ (g, h), (i, j) \})$$

b) σ_Ψ se evalúa como:

$$\sigma_\Psi \equiv \sigma_{\geq(\Theta^=(Z_Q, Z_R), \alpha_{QR})} = \sigma_{\geq(\Theta^=(\tilde{q}.Y, \tilde{r}.X), \beta)}$$

c) El cálculo del predicado \tilde{P} queda expresado mediante la relación (\tilde{p}, \tilde{P}) :

$$(\tilde{p}, \tilde{P}) = (\{ X, Y, G \}, \text{Cálculo}(P(X, Y, \gamma), \Lambda))$$

Se proyectan a continuación los atributos que permiten el acoplamiento entre las variables de cuerpo de la regla y la cabeza: $\tilde{q}.X$ y $\tilde{r}.Y$

$$\text{Cálculo}(P(X, Y, \gamma), \tilde{\Lambda}) = \Pi_{\tilde{q}.X, \tilde{r}.Y, \Phi(\Theta^=(\tilde{q}.Y, \tilde{r}.X))} \left(\sigma_{\geq(\Theta^=(\tilde{q}.Y, \tilde{r}.X), \beta)} (\tilde{q} \times \tilde{r}) \right)$$

De forma gráfica, este cálculo puede representarse como se muestra en la siguiente ilustración.

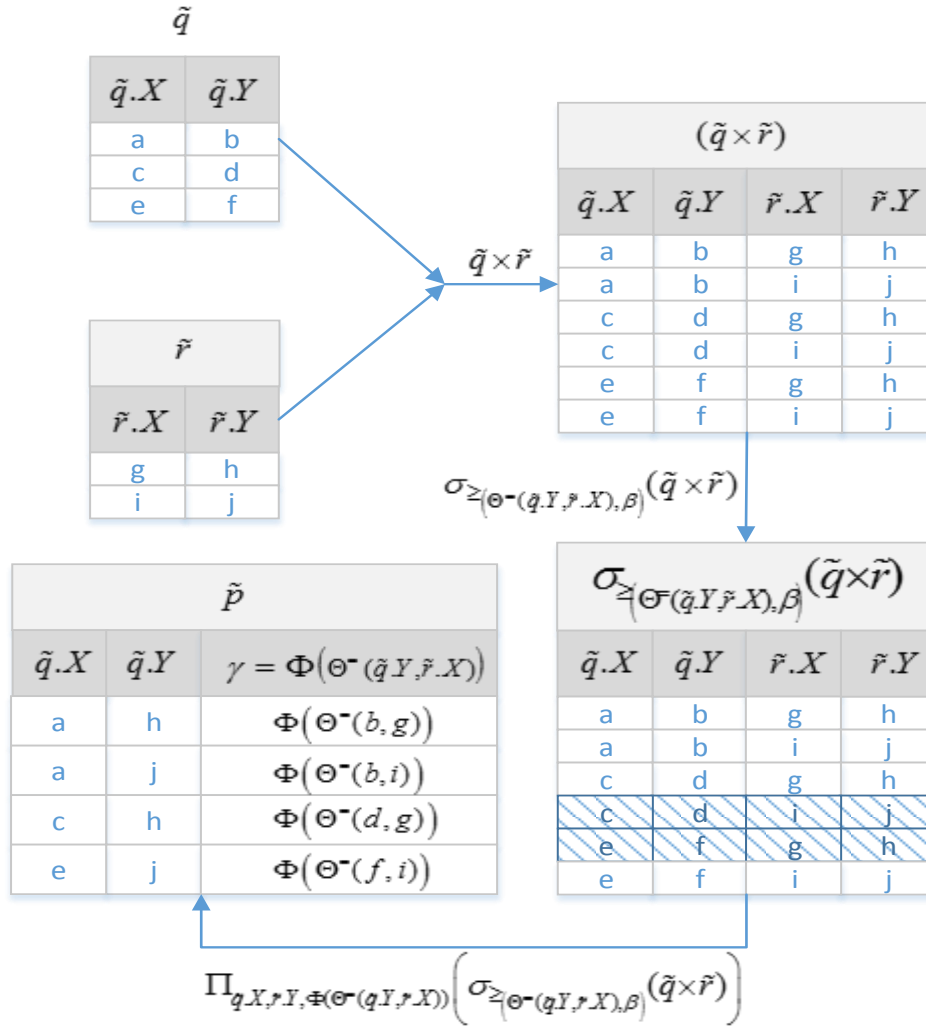


Ilustración 60. Ejemplo del algoritmo de deducción GDB.

Considerando que $\Theta^-(d, i) < \beta$ y que $\Theta^-(f, g) < \beta$ por los que no cumple la condición del operador de selección σ .

Por tanto, dado un predicado \tilde{P} , el algoritmo de deducción “abajo-arriba” que calcula los hechos verificados por \tilde{P} , es equivalente a la expansión del *predicado intensivo difuso* definido por la RGGA asociada a \tilde{P} , aplicado a la *base de datos de hechos extensivos*. El algoritmo finaliza con una única consulta, requiriendo consulta anidadas, sólo en el caso de que existan predicados definidos mediante RGGA en el cuerpo de la RGGA derivada de \tilde{P} .

5.1.2.2.1.1 Cálculo de reglas disyuntivas

La consulta asociada a un predicado definido mediante una disyunción de $\tilde{P}_i \mid i \in [1, r]$ reglas generalizadas con grado de acoplamiento es la siguiente:

$$\Pi_{X_1, \dots, X_n, \Omega(b_1, \dots, b_r)} \left(\bigcup_{i \in [1, r]} \sigma_{\Psi_i} \left(\times_{\forall j | (\tilde{q}_j, \tilde{Q}_j) \in BDHE_i} (\tilde{q}_j) \right) \right) \quad \text{Eq. (5.52)}$$

Donde $\Omega (b_1, \dots, b_r)$ se calcula en base a los $b_i \mid i \in [1, r]$ grados de acoplamiento de los hechos con el predicado \tilde{P}_i . Este valor b_i se evalúa como sigue:

$$b_i = \begin{cases} \Phi_i(\dots) & \text{si } \Pi \left\{ \sigma_{(X_1, \dots, X_n)=(a_1, \dots, a_n)} \wedge \Psi_i \left(\times_{\forall j | (\tilde{q}_j, \tilde{Q}_j) \in BDHE_i} (\tilde{q}_j) \right) \right\} \neq 0 \\ 0 & \text{en otro caso} \end{cases}$$

Eq. (5.53)

Siendo cada $\Phi_i(\dots)$ es el GA del hecho con el predicado P según la regla \tilde{P}_i .

5.1.2.2.1.2 Cálculo de reglas recursivas

El algoritmo Datalog “abajo-arriba” implementa el cálculo de un predicado mediante una consulta a una BDR. Cuando existen predicados recursivos (4.5.5.1) el algoritmo calcula la consulta, en la que aparecería de nuevo el predicado, que será sustituido por la misma consulta, y así sucesivamente hasta que la relación final no cambie. De forma más formal, el algoritmo de cálculo para reglas recursivas es:

- a) Calcular (\tilde{p}, \tilde{P})
- b) Mientras la instancia de \tilde{p} cambie:
 - 1) Sustituir la instancia (\tilde{p}, \tilde{P}) por la instancia de la relación resultante de la consulta relacional.

A modo de ejemplo, consideremos una BH extendida $\tilde{\Lambda}$ y el predicado \tilde{P} definido a través de las reglas generalizadas con grado de acoplamiento P_1 y P_2 .

$$\begin{aligned} \tilde{P}(X, Y, \gamma_p) &\leftarrow \tilde{Q}(X_{\tilde{Q}}, Z_{\tilde{Q}}, \gamma_{\tilde{Q}}) \wedge \Psi_1 \\ \gamma_p &= \Phi(\Theta^-(X, X_{\tilde{Q}}), \Theta^-(Y, Y_{\tilde{Q}}), \gamma_{\tilde{Q}}) \\ \Psi_1 &\equiv \geq (\Theta^-(X, X_{\tilde{Q}}), \alpha) \wedge \geq (\Theta^-(Y, Y_{\tilde{Q}}), \alpha) \\ \tilde{P}(X, Y, \gamma_p) &\leftarrow \tilde{Q}(X_{\tilde{Q}}, Z_{\tilde{Q}}, \gamma_{\tilde{Q}}) \wedge \tilde{P}(Z_{\tilde{P}}, Y_{\tilde{P}}, \gamma_{\tilde{P}}) \wedge \Psi_2 \\ \gamma_p &= \Phi(\Theta^-(X, X_{\tilde{Q}}), \Theta^-(Y, Y_{\tilde{P}}), \Theta^-(Z_{\tilde{Q}}, Z_{\tilde{P}}), \gamma_{\tilde{Q}}, \gamma_{\tilde{P}}) \\ \Psi_2 &\equiv \geq (\Theta^-(X, X_{\tilde{Q}}), \alpha) \wedge \geq (\Theta^-(Y, Y_{\tilde{P}}), \alpha) \wedge \geq (\Theta^-(Z_{\tilde{Q}}, Z_{\tilde{P}}), \beta) \\ \tilde{\Lambda} &= \{\tilde{Q}(a, b, 1), \tilde{Q}(a, c, 1), \tilde{Q}(c, g, 1), \tilde{Q}(d, e, 1), \tilde{Q}(d, f, 1)\} \end{aligned}$$

La consulta relacional asociada al predicado \tilde{P} es la siguiente:

$$\begin{aligned}
 (\tilde{p}, \tilde{P}) &= \Pi_{\tilde{q}.X, \tilde{q}.Y, \gamma_{\tilde{q}}}(\tilde{q}) \cup \Pi_{\tilde{q}.X, \tilde{p}.Y, \Phi}(\Theta = (\tilde{q}.Y, \tilde{p}.X), \tilde{q}.Y_{\tilde{q}}, \tilde{p}.Y_{\tilde{p}}) (\sigma_{\geq(\Theta = (\tilde{q}.Y, \tilde{p}.X), \beta)}(\tilde{q} \times \tilde{p})) = \\
 &= \Pi_A(\tilde{q}) \cup \Pi_B(\sigma_c(\tilde{q} \times \tilde{p}))
 \end{aligned}$$

Se verifica que $\gamma_{\tilde{q}} = 1$ dado que los hechos de \tilde{A} tienen un GA igual a 1 respecto al predicado \tilde{q} .

De forma gráfica, la aplicación del algoritmo sobre el ejemplo, se muestra en la siguiente ilustración, donde la instancia de la relación final no cambia hasta el final de la tercera iteración, y por tanto el algoritmo finaliza.

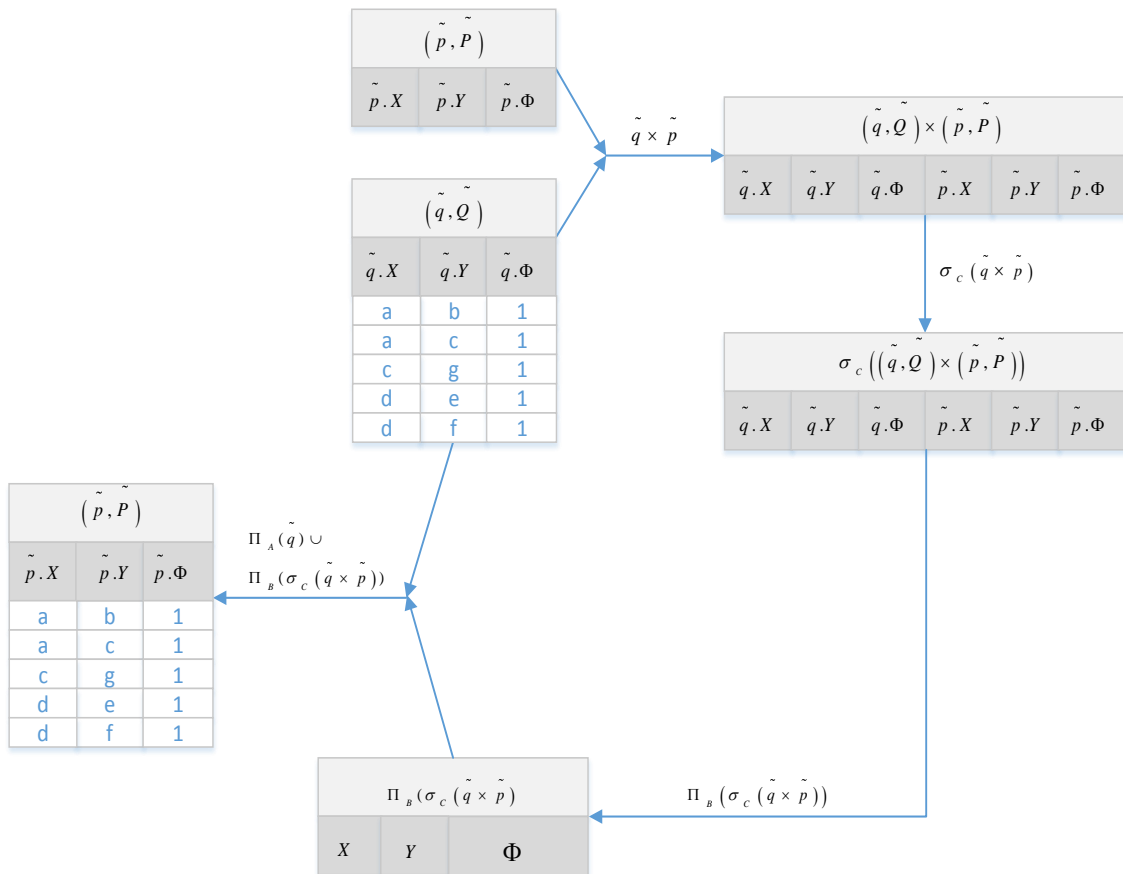


Ilustración 61. Paso 1 del algoritmo de deducción con reglas recursivas.

Considerandos que sólo $\Theta = (c, c) \geq \beta$ y $\Theta = (c, a) \geq \beta$ cumplen la condición del operador de selección σ_c .

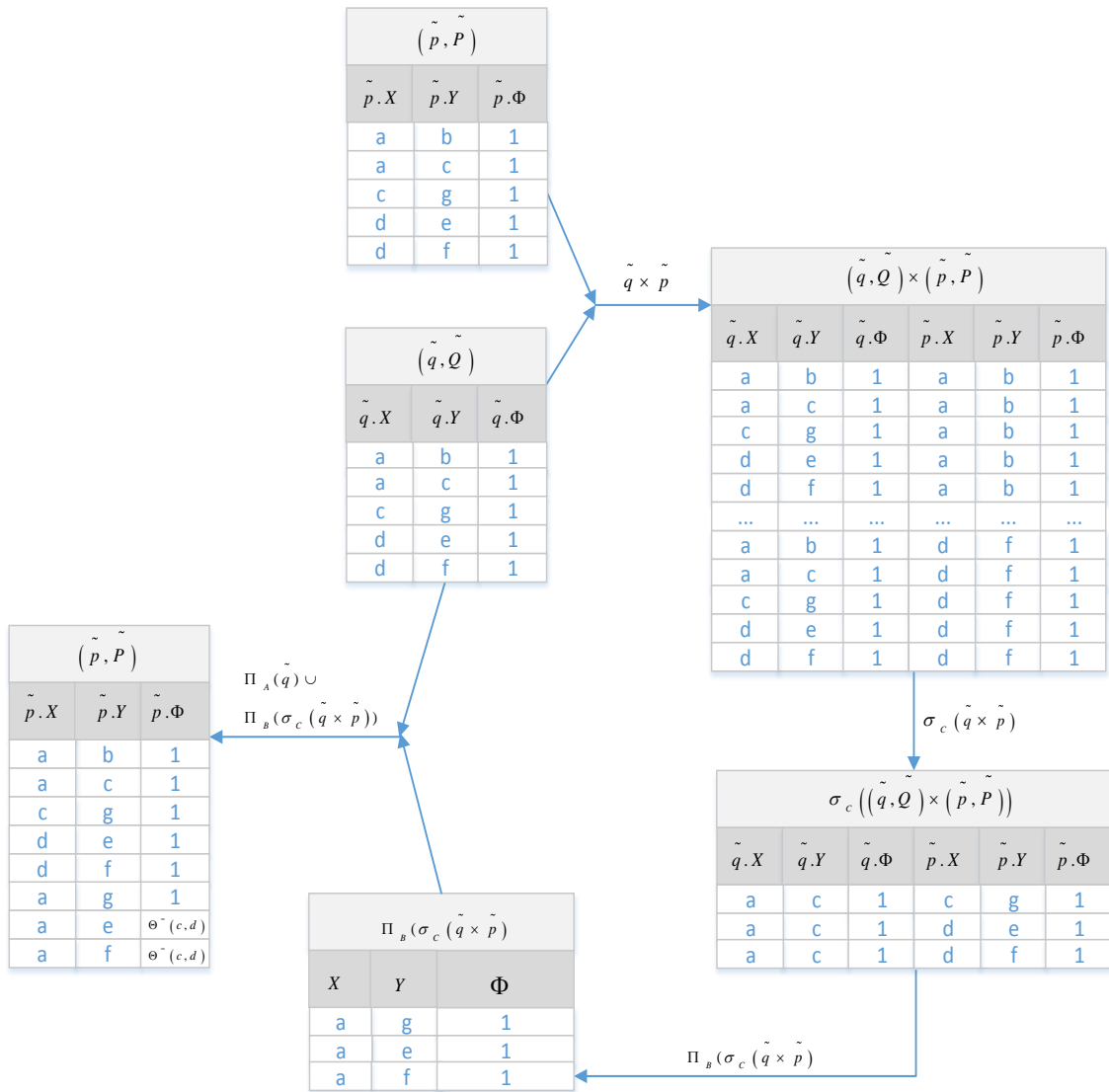


Ilustración 62. Paso 2 del algoritmo de deducción con reglas recursivas.

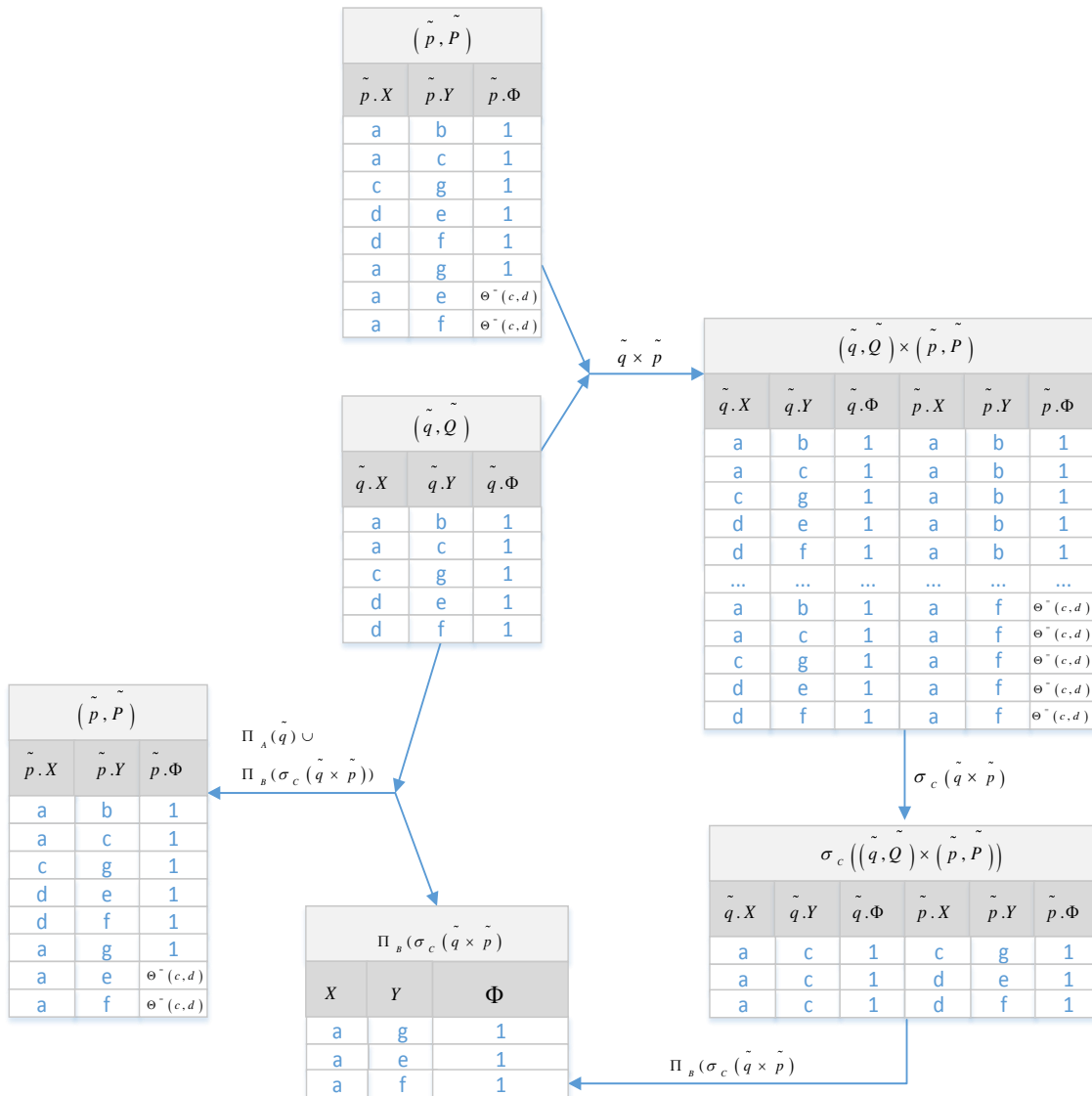


Ilustración 63. Paso 3 del algoritmo de deducción con reglas recursivas.

5.1.2.3 Base de meta-conocimiento difuso y base de reglas

En un SGBDR, el catálogo es el lugar donde se encuentra la información sobre los datos almacenados (los llamados metadatos o meta-conocimiento): usuarios, permisos, etcétera.

Este catálogo, que forma parte de la arquitectura de GDB (5.5), es extendido para almacenar:

- a) *La información difusa*. Utilizando la FMB descrita en 5.1.1.3.
- b) *La deductiva*. Mediante la denominada *base de reglas*

Es en la base de reglas (BR) donde se almacena toda la información necesaria para representar reglas, los predicados que las componen y las *tablas intensivas* asociadas. La BR fue formulada inicialmente en [Med97] implementada en FREDDI y extendida en [Bla01]. La composición de la *base de reglas* se puede expresar mediante un diagrama entidad-relación ([Che76] y [Bar92]). Este diagrama permite definir la estructura y la conexión entre las tablas. La definición de la *base de reglas* en este tipo de diagrama es la siguiente:

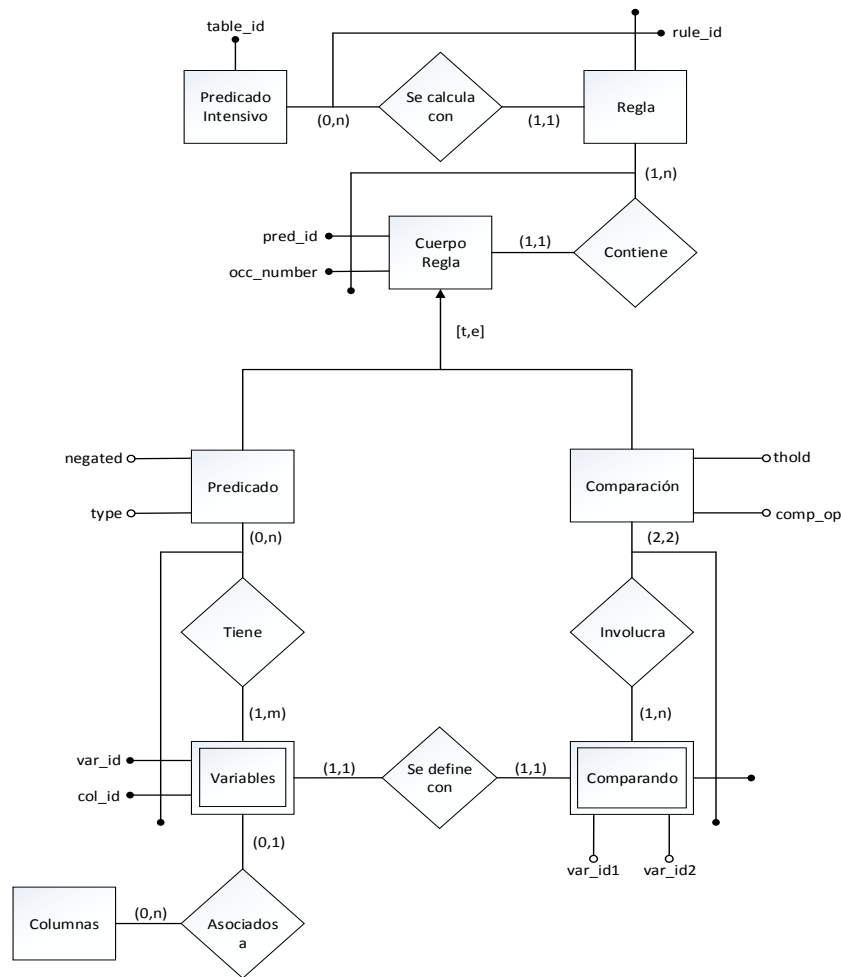


Ilustración 64. Modelo entidad-relación de la base de reglas de GDB.

Las tablas asociadas a este modelo entidad-relación son las siguientes (se pueden encontrar la descripción detallada en **[Bla01]**):

- DED_INT_TABLE_DESCRIPTION.** Almacena los predicados intensivos (`TABLE_ID`) definidos como la conjunción de reglas (`RULE_ID`).
- DED_RULE_DESCRIPTION.** Describe cada una de las reglas como una secuencia de predicados extensivos, intensivos y comparaciones concatenados con el operador conjunción. Cada predicado puede aparecer negado y ocurrir varias veces en la misma regla. El par (`TABLE_ID`, `RULE_ID`) identifica a la regla, el atributo (`PRED_ID`) identifica cada predicado de la regla, `OCC_NUMBER` establece la posición del predicado en la regla, `NEGATED` si el predicado está negado o no y `TYPE` el tipo de predicado (0 para extensivo, 1 para intensivo y 2 para comparación).
- DED_PREDICATE_DESCRIPTION.** Almacena el orden de las variables en cada uno de los predicados. La misma variable puede aparecer más de una vez en cada predicado, pero varias ocurrencias se distinguen por su posición dentro del predicado. Una variable (`VAR_ID`) ocupa una posición (`COL_ID`) dentro de un predicado (`PRED_ID`) que aparece en una posición dada (`OCC_NUMBER`) de una regla (`TABLE_ID`, `RULE_ID`).
- DED_COMPARISON_DESCRIPTION.** Describe las condiciones (tipo especial de predicados binarios) que sólo poseen dos variables. Una condición (`COMP_OP`) compara dos variables (`VAR_ID1` y `VAR_ID2`). Esta condición aparece en una posición dada (`OCC_NUMBER`) de una regla (`TABLE_ID`, `RULE_ID`) que define a un predicado

intensivo (PRED_ID). Los posibles valores de COMP_OP comienzan en cero y representan los cinco comparadores clásicos (=, <, >, <= y >=), más los 14 comparadores difusos generalizados vistos en [5.1.1.2](#).

5.1.2.4 DFSQL en el modelo GDB

Para dotar de las capacidades difusas y deductivas al modelo GDB, se parte de las modificaciones difusas realizadas al SQL en FREDDI (FSQL) y extendidas en **[Bla01]**, obteniendo un nuevo lenguaje llamado DFSQL.

El DFSQL se compone de un lenguaje para la definición de los objetos necesarios (DDL difuso deductivo) y otro lenguaje para la manipulación de la información (DML difuso deductivo),

Dentro de las posibilidades del lenguaje DFSQL, GDB-GUI sólo hará uso de ciertas sentencias y determinadas capacidades de ellas. A continuación se describen las sentencias de DDL y DML utilizadas.

5.1.2.4.1 DDL difuso deductivo

El DDL es el encargado de dar soporte a la definición de elementos deductivos. Se definen las siguientes sentencias del lenguaje: “create intensiva table”, “create rule”, “delete rule” y “drop intensiva table”

5.1.2.4.1.1 Sentencia “create intensiva table”

create_intensiva_table: CREATE INTENSIVA TABLE nombre_tabla ‘(‘ lista_columnas ‘)’ ‘;’

Donde el símbolo “lista_columnas” ya fue definido para el DDL de FSQL: ([5.1.1.4.1.1 Sentencia “create table”](#)).

Esta sentencia tiene asociada un conjunto de disparadores (secuencias de código que se ejecutan automáticamente en el seno de la BD cuando ocurre un determinado evento) que permiten la sincronizar la BD con reglas lógicas.

El modelo GDB implementa las *tablas intensivas* mediante una relación vacía (sin tuplas) que tiene asociada un conjunto de reglas (almacenadas en la *base de reglas* ver [5.5](#)) que generarán su contenido. La relación es eliminada al finalizar la consulta deductiva. Este es uno de los rasgos propios de las relaciones intensivas: son relaciones temporales. Los datos que resulten de una deducción (almacenados en una *tabla intensiva*) no se diferencian de los datos que están explícitamente almacenados en una relación clásica, excepto por el atributo extra que almacena el GA de los hechos (tuplas) con la regla.

5.1.2.4.1.2 Sentencia “create rule”

Esta sentencia asocia un predicado intensivo a una RGGA (que debe estar previamente creada).

create_rule: CREATE RULE FOR nombre_relación_intensiva ‘(‘ lista_variables_cabeza ‘)’ AS lista_predicados ‘;’

lista_variables_cabeza: identificador { ‘,’ identificador }+

lista_predicados: predicado {AND predicado}*

predicado: [NOT] (predicado_intensivo | predicado_extensivo | comparación)

predicado_intensivo: nombre_predicado '(' lista_variables_cuerpo ')'

lista_variables_cuerpo: identificador acoplamiento [grado] { ',' identificador acoplamiento [grado] }+

acoplamiento: (FEQ | NFEQ) identificador '.' Identificador

grado: THOLD numero

predicado_extensivo: nombre_predicado '(' lista_vars_pred_extensivo ')'

lista_vars_pred_extensivo: variable_pred_extensivo { ',' variable_pred_extensivo }*

variable_pred_extensivo: identificador SOURCE referencia_columna acoplamiento [grado]

comparación: identificador '.' identificador comparador identificador '.' identificador [grado]

comparador: FEQ | FGT | FGEQ | FLT | FLEQ | MGT | MLT | NFEQ | NFGT | NFGEQ | NFLT | NFLEQ | NMGT | NMLT

Las variables de los predicados están cualificadas con el nombre del predicado, por lo que no existen dos variables con el mismo nombre.

El símbolo "lista_variables_cuerpo" indica el conjunto de variables que aparecerán en el cuerpo de la regla, de modo que puedan ser instanciadas por los predicados del mismo. Si se desea unificar dos variables en el cuerpo de la regla, se puede utilizar el comparador clásico '=', o bien el CDG FEQ, si se quiere utilizar una igualdad difusa. En tal caso se puede establecer el umbral mínimo de cumplimiento 'THOLD' al valor deseado.

Las reglas son una conjunción de predicados difusos, ya sean intensivos o extensivos. Cuando el predicado sea extensivo, las variables deben enlazarse con atributos de sus relaciones extensivas asociadas, mediante la cláusula "SOURCE", de modo que el motor de inferencia sea capaz de obtener la información necesaria del predicado. En el caso de predicados intensivos, las variables se instanciarán con el resultado del cálculo del propio predicado.

5.1.2.4.1.3 Sentencia "delete rule"

Con la siguiente sentencia se permite el borrado de una regla (o todas utilizando "*"), previamente creadas:

delete_rule: DELETE_RULE ('*' | numero) FOR nombre_relacion_intensiva ';'

Esta sentencia borra la regla, pero no la tabla intensiva asociada. Para ello es necesario la sentencia "drop intensiva table".

5.1.2.4.1.4 Sentencia "drop intensiva table"

Al ejecutar la siguiente sentencia, se borrará la tabla intensiva, que previamente no debe estar asociada a ninguna regla.

drop_intensiva_table: DROP INTENSIVA TABLE nombre_relacion_intensiva ';'

5.1.2.4.2 DML difuso deductivo

Con DML es posible la manipulación de información deductiva. Se implementa a través de la sentencia “select” difusa (5.1.1.4.2.1) por lo que no necesita una sintaxis adicional. Al procesar la sentencia, se analizará la *base de meta-conocimiento difuso* y la *base de reglas* (5.5), para detectar los atributos con componentes difusas y deductivas y así desencadenar las acciones necesarias.

5.1.2.5 Implementación de intervalos: macros

En ciertas ocasiones es necesario recuperar información (eventos) que se encuentre, dado un valor inicial, en un intervalo determinado. Los eventos que pertenezcan a este entorno, deberán de cumplir una serie de condiciones, veamos como calcularlas.

❖ **Definición 5.18. Conjunto de eventos anterior y posterior a un evento dado en GIADA.** Sea una pareja (e, t^e) , donde e es un evento y t^e es el tiempo en el que ha ocurrido dicho evento. Los conjuntos de eventos $A = \{(a_1, t_1^a), (a_2, t_2^a), \dots, (a_n, t_n^a)\}$ y $B = \{(b_1, t_1^b), (b_2, t_2^b), \dots, (b_m, t_m^b)\}$ que han sucedido en unos entornos de tiempo $\varepsilon > 0$ y $\delta > 0$ antes de y después de e (respectivamente) pueden calcularse como:

$$A = \{(a_x, t_x^a) \mid t^e > t_x^a \wedge t^e < t_x^a + \varepsilon\} \quad \text{Eq. (5.18)}$$

$$B = \{(b_x, t_x^b) \mid t^e < t_x^b \wedge t^e > t_x^b - \delta\} \quad \text{Eq. (5.19)}$$

Este cálculo del conjunto de eventos dentro de un intervalo puede realizarse con una consulta SQL clásica, pero si se desea crear una regla que refleje esta restricción semántica, es necesario recurrir el DFSQL.

La sintaxis de DFSQL (5.1.2.4) no permite expresar el tipo de condiciones de los conjuntos A y B (Eq. (5.18) y Eq. (5.19)) que involucran a dos operadores: un comparador (“<” ó “>”) y un operador aritmético (“+” ó “-”). Por ello, es necesario realizar una extensión que recoja la sintaxis y la semántica necesaria.

Para minimizar cambios, la idea principal de implementación consiste en considerar la parte derecha de la comparación como una macro, que se expandirá (aplicará) antes de ejecutar el proceso de deducción. En el análisis semántico será ignorada. Durante la traducción a SQL, la macro será eliminada y añadida después al SQL resultante.

5.1.2.5.1 Modificaciones para el manejo de macros

A la hora de implementar las macros, es necesario realizar ciertas modificaciones: crear una tabla dentro del diagrama entidad-relación de la base de reglas (para almacenar la información de las macros), modificar la base de meta-conocimiento difuso y base de reglas (para definir la estructura y relación de la nueva tabla) y cambiar la sintaxis y la semántica del DFSQL (para implementar las operaciones necesarias).

5.1.2.5.2 Modificaciones al diagrama entidad-relación de la base de reglas

Partiendo del diagrama entidad-relación para el almacenamiento de reglas (5.1.2.3), se realiza una extensión para incluir una relación que almacene la información de las macros (abajo a la derecha en la ilustración):

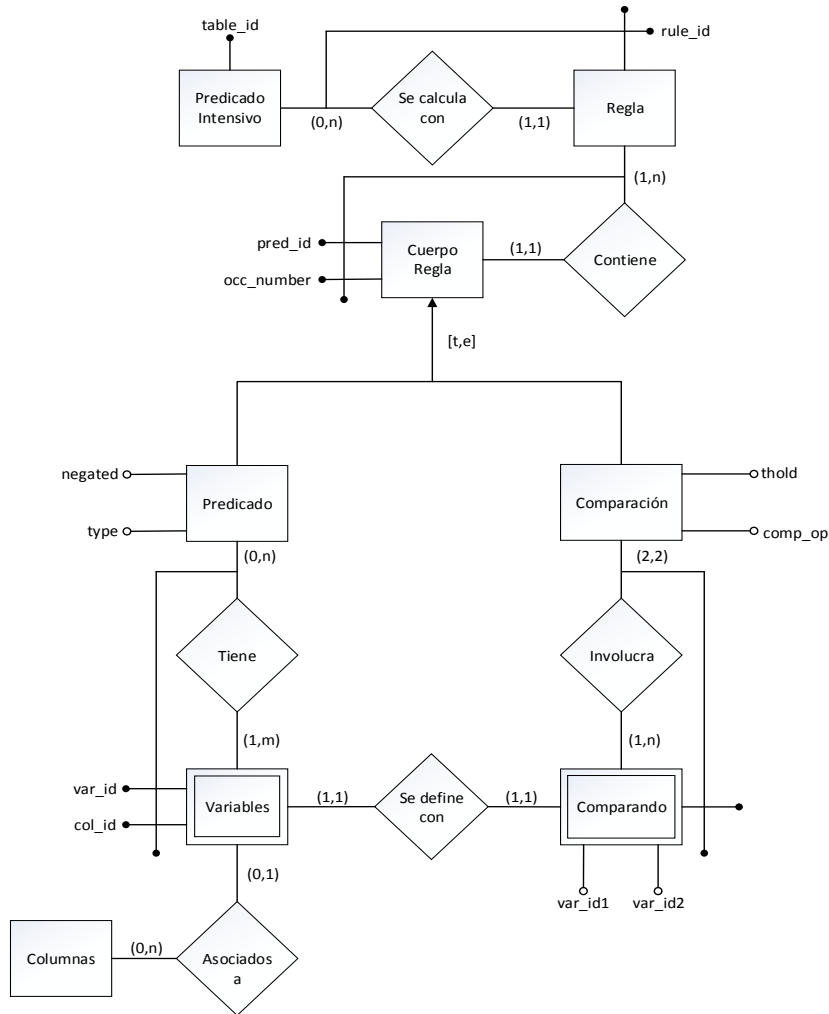


Ilustración 65. Modelo entidad-relación extendido de la base de reglas de GDB.

La nueva relación llamada `DED_MACRO_DESCRIPTION`, almacena la información relativa a las macros de cada predicado. Una variable (`VAR_ID`) ocupa una posición (`COL_ID`) dentro de un predicado (`PRED_ID`) que aparece en una posición dada (`OCC_NUMBER`) de una regla (`TABLE_ID`, `RULE_ID`). Se guarda el operador de la macro (`OP_MACRO`) y la constante numérica de esta macro (`MACRO_CONSTANT`). Operador y variable numérica serán las utilizadas en la expansión de la macro.

5.1.2.5.3 Modificaciones a la base de meta-conocimiento difusa y base de reglas

La nueva tabla definida en el punto anterior, pertenece a la base de meta-conocimiento deductivo y base de reglas (5.1.2.3) dentro de la arquitectura del modelo de datos de GDB (5.5). La tabla almacena los datos sobre macros y se denomina “`DED_MACRO_DESCRIPTION`”.

Una macro se define a través de un operador (`OP_MACRO` que puede ser: “+” ó “-”) y una constante numérica (`MACRO_CONSTANT`). La macro se aplica sobre una condición que aparece en una posición dada (`OCC_NUMBER`) de una regla (`TABLE_ID` y `RULE_ID`) que define a un predicado intensivo (`PRED_ID`).

La sentencia SQL que define esta relación es:


```
CREATE TABLE "SYS"."DED_MACRO_DESCRIPTION"
(
  "TABLE_ID" NUMBER,
  "RULE_ID" NUMBER,
  "PRED_ID" NUMBER,
  "OCC_NUMBER" NUMBER,
  "OP_MACRO" CHAR(1 BYTE),
  "MACRO_CONSTANT" VARCHAR2(100 BYTE),
  PRIMARY KEY ("TABLE_ID", "RULE_ID", "PRED_ID", "OCC_NUMBER"),
  FOREIGN KEY ("TABLE_ID", "RULE_ID", "PRED_ID", "OCC_NUMBER")
  REFERENCES "SYS"."DED_COMPARISON_DESCRIPTION" ("TABLE_ID",
  "RULE_ID", "PRED_ID", "OCC_NUMBER"),
  DELETE CASCADE ENABLE
);
```

5.1.2.5.4 Modificaciones a DDL de DFSQL

La modificación de la sintaxis del DFSQL se centra en la sentencia “create rule”. En concreto, en rodear con corchetes el operador derecho del comparador para definir una macro. Las modificaciones respecto a la definición inicial ([5.1.2.4.1.1](#)) se destacan en **rojo**.

create_rule: CREATE RULE FOR nombre_relación_intensiva ‘(lista_variables_cabeza ‘) AS lista_predicados ‘;’

lista_variables_cabeza: identificador { ‘,’ identificador }+

lista_predicados: predicado {AND predicado}*

predicado: [NOT] (predicado_intensivo | predicado_extensivo | comparación)

predicado_intensivo: nombre_predicado ‘(lista_variables_cuerpo ‘)’

lista_variables_cuerpo: identificador acoplamiento [grado] { ‘,’ identificador acoplamiento [grado]}+

acoplamiento: (FEQ | NFEQ) identificador ‘.’ Identificador

grado: THOLD numero

predicado_extensivo: nombre_predicado ‘(lista_vars_pred_extensivo ‘)’

lista_vars_pred_extensivo: variable_pred_extensivo { ‘,’ variable_pred_extensivo }*

variable_pred_extensivo: identificador SOURCE referencia_columna acoplamiento [grado]

comparación: identificador ‘.’ identificador comparador **operador_derecho** [grado]

operador_derecho: identificador ‘.’ identificador | ‘[’identificador ‘.’ identificador **operador_macro** ‘]’

operador_macro: ‘+’ | ‘-’

comparador: FEQ | FGT | FGEQ | FLT | FLEQ | MGT | MLT | NFEQ | NFGT | NFGEQ | NFLT | NFLEQ | NMGT | NMLT

5.1.2.5.5 Modificaciones a la semántica del DFSQL

En la fase de análisis semántico, la parte relacionado con la macro es ignorada. La implementación real se realiza tras la traducción a SQL, donde la macro es incorporada, como se verá en siguiente punto.

5.1.2.5.6 Ejemplo de macro

Veamos un ejemplo de uso de una macro. Se quiere obtener los eventos de IS en un entorno de 8ms respecto a un evento de GDS (una descripción detallada de cómo crear un ejemplo puede encontrarse en [5.4](#)).

La sintaxis en DFSQL para crear la tabla intensiva y la regla sería la siguiente:

```
//===== Intensional table =====
create intensional table GDB_INT_MACRO_EXAMPLE(
  GDS_EVENTTIME NUMBER(15),
  IS_EVENTTIME NUMBER(15));
//===== End of intensional table =====
//===== Rule =====
create rule for GDB_INT_MACRO_EXAMPLE (X,Y)
as
  GDB_GDSEVENT(X source gds_eventtime) AND
  GDB_ISEVENT(Y source is_eventtime) AND
  (X < [Y + 8]) AND
  (X > [Y - 8]);
//===== End of rule =====
```

La traducción de la regla a SQL clásico es la siguiente :

```
SELECT GDB_INT_MACRO_EXAMPLE.* FROM
( SELECT GDS_EVENTTIME GDS_EVENTTIME,
  IS_EVENTTIME IS_EVENTTIME,
  DFSQL_FUNCTIONS.MIN(1) "CDEG(*)"
FROM GIADA.GDB_GDSEVENT P1,
  GIADA.GDB_ISEVENT P2
WHERE GDS_EVENTTIME < IS_EVENTTIME+8 AND
  GDS_EVENTTIME > IS_EVENTTIME-8)
GDB_INT_MACRO_EXAMPLE
```

5.1.2.6 Implementación de constantes

La representación y manejo de constantes como parte de una RGGA, requiere de un predicado especial denominado predicado constante.

- ❖ **Definición 5.19. Predicado constante.** Sea un predicado $P(Z)$, donde Z es una variable. Se define el *predicado constante* $P^c(Z | a)$ asociado a la constante a como el predicado $P^c(Z)$ que verifica a un único hecho fijo y predeterminado $P^c(a)$. Siendo $Z | a$ la sustitución del término Z por la constante a .

Dado que a cada predicado, se le puede asociar una tabla, un predicado constante $P^c(Z | a)$ tendría asociada la siguiente relación:

Tabla 25. Relación asociada a un predicado constante.

$P^c(Z a)$
Z
a

Donde la primera fila indica el nombre de la tabla, la segunda el nombre del atributo y la tercera el valor del atributo. Se supone que el nombre del atributo coincide con el nombre de la variable del predicado.

La sustitución $Z | a$ ha de aplicarse en todas y cada una de las ocurrencias de Z . Más formalmente, sea un predicado definido como sigue:

$$R(X_1, X_2, \dots, X_n) \leftarrow Q_1(Y_{1,1}, \dots, Y_{1,n_1}) \wedge \dots \wedge Q_m(Y_{m,1}, Y_{m,2}, \dots, Y_{m,n_m}) \wedge P_1^c(Z_1 | a_1) \wedge \dots \wedge P_s^c(Z_s | a_s)$$

Siendo $P_x^c(Z_x | a_x)$, $x \in \{1, \dots, s\}$ predicados constantes. Si $Y_i | Z_x$ con $x \in \{1, \dots, s\}$ e Y_i es un término de $Q_r(Y_{r,1}, Y_{r,2}, \dots, Y_{i-1}, Y_i, Y_{i+1}, \dots, Y_{r,n_r})$ con $r \in \{1, \dots, m\}$, es posible obtener el predicado $Q_r(Y_{r,1}, Y_{r,2}, \dots, Y_{i-1}, Z_x, Y_{i+1}, \dots, Y_{r,n_r})$. Por tanto el predicado original queda transformado de la siguiente forma:

$$R(X_1, X_2, \dots, X_n) \leftarrow Q_1(Y_{1,1}, \dots, Y_{1,n_1}) \wedge \dots \wedge Q_r(Y_{r,1}, Y_{r,2}, \dots, Y_{i-1}, Z_x, Y_{i+1}, \dots, Y_{r,n_r}) \wedge \dots \wedge Q_m(Y_{m,1}, Y_{m,2}, \dots, Y_{m,n_m})$$

Al resolver, se realizará una reunión natural $Q_r(Y_{r,1}, Y_{r,2}, \dots, Y_{i-1}, Z_x, Y_{i+1}, \dots, Y_{r,n_r}) \times P_x^c(Z_x | a_x)$ sobre Z_x dado que $\Pi_{Z_x} Q_r = \Pi_{Z_x} P_x^c$, siendo Π el operador relacional de proyección.

El predicado constante $P_x^c(Z_x | a_x)$ verifica el hecho $P_x^c(a_x)$, por tanto debe estar en la BH, y además posee un GA igual a uno. Por tanto, basta extender el hecho con su GA: $P_x^c(Z_x | a_x, \gamma_{P_x^c}) \in \Lambda | \gamma_{P_x^c} = 1$, para acomodarlo dentro de la BHE y que así pueda adaptarse al modelo de datos de GDB.

5.2 Conjuntos de reglas

Supongamos un conjunto S de reglas previamente creadas: $S = \{R_1, \dots, R_n\}$. Si se desea aplicar el algoritmo deducción de GDB (5.1.2.2) sobre ellas, nos encontramos con un problema. El algoritmo es orientado a relaciones y ha de aplicarse de forma individual regla a regla, es decir, no es posible utilizar los datos calculados por R_1 sobre $R_i | i \in \{2, \dots, n\}$ o viceversa. El problema deriva del hecho de que el resultado del cálculo de regla es una relación obtenida mediante una única

sentencia SQL. Esta relación es intensiva y temporal (*5.1.2.4.1.1*), por lo que a priori, no podría ser utilizada como un predicado de otra regla.

Semánticamente, ¿qué significa calcular un conjunto de reglas? Calcular s , es equivalente a calcular una regla nueva R^* cuyo cuerpo se forma por la conjunción de todos los predicados presentes en el cuerpo de las reglas $R_i \mid i \in \{1, \dots, n\}$ y su cabeza por la unión de toda las variables de cabeza de R_i .

5.3 Semántica de la medida de calidad

Para poder entender la medida de calidad que asigna el modelo GDB ante una consulta, hay que contemplar dos escenarios diferentes: **a)** que exista componente difuso en la consulta o **b)** que no exista tal tipo de información.

En el caso **b)** nos encontramos ante una consulta clásica, el usuario desea conocer qué datos almacenados cumplen con las condiciones expresadas, obviando la imprecisión. En tal caso, el modelo siembre arroja una medida de calidad con el valor máximo (valor uno).

En el caso **a)** se utilizará como medida de calidad el GA (*4.5.4 Flexibilización de reglas*), ya que mide la ligadura de los hechos almacenados en la BH con la consulta (regla) realizada, o lo que es lo mismo, como es de compatible una tupla o atributo respecto al criterio de selección de la consulta.

5.4 Ejemplo de uso del modelo GDB

A fin de demostrar la aplicación del modelo de datos de GDB a un caso real, se necesita inicialmente una consulta sobre los datos almacenados que será transformada en una regla, que a su vez será calculada para obtener un resultado.

Partamos de la siguiente consulta “Se desea conocer todos los eventos que se han producido a la vez”. Normalmente, esta pregunta se realiza para conocer la densidad de eventos detectados por el instrumento en un momento dado.

Para poder transformar la consulta en una regla y verificar el resultado, es necesario realizar un proceso que consta de los siguientes pasos:

- 1) Inicialmente se debe determinar qué tipo de dato difuso, de los soportados por el modelo de datos GDB, representa mejor la información que se desea manejar.
- 2) Después hay que incluir este tipo de dato difuso como parte de un atributo de una tabla clásica.
- 3) Más tarde hay que definir la tabla intensiva y la regla que refleje la consulta.
- 4) Es necesario establecer la semántica del grado de acoplamiento (medida de calidad) que se obtenga.
- 5) Hay que proporcionar un conjunto de datos de entrada controlados (sintéticos), que nos permita de forma anticipada caracterizar la medida de calidad que se obtendrá.
- 6) Para concluir, hay que ejecutar la regla sobre el conjunto de datos controlados y compararlos con los esperados. Esto constituye la verificación de la regla.

5.4.1 Selección del tipo de dato difuso

Para procesar la consulta, es necesario analizar qué significa “a la vez” en el entorno de G. Por simplicidad, sólo se considerarán los eventos de GDS e IS.

Dado que la precisión de G. es de 4ms (3.12.3.1), cualquier evento en un rango de 4ms poseerá el mismo valor de tiempo. O dicho de otro modo, G. sólo genera marcas de tiempo en múltiplos de 4ms. Es interesante identificar además, los eventos cercanos con un entorno de un $\pm 4ms$ del valor considerado. Estos eventos no se han producido a la vez, sino que están “cerca”. Es aquí donde entra la componente difusa de la consulta. Queremos saber el grado (calidad) con el que los eventos se han producido al mismo tiempo: un mismo valor en los tiempos indicará una calidad 1, valores próximos irán reduciendo esta calidad.

Es necesario representar estas marcas de tiempo mediante un tipo de dato difuso (5.1.1.1) representable por el modelo de datos de GDB. Este tipo de dato permitirá aplicar operadores difusos que calculen esta proximidad entre valores.

Es importante indicar que los datos generados por G. son precisos (no difusos), por lo que los *difusos tipo 3* pueden ser descartarlos.

Una primera idea es utilizar *difusos tipo 1* que admiten representación precisa (clásica) y consulta difusa a través de *etiquetas lingüísticas*. Esto nos llevaría a definir la etiqueta lingüística “cerca en el tiempo”. El problema es que hay que definirla en términos de valores absolutos (*Ilustración 45*), independientemente del valor de tiempo del evento, lo que limita su aplicabilidad.

Restan los *difusos tipo 2*. Eliminadas las *etiquetas lingüísticas* por aplicarse a valores absolutos, podemos desechar también los *intervalos de proximidad*, ya que queremos conocer el grado de “cercanía” de dos valores, pero la representación de este tipo difuso en forma de rectángulo no lo permite (*Ilustración 49*).

La idea más natural es utilizar el tipo de dato difuso *valores aproximados* (*Ilustración 48*). En este tipo de dato, la altura (calidad) posee un valor de uno en un valor concreto de tiempo de evento, y asigna una calidad descendente, conformando un triángulo (trapezoide triangular), a medida que se aleja de dicho valor.

En nuestro ejemplo, aproximadamente 88 se representaría como:

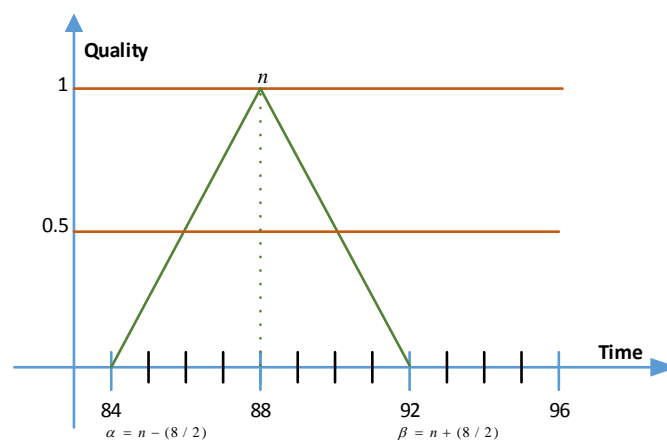


Ilustración 66. Representación de tipo difuso aproximadamente 88.

Según se describió en *5.1.1.4.1.1 Sentencia "create table"*, la definición de este tipo difuso requiere de dos parámetros: *'margen'* (base del triángulo en valor absoluto) y *'much'* (distancia mínima para que dos valores sean considerados no comparables).

En nuestro ejemplo *'margen'*=8ms (un entorno de 4ms relativo al valor de tiempo del evento) y *'much'*=9 (distancia a partir de la cual dos eventos nos son considerados simultáneos). Realmente, dada la precisión de 4ms, habría que establecer *'much'* =12, pero esto no varía el resultado final, por lo que se ha optado por mantener *'much'*=9 por simplicidad.

5.4.2 Definición de relaciones clásicas

Una vez seleccionado el tipo de difuso a utilizar, es necesario crear una tabla que incorpore un atributo con este tipo difuso y que almacene los datos de tiempo.

El paso inicial es localizar las tablas relacionales donde se encuentran originalmente definido el tiempo de los eventos. Utilizando la partición de telemetrías declarada en *Ilustración 39*, las tablas necesarias son:

Tabla 26. Atributos de la tabla GDB_GDSEVENT utilizados en el ejemplo de uso del modelo GDB.

GDB_GDSEVENT	
gds_item	gds_eventtime

Tabla 27. Atributos de la tabla GDB_ISEVENT utilizados en el ejemplo de uso del modelo GDB.

GDB_ISEVENT	
is_item	is_eventtime

La semántica detallada de estos campos, puede encontrarse en el *Apéndice A*. De forma resumida, *'x_item'* (donde *'x'* es *'is'* o *'gds'*) corresponde a la posición dentro del paquete de ciencia (*3.12.3*) del evento, y *'x_eventtime'* es el tiempo del sistema capturado por la rutina de interrupción.

Estos atributos, originalmente definidos como atributos clásicos relacionales, hay que extenderlos para incorporar el *difusos tipo 2* con *'margen'* =8 y *'much'* =9. Para ello se adapta la definición *5.1.1.4.1.1 Sentencia "create table"*. En concreto:

Definición de la tabla GDB_GDSEvent:

```
CREATE TABLE GDB_GDSEvent (
...
GDS_ITEM NUMBER(2) NOT NULL Constraint "GDS_ITEM"
CHECK (GDS_ITEM>=0) ,
...
GDS_eventTime FTYPE1(8,9) NUMBER(15) NOT NULL Constraint
"GDS_packetTime" CHECK (GDS_eventTime>=0 and
GDS_eventTime<=281474976710655) ,...)
```

Definición e la tabla GDB_ISEvent:

```
CREATE TABLE GDB_ISEvent (
...
IS_ITEM NUMBER(2) NOT NULL Constraint "IS_ITEM" CHECK (IS_ITEM>=0) ,
```

...
IS_eventTime FTYPE1 (8,9) NUMBER (15) NOT NULL Constraint, ...)

La definición completa de las tablas se encuentra en [Apéndice A](#).

La sintaxis de FSQL permite definir el *difusos tipo 2: valores aproximados* como *difusos tipo 1*. Dado que los datos son realmente precisos, se ha optado por esta notación en el ejemplo, aunque no existen diferencias en el cálculo de la consulta.

Estas dos definiciones de tablas serán enviadas al servidor DFSQL ([5.5 Arquitectura del modelo GDB](#)) y de ahí al servidor SQL difuso para ser traducido a una sentencia SQL válida del SGBDR Oracle ©. Además se añade información en la base de meta-conocimiento difuso ([5.1.2.3](#)) para almacenar información del tipo de dato difuso y su relación con las tablas clásicas. Para ello se completan las tablas FUZZY_COL_LIST y FUZZY_APPROX_MUCH como sigue:

Tabla 28. Tabla FUZZY_COL_LIST en el ejemplo de uso del modelo GDB.

FUZZY_COL_LIST				
OBJ#	COL#	F_TYPE	LEN	COM
230492	4	1	1	GDB_GDS_ISEVENT.GDS_IS_EVENTTIME
230480	4	1	1	GDB_ISEVENT.IS_EVENTTIME

Tabla 29. Tabla FUZZY_APPROX_MUCH en el ejemplo de uso del modelo GDB.

FUZZY_APROX_MUCH			
OBJ#	COL#	MARGEN	MUCH
230492	4	8	9
230480	4	8	9

5.4.3 Definición de la tabla intensiva y regla

El siguiente paso es traducir la consulta inicial “Se desea conocer todos los eventos que se han producido a la vez” en formato de una RGGa del modelo de datos de GDB. Para ello, necesitaremos modificar el *catálogo el sistema*, constituido por la *base de meta-conocimiento deductivo* y *base de reglas* ([5.1.2.3](#)), para crear una *tabla intensiva* (donde se almacenará el cálculo de la regla) y la regla propiamente dicha.

Recurrimos a la definición en DFSQL ([5.1.2.4](#)) para definir la tabla intensiva y la regla.

```
//===== Intensiva table =====
create intensiva table GDB_INT_TESIS_EXAMPLE (
  GDS_ITEM NUMBER(2),
  GDS_EVENTTIME NUMBER(15),
  IS_ITEM NUMBER(2),
  IS_EVENTTIME NUMBER(15));
//===== End of intensional table =====

//===== Rule =====
create rule for GDB_INT_TESIS_EXAMPLE (X,Y,Z,W)
```

```

as
GDB_GDSEVENT(X source gds_item, Y source gds_eventtime) AND
GDB_ISEVENT(Z source is_item, W source is_eventtime) AND
(Y FEQ W);
//===== End of rule =====

```

La sintaxis para la definición de una tabla intensiva y una regla puede encontrarse en [5.1.2.4.1.1](#) y [5.1.2.4.1.2](#), respectivamente.

Estas dos sentencias serán enviadas al *servidor DFSQL (5.5)* y de ahí al *servidor SQL deductivo* que realizará las modificaciones oportunas en *catálogo del sistema*.

En la definición de la *tabla intensiva*, se conforma la estructura de la tabla donde se almacenará el resultado de la consulta. Es por ello que se necesita acceder a la definición de los atributos de las tablas clásicas para mantener la compatibilidad de los atributos con la tabla intensiva.

Para construir la regla, hay que proporcionar el nombre de las tablas clásicas, los atributos involucrados, asignar un nombre de variable a los atributos e indicar las condiciones a aplicar (criterio de selección).

En el ejemplo, los atributos relevantes son `GDB_GDSEVENT.gds_eventtime` y `GDB_ISEVENT.is_eventtime`, a los que se les ha asignado las variables `Y` y `W`. La condición a aplicar para obtener “los eventos que se han producido a la vez” se basa en el operador clásico de igualdad (“`Y =W`”) entre los dos atributos.

Se puede utilizar el operador difuso de igualdad `FEQ (5.1.1.2.1)` en vez del operador clásico. Este operador proporciona los mismos resultados que el operador clásico más los eventos “cercaños” calificados con un valor de calidad.

Las modificaciones realizadas en el catálogo son las siguientes (la semántica de cada atributo fue definida en [5.1.2.3](#)):

Tabla 30. Tabla `DED_INT_TABLE_DESCRIPTION` en el ejemplo de uso del modelo GDB.

DED_INT_TABLE_DESCRIPTION	
TABLE_ID	RULE_ID
230492	4

Tabla 31. Tabla `DED_INTENSIVA_CATALOG` en el ejemplo de uso del modelo GDB.

DED_INTENSIVA_CATALOG		
IDPRED	MARCADO	NVARS
230950	1	4

Tabla 32. Tabla DED_COMPARISON_DESCRIPTION en el ejemplo de uso del modelo GDB.

DED_COMPARISON_DESCRIPTION							
TABLE_ID	RULE_ID	PRED_ID	OCC_NUMBER	VAR_ID1	VAR_ID2	COMP_OP	THOLD
230950	1	0	3	2	4	6	1

Tabla 33. Tabla DED_PREDICATE_DESCRIPTION en el ejemplo de uso del modelo GDB.

DED_PREDICATE_DESCRIPTION						
TABLE_ID	RULE_ID	PRED_ID	OCC_NUMBER	VAR_ID	COL_ID	SOURCE_COL
230950	1	230929	1	1	1	2
230950	1	230929	1	2	2	4
230950	1	230929	2	3	1	2
230950	1	230929	2	4	2	4

Tabla 34. Tabla DED_RULE_DESCRIPTION en el ejemplo de uso del modelo GDB.

DED_RULE_DESCRIPTION					
TABLE_ID	RULE_ID	PRED_ID	OCC_NUMBER	NEGATED	TYPE
230950	1	230929	1	0	0
230950	1	230925	2	0	0
230950	1	0	3	0	2

Es importante indicar que el atributo THOLD (θ) en la tabla DED_COMPARISON_DESCRIPTION, que indica el valor mínimo del grado de cumplimiento (CDEG) para una tupla aparezca como resultado en la relación, se deja a uno, su valor por defecto. Por simplicidad (en el caso de que existan múltiples condiciones difusas), GDB-GUI establece un valor de THOLD común para todas las condiciones de la regla. Este valor es asignado antes de realizar el cálculo de la regla, independientemente del valor THOLD almacenado.

5.4.4 Semántica de la medida de calidad

El valor de calidad se almacena en el atributo CDEG (θ) cuyo valor indica como es de compatible una tupla o atributo respecto al criterio de selección. De forma resumida, el valor de este atributo se calcula mediante las ecuaciones asociadas a los compradores difusos (5.1.1.2.1), a fin de obtener las intersecciones de las distribuciones de posibilidad trapezoidal asociados a los atributos difusos involucrados.

En el ejemplo, se ha definido dos atributos *difusos tipo 1* asociados a los tiempos de evento de GDS e IS. Así mismo, se ha creado una regla que establece una condición, basada en el operador difuso igualdad, sobre estos dos atributos.

Semánticamente, un valor uno en la calidad indicará que los dos tiempos de los eventos son exactamente los mismos. Se asignará un valor decreciente de calidad según los valores se alejen entre sí, de acuerdo al trapecoide triangular de base 8ms y altura 1.

De forma gráfica, el cálculo de la comparación de dos valores difusos, es equivalente a calcular el área de intersección de los dos trapezoides. En la siguiente ilustración, se calcula el valor de calidad para la comparación de igualdad difusa (FEQ) aplicada sobre los valores “aproximadamente 88” y “aproximadamente 92”.

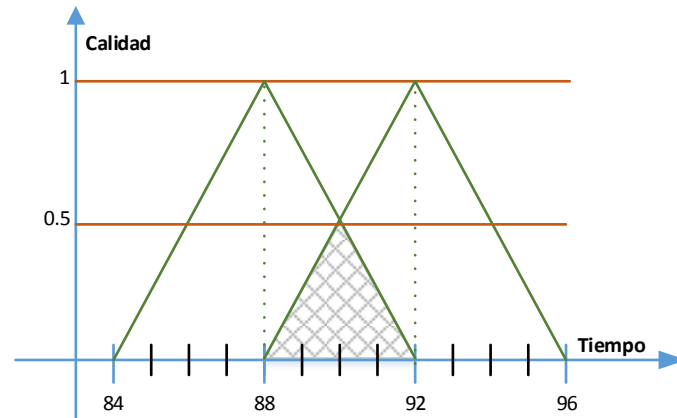


Ilustración 67. Intersecciones de los valores difusos aproximadamente 88 y 92.

Dado que los tiempos de los eventos son múltiplos de 4ms, el valor de la intersección (calidad) será como máximo de 0.5 (exceptuando cuando los valores sean iguales, donde la calidad a es 1), ya que no es posible encontrar valores intermedios. Además los valores más alejados de 4ms, generan una calidad 0, debido al parámetro ‘much’ en la definición del tipo difuso.

La comparación de los valores difusos aproximadamente 88 y 96, no produce ninguna intersección (como se muestra en la siguiente ilustración), por lo que la calidad calculada será igual a cero.

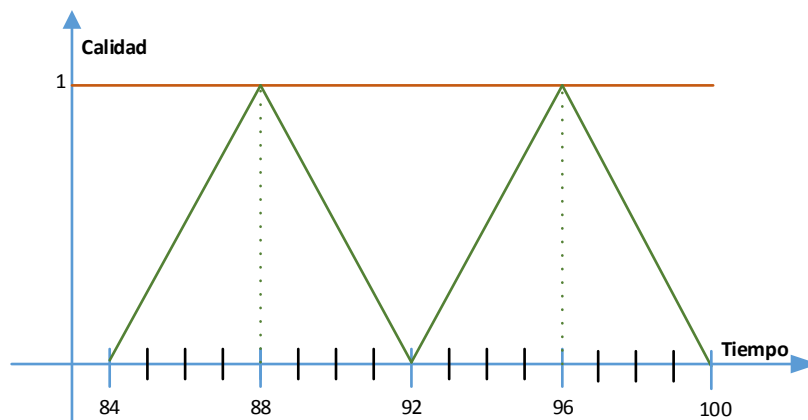


Ilustración 68. Intersecciones de los valores difusos aproximadamente 88 y 96.

5.4.5 Datos sintéticos

Para verificar la regla, se van a utilizar un conjunto de datos sintéticos. GDB-GUI proporciona un simulador de eventos (6.5) capaz de generar los datos requeridos de forma sencilla mediante un ‘script’.

Estos son los datos utilizados para verificar la regla construida:

Tabla 35. Tabla GDB_GDSEvent en el ejemplo de uso del modelo GDB.

GDB_GDSEVENT	
GDS_ITEM	GDS_EVENTTIME
4	88
5	92
6	96
7	100
8	100
9	104
10	108
11	112

Tabla 36. Tabla GDB_ISEvent en el ejemplo de uso del modelo GDB.

GDB_ISEVENT	
IS_ITEM	IS_EVENTTIME
0	88
1	92
2	96
3	100

Como se aprecia, siempre que sucede un evento de GDS desde 88ms a 100 ms, sucede uno de IS. Es más, en el valor de tiempo de 100ms se produce un evento triple: 2 GDS y un IS.

Por tanto es de esperar, a la hora de calcular la reglas, calidad 1 en los eventos que poseen igual valor numérico de tiempo, calidad 0.5 en los eventos alejados entre sí 4ms y calidad 0 en el resto.

5.4.6 Cálculo de la regla

El cálculo de la regla consiste en obtener las tuplas de la tabla intensiva que cumplen las condiciones establecidas, incluyendo un atributo extra, llamado CDEG que mide en qué grado se cumplen dichas condiciones. Para realizar esta labor, hay que enviar al *servidor DFSQL* la sentencia con la que obtendremos todos los atributos de la tabla intensiva:

```
Select * From GDB_INT_TESIS_EXAMPLE
```

El resultado obtenido lo proporciona el algoritmo de deducción “abajo-arriba” que se analizó en [5.1.2.2.1](#). En concreto, como resultado, se obtiene una consulta SQL (con una subconsulta añadida) que calcula las tuplas de la tabla intensiva, añade un atributo extra para almacenar la calidad (CDEG) y aplica el umbral de cumplimiento definido por el usuario (THOLD). La aplicación del umbral lo realiza GDB-GUI después de la traducción, en la cláusula where de la subconsulta.

Suponiendo una calidad asignada por el usuario de 0.4, la consulta SQL traducida sería:

```
SELECT GDB_INT_TESIS_EXAMPLE.*
FROM
  ( SELECT P1.GDS_ITEM GDS_ITEM,
          GDS_EVENTTIME GDS_EVENTTIME,
          P2.IS_ITEM IS_ITEM,
```

```

IS_EVENTTIME IS_EVENTTIME,
DFSQL_FUNCTIONS.MIN(
  FSQL_FUNCTIONS.FEQ_t1_t1(
    P1.GDS_EVENTTIME,
    P2.IS_EVENTTIME,8)) "CDEG(*)"
FROM
  GIADA.GDB_GDSEVENT P1,
  GIADA.GDB_ISEVENT P2
WHERE
  FSQL_FUNCTIONS.FEQ_t1_t1(
    P1.GDS_EVENTTIME,
    P2.IS_EVENTTIME
    ,8) >= 0.4)
GDB_INT_TESIS_EXAMPLE

```

La consulta SQL es directamente procesable por el SGDBR Oracle ©. La tabla intensiva obtenida es modificada por GDB-GUI para cambiar el nombre del atributo CDEG por Quality.

El cálculo difuso se realiza en uno de los procedimientos almacenados (funciones) llamado FSQL_FUNCTIONS.FEQ_t1_t1. Estas funciones fueron definidas al realizar la instalación de la implementación del modelo GDB en el SGBDR. En concreto, la función FSQL_FUNCTIONS.FEQ_t1_t1 calcula la intersección de los dos difusos tipo 1 utilizando el comprador difuso FEQ. Los dos primeros parámetros de la función, corresponden al nombre de los atributos difusos considerados y el último parámetro a la base del triángulo de la distribución de posibilidad.

Como resultado del ejemplo, se obtiene una tabla con cardinalidad (número de filas) 13:

Tabla 37. Tabla resultado GDB_INT_TESIS_EXAMPLE en el ejemplo de uso del modelo GDB.

GDB_INT_TESIS_EXAMPLE				
GDS_ITEM	GDS_EVENTTIME	IS_ITEM	IS_EVENTTIME	Quality
4	88	0	88	1
5	92	0	88	0.5
4	88	1	92	0.5
5	92	1	92	1
6	96	1	92	0.5
5	92	2	96	0.5
6	96	2	96	1
7	100	2	96	0.5
8	100	2	96	0.5
6	96	3	100	0.5
7	100	3	100	1
8	100	3	100	1
9	104	3	100	0.5

Recordemos que el algoritmo de deducción incorpora el operador relacional producto cartesiano, que implica aplicar la comparación difusa de cada fila de la relación GDB_GDSEvent con todas las filas de la relación GDB_ISEvent. Además como THOLD es 0.4, las tuplas con calidad menor a este valor no parecen en la tabla resultante.

En la siguiente ilustración se muestra la ejecución del ejemplo utilizando GDB-GUI:

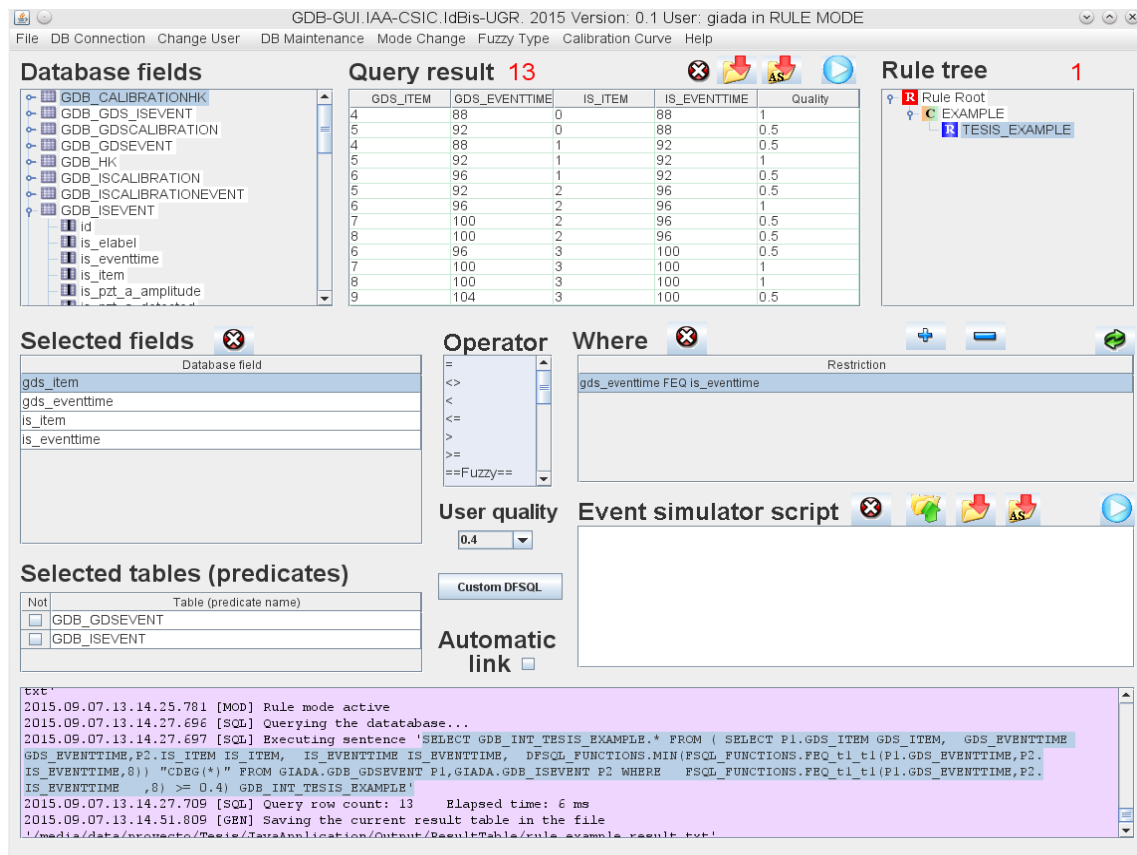


Ilustración 69. Resultado de la ejecución del ejemplo de uso del modelo de GDB en GDB-GUI.

El cambio del comparador difuso FEQ por un comparador clásico =, genera una tabla con cardinalidad 5, esto es, 8 filas menos que cuando se utilizó el comparador difuso. Estas 5 filas corresponden a los eventos que comparten exactamente el mismo el mismo valor de tiempo, como se aprecia en la siguiente tabla.

Tabla 38. Tabla resultado GDB_INT_TESIS_EXAMPLE en el ejemplo de uso del modelo GDB.

GDB_INT_TESIS_EXAMPLE				
GDS_ITEM	GDS_EVENTTIME	IS_ITEM	IS_EVENTTIME	Quality
4	88	0	88	1
5	92	1	92	1
6	96	2	96	1
7	100	3	100	1
8	100	3	100	1

5.4.7 Conclusiones del uso del modelo de datos de GDB

Partiendo de una consulta inicial “Se desea conocer todos los eventos que se han producido a la vez” se ha seguido un proceso para crear y calcular una regla en el modelo de datos de GDB. Esta regla se ha verificado utilizando como entrada un conjunto de datos sintéticos.

Se ha obtenido un valor numérico (calidad) proporcional a cómo los eventos se encuentran próximos entre sí. La calidad aumenta desde 0 (los eventos se consideran no simultáneos) a 1 (eventos simultáneos).

La relajación de la condición en la regla ha permitido conocer y valorar de forma numérica como los eventos están relacionados temporalmente. Esta nueva información incrementa el conocimiento obtenido respecto al obtenido con reglas con condiciones clásicas y nos proporciona una mejor comprensión del contexto de los resultados.

En concreto, en el ejemplo se ha calculado como realmente existen 13 eventos GDS y IS que pueden considerarse “cercaños”, lejos de los 5 que se obtendrían con la consulta clásica. Por tanto, la densidad de eventos detectados por instrumento es realmente casi el triple.

El ejemplo ha mostrado que la sintaxis de DFSQL empleada por las expresiones de regla y tabla intensiva es densa, propensa a errores y requiere de un conocimiento exacto de nombres de tablas, atributos y propiedades de los mismos. Cuando se requieren reglas más elaboradas que involucren a múltiples predicados, la complejidad aumenta rápidamente. Este es uno de los principales problemas que se resolverán utilizando GDB-GUI descrito capítulo 6: el usuario podrá crear de forma visual reglas sin tener en cuenta la sintaxis del DFSQL.

5.5 Arquitectura del modelo GDB

La implementación de GDB se ha realizado en el SBDR Oracle © y hereda de la realizada en **[Bla01]** y ésta a su vez se basa en la FREDDI (4.5.7) y FIRST (4.2.2.5.1).

Como puede apreciarse en FREDDI, la capacidad deductiva reside en un motor de inferencia fuera el SGDBR basado en PROLOG. Ello implica crear y mantener una interfaz entre ambos módulos, lo que a la larga resulta ineficaz e inflexible. Aprovechando el lenguaje de programación que se incluye en las últimas versiones del SGBDR Oracle©, es posible construir el motor de inferencia utilizando este lenguaje, lo que permite tener el módulo deductivo integrado en el SGBDR. Es más, el módulo difuso también puede implementarse con este lenguaje, lo que nos lleva a tener todos los módulos necesarios bajo el mismo SGDBR, dando uniformidad y solidez al sistema resultante. El esquema final de la arquitectura de GDB puede apreciarse en la siguiente ilustración:

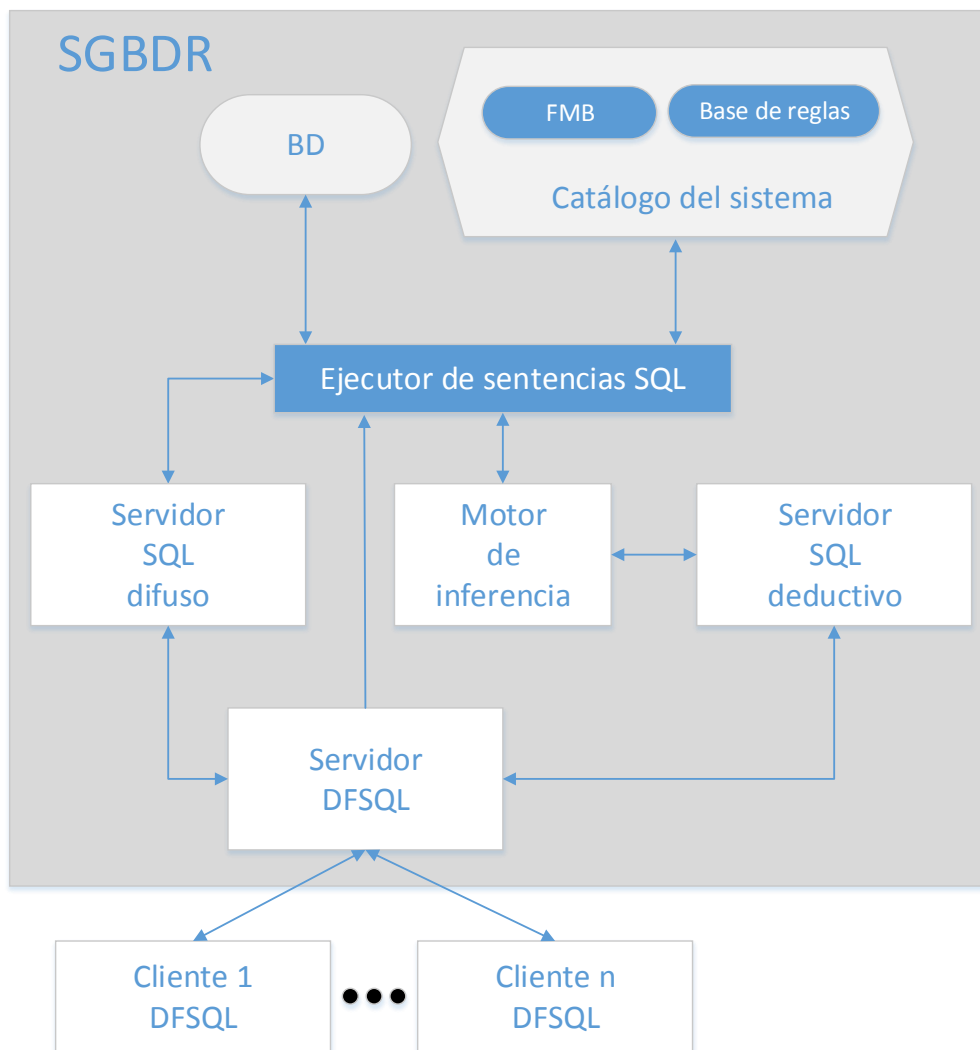


Ilustración 70. Arquitectura de la implementación de GDB.

La descripción de estos módulos es la siguiente:

- a) **SGBDR.** Es el sistema base que soporta a toda la arquitectura de GDB. Proporciona los recursos de gestión de la BDR y herramientas para programar aplicaciones SQL. Sobre la sintaxis de este SQL, se realizan las extensiones difusas (5.1.1) y deductivas (5.1.2).
- b) **Base de datos.** Almacena toda la información extensiva, difusa o no, dentro de un esquema relacional.
- c) **FMB(Fuzzy Meta-knowledge).** La base de meta-conocimiento difuso es una extensión del catálogo del SGBDR donde se almacena toda la información acerca de las estructuras difusas, datos y definiciones (ver 5.1.2.3).
- d) **Base de reglas.** Este es el módulo donde se almacenan las tablas intensivas referenciadas en las reglas (ver 5.1.2.3) y las propias reglas.
- e) **Servidor DFSQL.** En este módulo se procesan las consultas deductivas y difusas que se realicen desde los clientes, enviándolas al servidor adecuado (*SQL deductivo* o *SQL difuso*). Si desde el cliente DFSQL, se crea una consulta que no disponga de elementos difusos o deductivos (SQL clásico), se enviará dicha consulta directamente al *ejecutor de sentencias de SQL*.

- f) **Servidor SQL deductivo.** En este módulo se procesan las consultas deductivas a través del motor de inferencia, traduciéndolas al SQL del SGBDR.
- g) **Servidor SQL difuso.** En este módulo se procesan las consultas difusas, creando sentencias SQL clásicas que las enviará al ejecutor de sentencias SQL.
- h) **Ejecutor de sentencias de SQL.** Recibe consultas SQL clásicas y las ejecuta, accediendo para ello a los datos de la BD y al catálogo del sistema.
- i) **Motor de inferencia.** Realiza el proceso de deducción que permite obtener la información intensiva requerida. El servidor FSQL deductivo debe de determinar que reglas, tablas extensivas y definiciones debes ser enviadas al motor de inferencia para resolver estas consultas.
- j) **Ciente DFSQL.** Es la interfaz entre el usuario y el servidor de DFSQL. Es posible conectar más de un cliente a la vez al mismo servidor.

En esta arquitectura, la GDB-GUI se enmarca dentro de los clientes DFSQL, como interfaz de usuario a la implementación del modelo de datos de GDB y se detallará en *6 Interfaz de usuario GDB*.

5.6 Resumen y conclusiones

En este capítulo se ha detallado el modelo de datos de GDB, que hereda y adapta el desarrollado por **[Bla01]**, que su vez es una evolución de modelos e implementaciones anteriores (4.6).

En la siguiente ilustración se muestran los precedentes del modelo de datos de GDB, donde partiendo de modelos clásicos, se ha evolucionado para incorporar capacidades difusas y deductivas.

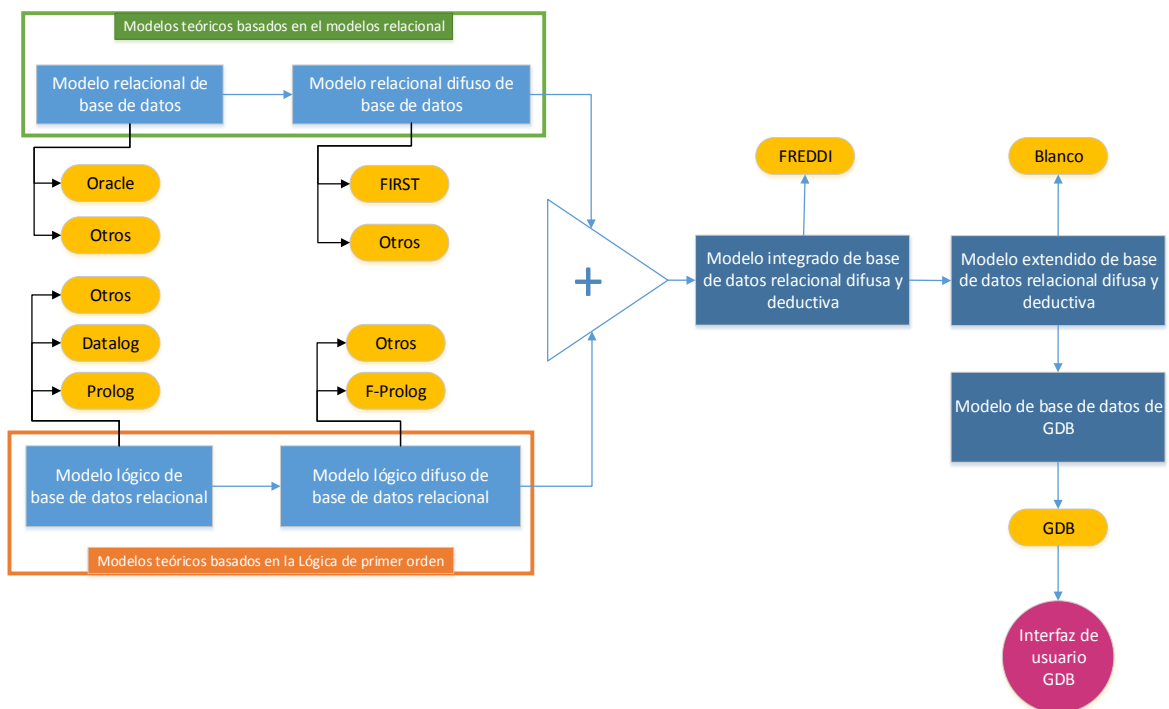


Ilustración 71. Evolución de los modelos teóricos e implementaciones hasta GDB.

Se ha desgranado como el modelo de datos de GDB maneja la información difusa, los tipos de datos difusos implementados, los comparadores difusos soportados y el funcionamiento de la base de meta-conocimiento difuso.

Así mismo se ha revisado cómo el modelo manipula la información deductiva, presentando la regla generalizada con grado de acoplamiento. Se ha detallado cómo se calcula este tipo de regla cuando aparecen en ella distintos tipos de predicados. A continuación se ha indicado cómo funciona el algoritmo de deducción “abajo-arriba” orientado a relaciones, que permite calcular una regla mediante una única consulta relacional. Seguidamente, se ha analizado la composición de la base de meta-conocimiento difuso y base de reglas y la sintaxis del lenguaje DFSQL que permite representar, manipular y consultar la información difusa y deductiva en un SGDBR.

Concluyendo el capítulo, se han tratado ciertos aspectos útiles para una aplicación práctica del modelo: constantes, macros, intervalos, conjunto de reglas y semántica de la medida de calidad. Así mismo detallado un ejemplo concreto de regla y se ha presentado al arquitectura del modelo.

El modelo de datos de GDB es una herramienta útil para manejar los datos enviados de G. No sólo permite una explotación de la información en el sentido clásico, si no que aporta nueva luz sobre los datos al relajar las consultas, permitiendo conocer y cuantificar la información “próxima” a la buscada. Además, el modelo, proporciona capacidades deductivas, lo que facilita al usuario expresar su conocimiento de una forma más próxima a su razonamiento (mediante reglas), además de poder calcular nueva información a partir de la ya almacenada.

La aplicación del modelo de datos GDB a la información proporcionada por el instrumento G. brinda una oportunidad excepcional de poder interpretar la información existente que no encaja de forma exacta en los modelos de comportamiento de polvo cometario. Con la ventaja adicional de poder expresar mediante reglas, el conocimiento obtenido durante los años de estudio de los datos. Se podrá obtener así, gracias a las capacidades deductivas del modelo, nuevo conocimiento acerca del polvo y el comportamiento del instrumento (como se verá en el capítulo [*7 Verificación y análisis de resultados*](#)).

En el capítulo [6](#) se describirá una interfaz visual (GDB-GUI) al modelo de GDB.

6 Interfaz de usuario GDB-GUI

La interfaz de usuario GDB (GDB-GUI) realiza el papel de cliente DFSQL en el contexto de la arquitectura del modelo de base de datos de GDB (5.5), como se muestra en la *Ilustración 70*. Fue inicialmente presentada en **[Mor08]**.

Esta interfaz ha sido diseñada con un triple objetivo: *importar*, *explotar* y *verificar* los datos provenientes del instrumento GIADA.

Por un lado permite cargar (*importar*) y almacenar en una base de datos relacional la información generada por G. Esto incluye la información obtenida en la fase de calibración del instrumento, los datos reales en vuelo y los datos simulados.

Esta información puede ser consultada (*explotada*) mediante dos interfaces complementarias. La primera permite construir sentencias clásicas SQL (6.3) y la segunda (6.4) facilita la construcción de reglas con componentes difusos (DFSQL) según el modelo de datos de GDB.

Tanto las sentencias como las reglas, pueden ser *verificadas* mediante el uso de datos sintéticos, esto es, conjuntos de datos de entrada precisos y controlados que permiten calcular anticipadamente el resultado, para, posteriormente, compararlo con el devuelto por GDB-GUI. Esta labor de generación de datos la realiza el simulador de eventos.

Se ha realizado un diseño que potencia la parte visual de GDB-GUI. La mayor parte del trabajo de creación de reglas o sentencias se realiza mediante ratón. No es necesario que el usuario conozca la sintaxis de SQL o DFSQL para poder realizar consultas sobre los datos almacenados.

En este capítulo se detallará el flujo de datos científicos susceptible de ser procesado por GDB-GUI. A continuación se describirán las características comunes de ambas interfaces (SQL y DFSQL) para continuar con las particulares de cada interfaz, concluyendo con una descripción del simulador de eventos.

6.1 Flujo de datos de GDB

En la siguiente ilustración se muestran los principales flujos de datos relacionados con G.

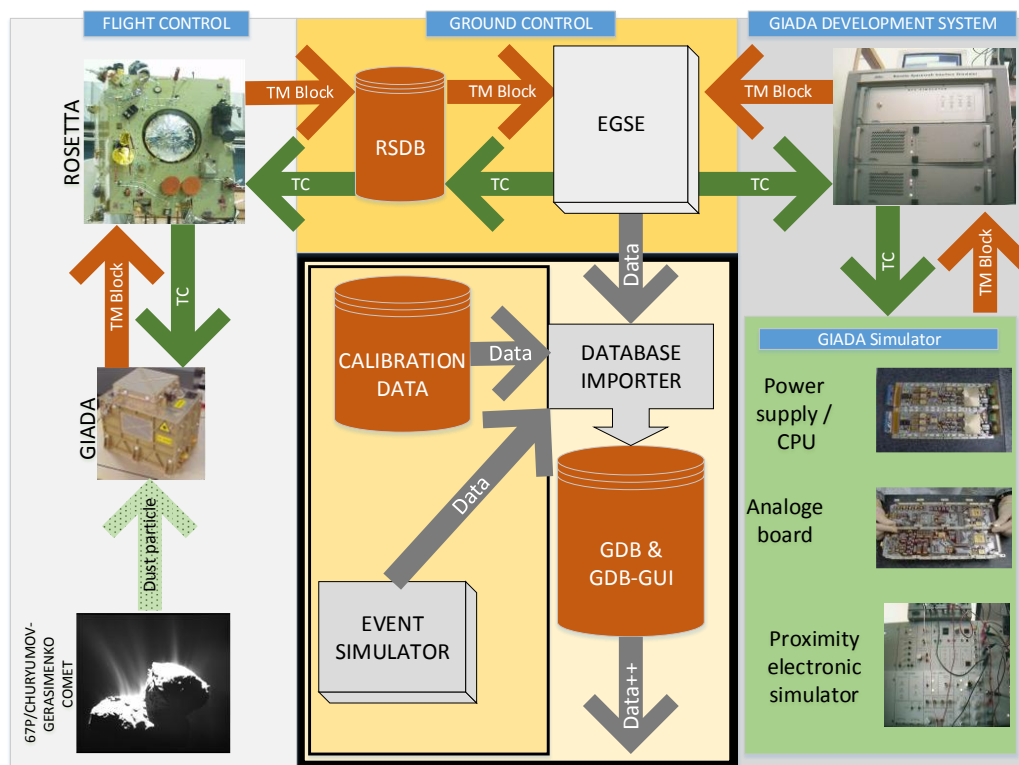


Ilustración 72. Flujo de datos en GIADA.

Durante una observación científica, los datos de partículas reales (*TM blocks*) procedentes del cometa son generados por G. La observación se configura mediante un conjunto de comandos (*TC*) enviados desde Rosetta. Estos datos serán encaminados, mediante las antenas de alta ganancia de Rosetta, al control terrestre. La base de datos de Rosetta (RSDB) es la encargada de enviar comandos, recepcionar los datos y distribuirlos a los computadores (EGSE) de los distintos instrumentos a lo largo del planeta. En la *Ilustración 37* se muestra con más detalle la gestión del flujo de datos en control terrestre.

El EGSE es el computador encargado de recibir los datos de G. y de enviar las órdenes oportunas para el control de la observaciones científicas en vuelo. Este computador también es el utilizado para centralizar el trabajo en el sistema de desarrollo de G., encargado de simular en Tierra el comportamiento del instrumento en vuelo a fin de depurar posibles problemas y encontrar soluciones (parches al software embarcado). Para ello dispone de réplicas de Rosetta (RO-SIS), de las tarjetas que componen GIADA (“Power supply/CPU” y “Analogue board”) y de simuladores de los sensores de G. (“Proximity electronic simulator”). En el capítulo 2 se explicó con más detalle el sistema de desarrollo de GIADA.

Existe una segunda fuente de datos provenientes de partículas procesadas por G. Son los datos recopilados en la fase de calibración del instrumento antes de su integración en Rosetta. Estas pruebas estaban destinadas a medir el comportamiento de G. ante un conjunto de estímulos controlados, como se describió en *2.2.3.1.1.1*, *2.2.3.1.1.2* y *2.2.3.1.3*.

Los datos simulados constituyen una tercera fuente de datos de partículas. El simulador de eventos crea bloques de datos que imitan la detección de una partícula por parte de G. Es posible establecer de forma sencilla un valor concreto para determinado sensor o estructura. No existe diferencia de formato entre los datos generados por el simulador, los obtenidos mediante calibración y los enviados desde Rosetta.

Los datos procesados por las tres fuentes: datos reales de G., datos de calibración y datos simulados, son procesados por el importador de datos de GDB-GUI y almacenados en GDB. Esto

implica que, los datos procedentes de las distintas fuentes, después de su procesamiento, se encuentran almacenadas de forma uniforme, en la misma BD relacional y disponibles para su explotación.

6.2 Características comunes a las interfaces SQL y DFSQL

GDB-GUI posee dos interfaces que trabajan sobre la misma estructura relacional. Ambas comparten una arquitectura y aspecto visual común:

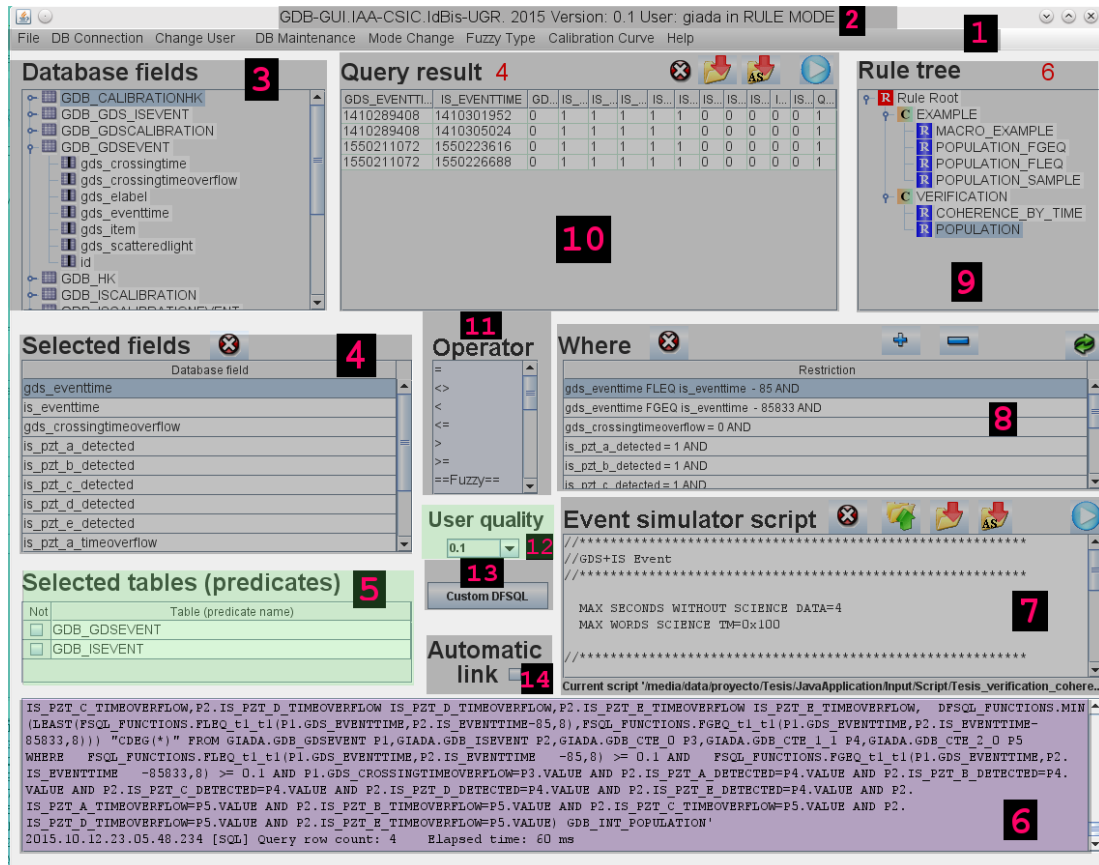


Ilustración 73. Pantalla principal de GDB-GUI.

- ❖ **Área 1.** Aquí se sitúa el menú principal. Las opciones que incorpora se describen en la sección 6.2.2.
- ❖ **Área 2.** En este área podemos encontrar información acerca de GDB-GUI. Se incluye el nombre del programa, la versión, el nombre de usuario y el modo de operación: SQL o DFSQL. El modo DFSQL también es denominado “Rule Mode” o modo regla.
- ❖ **Área 3.** Es dónde se listan todas las relaciones clásicas y sus atributos, Su gestión se detallará en 6.2.3.
- ❖ **Área 4.** Destinada a albergar los atributos seleccionados por el usuario de GDB-GUI.
- ❖ **Área 5.** Este es un área sólo visible en modo DFSQL e indica los predicados (relaciones) que componen la regla.
- ❖ **Área 6.** Es donde se encuentra el “log” (histórico) del sistema, que incluye todos los hechos relevantes acaecidos en GDB-GUI (6.2.8) desde el inicio del programa.
- ❖ **Área 7.** Está dedicada a la gestión del simulador de eventos (6.5).
- ❖ **Área 8.** Es aquí donde se mostrarán las condiciones establecidas por el usuario sobre los atributos de las relaciones (6.2.3.3).
- ❖ **Área 9.** Muestra las reglas o sentencias SQL almacenadas en el sistema.

- ❖ **Área 10.** En esta área se localizan los resultados de la aplicación de una regla o una sentencia SQL.
- ❖ **Área 11.** Se detalla la lista de los operadores disponibles para establecer condiciones sobre los atributos.
- ❖ **Área 12.** Sólo es visible en el modo regla y sirve para establecer la calidad mínima de las tuplas del resultado.
- ❖ **Área 13.** Es aquí donde el usuario puede introducir manualmente una sentencia SQL o DFSQL sin necesidad de utilizar la interfaz visual.
- ❖ **Área 14.** Establece si se desea realizar un enlace automático de relaciones a través de atributos (6.2.13).

El funcionamiento general de GDB-GUI consiste en crear una regla o una sentencia SQL que representa una pieza de conocimiento experto y que será aplicada sobre un conjunto de datos simulados (área 7), reales o de calibración. Para ello se utilizan las relaciones (área 3) para seleccionar un conjunto de atributos (área 4). Sobre estos atributos se establecen un conjunto de condiciones (área 8) que incluyen a operadores (área 11). La regla o sentencia se guardará en el sistema (área 9) y será posible aplicarla sobre los datos obteniendo un resultado (área 10). Este proceso se ampliará en [6.2.3](#).

GDB-GUI ha sido documentada e implementada en inglés para facilitar su uso a los diferentes miembros internacionales del consorcio de “GIADA team”.

Los accesos al SBDR se han minimizado a costa de un trabajo mayor desde GDB-GUI. Todas las tablas de usuario se cargan en memoria al inicio de la sesión y sólo se accede al SBDR en caso de modificaciones. Esto favorecerá la agilidad la respuesta cuando existan múltiples usuarios conectados a la vez, pero incrementa los requerimientos de memoria del computador que ejecute GDB-GUI.

6.2.1 Diseño de la BDR de GDB

El diseño del esquema relacional, que almacena todos los datos de GDB, expresado mediante el modelo entidad-relación, deriva directamente de la estructura jerárquica de las telemetrías definida en [Ilustración 39 \(3.12 Datos generados por GIADA\)](#).

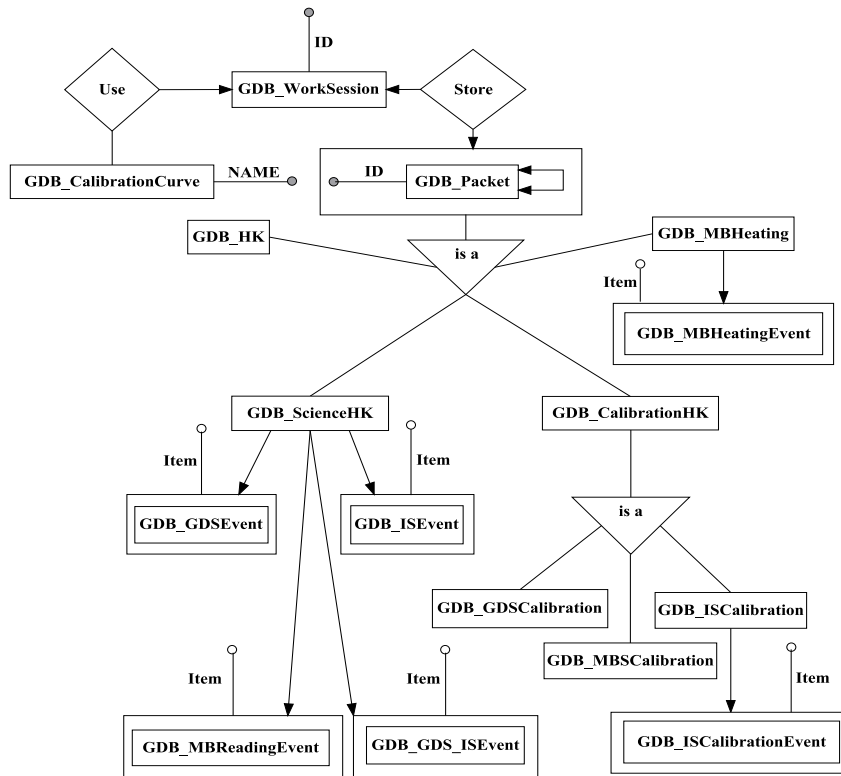


Ilustración 74. Diagrama entidad-relación de la base de datos relacional de GIADA.

Las tablas resultantes de este diseño, la descripción sintáctica y semántica de cada atributo, puede encontrarse en el [Apéndice A](#).

6.2.2 Menú principal

Se pasa a describir las opciones presentes en el menú principal (área 1 en la [Ilustración 73](#)):

1. **“File”**. Da acceso al importador de datos (6.2.5) de ficheros de EGSE y ROSIS.

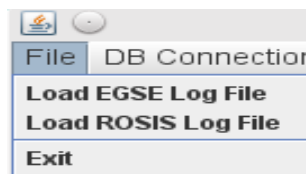


Ilustración 75. GDB-GUI. Menú principal-File.

La opción “Exit” permite finalizar la conexión con el SGBDR y cerrar GDB-GUI.

2. **“DB connection”**. Permite conectar y desconectar del SGBDR.

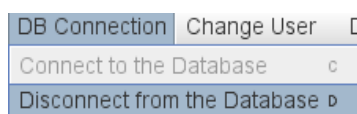


Ilustración 76. GDB-GUI. Menú principal-DB Connection.

3. **“Change user”**. Con esta opción es posible cambiar entre usuarios. La gestión de usuarios se detalla en [6.2.6](#).

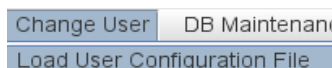


Ilustración 77. GDB-GUI. Menú principal-Change user.

4. **“DB maintenance”**. Aquí se encuentran las acciones posibles sobre las tablas relacionales, difusas y deductivas de GDB-GUI. En [6.2.9](#) se amplía la descripción de estas acciones.

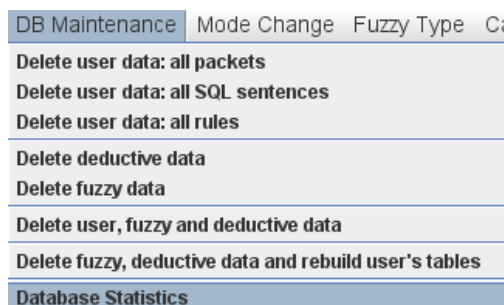


Ilustración 78. GDB-GUI. Menú principal-DB Maintenance.

5. **“Mode change”**. Con esta opción es posible cambiar entre los dos modos de funcionamiento de GDB-GUI: “SQL mode” y “Rule mode”.

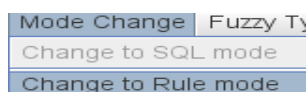


Ilustración 79. GDB-GUI. Menú principal-Mode Change.

6. **“Fuzzy type”**. Muestra la información sobre los tipos de datos difusos presentes en GDB-GUI ([6.4.1](#)).

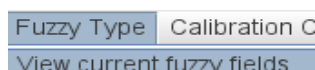


Ilustración 80. GDB-GUI. Menú principal-Fuzzy type.

7. **“Calibration curve”**. Permite aplicar las curvas de calibración ([6.2.7](#)) a los resultados y acceder a la calculadora de curvas.

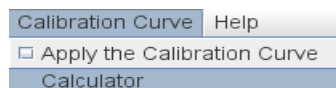


Ilustración 81. GDB-GUI. Menú principal-Calibration curve.

8. **“Help”**. Muestra información del autor, de la GDB-GUI y de la estructura de la BDR ([6.2.1](#)).

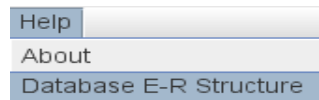


Ilustración 82. GDB-GUI. Help.

6.2.3 Creación y ejecución de consultas

Se denominará *consulta* a una regla o sentencia SQL en GDB-GUI.

A la hora de crear y ejecutar una consulta, se debe de seguir un proceso que consta de los siguientes pasos:

- a) Construcción de una consulta vacía.
- b) Selección de las relaciones y atributos de la consulta.
- c) Creación de las condiciones sobre atributos.
- d) Aplicación de la consulta y obtención de resultados.

Veamos en detalle cada uno de ellos:

6.2.3.1 Construcción de una consulta vacía

Inicialmente, se ha de disponer de una pieza de conocimiento, normalmente extraída de un documento, que se desea representar mediante una consulta. Hay que indicar a GDB-GUI cómo se llamará consulta que deseamos crear. Con esta información, se realiza las operaciones oportunas (de forma transparente al usuario) para crear una estructura vacía que soporte la nueva información. En [6.3](#) y [6.4](#) se describirán las tablas asociadas al almacenamiento de una sentencia o regla, respectivamente.

Las consultas se almacenan en categorías. Veamos cómo se realiza la gestión tanto de categorías como de consultas.

6.2.3.1.1 Gestión de categorías

GDB-GUI guarda las consultas en un árbol de categorías (área 9 en la [Ilustración 73](#)). Cada categoría (también llamada nodo) puede contener un conjunto de sub-categorías y/o un conjunto de consultas. Una categoría o una consulta sólo pueden tener a una categoría padre. Existe una categoría llamada raíz (“root”) que es el ancestro de cualquier otra categoría o consulta.

Un usuario puede crear, borrar, editar, mover y pedir información sobre una categoría.

Para crear, borrar o pedir información de una categoría, basta con pulsar con el botón derecho de ratón sobre ella, como se muestra en la siguiente ilustración.

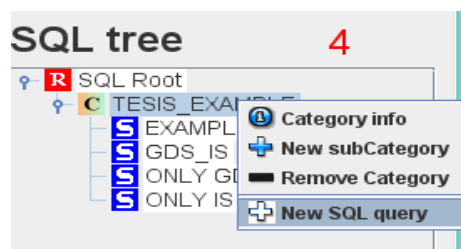


Ilustración 83. GDB-GUI. Acciones posibles sobre una categoría.

El número 4 en rojo, indica el número de consultas en total almacenadas en el nodo raíz.

El usuario puede crear una nueva categoría (seleccionando la opción “New subCategory”). En este momento, podrá introducir el nombre de la nueva categoría y opcionalmente una descripción de la misma, como se muestra en la siguiente ilustración:

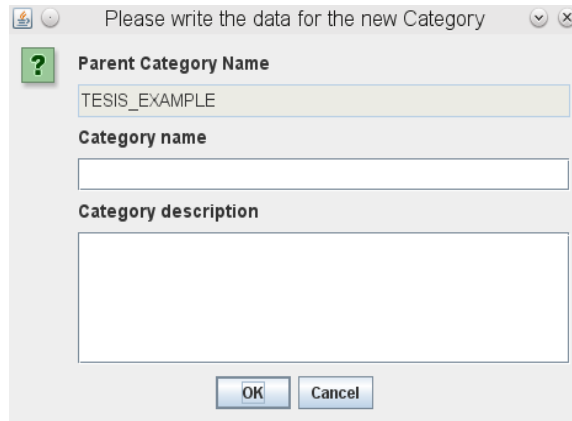


Ilustración 84. GDB-GUI. Creación de una categoría.

Para borrar una categoría (opción “Remove Category”), el usuario debe confirmar que realmente desea realizar la operación, como se indica en la siguiente ilustración:

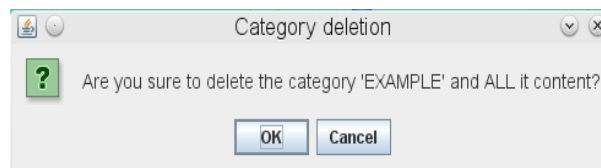


Ilustración 85. GDB-GUI. Borrado de una categoría.

El borrado de una categoría eliminará todas las consultas y categorías que almacene de forma recursiva.

La opción: “Category info” muestra la siguiente información sobre una categoría:

- a) Identificador de la categoría padre.
- b) Nombre de la categoría padre.
- c) Nombre de la categoría.
- d) Identificador de la categoría.
- e) Conteo del número y tipo de hijos que almacena.
- f) Usuario que creó la categoría.
- g) Cuando se creó la categoría.
- h) Descripción de su contenido.

Un ejemplo de esta información se muestra a continuación:

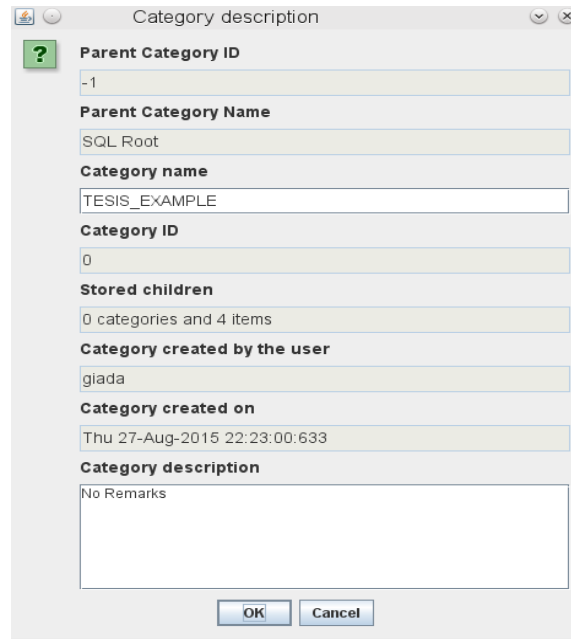


Ilustración 86. GDB-GUI. Información acerca de una categoría.

El usuario puede editar en esta ventana el nombre de la categoría o la descripción.

GDB-GUI permite mover una categoría o consulta. Para ello basta con pulsar botón izquierdo sobre el elemento a mover y arrastrar y soltar sobre el destino. El cursor del ratón cambiará a un rectángulo vacío durante la operación.

6.2.3.1.2 Gestión de consultas

Las acciones posibles sobre consultas son: creación, borrado, edición, movimiento y petición de información.

En el caso de que se desee crear una consulta, el usuario debe seleccionar inicialmente la categoría que la alojará (ver *Ilustración 83*), usando el botón derecho del ratón y seleccionar la opción "Add rule" o "New SQL query". Se debe establecer el nombre de la consulta y la fuente del conocimiento (sólo visible en reglas) que se desea reflejar.

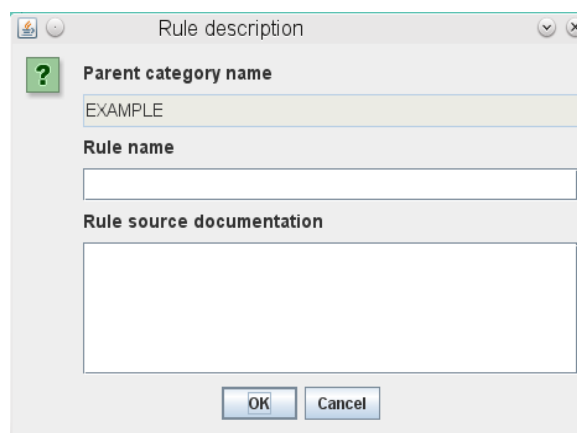


Ilustración 87. GDB-GUI. Creación de una consulta.

Utilizando el botón derecho sobre una consulta ya creada, se crea un menú con las opciones de información y borrado, como se muestra en la siguiente ilustración.

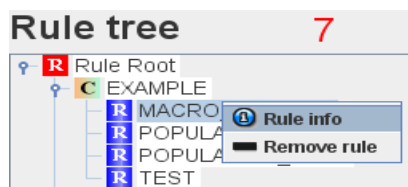


Ilustración 88. GDB-GUI. Información y borrado consulta.

Si se elige el borrado (“Remove rule” o “Remove SQL sentence”), aparecerá una petición de confirmación antes de realizar la tarea, como ya se hizo en el borrado de categorías (*Ilustración*).

Con la petición de información (“Rule info” o “SQL sentence info”), se muestra la siguiente información sobre una consulta:

- a) Nombre de la categoría padre.
- b) Nombre de la consulta.
- c) Tabla intensiva (sólo en el caso de reglas).
- d) Identificador de la consulta.
- e) Sentencia SQL o DFSQL de la consulta.
- f) Fuente del conocimiento (sólo en el caso de reglas).
- g) Historia de cambios de la consulta.

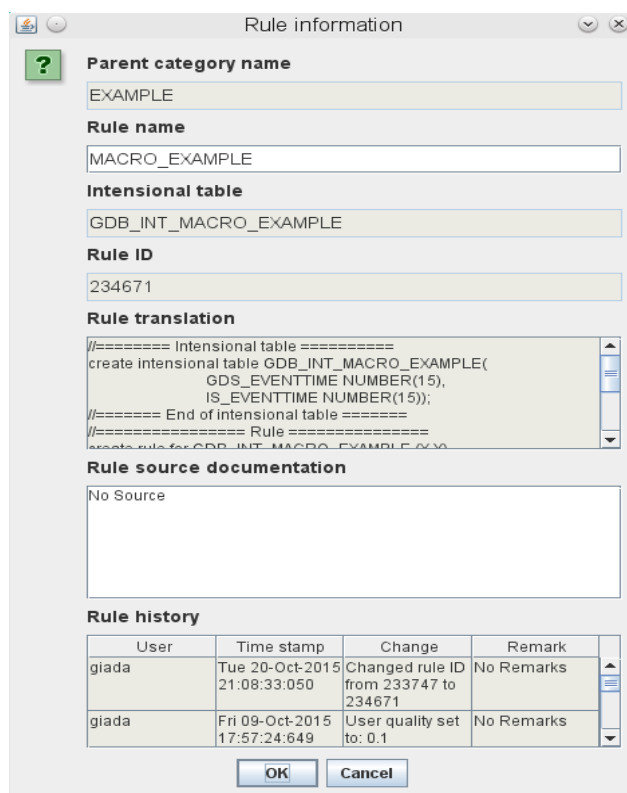


Ilustración 89. GDB-GUI. Información de una consulta.

Cualquier cambio que afecte a una consulta se verá reflejado en su historia. Las entradas en la tabla de historia se realizan de forma automática por GDB-BUI. La historia de una consulta permite conocer qué cambio se ha producido, quien lo ha realizado y cuando, lo que posibilita mantener una trazabilidad de la consulta, y en definitiva la trazabilidad del conocimiento almacenado.

En esta ventana de información, se podrá actualizar el nombre y la fuente (sólo en el caso de la regla).

Se puede mover una consulta de una categoría a otra arrastrando y soltando el botón izquierdo del ratón, como ya se indicó en [6.2.3.1.1](#).

6.2.3.2 Selección de las relaciones y atributos

A la hora de expresar la pieza de conocimiento deseada, el usuario de GDB-GUI tiene a su disposición la lista de relaciones y los atributos correspondientes (área 3 en la [Ilustración 73](#)).

Al pulsar con el ratón sobre una relación, aparecerán los atributos que almacena. Para cada atributo, pulsando el botón derecho del ratón, aparece una ventana con los siguientes datos:

- a) Nombre del atributo.
- b) Valores mínimo y máximo.
- c) Curva de calibración (sólo si tiene una asignada a ella).
- d) Descripción del atributo.

En la siguiente ilustración se muestra la información sobre el atributo `gds_eventtime` en la relación `GDB_GDSEVENT`.

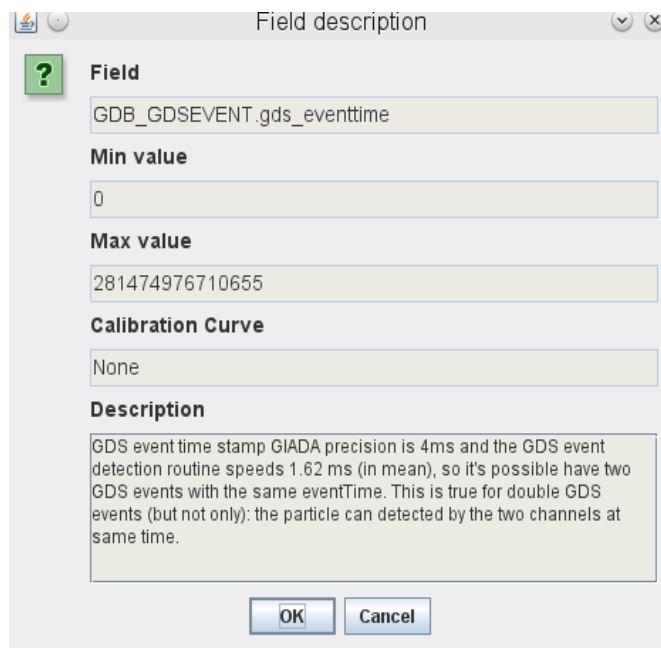


Ilustración 90. GDB-GUI. Información acerca de un atributo.


La gestión de las curvas de calibración se detallará en [6.2.7](#). Tanto la descripción de los atributos como las curvas de calibración son construidas por GDB-GUI a partir un conjunto de ficheros de entrada descritos en [6.2.8](#).

La información de cada atributo, ayudará al usuario en su labor de creación de reglas y sentencias.

El usuario no podrá crear nuevas relaciones o atributos al margen de los ya definidos. En caso de que sea necesario para el cálculo de la consulta, GDB-GUI lo hará de forma transparente. En cambio, sí podrá borrar su contenido o reconstruir la tablas, como se detallará en [6.2.9](#).

Existe una forma de evitar la interfaz visual y ejecutar una consulta directamente en SQL o DFSQL. Para ello es necesario usar el botón del área 13 ([Ilustración 73](#)).

6.2.3.3 Creación de las condiciones sobre atributos

Una vez seleccionados los atributos, es posible cambiar el orden de aparición (que será el orden en el que aparecerán en el resultado) mediante el botón izquierdo del ratón, arrastrando y soltando el atributo elegido. También es posible borrar un atributo concreto o todos los ya elegidos utilizando el botón derecho y pulsando el botón .

En la siguiente ilustración se muestran el menú para borrar un atributo previamente seleccionado.

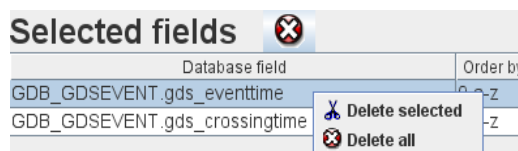


Ilustración 91. GDB-GUI. Borrado de un atributo.

Si se está en el modo de sentencias SQL, es posible establecer el orden (ascendente: a-z o descendente: z-a) en el que aparecerán las tuplas en la relación. GDB-GUI sólo permite dos niveles de ordenación de atributos (0 y 1) en la misma consulta ([6.2.8](#)). En la siguiente imagen se muestra el menú para ordenar por atributos en sentencias SQL.

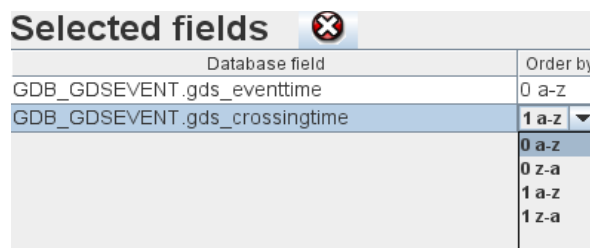


Ilustración 92. GDB-GUI. Menú de ordenación por atributos en sentencias SQL.

El usuario puede establecer una lista de condiciones sobre los atributos seleccionados. Para ello se utilizan comparadores binarios (área 11 en la [Ilustración 73](#)). Este conjunto se amplía con los comparadores difusos generalizados ([5.1.1.2](#)) en caso de que GDB-GUI esté en el modo de regla.

Se podrá elegir un único atributo y un comparador. En este caso, aparecerá un diálogo adicional para establecer el valor de una constante que se utilizará como parte derecha del comparador (la izquierda será el atributo seleccionado). En la siguiente ilustración se muestra como establecer una comparación `gds_eventtime > 100`:

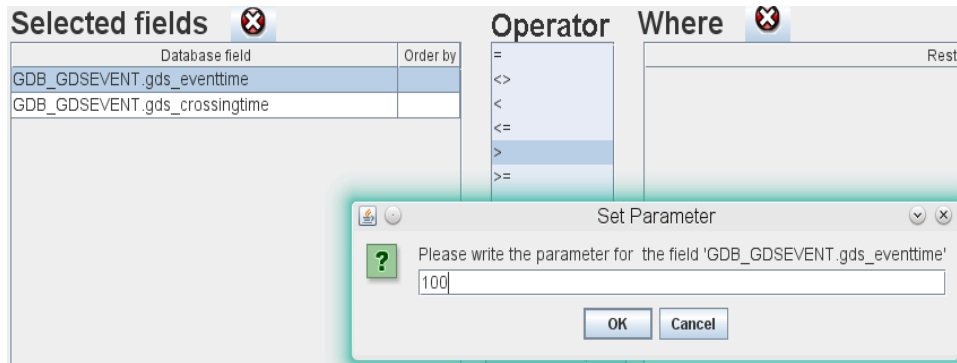



Ilustración 93. GDB-GUI. Creación de una comparación con un único atributo.

Una vez creada la comparación, ésta aparecerá reflejada en el área 8 (*Ilustración 73*), como se muestra en la siguiente ilustración:



Ilustración 94. GDB-GUI. Almacenamiento de una comparación con un único atributo.



El botón  permite añadir un segundo operador ('+' o '-') y un operando a la parte derecha de la comparación, lo que permite implementar intervalos (5.1.2.5).

En la siguiente ilustración, se ha añadido el operador '-' y el operando 12 a la condición `gds_eventtime > 100`.



Ilustración 95. GDB-GUI. Operador extra a la derecha de la comparación.

El botón  borra, si existe, el operador de la parte derecha de la comparación.

Si GDB-GUI está en el modo de sentencia SQL, aparecen dos botones adicionales. El primero  permite cambiar el operador entre condiciones. Por defecto se utiliza un 'y' lógico ("and"), aunque con este botón se puede utilizar un 'o' ("or") o bien revertir los cambios. El segundo botón () intercambia el operador derecho e izquierdo de la comparación.

También es posible seleccionar dos atributos y utilizar un comparador como se indica en la siguiente ilustración:

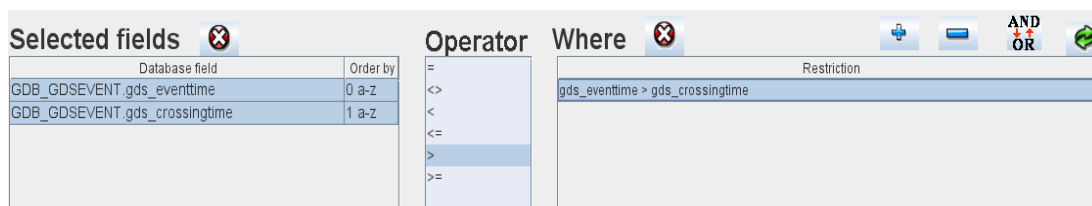



Ilustración 96. GDB-GUI. Creación de una comparación con dos atributos.

6.2.3.4 Aplicación de la consulta y obtención de resultados

Una vez creada la consulta, se puede aplicar sobre los datos almacenados. Para ello es necesario utilizar el botón  del área 10 de la *Ilustración 73*. Un ejemplo de aplicación se muestra en la siguiente ilustración:

GDS_EVENTTIME	GDS_CROSSINGTIME
5508943	10
5651616	10
5763377	10
5845796	10
5900384	10
6055512	10
6139594	10





Ilustración 97. GDB-GUI. Resultado de una consulta.

El número **7** en rojo indica el número de tuplas de la relación que satisfacen las condiciones establecidas.

Los atributos y el orden de aparición en la tabla resultado es el indicado en la selección de atributos de la consulta.

En el modo de sentencias SQL, las tuplas se ordenarán según lo haya establecido el usuario. En el modo de reglas, aparecerá un atributo adicional llamado “Quality” con la calidad de cada tupla (5.4.6).

Con el resultado en pantalla, el usuario puede optar por varias opciones:

- Borrar todas las tuplas con botón .
- Salvar el resultado en un fichero separado con tabuladores con el botón  usando el nombre de fichero actual.
- Salvar el resultado en un fichero y asignarle un nombre con .
- Ejecutar la consulta de nuevo con el botón .

El fichero con el resultado (separado con tabuladores) puede ser cargado en hojas de cálculo u otras bases de datos.

Después de ejecutar la consulta, aparece una ventana indicando la cantidad de tuplas devueltas y el tiempo empleado para obtener el resultado:

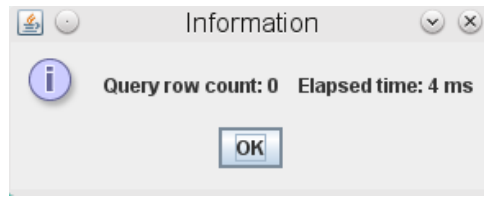


Ilustración 98. GDB-GUI. Ventana de resumen tras la ejecución de una consulta.

6.2.4 Almacenamiento y recuperación de consultas

Cualquier consulta (regla o sentencia SQL) se almacena (y se recupera) desde un conjunto de relaciones del SGBDR. Estas relaciones se enumerarán en [6.3.1](#) (sentencias) y [6.4.4](#) (reglas).

Existe un esquema general de almacenamiento y recuperación, independientemente del tipo de consulta:

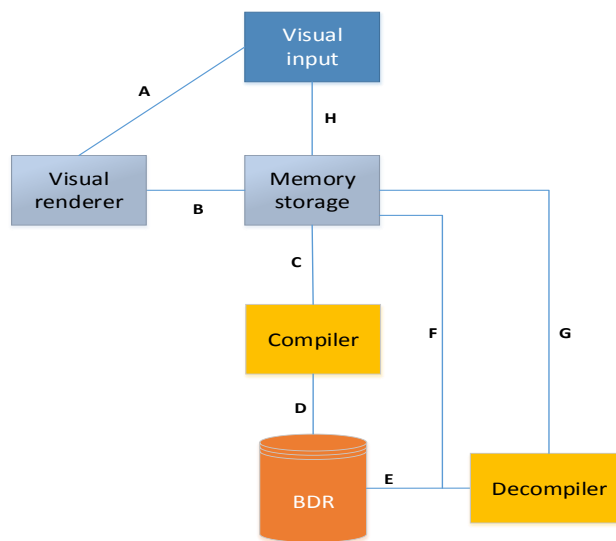


Ilustración 99. GDB-GUI. Esquema general del almacenamiento y recuperación de consultas.

La interfaz de GDB-GUI ([Ilustración 73](#)), consta de un conjunto de componentes visuales (cuadros de texto, listas, combos, etcétera). El usuario utilizará esta interfaz para crear una consulta (“visual input”) formada por un conjunto de relaciones, atributos y condiciones ([6.2.3](#)).

Inicialmente, una consulta es almacenada (arista **A**) en componentes visuales, que son analizados (“Visual renderer”) en un determinado orden, con el fin de completar (arista **B**) una serie de estructuras de datos en memoria (“Memory storage”). Estas estructuras son examinadas (arista **C**) y pasadas a un compilador (“Compiler”) que construirá una expresión válida en SQL o DFQL que representa la consulta del usuario. Esta expresión será almacenada (arista **D**) en la base de datos relacional (BDR) junto con otras propiedades de la consulta: categoría padre, usuario, etcétera (ver [6.3.1](#) y [6.4.4](#)).

Al iniciar el sistema, todas las consultas son recuperadas de la BDR (arista **E**). Esta información es volcada en el “Memory storage” (arista **F**). La expresión asociada a la consulta, recibe un tratamiento especial. Es analizada por el deconstructor (“Decompiler”) que extrae los componentes de la consulta (relaciones, atributos y condiciones) antes de almacenarlos (arista **G**). Los componentes visuales de la interfaz son asignados con los valores recopilados del “Memory storage” (arista **H**) y mostrados al usuario.

6.2.5 Importador de datos

GDB-GUI incluye el *importador de datos* (*Ilustración 72*), que permite cargar en la BDR de GDB-GUI la información sobre partículas proveniente de ROSIS, EGSE o del simulador de eventos.

En el caso de ROSIS y EGSE, el formato de los datos aceptado por el importador se basa en ficheros de texto con datos hexadecimales. Estos ficheros, tras ser procesados, generan un flujo de datos hexadecimal que llega al importador. Este flujo es analizado y diseccionado hasta obtener los valores individuales de cada campo, según el formato definido por el SW de GIADA (*3.12*). Estos campos serán utilizados como valores de los atributos de las relaciones definidas en la BDR.

El simulador de eventos (*6.5*), en última instancia, también genera un flujo de datos hexadecimales, que será procesado de forma análoga.

Es aspecto de una telemetría de EGSE es la siguiente:

```
Sat Apr 13 08:42:41 2002
TC Packet Sent to GIADA:
  APID = 90, 12 (PRIVATE)
  Packet Length: 145
  Ack: 0001
  Packet Type, Subtype: 18, 3
-----
1D AC C0 00 00 91 11 12 03 00 00 01 00 00 00 1E
00 00 05 A1 05 C0 00 00 00 3C 06 63 00 00 02 58
00 32 00 A3 00 A3 00 00 00 78 00 00 00 78 AF 03
26 72 F5 78 FE 20 00 00 00 00 00 00 0E 10 9F 1F
1A 10 00 00 00 4B 00 0F 0F 0F 0F 0F 00 00 0E 10
19 02 9F 00 0A C1 F8 00 00 00 01 2C 0A 10 00 00
01 68 00 00 0E 10 19 C5 1E 07 1A 25 00 00 00 00
00 00 00 00 00 00 00 05 00 00 00 28 00 00 00 28
00 00 00 00 00 00 00 00 00 28 00 28 00 00 00 00
00 00 00 00 82 E3
=====
```

Y la misma telemetría en el formato de ROSIS tiene el siguiente aspecto:

```
2002,103,08,42,41,885 :
TYP FIELD_1 FIELD_2 FIELD_3A FIELD_3B LEN
TM SCOE 00000000 00000000 000000CD
0000: 03 7F 00 00 00 00 00 00 32 30 30 32 2C 31 30 33 2C 30 38
2C 34 32 2C 34 31 2C 38 38 35 20 00 00
0020: 0F EE C0 00 00 A6 3C B7 EF 81 03 75 00 AA 02 00 06 01 80
54 03 1D AC C0 00 00 91 11 12 03 00 00
0040: 01 00 00 00 1E 00 00 05 A1 05 C0 00 00 00 3C 06 63 00 00
02 58 00 32 00 A3 00 A3 00 00 00 78 00
0060: 00 00 78 AF 03 26 72 F5 78 FE 20 00 00 00 00 00 00 0E 10
9F 1F 1A 10 00 00 00 4B 00 0F 0F 0F 0F
0080: 0F 00 00 0E 10 19 02 9F 00 0A C1 F8 00 00 00 01 2C 0A 10
00 00 01 68 00 00 0E 10 19 C5 1E 07 1A
00A0: 25 00 00 00 00 00 00 00 00 00 00 00 05 00 00 00 28 00 00
00 28 00 00 00 00 00 00 00 00 28 00
00C0: 28 00 00 00 00 00 00 00 00 82 E3 00 00
== EOM ==
```

6.2.6 Gestión de usuarios

GDB-GUI permite acceder a múltiples usuarios simultáneamente al mismo SBDR. Cada usuario es identificado al inicio de una sesión de trabajo en GDB_GUI y sus preferencias de configuración son aplicadas (6.2.8).

Al crear un usuario, no existen reglas ni sentencias SQL inicialmente almacenadas. Las tablas relacionales con los datos clásicos y las requeridas por el modelo GDB para manejar la imprecisión y la deducción, son construidas, pero con contenido vacío.

Cada usuario podrá disponer de sus propios datos de partículas, reglas y sentencias SQL.

6.2.7 Curvas de calibración

Las curvas de calibración, su necesidad y su aplicación en G. fueron tratadas en la sección en 2.2.3.2. De forma resumida, estas curvas convierten valores de los sensores expresado en cuentas digitales en unidades físicas. G. posee dos cadenas de adquisición de datos (*principal* y *redundante*) con sus propias curvas de calibración.

El EGSE dispone de dos archivos con la definición de las curvas de calibración de cada cadena de adquisición (6.2.8). Estos archivos son cargados y procesados al inicio de GDB-GUI. A continuación se muestra el aspecto de la curva de calibración (expresada como un polinomio de grado 5) para la temperatura del láser 1 del GDS en la cadena principal de adquisición:

```
[GDS Laser 1 Temperature]
REM Laser 1_1
REM Coefficients are for N<->°C conversion
REM updated 10/06/02
a=-3.2970E-15
b=3.3914E-11
c=-1.3919E-7
d=2.8730E-4
e=-3.3101E-1
f=230.50
a1=-2.8589E-07
b1=5.0790E-05
c1=3.7055E-04
d1=-3.0852E-01
e1=-2.0317E+01
f1=3.2314E+03
min=-30
max=60
```

En esta curva, a es el coeficiente de grado 5 del polinomio y f el de grado 0, mientras que a_1, \dots, f_1 son los coeficientes del polinomio utilizado en la conversión inversa (de unidades físicas a cuentas). Los valores min y max representan el intervalo válido dónde aplicar la curva de calibración.

GDB-GUI implementa dos utilidades (ver *Ilustración 81*) relacionados con las curvas de calibración a fin de facilitar la creación de consultas. La primera es una calculadora, que permite al usuario convertir cuentas en unidades físicas, y viceversa, aplicando los ficheros de curvas del EGSE.

En la siguiente ilustración se muestra la calculadora en funcionamiento:

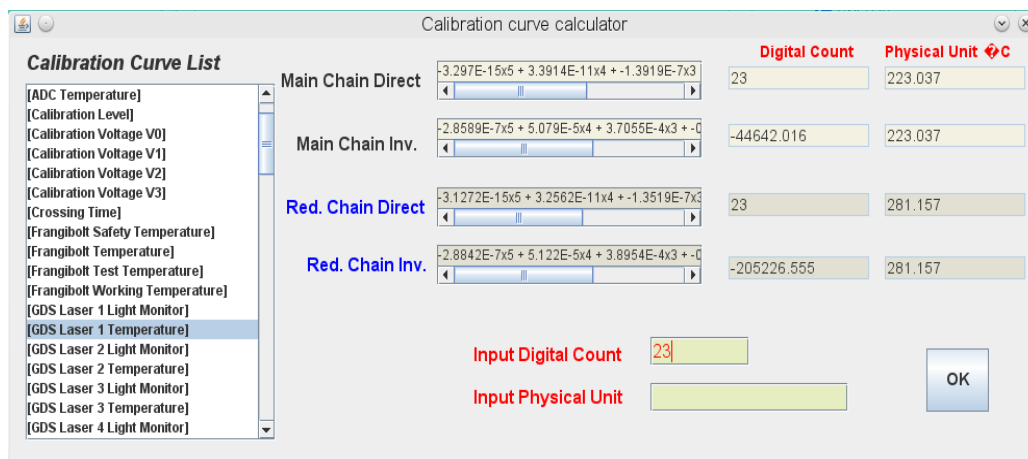


Ilustración 100. GDB-GUI. Calculadora de curvas de calibración.

La segunda utilidad permite activar o desactivar la conversión a unidades de los valores en cuentas presentes en la tabla de resultado. Para ello se utiliza las curvas de calibración asignadas a cada atributo (6.2.3.2). A modo de ejemplo se muestran dos resultados de una consulta, una en cuentas y la otra con la conversión de unidades activada.

GDS_EVENTTIME	GDS_CROSSINGTIME
5508943	10
5651616	10
5763377	10
5845796	10
5900384	10
6055512	10
6139594	10

Ilustración 101. GDB-GUI. Tabla sin la conversión de curvas de calibración aplicada.

GDS_EVENTTIME	GDS_CROSSINGTIME_us
5508943	100.0
5651616	100.0
5763377	100.0
5845796	100.0
5900384	100.0
6055512	100.0
6139594	100.0

Ilustración 102. GDB-GUI. Tabla con la conversión de curvas de calibración aplicada.

6.2.8 Ficheros de configuración, directorios de entrada y salida

GDB-GUI utiliza un conjunto de ficheros de configuración y directorios de entrada y salida para poder ejecutarse. La descripción de todos ellos se detalla a continuación:

Ficheros de configuración:

a) SystemConfiguration.txt. Contiene los siguientes entradas:

1. Descripción de GDB-GUI.
2. Localización del fichero de usuario (por defecto UserConfiguration.txt).

3. Localización del fichero utilizado para reconstruir las tablas clásicas que incorporan atributos difusos ([6.2.9](#)).
4. Localización del directorio de log.
5. Información sobre el árbol de llaves externas ([6.2.13](#)).

b) UserConfiguration.txt. Dispone de los siguientes ítems:

1. Datos de conexión con el SGDBR.
2. Modo de GDB-BUI por defecto (modo regla o modo SQL).
3. Directorios de entrada (ver más adelante en esta sección).
4. Directorios de salida (ver más adelante en esta sección).
5. Número de niveles de ordenamiento de atributos ([6.2.3.3](#)).
6. Número de decimales usados en la calculadora de curvas de calibración ([6.2.7](#)).

Directorios de entrada:

- a) **Calibration_curve.** Guarda los ficheros “ConvFactorsMainFS.ini” y “ConvFactorsRedundantFS.ini” que albergan las curvas de calibración ([2.2.3.2](#) y [6.2.7](#)) de las cadenas de adquisición principal y redundante, respectivamente.
- b) **EGSE.** Almacena los ficheros hexadecimales del EGSE ([6.2.5](#)).
- c) **Field_description.** Alberga la descripción de los atributos ([6.2.3.2](#)).
- d) **ROSIS.** Contiene los ficheros hexadecimales de ROSIS ([6.2.5](#)).
- e) **Script.** Guarda los ficheros usados como “scripts” por el simulador de eventos (6.5).
- f) **Table_definition.** Almacena el fichero usado para reconstruir las tablas clásicas que incorporan atributos difusos ([6.2.9](#)).

Directorios de salida:

- a) **Logger.** Es en este directorio donde se almacenarán los ficheros con los históricos de GDB-GUI.
- b) **ResultTable.** Este es el directorio donde se guardarán por defecto los ficheros con el resultado de las consultas exportados por el usuario ([6.2.3.4](#)).

6.2.9 Gestión de la base de datos relacional

GDB-GUI ofrece al usuario un conjunto de utilidades para manejar las relaciones con información clásica, difusa y deductiva (ver *Ilustración 78*) en el menú principal (“DB maintenance”):

- a) **“Delete user data: all packets”.** Borrará el contenido de los datos clásicos sobre partículas.
- b) **“Delete user data: all SQL sentences”.** Eliminará todas las sentencias SQL creadas.
- c) **“Delete user data: all rules”.** Suprimirá todas las reglas creadas.
- d) **“Delete deductive data”.** Borrará el contenido de las tablas con información deductiva ([5.1.2.3](#)).
- e) **“Delete fuzzy data”.** Eliminará el contenido de las tablas con información difusa ([5.1.1.3](#)).
- f) **“Delete user, fuzzy and deductive data”.** Realizará las acciones **d)** y **e)**.
- g) **“Delete user, fuzzy, deductive data and rebuild user’s table”.** Procederá a ejecutar **f)** y además destruirá y creará las tablas con información clásica.
- h) **“Database statistics”.** Informa acerca del número de paquetes almacenados

En g), la reconstrucción de tablas clásicas se realiza desde un fichero externo que almacena su descripción (6.2.8). Esta descripción incluye los atributos clásicos que tienen asociado un tipo de dato difuso.

6.2.10 Gestión del SGBDR

El modelo GDB y GDB-GUI dispone de un conjunto de ficheros de instalación y configuración (ver *Apéndice B*) del SGBDR Oracle©, que pasa a describirse:

- a) **“Instalación Oracle y DFSQL.doc”**. Documento de descripción del proceso de instalación de Oracle© y de los servidores de deductivos y difusos en un computador.
- b) **“DeductiveFuzzyServer”**. Este directorio contiene los ficheros necesarios para instalar el servidor deductivo difuso en el SGBDR Oracle©.
- c) **“FuzzyServer”**. Se almacena en este directorio los ficheros de instalación el servidor difuso para el SGBDR Oracle©.
- d) **“CreateRol.sql”**. Este script crea un “rol” en Oracle© que habilita los permisos de acceso a los usuarios de GDB-GUI a las tablas deductivas y difusas.
- e) **“RuleStorage.sql”**. Script de creación de las tablas necesarias para almacenar reglas.
- f) **“SQL_Storage.sql”**. Este script crea las tablas que almacenarán las sentencias SQL.

6.2.11 Adaptación a otros sistemas

La configuración de GDB-GUI ha sido diseñada específicamente para el instrumento GIADA como interfaz del modelo de datos de GDB. La adaptación a otro sistema/instrumento, pasa por redefinir los siguientes ficheros de entrada (6.2.8):

- a) **Calibration_curve**. Con las curvas del nuevo sistema.
- b) **Field_description**. Descripción de los atributos de todas las tablas.
- c) **Table_definition**. Definición de las tablas clásicas.

En c) habría además que incorporar a los atributos que necesiten un tratamiento difuso, el tipo de dato difuso apropiado.

Opcionalmente, habría que actualizar el fichero utilizado para el enlace automático de tablas (6.2.13) si se desea incorporar esta capacidad en el nuevo sistema.

6.2.12 Nomenclatura de las tablas

Todos los nombres de las tablas relacionadas con GDB_GUI poseen el prefijo “GDB_”. Existen además prefijos adicionales para las tablas de constantes: “GDB_CTE_” y para las tablas intensivas: “GDB_INT_”.

La tabla intensiva construye su nombre con el prefijo mencionado más el nombre de la regla asignado por el usuario.

6.2.13 Enlace automático de tablas

El diseño de la BDR deriva directamente de la estructura jerárquica en forma de árbol de las telemetrías, lo que implica que también existe una estructura jerárquica en las relaciones, como se muestra en la siguiente ilustración:

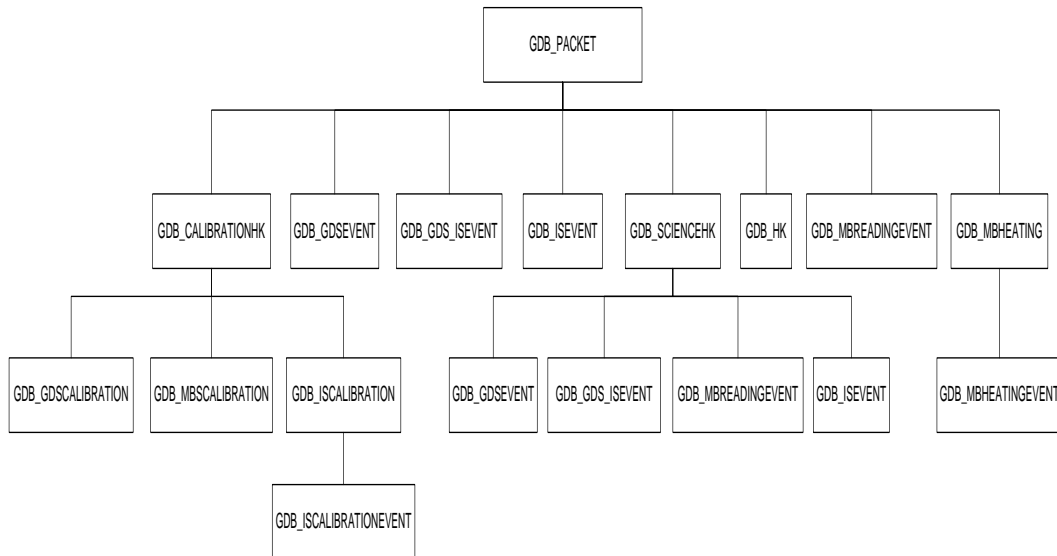


Ilustración 103. GDB-GUI. Estructura jerárquica de las relaciones clásicas.

Un atributo, presente en todas las relaciones, hace de nexo de unión entre todos los nodos del árbol. Este atributo es el ID del paquete en el que los datos llegaron al importador de GDB-GUI (6.2.5).

Esta jerarquización, permite enlazar (utilizar el operador *equi-reunión* 4.1.4) cualquier conjunto de relaciones dentro del árbol. GDB-GUI utiliza un fichero (6.2.8) para indicar la jerarquía de las relaciones y el atributo de unión entre ellas.

GDB-GUI permite al usuario activar o desactivar el enlace automático de relaciones (área 14 en la *Ilustración 73*).

A modo de ejemplo, supongamos que queremos crear una consulta que incluya atributos de las tablas GDB_Packet, GDB_ISCalibrationEvent y GDB_MBHeatingEvent.

Esta consulta, sin enlazar las relaciones, puede expresarse como:

```

Select GDB_ISCALIBRATIONEVENT.is_cal_eve_pzt_a_amplitude,
       GDB_MBHEATINGEVENT.mb_heat_eve_frequency,
       GDB_PACKET.packet_crc
From GDB_ISCALIBRATIONEVENT,
     GDB_MBHEATINGEVENT,
     GDB_PACKET;
  
```

Utilizando el enlace automático, la consulta quedaría como sigue (en rojo la parte de la sentencia añadida automáticamente):

```

Select GDB_ISCALIBRATIONEVENT.is_cal_eve_pzt_a_amplitude,
       GDB_MBHEATINGEVENT.mb_heat_eve_frequency,
       GDB_PACKET.packet_crc
From GDB_ISCALIBRATIONEVENT,
     GDB_MBHEATINGEVENT,
     GDB_PACKET
Where GDB_ISCALIBRATIONEVENT.ID = GDB_MBHEATINGEVENT.ID AND
     GDB_MBHEATINGEVENT.ID = GDB_PACKET.ID;
  
```

Sin el enlace automático, debe ser el usuario el que establezca a mano cada uno de los enlaces, lo que es engorroso y propenso a errores.

6.2.14 Herramientas de desarrollo de GDB-GUI

Las herramientas seleccionadas para el desarrollo GDB-GUI se han elegido para que permitan una ejecución multiplataforma. En concreto, trabaja en todas las plataformas que soporten la ejecución de una máquina virtual de Java: Windows, Linux y MacOS.

- a) **Lenguaje de programación Java**, para el desarrollo de GDB-GUI.
- b) **Java 1.8.0_31** como máquina virtual para ejecutar Java.
- c) **SQLJ** para integrar el uso de SQL en java.
- d) **Javacc 4.0-129.1.3** en la construcción del simulador de eventos.
- e) **Oracle JDeveloper 12.c** como entorno de desarrollo de GDB-GUI.
- f) **Oracle Database 10g Release 2 Enterprise Edition Version 10.2.0.1.0 64 bits**, como SGBDR.
- g) **Oracle SQL Developer 4.1.0.17** para acceder a las relaciones del SGBDR.

6.3 Interfaz SQL

La interfaz SQL permite al usuario un acceso clásico a los datos almacenados, pudiendo crear y ejecutar de forma visual sentencias SQL.

Esta interfaz hereda todas las características comunes a la creación y ejecución de consultas descritas en [6.2.3 Creación y ejecución de consultas](#).

Se pasan a detallar las particularidades a esta interfaz.

6.3.1 Almacenamiento de sentencias SQL

En la siguiente ilustración se muestra el esquema relacional que alberga la información de las sentencias SQL expresado mediante el modelo entidad-relación:

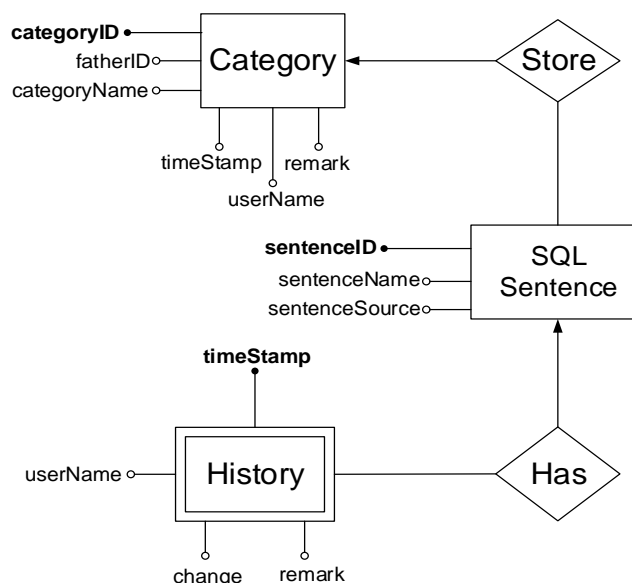


Ilustración 104. GDB-GUI. Diagrama entidad-relación del almacenamiento de sentencias SQL.

Este diseño genera tres relaciones que pasan a describirse.

La relación "GDB_SQL_CATEGORY" almacena la información relativa de la sentencia SQL en el árbol de telemetrías: posición (atributos: "CATEGORYID", "CATEGORYFATHERID" y "CATEGORYNAME"), fecha de creación ("TIMESTAMP"), usuario que creó la sentencia ("USERNAME") y comentarios del usuario ("REMARK").

En la relación "GDB_SQL_DATA" se encuentran las características principales de una sentencia: un identificador único ("SENTENCEID"), el identificador de la categoría que almacena la sentencia ("CATEGORYID"), el nombre que el usuario dio a la sentencia ("SENTENCE-NAME"), la sentencia en SQL ("SENTENCESOURCE") y el estado del enlace automático ("LINKSTATUS" [6.2.13](#)).

La relación "GDB_SQL_HISTORY" guarda el historial de cambios realizados sobre una sentencia: identificador de la sentencia ("SENTENCEID"), cuando se realizó el cambio ("TIMESTAMP"), qué usuario lo hizo ("USERNAME"), que cambio se produjo ("CHANGE") y comentarios "REMARK".

Los detalles de las tres relaciones pueden encontrarse en el DVD anexo, en el directorio "Instalación" fichero "SQL_Storage.sql" ([Apéndice B](#)).

6.4 Interfaz DFSQL

Con esta interfaz, el usuario puede crear y ejecutar de forma visual reglas que según el modelo de GDB.

Esta interfaz hereda todas las características comunes a la creación y ejecución de consultas descritas en [6.2.3 Creación y ejecución de consultas](#).

A continuación se detallan las particularidades de esta interfaz.

6.4.1 Atributos clásicos con componente difusa

La magnitud "tiempo" es una candidata natural en G. a ser tratada de forma difusa debido a las propias imprecisiones del instrumento en las marcas de tiempo (4ms ver [3.12.1](#)) y al método de detección de partículas ([7.1](#)). Es posible usar imprecisión en el resto de magnitudes (temperatura, voltaje, flujo de polvo, etcétera), pero, inicialmente, GDB-GUI se centrará sólo en el tiempo.

Todos los campos con información temporal de la BDR incorporan un tipo de dato difuso, por lo que son susceptibles de ser incorporados a una expresión que involucre comparadores difusos. La lista y la definición de estos campos se encuentra en el [Apéndice A](#) y puede ser consultada en el área 1 de la [Ilustración 73](#):

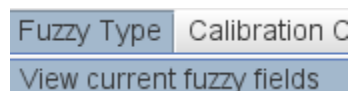


Ilustración 105. GDB-GUI. Menú para obtener la lista de atributos clásicos con componente difusa.

Como resultado se obtienen 25 atributos clásicos con componente difusa definidos en GDB-GUI, como se muestra en la siguiente ilustración:

Query result 25

FIELD	F_TYPE	MARGEN	MUCH
GDB_CALIBRATIONHK.CAL_HK_EVENTTIME	1	8	9
GDB_GDSEVENT.GDS_CROSSINGTIME	1	8	9
GDB_GDSEVENT.GDS_EVENTTIME	1	8	9
GDB_GDS_ISEVENT.GDS_IS_CROSSINGT...	1	2	3
GDB_GDS_ISEVENT.GDS_IS_EVENTTIME	1	8	9
GDB_GDS_ISEVENT.GDS_IS_FLIGHTTIME	1	2	3
GDB_GDS_ISEVENT.GDS_IS_PZT_A_TIME	1	8	9
GDB_GDS_ISEVENT.GDS_IS_PZT_B_TIME	1	8	9
GDB_GDS_ISEVENT.GDS_IS_PZT_C_TIME	1	8	9
GDB_GDS_ISEVENT.GDS_IS_PZT_D_TIME	1	8	9
GDB_GDS_ISEVENT.GDS_IS_PZT_E_TIME	1	8	9
GDB_ISCALIBRATIONEVENT.IS_CAL_EVE...	1	8	9
GDB_ISCALIBRATIONEVENT.IS_CAL_EVE...	1	8	9

Ilustración 106. GDB-GUI. Lista de atributos clásicos con componente difusa.

6.4.2 Creación de constantes

Cuando el usuario utilice una constante en la construcción de una regla, se creará una tabla constante, como indican las directrices del modelo GDB (5.1.2.6). Esta tabla incorpora un atributo que identifica la regla que originó su creación. La tabla será borrada cuando lo sea su regla asociada.

6.4.3 Limitaciones del servidor DFSQL

El servidor de SQL no soporta cierto tipo de expresiones SQL: “order by”, “purge”, “check”, “group”, “having”, etcétera. Estas expresiones no pueden utilizarse en la creación de una regla y por ello no parecen como opciones en GDB-GUI.

6.4.4 Almacenamiento de reglas

Se muestra en la siguiente ilustración el esquema relacional que almacena la información de las reglas expresado mediante el modelo entidad-relación:

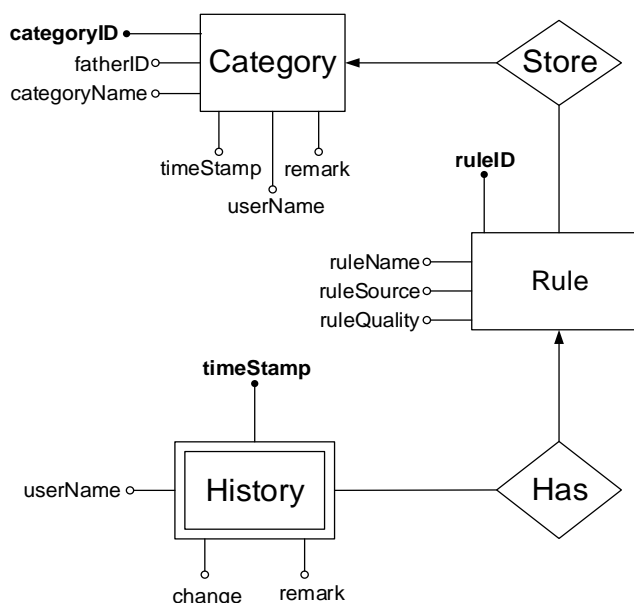


Ilustración 107. GDB-GUI. Diagrama entidad relación para el almacenamiento de reglas.

Este diseño genera tres relaciones, dos de ellas "GDB_RULECATEGORY" y "GDB_RULEHISTORY" son equivalentes, respectivamente, a las relaciones "GDB_SQL_CATEGORY" y "GDB_SQL_HISTORY" ya definidas en [6.3.1](#).

En la tercera relación "GDB_RULEDATA" se encuentran la características principales de una regla: un identificador unívoco ("RULEID"), el identificador de la categoría que almacena la regla ("CATEGORYID"), el nombre que el usuario dio a la regla ("RULENAME"), la sentencia en DFQL ("RULESOURCE"), el estado del enlace automático ("LINKSTATUS" [6.2.13](#)) y la calidad mínima de la regla asignada por el usuario ([6.2.3.4](#) "USERQUALITY").

La relación entre este esquema y el esquema de almacenamiento de reglas del modelo de datos GDB ([5.1.2.5.2](#)) se encuentra al ser el atributo "RULEID" (en la relación "GDB_RULEDATA") llave externa ([4.1.2](#)) de la relación "DED_INTENSIVA_CATALOG" (con llave primaria "idpred").

Los detalles de las tres relaciones pueden encontrarse en el DVD anexo, en el directorio "Instalación" fichero "RuleStorage.sql" ([Apéndice B](#)).

6.5 Simulador de eventos

Para poder validar el modelo de datos de GDB, así como su interfaz GDB-GUI ([Z](#)), es necesario verificar las reglas construidas con el modelo utilizando esta misma interfaz. Para ello se requiere de un conjunto de datos perfectamente controlado, que permita anticipar el resultado, que será contrastado con el calculado por GDB-GUI. Los conjuntos de datos de G. provienen de dos fuentes:

- a) Datos de la campaña de calibración de G.**
- b) Datos reales del cometa.**

Aunque estos datos son muy valiosos, a priori, no servirán a la hora de verificar reglas, ya que fueron generados u obtenidos con otros fines. Por ejemplo, supongamos que existe una regla que, aplicada a las fuentes **a)** y **b)** no genera resultados. Esta ausencia de datos, no indica que exista un error en la construcción de la regla, sino que simplemente no existen atributos que cumplan con las condiciones requeridas.

Por tanto, para la verificación, será necesario generar un conjunto de datos que cubran específicamente las necesidades de la regla. La regla, una vez verificada, podrá aplicarse sobre las fuentes **a)** y **b)**, utilizando los datos simulados como referencia para explicar los resultados obtenidos.

Más aún, debe de conocerse en profundidad el conjunto de datos aplicable a la regla, no solo la cualidad de los mismos (tipos de eventos) sino también la cantidad de ellos, a fin de realizar una comprobación exhaustiva.

Aunque se dispone de un modelo de G. funcionalmente equivalente al que se encuentra en Rosetta ([3.1](#)), no dispone de sensores reales, sólo de una simulación de los mismos que no permite una generación precisa de datos.

En consecuencia, es necesario construir un "simulador de eventos" que nos proporcione un conjunto de datos con las cualidades y cantidades buscadas y que no necesite de un HW adicional.

El diseño del simulador de eventos se ha enfocado como un lenguaje de generación de datos con el que el usuario creará pequeños programas o "scripts" utilizando un editor de textos. El

script será compilado y ejecutado dentro de GDB-GUI produciendo como salida un conjunto de datos que será insertado dentro del SGBDR. Este proceso se muestra en la siguiente ilustración.

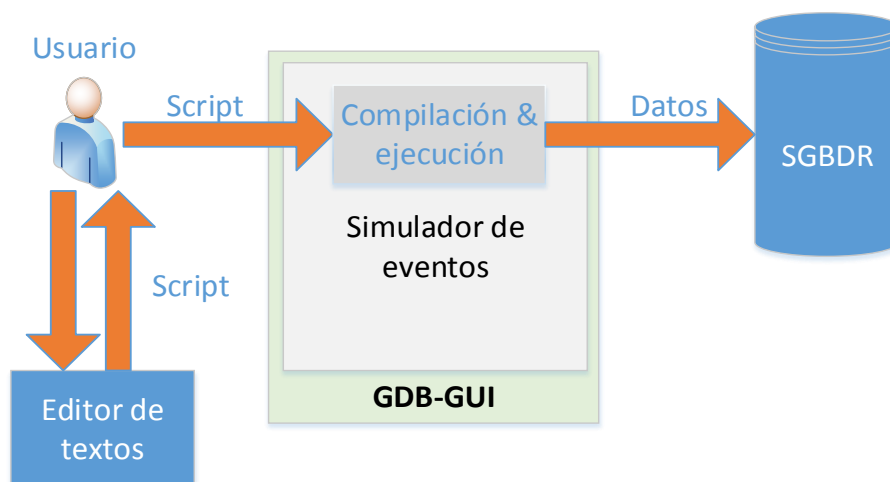


Ilustración 108. GDB-GUI. Generación de datos con el simulador de eventos.

El lenguaje permite asignar un valor a todos y cada uno de los atributos de las tablas que almacenan los datos científicos de G. (*Apéndice A*). Así mismo, permite controlar de forma exacta la cantidad de datos generados, el tamaño de los paquetes y el flujo de los mismos (paquetes/segundo).

El lenguaje no incorpora variables ni funciones y la única sentencia de control es la de iteración de bloques de sentencias. La gramática se ha especificado con javacc (<https://javacc.dev.java.net/>) creando el analizador léxico y el sintáctico del lenguaje ([Aho86]). La generación de código de estos dos analizadores se realiza en lenguaje java, por lo que pueden ser compilados dentro de GDB. La gramática del lenguaje puede encontrarse en el DVD anexo (*Apéndice B*).

El control de la ejecución del script reside en un “ejecutor” o “runner”, produciendo como salida, eventos científicos cuya estructura y valor han sido especificados en el script. El “runner” también implementará las directrices del tamaño del paquete de G. A fin de poder implementar correctamente el control de flujo, se ha construido un planificador o “scheduler” que acumulará los paquetes hasta que se cumplan las condiciones temporales establecidas en el script. Cuando este hecho se produzca, el grupo de paquetes será enviado al importador de datos (6.2.5) en forma de flujo de datos hexadecimal. El importador, una vez procesado el flujo de datos, se encargará de insertarlo en la BDR. Todo este proceso puede verse en la siguiente ilustración:

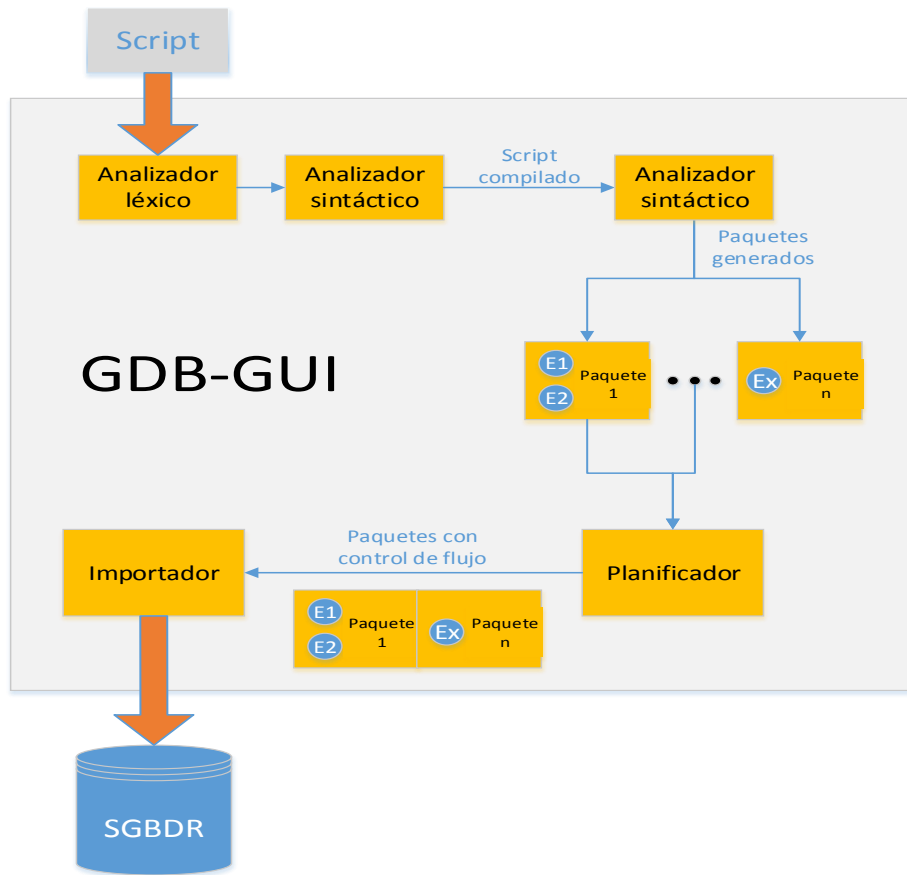


Ilustración 109. GDB-GUI. Proceso de ejecución de un script en el simulador de eventos.

Para mostrar toda la capacidad del lenguaje, se creará un script de ejemplo. El objetivo es generar 200 eventos GDS, agrupados en paquetes de tamaño máximo de 10 bytes, con un flujo de dos paquetes por segundo. El flujo y el tamaño de los paquetes se indican con las sentencias `MAX SECONDS WITHOUT SCIENCE DATA=2` y `MAX WORDS SCIENCE TM=0xA` respectivamente. La repetición de eventos se realiza con la sentencia `REPEAT` indicando las iteraciones requeridas entre paréntesis y encerrando entre llaves el bloque de sentencias a reproducir. Se permite el anidamiento de sentencias `REPEAT`. Se puede añadir la sentencia `DELAY` para provocar un retardo (en segundos) que se verá reflejado en las marcas temporales de los eventos y de los paquetes. Las líneas que comienzan con los caracteres `//` se considerarán comentarios y serán ignoradas.

A fin de compartir datos entre diferentes scripts, se incluye la sentencia `#include` seguido de un nombre de fichero. El procesador de scripts cuando encuentra este tipo de sentencia, la sustituye por el contenido del fichero y continúa con el procesamiento.

Reuniendo toda esta información, el script de ejemplo es el siguiente:

```
Fichero: <Example.txt>
//*****
//Example
//*****

MAX SECONDS WITHOUT SCIENCE DATA=2 //s
MAX WORDS SCIENCE TM=0xA //bytes

REPEAT (2)
{
REPEAT (100)
```

```

    {
        #include <GDSEvent/GDSEvent_1/GDS_Event.txt>
    }
}

//*****

```

Es en el fichero `GDSEvent/GDSEvent_1/GDS_Event.txt` donde se indicarán la estructura de paquete y los valores de los datos.

Fichero: **<GDSEvent/GDSEvent_1/GDS_Event.txt>**

```

//*****
//GDSEvent Event
//*****

#include <Packet.txt>
#include <SciHK.txt>

#include <GDS.txt>

//*****

```

Se muestran a continuación los ficheros `Packet.txt`, `SciHK.txt` y `GDS.txt`. La lista de campos que contiene puede consultarse en [3.12.3](#). Algunos valores de los campos admiten la constante `CALCULATED` lo que indica que será el propio GDB-GUI el que le asigne el valor. Esto es precisamente útil para las marcas temporales o el tamaño.

Fichero: **<Packet.txt>**

```

//*****
//PACKET HEADER
//*****

Version Number          =0
Type                    =0
Datafield Header Flag   =1
APID                    =1452

Segmentation Flags      =3
Source Sequence Count   =CALCULATED
Packet Length           =CALCULATED

Packet Time             =CALCULATED
PUS Version              =0
Checksum Flag           =0
Spare                   =0

Service                  =20
SubService               =3

PAD                      =0

//*****

```

Algunos valores de campos pueden agruparse y expresarse como una tabla, si bien, pueden asignarse uno a uno como se ha visto en los ficheros anteriores. El objetivo es facilitar al usuario la introducción de datos de forma tabular.

Fichero: **<ScienceHK.txt>**

```
//*****
//Science HK
//*****

HK Science Label=18507

#include <LASERHK_BY_TABLE.txt>

IS Temperature=4095

//*****
```

Fichero: <LASERHK_BY_TABLE.txt>

```
//*****
//Laser HK
//*****

LASERS TEMPERATURE = Table <Table_Laser_Temperature.txt>
LASERS LIGHT        = Table <Table_Laser_Light.txt>

//*****
```

Fichero: <Table_Laser_Temperature.txt>

```
//*****
//LT1      LT2      LT3      LT4
  1992      972      632      62
  123      452      456      632

//*****
```

Fichero: <Table_Laser_Light.txt>

```
//*****
//LL1      LL2      LL3      LL4
  4095      2497      2327      2160

//*****
```

Aunque el campo del tiempo es normalmente calculado, es posible asignarle un valor fijo, como se aprecia en el siguiente fichero:

Fichero: <GDS.txt>

```
//*****
//GDS EVENT
//*****

GDS Label      =LEFT
Event Time     =3433::00

#include <GDS_EVENT_BY_TABLE.txt>

//*****
```

Fichero: <GDS_EVENT_BY_TABLE.txt>

```
//*****
//GDS Event
//*****

GDS Event      = Table <Table_GDS_Event.txt>






//*****
```

```
Fichero: <Table_GDS_Event.txt>
//*****
//Scattered Light Crossing Time Overflow Crossing Time

4095          0          10

//*****
```

El control del simulador de eventos se realiza desde el área 7 en la *Ilustración 73*. Las acciones posibles son:

- a) **Cargar un fichero de script.** Pulsando el botón .
- b) **Salvar el fichero de script con el nombre actual.** Con el botón .
- c) **Salvar el fichero de script con un nombre nuevo.** Usando el botón .
- d) **Ejecutar el script.** Pulsando el botón .
- e) **Borrar el script.** Con el botón .

En e) el borrado del script sólo es visual, no se borrará el fichero.

Tras pulsar el botón de ejecución del script, el usuario debe de indicar la cadena de adquisición a utilizar en las curvas de calibración (*6.2.7*):

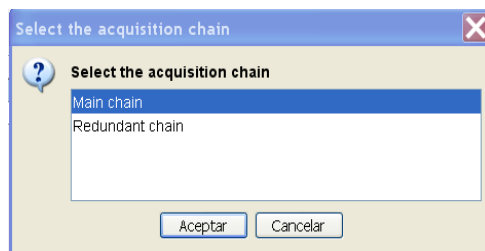


Ilustración 110. GDB-GUI. Selección de la cadena de adquisición antes de la ejecución de un script en el simulador de eventos.

Tras la ejecución aparece una ventana de resumen que informa acerca de los paquetes y eventos creados:

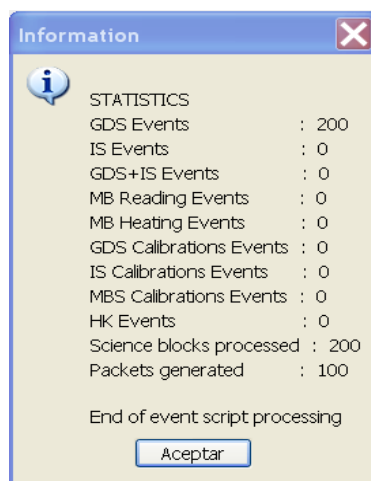


Ilustración 111. GDB-GUI. Resumen de los resultados de la ejecución de un script.

La principal misión del simulador de eventos es la de validar las reglas construidas, aunque tiene otras aplicaciones importantes. Dado un conjunto de datos reales, será posible construir un script que simule su generación. Con este script se obtendrá una idea clara de la estructura de la secuencia de adquisición de los eventos, que permitirá estudiar tanto el comportamiento del instrumento como el del cometa. Este análisis será especialmente valioso cuando se produzcan avalanchas de partículas cometarias, donde G. estará sometida a estrés. Otro campo de aplicación viene dado por su facilidad de introducir datos en el SGBDR de forma flexible y controlada, ya que permitirá generar casos de prueba para otros elementos de la cadena de datos de GIADA, no sólo GDB-GUI.

6.6 *Resumen y conclusiones*

Se ha presentado en este capítulo GDB-GUI, la interfaz de usuario para el modelo de base de datos GDB.

GDB-GUI permite construir, tanto sentencias clásicas en SQL, como reglas (acorde con el modelo GDB) con el objetivo de explotar los datos almacenados sobre partículas. Un usuario de GDB-GUI podrá realizar estas actividades de forma visual, sin conocer los lenguajes subyacentes: SQL y DFSQL, permitiendo exportar los resultados obtenidos a hojas de cálculo u otras bases de datos.

El usuario dispone de un conjunto de herramientas que le ayudarán a construir las consultas: información sobre esquema de la base de datos, significado de cada atributo, calculadora de curvas de calibración, conversión automática de valores de atributo a unidades físicas, enlace automático de tablas, trazabilidad de reglas, importador de datos y simulador de eventos.

Los cambios realizados en las consultas, y por tanto también al conocimiento que ellas reflejan, quedan almacenados de forma automática en el listado histórico de cambios que guarda GDB-GUI. Es decir, se mantiene una trazabilidad del conocimiento almacenado, lo que facilitará el análisis de su evolución, además de precisar qué conocimiento fue utilizado para procesar determinadas datos.

El importador de datos permite introducir en el SGBDR la información proveniente de partículas reales del cometa, la obtenida en la calibración del instrumento o la creada con el simulador de eventos. Un usuario de GDB-GUI podrá utilizar indistintamente estas tres fuentes de datos, con la misma interfaz y de forma centralizada en la misma base de datos.

La labor de verificación de las consultas creadas requiere del simulador de eventos. El simulador permite la generación de información sobre partículas de forma controlada a través de scripts de usuario, por lo que será posible calcular anticipadamente el resultado y compararlo con el devuelto por el interfaz. Este simulador será utilizado en la verificación del modelo de datos y del propio GDB-GUI, como se verá en el capítulo 7. Pero tiene más aplicaciones: imitar la generación de flujos de partículas reales (lo que permite analizar la respuesta del sistema de adquisición de GIADA) y producir conjuntos de datos diseñados para validar otras herramientas (distintas a GDB-GUI) dentro de la cadena de datos del instrumento.

La adaptación de GDB-GUI a otro sistema/instrumento se centra en redefinir cuatro ficheros de texto usados como entrada: el de curvas de calibración, el utilizado en la descripción de atributos, el que contiene la definición de las tablas (indicando los atributos difusos), y opcionalmente, el relacionado con el enlace automático de tablas.

7 Verificación y análisis de resultados

El objetivo de este capítulo es mostrar cómo, tanto el modelo de base de datos de GDB (5), cómo su interfaz de usuario GDB-GUI (6) son capaces de crear reglas basadas en conocimiento experto, aplicarlas a un conjunto de datos de entrada, proporcionar unos datos de salida calificados mediante una medida de calidad y cómo estas actividades se ajustan al resultado esperado.

A continuación, se detallan los pasos necesarios (como ya se hizo en 5.4 *Ejemplo de uso del modelo GDB*) para la creación, ejecución y análisis de resultados de una regla:

1. Determinación del conocimiento a representar,
2. Selección de datos difusos,
3. Definición de relaciones clásicas e intensivas
4. Creación de la regla,
5. Generación de datos sintéticos,
6. Semántica de la medida de calidad,
7. Cálculo de la regla,
8. Conclusiones de la aplicación de la regla

Los datos de entrada son sintéticos, es decir, creados ad hoc para comprobar la aplicación de la regla en un contexto controlado. El formato de estos datos se ajusta a los requerimientos de G. y para crearlos se ha utilizado el simulador de eventos de GDB-GUI (6.5).

Se crearán dos reglas centradas en los eventos GDS+IS. De todos los posibles eventos detectados por GIADA, el evento doble GDS+IS es el que aporta mayor conocimiento a nivel científico, esto es, es el que más ayuda a conocer la dinámica del polvo del cometa. Es por ello que se requiere un análisis más profundo a nivel de HW y de SW para detallar cómo se genera un evento GDS+IS. Será, precisamente este análisis, el que proporcione el conocimiento experto para crear ambas reglas.

La primera regla se dedica a la verificación de la coherencia de eventos GDS+IS utilizando las distintas formas de medir la velocidad de una partícula y la segunda se ocupa de calcular poblaciones de GDS e IS candidatas a crear eventos doble GDS+IS. En ambas reglas se utilizará la información sobre la detección de eventos individuales GDS, IS y la asociación de eventos GDS+IS. Este conocimiento, que fue inicialmente descrito en 2 y 2, será ampliado en este capítulo.

Una vez creadas ambas reglas, se aplicarán sobre los datos de calibración del modelo de vuelo de G. y posteriormente se analizarán los resultados obtenidos.

7.1 Detección de eventos GDS+IS

El esquema simplificado de la detección de un evento GDS+IS se muestra en la siguiente ilustración:

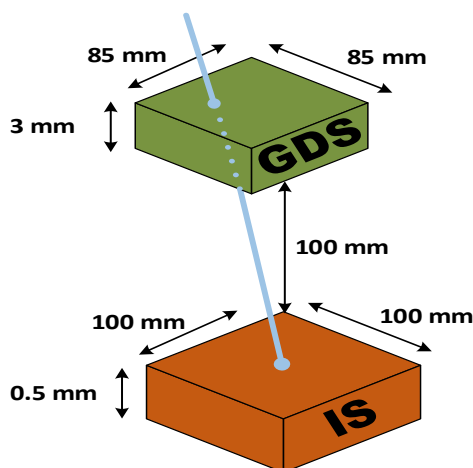


Ilustración 112. Detección de un evento de GDS+IS.

Cuando una partícula entra en el sistema de detección de G., ésta es detectada por la cortina láser del GDS, inicializando un contador de tiempo que será detenido al impactar dicha partícula en la membrana del IS. Dado que las dimensiones físicas del sistema de detección son conocidas, es posible medir la velocidad y el momento lineal de la partícula mediante los sensores de IS (*2.2.4 Prestaciones de GIADA*).

Este esquema de detección (*Ilustración 112*) brinda la posibilidad de calcular velocidad de una partícula de dos formas diferentes e independientes:

- a) **Mediante el tiempo en cortina.** Este es el tiempo durante el cual la partícula atraviesa la cortina láser (*2.2.3.1.1.1 GDS*). El contador HW que proporciona este tiempo está asociado al campo “crossingtime” del evento de GDS definido en el *Apéndice A*.
- b) **Mediante el tiempo de vuelo.** Existe un contador HW que se inicia al ser detectada la partícula en la cortina. Este contador se detiene al impactar en el IS y ser detectada por uno de sus PZT. Esto corresponde al tiempo de vuelo desde GDS a IS (*2.2.3.1.1.2 IS*). El campo del evento de IS asociado a este contador es el “flight-time” (*Apéndice A*).

Con estos dos tiempos y teniendo en cuenta la definición de los campos y las dimensiones del instrumento, es posible calcular los límites de tiempo de vuelo y velocidad, como se muestra en la siguiente tabla.

Tabla 39. Rango de medidas de tiempo y velocidad obtenidas partiendo del tiempo en cortina láser y del tiempo de vuelo GDS-IS.

	Cortina láser	Vuelo GDS-IS ⁽¹⁾
	“crossingtime”	“flighttime”
Mínimo tiempo medible (s) ⁽⁰⁾	$3 \times 10^{-5(4)}$	1×10^{-5}
Tiempo por cuenta	1×10^{-5}	1×10^{-5}
Máximo tiempo medible (s) ⁽⁰⁾	1022×10^{-5}	32766×10^{-5}
Máximo tiempo medible (cuentas) ^{(0) (2)}	1022	32766
Mínima velocidad (ms⁻¹)	$\frac{3 \times 10^{-3} \text{ m}}{1022 \times 10^{-5} \text{ s}} \approx 0.294$	$\frac{100 \times 10^{-3} \text{ m}}{32766 \times 10^{-5} \text{ s}} \approx 0.305$

Máxima velocidad (ms⁻¹)	$\frac{3 \times 10^{-3} \text{ m}}{3 \times 10^{-5} \text{ s}} = 100$	300 ⁽³⁾
---	---	--------------------

⁽⁰⁾ Se asume una incertidumbre de 9µs en la medida de tiempo debido a la resolución del sistema de medida. Por simplicidad, esta imprecisión no se tendrá en cuenta en los cálculos.

⁽¹⁾ Se asume un ángulo de incidencia de la partícula respecto al plano de detección del GDS de 90 grados.

⁽²⁾ Una cuenta por encima del valor máximo produce el desbordamiento del contador de tiempo (*Apéndice A*).

⁽³⁾ La velocidad máxima detectable medida en laboratorio es de 300ms⁻¹ (ver *2.2.4*).

⁽⁴⁾ Mínimo número de cuentas necesario para que una partícula sea detectada. Este valor puede establecerse entre 1 y 3, siendo el valor actual 3 cuentas. El valor se fija en el FC.

Ante una misma partícula, los valores de velocidad calculados a través del *crossingtime* y *flighttime* deben ser coherentes entre sí para garantizar la validez de la medida realizada. Esto será la base para crear la primera regla de verificación (*7.2.1*).

Otro aspecto a tener en cuenta al tratar eventos GDS+IS, es el procedimiento utilizado para la asociación de eventos. Esta asociación se realiza a través de la detección de eventos individuales e independientes de GDS y de IS. Sólo en determinadas situaciones, el SW de G. es capaz de identificar con éxito un evento GDS+IS, necesitando para el resto de ocasiones, un procesado en Tierra. La lógica que determina cómo emparejar los eventos GDS+IS, depende directamente de cómo se han detectado y procesado los eventos individuales.

A continuación se amplía la información proporcionada en *3.6* y *3.7* para detallar cómo el SW realiza la adquisición de eventos individuales GDS e IS mediante la *rutina de atención a la interrupción* y el *procesamiento de la cola de eventos*, para finalmente detallar el algoritmo de asociación de eventos GDS+IS.

7.1.1 Rutina de atención a la interrupción

Cuando el HW detecta un determinado evento, envía una interrupción (“IRQ” en inglés) al procesador a través del controlador de interrupciones, con el fin de procesar la información recogida en determinados registros de HW. Esta petición “interrumpe” el funcionamiento normal del programa que se esté ejecutando y pasa a ser atendida en la rutina de atención de interrupción (“ISR” en inglés). Cuando ésta finaliza, se vuelve al mismo punto donde se dejó el programa interrumpido.

Podemos definir ‘programa’ (también llamado rutina o código) como una secuencia de operaciones en el espacio de direcciones del procesador de un computador. No todo el código puede ser interrumpido, y existe además prioridad sobre qué rutinas pueden interrumpir a otras. En el procesador utilizado en G., sólo las rutinas con mayor valor de prioridad pueden interrumpir a rutinas de menor prioridad (*Tabla 7. Tabla de interrupciones y prioridades en GIADA*). Normalmente se identifica el número de interrupción con su prioridad.

Existen dos fenómenos que afectan a las IRQ: *anidamiento* y *avalanchas*

- a) **Anidamiento de IRQ.** Si se está ejecutando una ISR_x y se produce una nueva IRQ_z con z > x, la ISR_x se detendrá hasta que finalice ISR_z.

- b) **Avalancha de IRQ.** Si de forma permanente se produce una IRQ_x , las interrupciones menores que x no serán atendidas, produciéndose una generación constante de ISR_x .

Los anidamientos y avalanchas afectarán a la asociación de eventos GDS+IS, como veremos más adelante.

En ambos casos **a)** y **b)**, el SW de G. incorpora un mecanismo que deshabilita la IRQ después de “n” atenciones consecutivas de la misma ISR en un intervalo de tiempo de un segundo. La IRQ será rehabilitada después de un segundo. Se ha medido de forma experimental que G. puede procesar de forma sostenida hasta 40 eventos (20 GDS más de 20 IS) por segundo (2.2.4 Prestaciones de GIADA).

En el caso que nos atañe, la IRQ de GDS tiene mayor prioridad que la de IS, por lo que, aunque los eventos sean detectados a la vez, se procesará primero la interrupción de GDS. Esto es aplicable incluso, si se detecta un evento de GDS mientras se procesa la ISR de IS.

Veamos en detalle el funcionamiento de las ISR de GDS e IS. Las rutinas están escritas originalmente en lenguaje ensamblador, se ha hecho una traducción a un diagrama de flujo en inglés para facilitar su lectura a usuarios de GDB-GUI. En el diagrama de flujo se ha implementado el siguiente código de colores y formas:

- a) Líneas azules indican el siguiente paso dentro del diagrama.
- b) Líneas verdes indican la salida “sí” ante una condición.
- c) Líneas rojas indican la salida “no” ante una condición.
- d) El punto verde indica la entrada a la ISR o rutina y el rojo su salida.

7.1.1.1 ISR para el GDS

Los campos obtenidos en la ISR para un evento de GDS son los siguientes: “elabel”, “eventtime”, “scatteredlight”, “crossingtimeoverflow”, “crossingtime” y “flighttime”. El significado de cada campo puede encontrarse en la definición de la tabla “GDB_GDSEvent” en Apéndice A. El valor del “flighttime” es añadido a la información del evento por si es necesario construir un evento GDS+IS en el procesamiento de la cola de eventos. Cada unidad de contador de “crossingtime” equivale a 10µs de tiempo.

La rutina de GDS (3.13 Prestaciones del software de GIADA) tiene una duración media de 1,62 ms para un evento individual de GDS (detectado por el canal izquierdo o derecho. Ver 2.2.3.1.1) y 2,92ms para un evento doble (izquierdo y derecho).

De manera resumida, la ISR actúa como sigue. Si no se ha detectado nada en el canal izquierdo, se pasa al canal derecho. Se analiza el canal actual durante al menos 11ms. Si se observa como una partícula abandona la cortina laser o se desborda el contador de tiempo en cortina o se sobrepasan estos 11ms, entonces se adquieren los datos del evento, se introducen en la cola de eventos y se pasa analizar el siguiente canal (si el canal actual es el derecho finaliza la ISR).

De forma detallada, la implementación de la ISR de GDS se muestra en la siguiente ilustración:

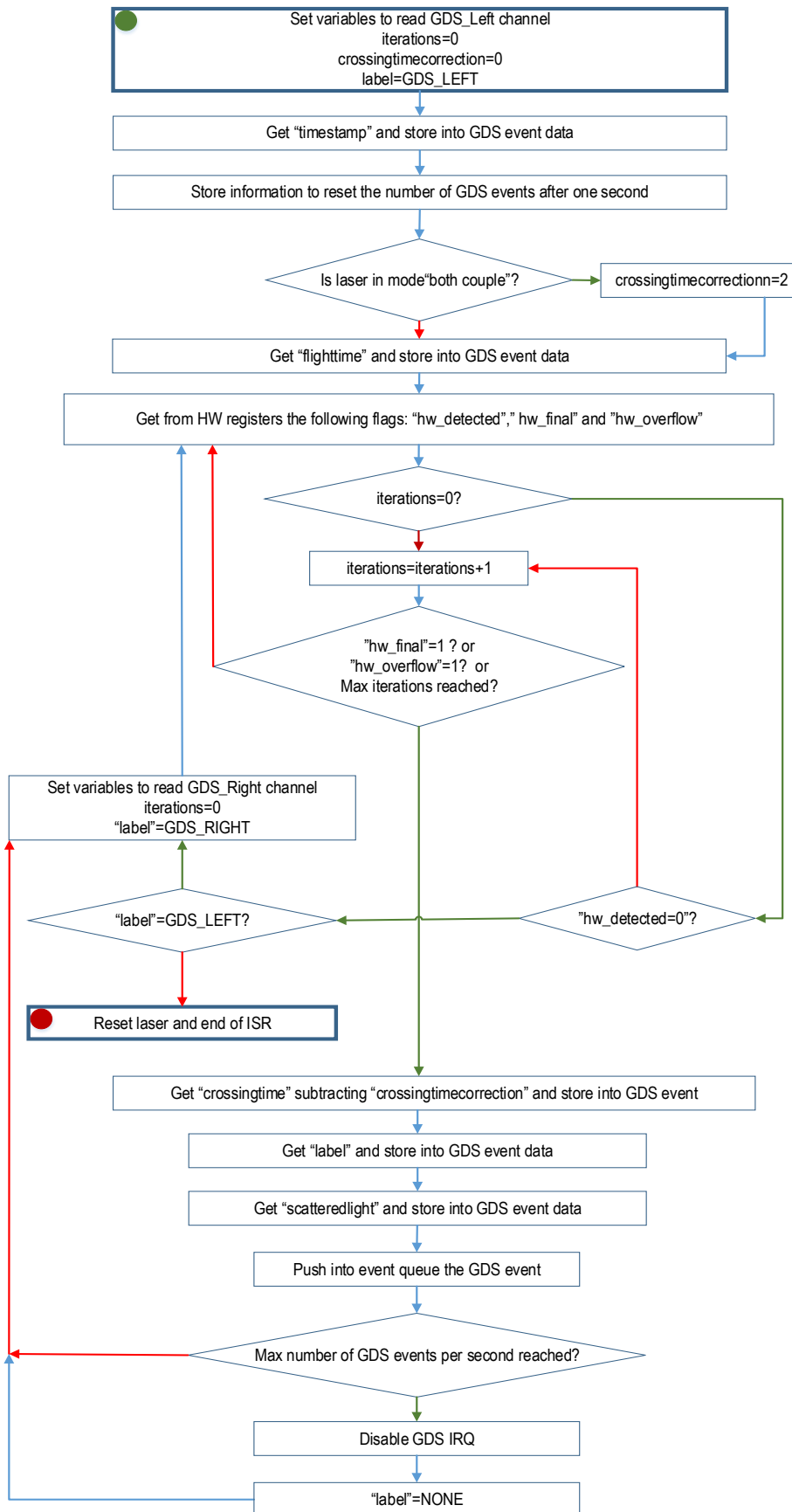


Ilustración 113. Diagrama de flujo de la ISR de GDS.

7.1.1.2 ISR para el IS

La ISR de IS obtiene los siguientes campos para un evento (donde *_X* es uno de los 5 PZT): “elabel”, “eventtime”, “PZT_X_detected”, “PZT_X_amplitude”, “PZT_X_timeoverflow” y “PZT_X_time”. Los valores de “PZT_X_range”, “PZT_X_gain” se asignarán en el procesamiento del evento en la cola de eventos. El significado de cada campo puede encontrarse en la definición de la tabla “GDB_ISEvent” en el *Apéndice A*.

La rutina de IS (*3.13 Prestaciones del software de GIADA*) tiene una duración media de 6,6ms para un evento sin corrección de autogancia y 7.4ms para un evento con corrección.

Resumiendo, la ISR realiza la siguiente actividades: se obtienen los datos del evento para cada PZT (habilitado o no) y se introducen en la cola de eventos. La activación de la autogancia se realiza si el PZT_A está habilitado y su amplitud saturada. En tal caso se vuelve a medir su amplitud, variando la ganancia antes de pasar los datos a la cola de eventos.

De forma detallada, la implementación de la ISR de IS se muestra en la siguiente ilustración:

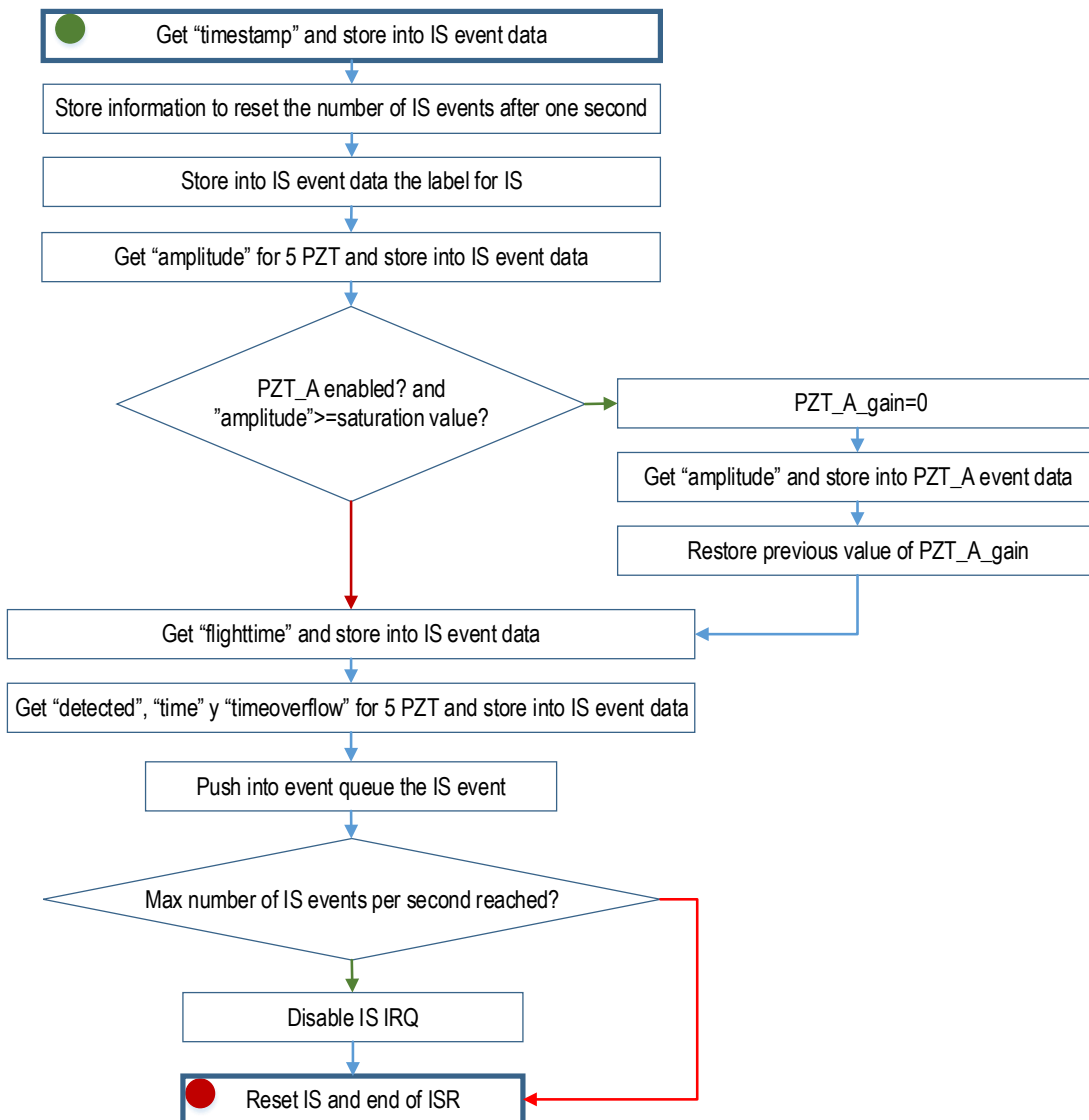


Ilustración 114. Diagrama de flujo de la ISR de IS.

7.1.2 Procesamiento de la cola de eventos

Una vez capturados los eventos, se introducen en una cola de procesamiento que es atendida en la línea de ejecución principal del SW de G. Es aquí donde se realiza la asociación de eventos individuales GDS e IS para generar un evento doble GDS+IS.

Para comunicar las rutinas de procesamiento de GDS e IS, se utiliza una zona de memoria llamada "buffer de GDS+IS". Es en este buffer donde el procesamiento del GDS almacena los datos del último evento adquirido y donde la rutina de procesado del IS busca para intentar crear un evento doble GDS+IS.

Es importante recordar que los eventos de G. se acumulan en un bloque de telemetrías de ciencia ([3.12.2](#)). Este bloque se envía al SC si sobrepasa un determinado tamaño o un determinado valor de tiempo (60s por defecto) sin añadir nuevos eventos. Realizado el envío, se borra el buffer GDS+IS. Para analizar periódicamente el estado del bloque (y enviarlo si se cumplen las condiciones) se crea una tarea en el SW de G.

Se pasa a detallar cada uno de los dos procesamientos de eventos relevantes: GDS e IS. Las rutinas originales de procesamiento se diseñaron en el lenguaje de programación C. Al igual que al tratar las rutinas de ISR, se ha realizado una traducción al inglés del diagrama de flujo, manteniendo el código de colores y formas previamente indicado.

7.1.2.1 Procesamiento del evento GDS

Esta rutina es llamada cuando al inicio de la cola de eventos se encuentra un evento de GDS (izquierdo o derecho).

El tiempo medio desde la interrupción hasta su procesamiento es de 19.62 ms ([3.13 Prestaciones del software de GIADA](#)), pero este lapso de tiempo no afecta al procesamiento, ya que los datos ya han sido adquiridos y almacenados.

De forma resumida, esta rutina recoge los valores almacenados por la ISR y los deshecha si el canal por el que fueron obtenidos está deshabilitado en el CF o si existe algún error en la adquisición. Si el buffer para el GDS+IS no está vacío, se añade el contenido del buffer al bloque de ciencia como un GDS individual. En otro caso, se crea una tarea de volcado para realizar esta misma operación: añadir al bloque de ciencia el buffer GDS+IS para después borrar dicho buffer. Antes de finalizar la rutina, se añade el evento al buffer GDS+IS. El buffer de GDS+IS sólo almacena un único evento de GDS: el último detectado.

La descripción más detallada de este proceso se realiza en la siguiente ilustración.

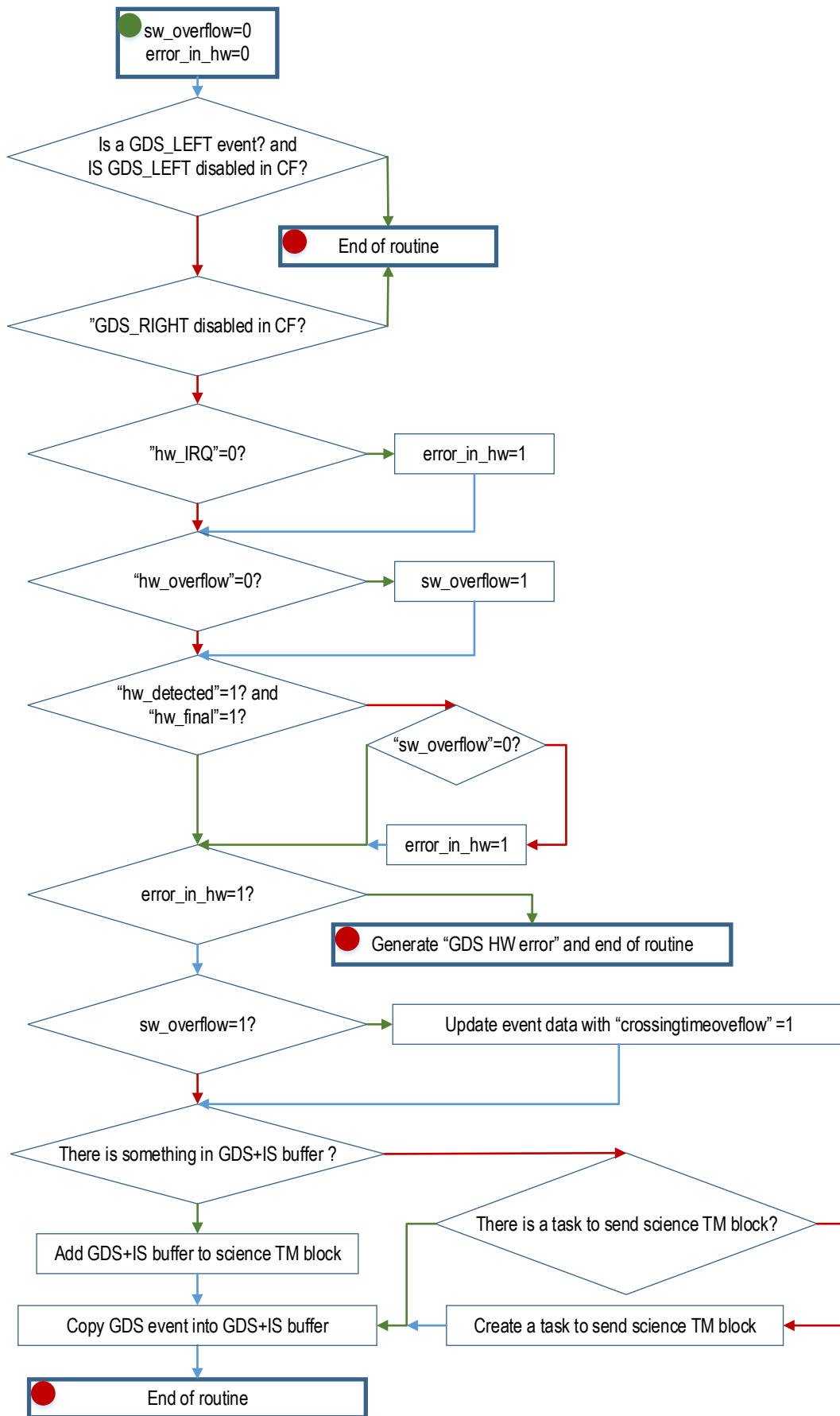


Ilustración 115. Diagrama de flujo del procesamiento del evento de GDS.

Prestando mayor atención a la tarea de volcado (que recibe el nombre de “ScienceReportTimeout”), ésta se crea en la rutina de procesamiento de GDS y se ejecuta tras 60s (dependiendo de un parámetro del FC). Esta temporización es necesaria para mantener el requerimiento de bloque de ciencia, según el cual, debe ser enviado si no existen más eventos tras 60s.

En la ejecución de “ScienceReportTimeout” se analiza el contenido del buffer GDS+IS. Si no está vacío, añade su contenido al final del bloque de ciencia y se borra el buffer GDS+IS. En cualquier caso, se envía el bloque de ciencia al SC.

7.1.2.2 Procesamiento del evento IS

La rutina de procesamiento de IS es ejecutada cuando al inicio de la cola de eventos se encuentra un evento de tipo IS.

El tiempo medio desde la interrupción hasta su procesamiento es de 19.94 ms (3.13).

A modo de resumen, en esta rutina se obtienen los valores asociados a cada PZT y los almacena en los datos del evento IS. El evento es descartado si se detecta algún error en la cadena de adquisición. Si el tiempo de vuelo desde el GDS (“flighttime”) es cero, se añade un evento individual de IS al bloque de telemetrías. En otro caso, se busca en el buffer para construir un evento doble GDS+IS que será añadido al bloque de telemetrías.

La descripción más detallada de este proceso se realiza en la siguiente ilustración.

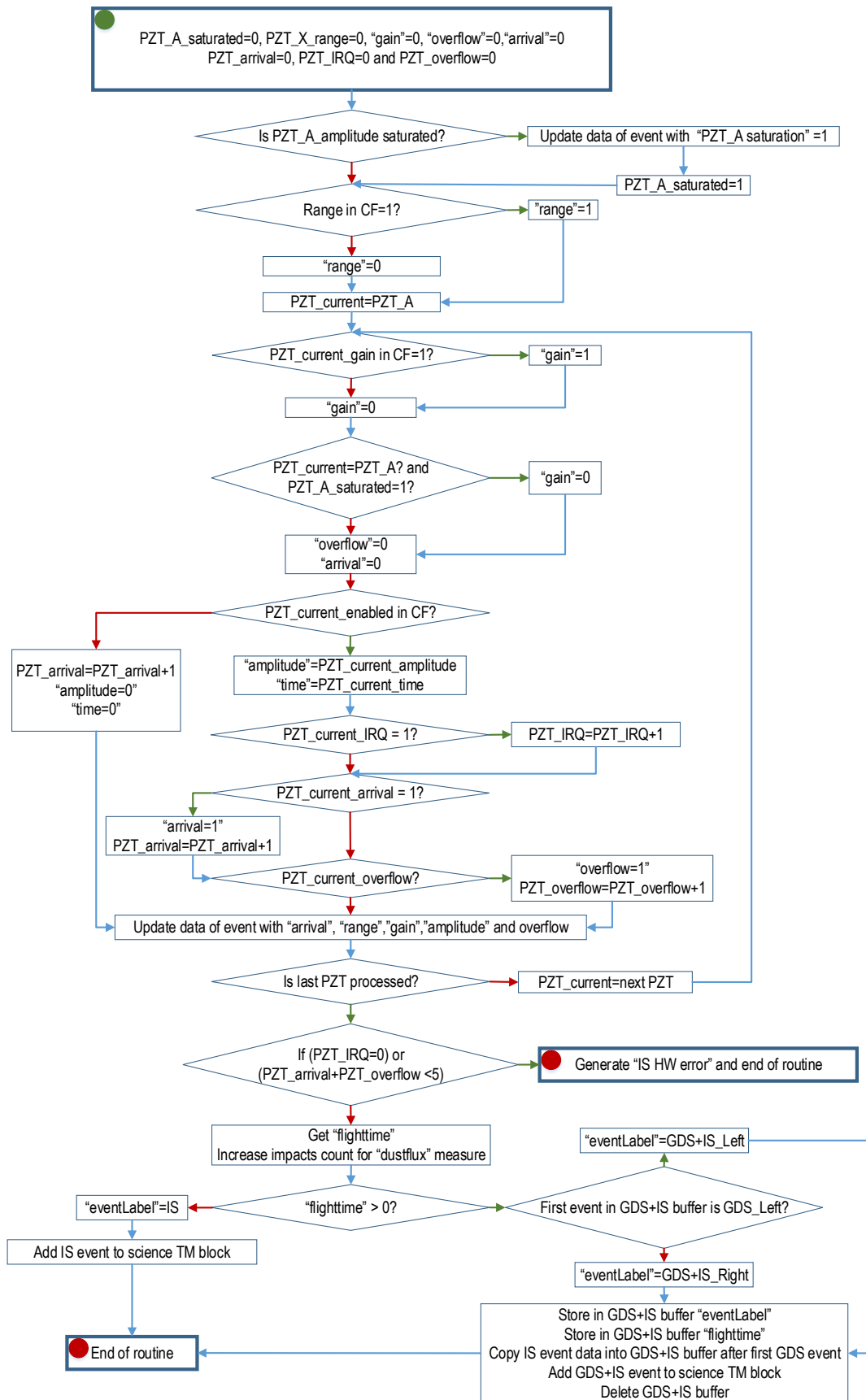


Ilustración 116. Diagrama de flujo del procesamiento del evento de IS.

7.1.3 Algoritmo de asociación de eventos GDS+IS

Después de analizar las rutinas de ISR y procesamiento de GDS e IS, se resume en la siguiente ilustración cómo el SW de G. las interrelaciona para generar un evento GDS+IS.

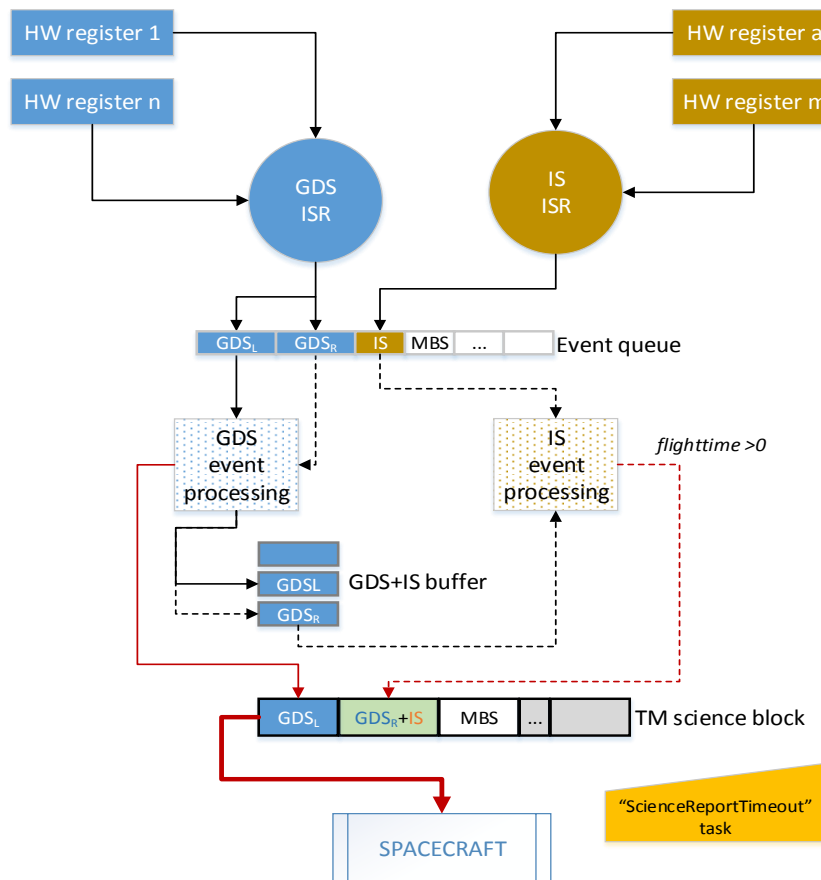


Ilustración 117. Procesamiento de un evento GDS+IS.

Una ISR (GDS o IS) es ejecutada cuando es detectada por el HW. La información acerca de la detección se almacena en registros HW independientes. Dentro de la ISR, se recaban los datos de los registros HW, más algunos adicionales, y se crea un evento (GDS o IS) que se añade a una cola de eventos para su posterior procesamiento. En la ISR de GDS, el canal izquierdo es prioritario respecto al derecho, es decir, se detectará y almacenará primero.

La rutina de procesamiento de eventos GDS añade al bloque de ciencia el propio evento GDS si el buffer GDS+IS está vacío. Si está lleno (el buffer sólo almacena un único evento) se crea una tarea, *ScienceReportTimeout*, que se ejecuta a los 60s para añadir al bloque de ciencia el contenido del buffer GDS+IS. En cualquier caso, la rutina copia el evento detectado en el buffer GDS+IS.

La rutina de procesamiento de IS creará un evento GDS+IS con el contenido del buffer si el *flighttime* es mayor que cero. En otro caso creará un evento IS individual. En otras palabras, la asociación de GDS+IS se realiza entre el último evento de GDS detectado y el primer IS procesado.

En una situación normal, una partícula será detectada por los canales derecho e izquierdo del GSD y por el IS. Se producirá un evento individual GDS izquierdo y el GDS derecho se unirá con el IS para construir un evento doble.

7.1.3.1 Análisis de la asociación de eventos GDS+IS

En la siguiente ilustración se muestran las relaciones entre ISR, procesamiento de eventos y tareas de volcado relativos a la detección de partículas en el GDS.

Las flechas azules indican posibles caminos de ejecución entre diferentes procesos que se ejecutarán de forma secuencial en el bucle principal del programa. Las flechas rojas implican llamadas a la ISR. El aspa (X) indica que el contenido no es relevante y un recuadro vacío indica que no existe información.

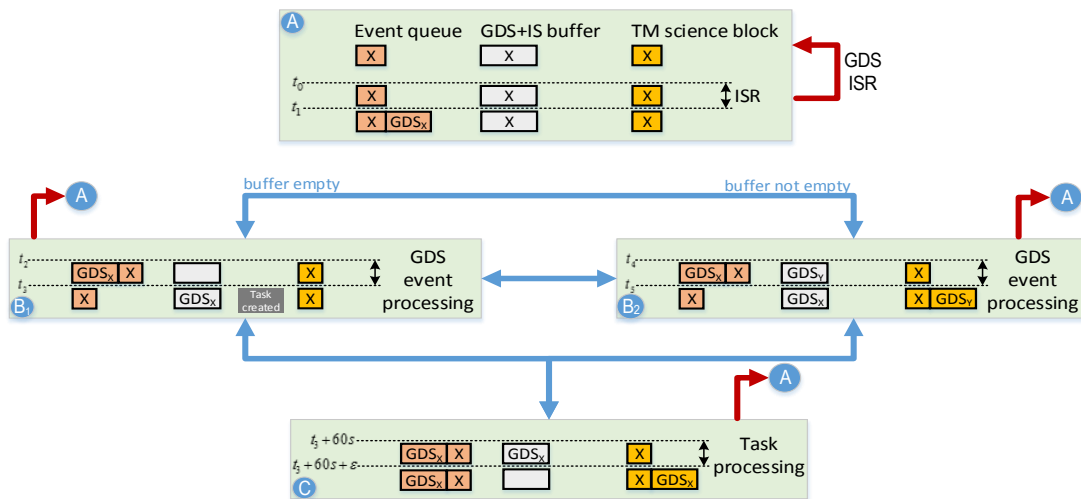


Ilustración 118. Relación entre ISR, procesamiento de eventos y tarea en GDS.

El recuadro A indica el procesamiento de una ISR de GDS y su efecto en la cola de eventos. La flecha roja en el recuadro A indica una avalancha o un anidamiento de interrupciones en el GDS (antes de que la IRQ sea bloqueada por el SW) que se traducen en una escritura secuencial de datos de eventos en la cola de procesamiento. Esta actividad se produce incluso mientras se realiza el procesamiento en la cola o en la tarea. Las dos acciones posibles de la cola ante un evento GDS (recuadros B₁ y B₂ en la ilustración), como ya se indicó, depende del estado del buffer. Si está vacío se crea la tarea de volcado, en otro caso se añade el evento del GDS al bloque de telemetrías.

La tarea encargada del volcado del buffer se refleja en el recuadro C.

Análogamente, la relación entre ISR y procesamiento de IS se detalla a continuación:

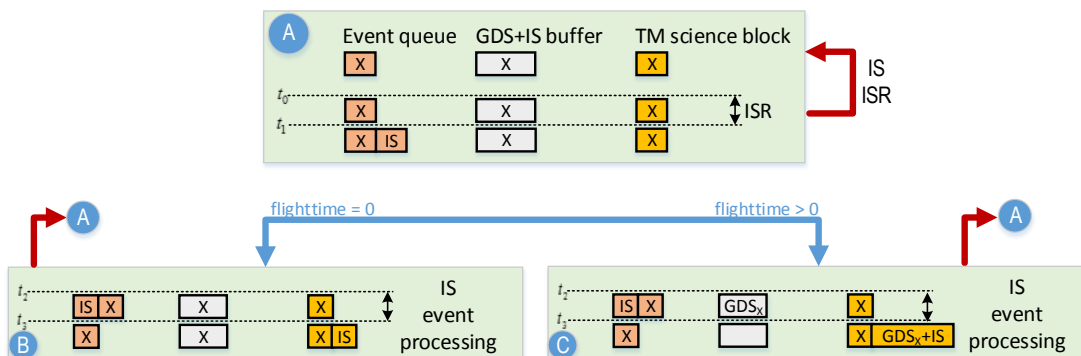


Ilustración 119. Relación entre ISR, procesamiento de eventos y tarea en IS.

En el recuadro A se encuentran las actividades realizadas en la ISR de IS. Una avalancha o anidamiento de IS, conduce a una escritura secuencial de datos del evento. Cualquier ISR de IS quedará suspendida hasta que terminen todas las interrupciones de GDS. Las dos posibles actividades en el procesamiento de eventos de IS, quedan reflejadas en los recuadros B y C: si el *flighttime* es cero se crea un evento IS, en otro caso un evento GDS+IS.

El sistema de asociación de eventos GDS+IS implementado por el SW de G. funciona correctamente cuando una partícula genera un evento GDS y después un evento IS, que es el comportamiento esperado. Pero dadas las limitaciones y las componentes de ruido propias del sistema de detección, es posible que el evento GDS o IS no sea detectado (2.2.3.1.1) o bien que existan chorros de partículas expulsadas a gran velocidad (“jets”). Es por ello que hay que profundizar en el algoritmo de asignación de eventos GDS+IS teniendo en cuenta los diferentes casos posibles de llegada de interrupciones a GDS e IS.

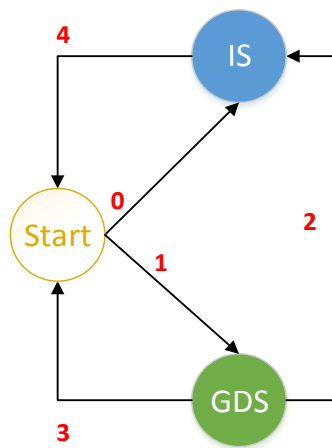


Ilustración 120. Proceso de asignación de eventos GDS+IS.

El nodo “start” refleja la situación donde no es posible generar un evento GDS+IS. En este estado, puede llegar un evento IS (arista 0) o GDS (arista 1). Los nodos “GDS” e “IS” indican que se ha detectado un evento “GDS” e “IS”, respectivamente. Una arista indica la detección de un evento del tipo GDS (aristas 1 y 3) o IS (aristas 0,2 y 4). Sólo la arista 2 producirá un evento GDS+IS uniendo el último evento GDS detectado con el IS situado en la cola de eventos (suponiendo un *flighttime* distinto de cero). El resto de aristas generan eventos individuales. Las aristas (0,4) y (1,3) representan una avalancha o bien un anidamiento de interrupciones IS o de GDS respectivamente. Los eventos individuales serán procesados en la tarea o directamente en la rutina de procesamiento.

En caso de que G. encuentre un jet, se producirá una avalancha y/o anidamiento de interrupciones de GDS e IS. Esto se traduce en un flujo (detección continuada de eventos del mismo tipo) de eventos GDS e IS. Ya se indicó la prioridad del GDS sobre el IS, por lo que se podrán recibir flujos de GDS en medio del procesamiento de un flujo de IS. En esta situación se sigue aplicando el proceso de asignación de eventos descrito en la *Ilustración 120*, pero con un problema larvado. Como ya se indicó, la asociación GDS+IS realizada por el SW utilizará el último evento de GDS detectado en el flujo y el primer IS procesado. En el caso de un jet, los eventos GDS previos al utilizado en la asociación serán procesados como eventos individuales. Los eventos de IS sucesivos al que compone el GDS+IS serán considerados dependiendo de una de las siguientes situaciones:

- a) No existen más eventos GDS. En este caso los eventos del flujo de IS serán tratados como eventos individuales.
- b) Existe un flujo nuevo de eventos GDS. Estos eventos habrán sido detectados después de realizar la asociación GDS+IS y antes de procesar el siguiente IS. Esto lleva a crear un nuevo evento GDS+IS con el ultimo evento GDS detectado, generando

eventos GDS simples con los anteriores, lo que reinicia el problema de la asociación de eventos.

Por tanto, la presencia de un jet generará muy pocos eventos GDS+IS y múltiples eventos individuales, pero que son claramente candidatos a crear eventos dobles. Sólo mediante un análisis en Tierra, basándose en las marcas temporales de los eventos individuales, se podrá conocer la población de eventos GDS e IS candidatos a crear un evento GDS+IS. Esto será el fundamento para la segunda regla de verificación.

7.2 Reglas de verificación de eventos GDS+IS

Se detallan a continuación como se realiza la definición y el cálculo de dos reglas centradas en la gestión de eventos GDS+IS. La primera regla se encargará de verificar la coherencia de la velocidad de eventos GDS+IS previamente detectados, y la segunda buscará poblaciones de eventos individuales de GDS e IS candidatos a formar un evento doble GDS+IS.

7.2.1 Regla para el cálculo de la coherencia de eventos GDS+IS mediante velocidades

El objetivo es crear una regla que establezca la coherencia de eventos GDS+IS a partir de la velocidad de la partícula, como se analizó en [7.1](#). Esto es: “la velocidad de una partícula es determinada mediante dos sistemas independientes de medición de tiempo: *crossingtime* y *flighttime*. Ambos sistemas deben de medir la misma velocidad, cuando ésta se encuentre dentro del rango dinámico de velocidades válido para ambos sistemas”.

Según lo indicado en la [Tabla 39](#), el rango de velocidades medido con el *crossingtime* es de: 0.294 ms^{-1} a 100 ms^{-1} y para el *flighttime* de 0.305 ms^{-1} hasta 300 ms^{-1} . El rango dinámico común de medida queda fijado por el rango más pequeño, en este caso el del *crossingtime*: $[0,100] \text{ ms}^{-1}$.

Partículas con velocidades superiores a 100 ms^{-1} no serán detectadas por el GDS y por lo tanto no generarán eventos GDS+IS. Este límite de 100 ms^{-1} se puede modificar actualizado el número de pulsos mínimo necesario para detectar una partícula en el GDS ([2.2.3.1.1.1](#)). El número de pulsos es por defecto 3 y se establece en el FC.

Por simplicidad se ha supuesto que el ángulo de incidencia de la partícula respecto al plano de detección el GDS es de 90 grados.

A continuación se seguirán los pasos descritos en [5.4 Ejemplo de uso del modelo GDB](#) para crear la regla que representa esta pieza de conocimiento.

7.2.1.1 Selección del tipo de dato difuso

La velocidad de 1 ms^{-1} corresponde a 300 cuentas en el contador de *crossingtime*, o lo que es lo mismo aproximadamente el 30% del rango de medida en cuentas. Se asume que 1 ms^{-1} es la diferencia de velocidad a partir de la cual se puede considerar que las dos velocidades en cuestión son demasiado distintas para ser comparadas y que serán menos “iguales” conforme se alejen entre sí.

El tipo de dato difuso que mejor encaja para representar esta comparación de velocidades es de *valores aproximados* (ver [5.4.1](#)). En la definición del tipo de dato se ha de incluir un ‘margen’=2 (un entorno de 1 ms^{-1} relativo al valor de velocidad calculada con el *crossingtime*) y ‘much’=3.

Por ejemplo, una velocidad de “aproximadamente 95 ms⁻¹” quedaría representada como sigue:

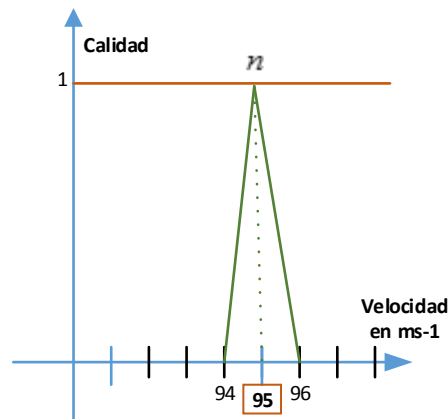


Ilustración 121. Representación del tipo difuso velocidad aproximadamente 95 ms⁻¹.

7.2.1.2 Definición de relaciones clásicas

Las dos velocidades a tratar (*crossingtime* y *flighttime*) las podemos encontrar como atributos de la tabla `GDB_GDS_ISEvent`. Además, necesitaremos el atributo `IS_ITEM` para identificar cada evento GDS+IS de forma unívoca. La semántica detallada de estos campos, puede encontrarse en el [Apéndice A](#).

Es necesario adaptar la definición de la tabla para que acoja al tipo de dato difuso que se requiere:

Definición de la tabla `GDB_GDS_ISEvent`:

```
CREATE TABLE GDB_GDSEvent (
...
GDS_ITEM NUMBER(2) NOT NULL Constraint "GDS_IS_ITEM"
CHECK(GDS_IS_ITEM >=0),
...
GDS_IS_crossingTime FTYPE1(2,3) NUMBER(5) NOT NULL
"GDS_IS_crossingTime" CHECK(GDS_IS_crossingTime>=0 and
GDS_IS_crossingTime<=1023)
...
GDS_IS_flightTime FTYPE1(2,3) NUMBER(5) NOT NULL Constraint
"GDS_IS_flightTime" CHECK(GDS_IS_flightTime>=0 and
GDS_IS_flightTime<=32767) , ...)
```

Se ha utilizado la notación de *difusos tipo 1* para representar el tipo de dato de valores de *valores aproximados*, de la misma forma que se aplicó en el ejemplo de regla (5.4.2).

Se aprovechará la definición de los atributos `GDS_IS_crossingTime` y `GDS_IS_flightTime` (que originalmente almacenaban cuentas) para almacenar los valores de velocidad. Esto implica que será necesario realizar una conversión del valor en cuentas a velocidad antes de insertar los valores en las tablas. La parte fraccionaria de la conversión será ignorada ya que ambos atributos son manejados como enteros.

7.2.1.3 Definición de la tabla intensiva y regla

Ya seleccionados los atributos de los que se obtendrá la información requerida, los relacionaremos con el operador de la igualdad difusa (FEQ). Este operador nos proporcionará la medida de calidad según la cual las dos magnitudes de velocidad están alejadas entre sí.

La regla expresada en DFSQL queda como sigue:

```
//===== Intensional table =====
create intensional table GDB_INT_COHERENCE_BY_TIME (
  GDS_IS_ITEM NUMBER(2),
  GDS_IS_CROSSINGTIME NUMBER(5),
  GDS_IS_FLIGHTTIME NUMBER(5),
  GDS_IS_CROSSINGTIMEOVERFLOW NUMBER(1),
  GDS_IS_FLIGHTTIMEOVERFLOW NUMBER(1));
//===== End of intensional table =====
//===== Rule =====
create rule for GDB_INT_COHERENCE_BY_TIME (X,Y,Z,W,P)
as
  GDB_GDS_ISEVENT(X source gds_is_item, Y source
gds_is_crossingtime, Z source gds_is_flighttime, W source
gds_is_crossingtimeoverflow, P source
gds_is_flighttimeoverflow) AND
  GDB_CTE_0_0(Q source value, R source value) AND
  GDB_CTE_100(S source value) AND
  (Y FEQ Z) AND
  (W = Q) AND
  (P = Q) AND
  (Z <= S);
//===== End of rule =====
```

Se han eliminado los eventos con desbordamiento (*overflow*). Para ello se ha de incluido un predicado en el que los atributos `gds_is_crossingtimeoverflow` y `gds_is_flighttimeoverflow` sean cero. La implementación de este predicado requiere del predicado constante cero `GDB_CTE_0_0` ([5.1.2.6 Implementación de constantes](#)).

Se ha incluido un predicado adicional para restringir los valores del *flighttime* al rango común de medida. Para ello se ha utilizado el predicado constante `GDB_CTE_100`. Este valor 100 es la velocidad en ms^{-1} del límite de velocidad máximo permitido para el *flighttime*.

7.2.1.4 Semántica de la medida de calidad

La coherencia entre las dos velocidades nos indica que, en el rango de medida común, la calidad de la medida aumentará cuanto más iguales lo sean los valores que tomen. Dos valores iguales proporcionan la máxima coherencia (calidad 1), valores separados más de 1ms^{-1} indican una incoherencia, es decir, calidad 0. Este valor de calidad cero significa que la diferencia entre ambas velocidades es mayor que el 30% del rango de medida común, y por tanto el evento GDS+IS debería ser ignorado.

Será el operador FEQ el que evalúe la coherencia mediante la intersección de los conjuntos difusos asociados a cada valor de velocidad.

Los posibles valores de calidad (área rayada) respecto a la distancia relativa entre valores de velocidad (*crossingtime* en color verde y *flighttime* en color marrón), se muestra en la siguiente ilustración:

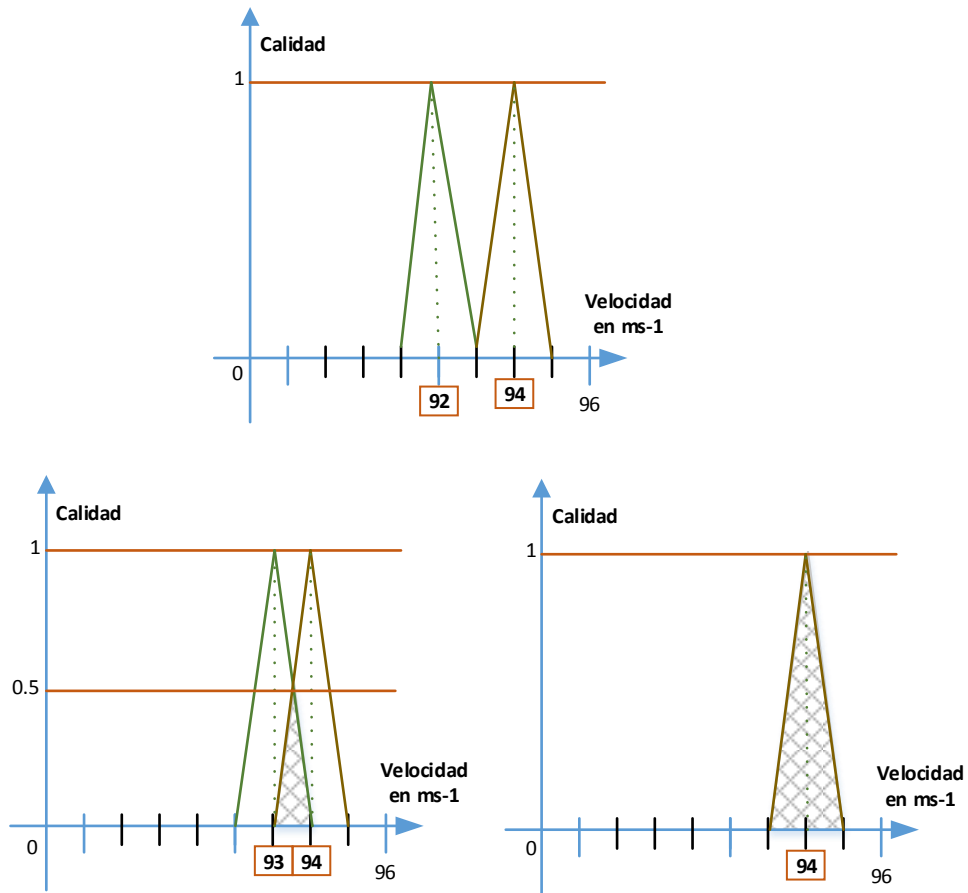


Ilustración 122. Cálculo de calidad para distintas diferencias de velocidad.

7.2.1.5 Datos sintéticos

La generación de datos sintéticos se centra en crear eventos GDS+IS donde el valor de *flighttime* es fijo y los valores de *crossingtime* se distribuyen entorno a él, con el fin de obtener una calidad no nula (ver 7.2.1.4). Esto es, los valores de *crossingtime* se centran en un entorno de $\pm 1 \text{ ms}^{-1}$ respecto a un *flighttime* fijo.

En la siguiente tabla se muestran los valores de velocidad en cuentas y sus correspondientes velocidades.

Tabla 40. Tabla GDB_GDSISEvent con datos sintéticos para la verificación de la regla de coherencia.

GDB_GDS_ISEVENT		
GDS_ITEM	GDS_IS_CROSSINGTIME cuentas/ms-1	GDS_IS_FLIGHTTIME cuentas/ms-1
0	25/12.00	1000/10
1	26/11.54	1000/10
2	27/11.11	1000/10

3	28/10.71	1000/10
4	29/10.34	1000/10
5	30/10.00	1000/10
6	31/9.68	1000/10
7	32/9.38	1000/10
8	33/9.09	1000/10
9	34/8.82	1000/10
10	35/8.57	1000/10

7.2.1.6 Cálculo de la regla

Asumiendo una calidad asignada por el usuario de al menos 0.5, la ejecución de la regla creada sobre los datos sintéticos, arroja los siguientes 8 tuplas.

Tabla 41. Resultado de la aplicación de la regla de coherencia con calidad mínima 0.5.

GDB_INT_COHERENCE_BY_TIME			
GDS_IS_ITEM	GDS_IS_CROSSINGTIME (ms-1)	GDS_IS_FLIGHTTIME (ms-1)	Calidad
1	11	10	0.5
2	11	10	0.5
3	10	10	1
4	10	10	1
5	10	10	1
6	9	10	0.5
7	9	10	0.5
8	9	10	0.5

Por claridad, se han omitido los atributos de desbordamiento de *crossingtime* y *flighttime* que se definieron en la tabla intensiva (7.2.1.3). Si alguno de los valores de estos dos atributos es igual a uno, implicaría que la tupla correspondiente no aparecería en el resultado.

Puede apreciarse cómo, respecto a la calidad máxima en los valores la tupla con ítems 3,4 y 5, la calidad disminuye progresivamente conforme los valores del *crossingtime* se alejan. Los eventos con ítems 0, 9 y 10 han sido descartados ya que se encontraban demasiado alejados del valor de referencia de *flighttime*.

Las medidas de calidad obtenidas, coinciden con las calculadas teóricamente en la *Ilustración 122*, considerando las diferencias entre velocidades.

La ejecución de la regla en GDB-GUI se muestra a continuación:

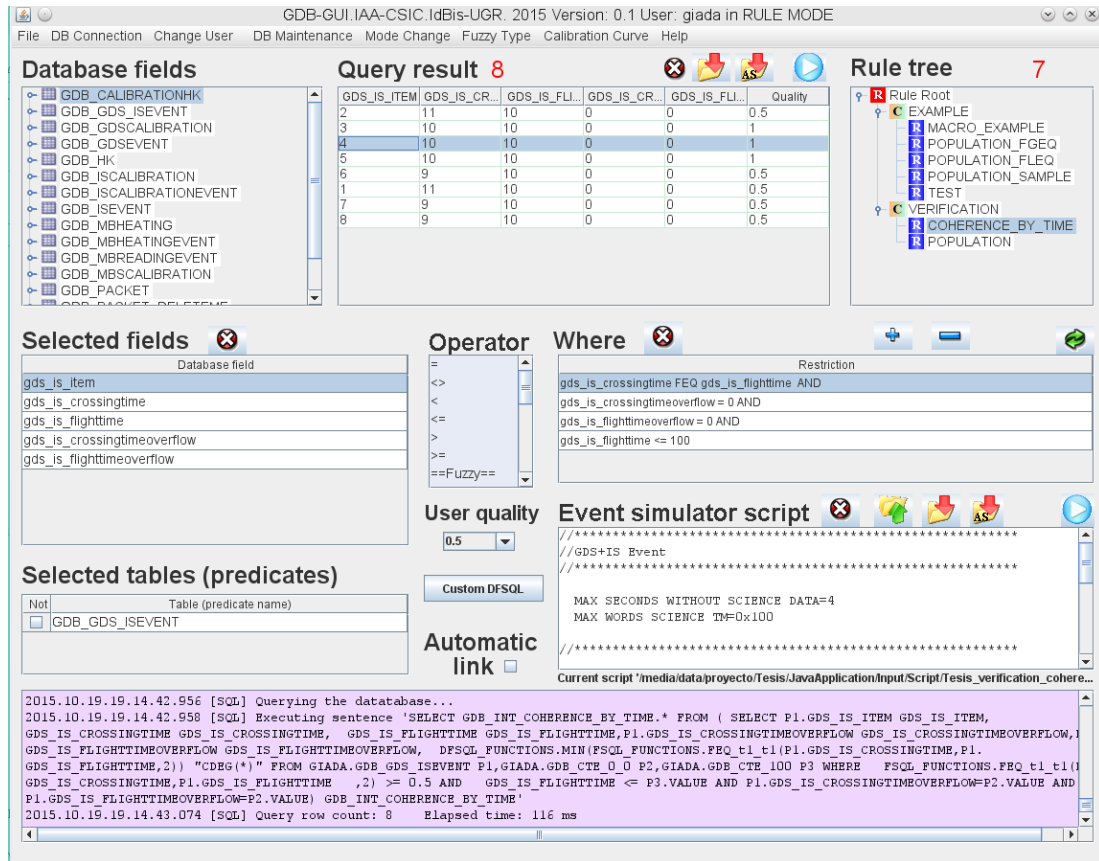


Ilustración 123. Ejecución de la regla de coherencia por tiempo en GDB-GUI.

7.2.1.7 Conclusiones

Dado que es posible medir la velocidad de una partícula de dos formas diferentes (*flight-time* y *crossingtime*) con los datos que proporciona un evento GDS+IS, se ha analizado la coherencia de dichas medidas. Se ha asumido que una diferencia mayor de 1ms^{-1} entre ambas (30% del rango común de medida), implicaba que el evento GDS+IS no había sido correctamente detectado y debería desecharse.

En el ejemplo de aplicación de la regla, ha aparecido eventos GDS+IS directamente descartados (calidad 0) por estar sus velocidades de *crossingtime* y *flighttime* demasiado alejadas entre sí. Sin embargo otros eventos han mejorado su calidad conforme disminuía su diferencia relativa.

Con esta información, deberá ser el científico responsable del análisis de datos el encargado de desechar de sus estadísticas los eventos con calidad cero, así como evaluar qué hacer con los eventos con calidad 0.5.

Esta regla puede ser utilizada para analizar eventos GDS+IS previamente detectados (en la calibración de G. o en los datos reales) y descubrir falsos positivos; es decir, eventos GDS+IS identificados, pero con calidad 0, por lo que deberían ser desechados debido a la falta de coherencia en las medidas de velocidad.

Un posible refinamiento adicional de esta regla consistiría en considerar distintos ángulos de incidencia de la partícula respecto al plano de detección. Este nuevo conocimiento implicaría relajar la diferencia entre velocidades y añadir restricciones sobre velocidades no compatibles con el ángulo de entrada.

Existe una forma de “rescatar” eventos GDS+IS descartados por la aplicación de la regla de coherencia en el caso de una doble detección GDS (2.2.3.1.1). En un evento doble en el GDS (por el canal izquierdo y derecho), sólo uno de ellos (el derecho, ver 7.1.3) se utiliza para crear un evento GDS+IS. Sería plausible considerar el otro evento GDS (el izquierdo) cómo un sustituto válido. Esto es, habría que desasociar el evento GDS presente en GDS+IS y sustituirlo por su evento GDS en el otro canal. Para realizar esta operación habría que localizar qué evento GDS es un buen candidato. Esta tarea puede complicarse en el caso de una entrada masiva de partículas, donde hay múltiples opciones para reemplazar al GDS defectuoso. En tal caso, habrá que evaluar la idoneidad de cada GDS candidato para emparejarse con el IS. La regla de cálculo de poblaciones de eventos GDS e IS (7.2.2) ayudará a realizar esta labor.

7.2.2 Regla para el cálculo de poblaciones de eventos GDS e IS

Como se indicó en 7.1.3 el SW de G. produce poblaciones de eventos individuales GDS e IS candidatas a formar eventos compuestos GDS+IS. La única forma de identificar estas parejas de eventos es comprobar que el tiempo de detección entre ellas se encuentra dentro de los límites detectables por el instrumento.

La velocidad mínima medible es de 0.3 ms^{-1} (asumiendo la misma velocidad mínima en la cortina y en el tránsito al IS) y la máxima de $0.3 \times 10^3 \text{ ms}^{-1}$ (Tabla 39). El tiempo máximo y mínimo que tardaría una partícula (asumiendo un grado de incidencia de 90 grados respecto al plano de detección del GDS) en producir un evento GDS e IS es el siguiente:

$$\text{max tiempo} = \frac{100 \times 10^{-3} \text{ m}}{0.3 \text{ m s}^{-1}} \approx 333 \text{ m s}$$

$$\text{min tiempo} = \frac{100 \times 10^{-3} \text{ m}}{300 \text{ m s}^{-1}} = 0.333 \text{ m s}$$

Por tanto, el tiempo transcurrido entre dos eventos individuales IS y GDS para que sean considerados como un evento GDS+IS plausible, debe estar entre 0ms y 333ms. La resolución de G. es de milisegundos, por lo que las fracciones de esta unidad son ignoradas.

7.2.2.1 Selección del tipo de dato difuso

La labor a realizar requiere la comparación de dos tiempos, con una diferencia relativa entre ellos en el intervalo de [0,333] ms, o lo que es lo mismo [0,83] cuentas (cada cuenta equivale a 4ms). Se asume que una diferencia de 4 cuentas es suficientemente grande como para considerar dos tiempos distintos, es decir cuatro veces la resolución del tiempo de evento de G.

El tipo de dato difuso *valores aproximados* es el que mejor encaja para para representar esta información. En la definición del tipo de dato utilizaremos un ‘margen’ de 8 cuentas (un entorno de 4 cuentas relativo al valor del tiempo) y un ‘much’ de 9 cuentas.

A modo de ejemplo, el valor de marca de tiempo “aproximadamente 300 cuentas” se representaría de la siguiente forma:

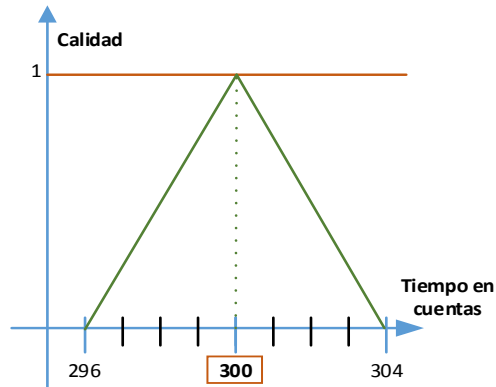


Ilustración 124. Representación de tipo difuso aproximadamente 300 cuentas.

7.2.2.2 Definición de relaciones clásicas

Las dos marcas de tiempo necesarias (`gds_eventtime` e `is_eventtime`) las encontramos como atributos en las tablas `GDB_GDSEVENT` y `GDB_ISEVENT`, respectivamente (ver [Ilustración 39](#)). La semántica detallada de estos campos, se encuentra en el [Apéndice A](#). Adicionalmente, se necesitarán los atributos `GDS_ITEM` e `IS_ITEM` para identificar cada evento de forma única.

Los atributos clásicos con las marcas de tiempo hay que extenderlos para incorporar el tipo difuso de *valores aproximados* con 'margen' =8 y 'much' =9.

Definición de la tabla `GDB_GDSEvent`:

```
CREATE TABLE GDB_GDSEvent (
...
GDS_ITEM NUMBER(2) NOT NULL Constraint "GDS_ITEM"
CHECK (GDS_ITEM >= 0),
...
GDS_eventTime FTYPE1(8,9) NUMBER(15) NOT NULL Constraint
"GDS_packetTime" CHECK (GDS_eventTime >= 0 and
GDS_eventTime <= 281474976710655), ...)
```

Definición de la tabla `GDB_ISEvent`:

```
CREATE TABLE GDB_ISEvent (
...
IS_ITEM NUMBER(2) NOT NULL Constraint "IS_ITEM"
CHECK (IS_ITEM >= 0),
...
IS_eventTime FTYPE1(8,9) NUMBER(15) NOT NULL Constraint, ...)
```

Se ha optado por el uso de *difusos tipo 1* para representar el tipo de dato de valores de *valores aproximados*, al igual que se aplicó en el ejemplo de regla ([5.4.2](#)).

7.2.2.3 Definición de la tabla intensiva y regla

Las poblaciones de eventos buscadas deben de cumplir las restricciones impuestas por el método de detección de G. Esto es, la diferencia de tiempo entre el evento de IS y el de GDS debe

estar dentro del rango (intervalo) de medición: entre 0 y 83 cuentas. Es más, también sería interesante calcular los eventos que se encuentren cerca de este intervalo y en qué grado lo están.

Una forma de encontrar a estos tipos de eventos sería la siguiente: fijado un evento de IS, habría que examinar los eventos GDS que se encuentren dentro del intervalo de medición definido por el tiempo de evento de IS. Esta labor debería de repetirse para todos los eventos de IS. Para ello utilizaremos las macros (5.1.2.5), que fueron incluidas en el modelo de datos GDB para trabajar con intervalos. Se utilizarán los operadores difusos (5.1.1.1) “*mayor o igual difuso que*” (FGEQ) y “*menor o igual difuso que*” (FLEQ) para establecer la pertenencia difusa al intervalo.

La regla que implementa estas restricciones se describe a continuación (expresada en DFSQL):

```
//===== Intensional table =====
create intensional table GDB_INT_POPULATION(
  GDS_ITEM NUMBER(2),
  GDS_EVENTTIME NUMBER(15),
  GDS_CROSSINGTIMEOVERFLOW NUMBER(1),
  IS_EVENTTIME NUMBER(15),
  IS_ITEM NUMBER(2));
//===== End of intensional table =====
//===== Rule =====
create rule for GDB_INT_POPULATION (X,Y,Z,W,P)
as
  GDB_GDSEVENT(X source gds_item, Y source gds_eventtime, Z
source gds_crossingtimeoverflow) AND
  GDB_ISEVENT(W source is_eventtime, P source is_item) AND
  GDB_CTE_4_0(Q source value) AND
  (Y FLEQ [W - 0]) AND
  (Y FGEQ [W - 83]) AND
  (Z = Q) AND
  (Y <= W);
//===== End of rule =====
```

Los predicados (Y FLEQ [W - 0]) AND (Y FGEQ [W - 83]) implementan la pertenencia al intervalo difuso [0,83] cuentas y el predicado (Y <= W) garantiza que no aparezcan tuplas que incluyan GDS sucedidos después de un IS. Se ha incluido un predicado para desechar los eventos GDS que tengan desbordamiento en el *crossingtime*, es decir, se ha de verificar: *gds_crossingtimeoverflow*=0.

7.2.2.4 Semántica de la medida de calidad

La regla calcula el grado de pertenencia (calidad) de un conjunto de eventos GDS respecto a un intervalo definido por el tiempo de evento IS. Cuanto mayor sea el grado obtenido, mayor será la confianza en que el evento de GDS más el IS pueda conformar un evento doble GDS+IS.

La pertenencia a un intervalo es un tanto diferente a la considerada en 5.1.1.1. No se define un entorno respecto a un evento IS (eventos GDS antes y después) sino un rango de tiempo antes del evento IS (los eventos GDS detectados justo después del IS no pertenecen al rango buscado). En la siguiente ilustración, el evento de IS en el marca de tiempo X, define un intervalo de tiempo (en gris) para los eventos GDS candidatos a crear un evento GDS+IS.

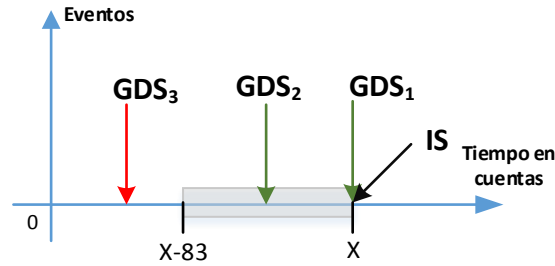


Ilustración 125. Intervalo de población de eventos GDS definido por un evento IS.

En la definición de la regla se incluyen dos comparadores difusos $FGEQ$ y $FLEQ$, por lo que habrá que seguir el proceso de definido en 5.1.1.2.1 para realizar su cálculo.

A modo de demostración, se analiza el cálculo de la regla de uno de los dos eventos IS que serán creados como datos sintéticos (7.2.2.5). En concreto, se analizará el evento IS creado con un tiempo de 683 cuentas, que genera un intervalo de tiempo para los eventos GDS de [600-683] cuentas. El cálculo con el otro evento IS sintético es análogo. El resultado de la aplicación de la regla sobre estos valores sintéticos se muestra en la siguiente tabla.

Tabla 42. Calidad de los eventos GDS a la izquierda y a la derecha del intervalo definido por un evento IS con tiempo 683.

GDB_GDSEVENT			
GDS_EVENTTIME (cuentas)	IS_EVENTTIME (cuentas) Límite izquierdo -derecho	Calidad a la izquierda del intervalo GDS_EVENTTIME FGEQ 600	Calidad a la derecha del intervalo GDS_EVENTTIME FLEQ 683
592	600-683	0	1
593	600-683	0.13	1
594	600-683	0.25	1
595	600-683	0.38	1
596	600-683	0.5	1
597	600-683	0.63	1
598	600-683	0.75	1
599	600-683	0.88	1
600	600-683	1	1
601	600-683	1	1
682	600-683	1	1
683	600-683	1	1

Nótese cómo los valores con marca de tiempo de GDS {600,601,682,683} que se sitúan en límite o dentro del intervalo definido [600,683] poseen una calidad máxima. La calidad calculada por el comparador FGEQ se va degradando hasta llegar a cero, según se alejen los tiempos de GDS del límite izquierdo del intervalo.

La regla descartará cualquier evento GDS posterior al IS, en este caso, posterior a 683 cuentas por lo que no aparecerán en el resultado. Esto implica que el cálculo con FLEQ siempre devolverá uno ya que los tiempos de los eventos de GDS siempre serán menores a los de IS debido a las restricciones impuestas en la definición de la regla.

Se detalla a continuación el cálculo del comparador FGEQ sobre los valores en cuentas de $GDS_EVENTTIME=295$ y $IS_EVENTTIME=300$, es decir $295 \text{ FGEQ } 300$. El valor de $IS_EVENTTIME$ ya ha sido decrementado en 83 cuentas como indica el predicado $(Y \text{ FGEQ } [W - 83])$.

En la siguiente ilustración se muestran las distribuciones de posibilidad asociadas a ambos valores de acuerdo al tipo de dato difuso *valores aproximados*. En verde el tiempo de evento de GDS y en marrón el de IS.

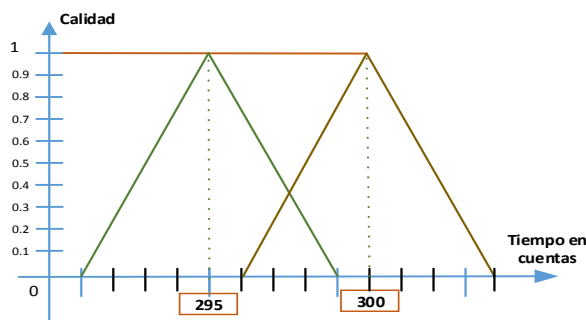


Ilustración 126. Representación de los tipos difusos aproximadamente 295 y 300.

Como se indicó en 5.1.1.2.1, en el cálculo del comparador se requiere modificar la distribución de posibilidad de la parte derecha de la comparación dependiendo del operador (en este caso FGEQ), para a continuación calcular la intersección.

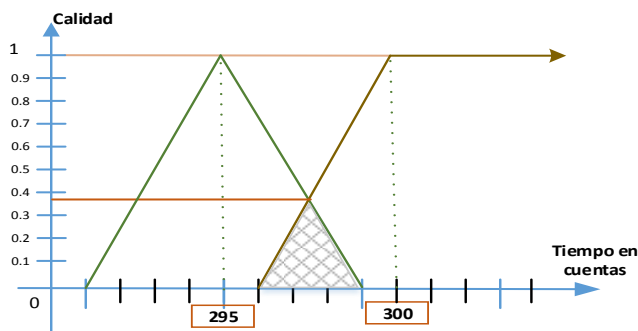


Ilustración 127. Cálculo del grado de compatibilidad de 295 FGEQ 300.

El valor de tiempo calculado por GDB $295 \text{ FGEQ } 300=0.38$ coincide al calculado gráficamente.

Como puede apreciarse, la modificación de la parte derecha de la distribución de posibilidad, es equivalente al cálculo de la intersección de las dos distribuciones de posibilidad sin aplicar dicha modificación.

En la siguiente ilustración se muestra todos los posibles valores de calidad (área rayada) resultante del cálculo de la comparación entre un tiempo de evento de IS de 300 (color marrón) y diferentes tiempos de GDS (color verde), correspondiente a eventos acaecidos con anterioridad. Se asume que el valor de tiempo de IS ya está decrementado en 83 cuentas.

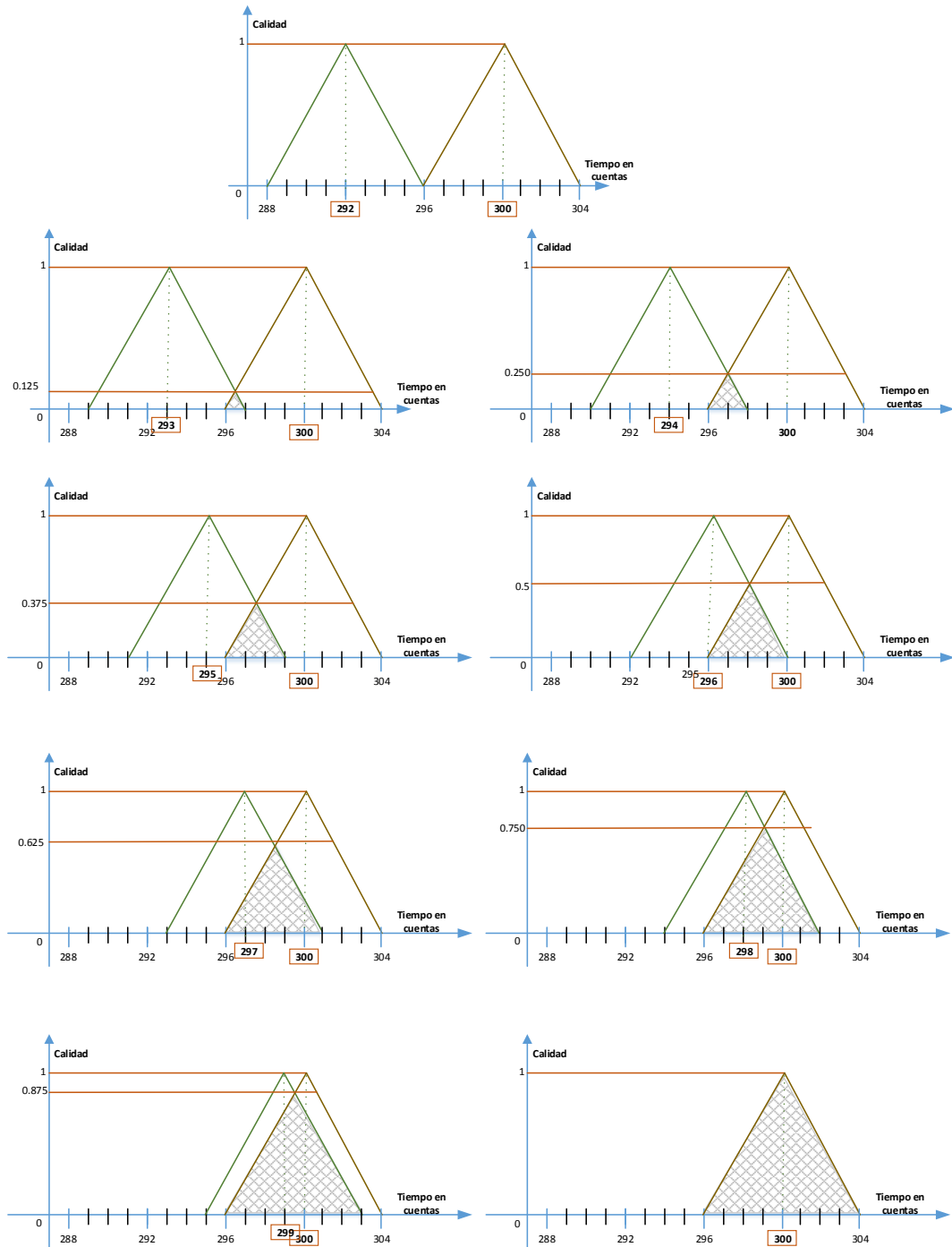


Ilustración 128. Cálculo de la calidad asignada para diferentes valores de tiempo de eventos de GDS respecto a un tiempo de evento IS.

Los valores de tiempo de GDS dentro del intervalo definido por el tiempo de IS se calcularán siempre con calidad 1. En la siguiente ilustración se calcula 301 FGEQ 300. El valor de tiempo de IS se indicará en color marrón y el del tiempo de GDS en color verde.

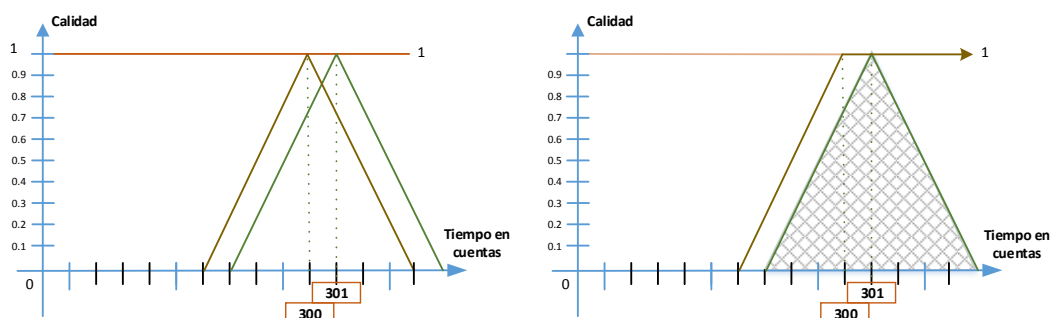


Ilustración 129. Cálculo del grado de compatibilidad de 301 FGEQ 300.

La regla creada es una RGA que alberga dos predicados con comparadores difusos, por lo que debe calcularse aplicando las operaciones definidas en 5.1.2.1.1 Cálculo de reglas generalizadas con grados de acoplamiento. De forma resumida, esto implica que la calidad final (grado de acoplamiento) será el mínimo del obtenido por el conjunto de predicados. Por lo que la tabla de eventos de GDS quedaría como sigue:

Tabla 43. Cálculo de la regla de poblaciones para un evento IS con tiempo 683 cuentas.

GDB_GDSEVENT		
GDS_EVENTTIME (cuentas)	IS_EVENTTIME (cuentas) Límite izquierdo -derecho	Calidad
592	600-683	0
593	600-683	0.13
594	600-683	0.25
595	600-683	0.38
596	600-683	0.5
597	600-683	0.63
598	600-683	0.75
599	600-683	0.88
600	600-683	1
601	600-683	1
682	600-683	1
683	600-683	1

7.2.2.5 Datos sintéticos

Se simulará un flujo de 15 eventos GDS y dos de IS. Se ignorarán los eventos GDS+IS, ya que estos podrán ser verificados aplicando al regla de coherencia por tiempo (7.2.1).

Los eventos sintéticos de GDS se distribuyen antes, durante y después del intervalo definido por el evento IS con tiempo 683. Es decir, se ha fijado un intervalo de tiempo para los eventos GDS definido por $[600=683-83, 683=683-0]$, como se muestra en la figura.

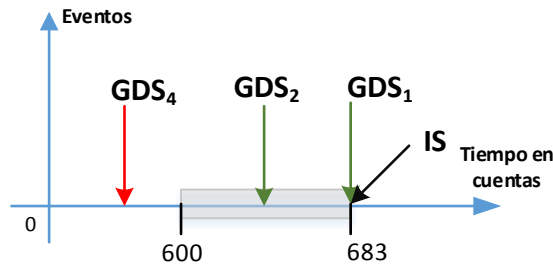


Ilustración 130. Intervalo de población de eventos GDS definido por un evento IS con tiempo de evento 683 cuentas.

Se muestra en la siguiente tabla los 15 eventos GDS generados:

Tabla 44. Tabla GDB_GDSEvent para la regla de poblaciones.

GDB_GDSEVENT	
GDS_ITEM	GDS_EVENTTIME (cuentas)
2	592
3	593
4	594
5	595
6	596
7	597
8	598
9	599
10	600
11	601
12	682
13	683
14	684
15	685
16	686

Los dos eventos de IS se diferencian en una cuenta de marca de tiempo.

Tabla 45. Tabla GDB_ISSEvent para la regla de poblaciones.

GDB_ISEVENT	
IS_ITEM	IS_EVENTTIME (cuentas)
0	683
1	684

7.2.2.6 Cálculo de la regla

Por claridad en el resultado, se ha omitido los atributos de desbordamiento del evento GDS que fueron definidos en la tabla intensiva. Cualquier tupla con desbordamiento con valor uno sería descartada.

Para un grado de calidad mínimo de 0.5, se han recuperado 8 eventos GDS candidatos a formar un evento GDS+IS con el evento IS con marca de tiempo 683 cuentas. Así mismo, se han detectado 8 eventos GDS para el evento IS con tiempo 684 cuentas.

Los dos eventos IS con tiempo 683 y 684 han generado dos intervalos para eventos GDS: [600,683] y [601,684] respectivamente. La siguiente tabla muestra la evaluación de cada uno de los eventos GDS presentes respecto a cada uno de los dos intervalos definidos.

Tabla 46. Resultado de la aplicación de la regla de poblaciones con calidad mínima 0.5.

GDB_INT_POPULATION				
GDS_ITEM	GDS_EVENTTIME (cuentas)	IS_ITEM	IS_EVENTTIME (cuentas)	Calidad
6	596	0	683	0.5
7	597	0	683	0.63
8	598	0	683	0.75
9	599	0	683	0.88
10	600	0	683	1
11	601	0	683	1
12	682	0	683	1
13	683	0	683	1
7	597	1	684	0.5
8	598	1	684	0.63
9	599	1	684	0.75
10	600	1	684	0.88
11	601	1	684	1
12	682	1	684	1
13	683	1	684	1
14	684	1	684	1

Estos valores de calidad coinciden, atendiendo a las diferencias relativas entre tiempos, a los estimados en 7.2.2.4 (Ilustración 128), pero asumiendo un redondeo al alza.

Para los eventos GDS con valor igual al límite del intervalo o dentro del mismo, GDB les ha asignado calidad máxima: {10,600}, {11,601}, {12,682} y {13,683} respecto al IS 683 y {11,601}, {12,682}, {13,683} y {14,684} para el IS 684. La pareja {x, y} indica los valores de los atributos x=GDS_ITEM y=GDS_EVENTTIME.

Existen eventos GDS que poseen máxima calidad para los dos intervalos, justo aquellos que se encuentran dentro de su intersección: {11,601}, {12,682}, {13,683}.

Los eventos GDS con tiempos demasiado alejados del IS, no han aparecido en el resultado del cómputo, al tener una calidad menor a 0.5: {2,592}, {3,593}, {4,594} y {5,595}.

Los eventos de GDS posteriores a los dos eventos IS no han aparecido en el resultado: {15,685} y {16,686}.

Los valores de tiempo de GDS han ido perdiendo calidad conforme se alejaban de los límites izquierdos de los intervalos.

Un mismo evento GDS {10,600} ha resultado tener calidad máxima para IS 683 (ya que se encuentra en el límite de su intervalo), pero una calidad inferior: 0.88 para IS 684 (puesto que no pertenece su intervalo pero está “próximo”).

El resultado de la ejecución de la regla en GDB-GUI se muestra a continuación:

The screenshot shows the GDB-GUI interface with the following components:

- Database fields:** A list of fields including GDB_CALIBRATIONHK, GDB_GDS_ISEVENT, GDB_GDSCALIBRATION, GDB_GDSEVENT, GDB_HK, GDB_ISCALIBRATION, GDB_ISCALIBRATIONEVENT, GDB_ISEVENT, GDB_MBHEATING, GDB_MBHEATINGEVENT, GDB_MBREADINGEVENT, GDB_MBSALIBRATION, and GDB_PACKET.
- Query result 16:** A table with columns GDS_ITEM, GDS_EVENT., GDS_CROS., IS_EVENTIT., IS_ITEM, and Quality. It contains 12 rows of data.
- Rule tree 7:** A tree structure showing rules like EXAMPLE, MACRO_EXAMPLE, POPULATION_FGEQ, POPULATION_FLEQ, TEST, VERIFICATION, COHERENCE_BY_TIME, POPULATION, and POPULATION_SAMPLE.
- Selected fields:** A list of fields including gds_item, gds_eventtime, gds_crossingtimeoverflow, is_eventtime, and is_item.
- Operator:** A dropdown menu set to "Fuzzy==".
- Where:** A text area containing the restriction: "gds_eventtime FLEQ is_eventtime - 0 AND gds_eventtime FGEQ is_eventtime - 83 AND gds_crossingtimeoverflow = 0 AND gds_eventtime <= is_eventtime".
- User quality:** A dropdown menu set to "0.5".
- Selected tables (predicates):** A list of tables including GDB_GDSEVENT and GDB_ISEVENT.
- Event simulator script:** A text area containing the script content, including "MAX SECONDS WITHOUT SCIENCE DATA=4" and "MAX WORDS SCIENCE TM=0x100".
- Automatic link:** A checkbox that is currently unchecked.
- Output log:** A text area at the bottom showing the execution of a SQL query and the resulting output, including row counts and file paths.

Ilustración 131. Ejecución de la regla de poblaciones en GDB-GUI.

7.2.2.7 Conclusiones

La regla propuesta para la detección de poblaciones de GDS e IS candidatas a crear un evento doble GDS+IS se ha aplicado sobre datos que simulan un jet de partículas. En concreto, se ha generado un flujo de 15 eventos GDS y otro de dos eventos IS. En esta situación, el SW de

GIADA generaría únicamente dos eventos GDS+IS y 13 eventos GDS individuales. GDB ha conseguido identificar, para cada uno de los dos eventos IS, 8 posibles eventos GDS candidatos a producir un evento doble GDS+IS.

Esto implica que, cada uno de los eventos GDS+IS es sólo uno de los 8 candidatos plausibles dentro de la población de GDS detectados por el instrumento. Sólo un análisis más en profundidad de estas poblaciones, podría arrojar más luz sobre qué eventos GDS poseen más fiabilidad que otros. En concreto, se podría crear una regla para aprovechar el tiempo de cruce de la cortina (*crossingtime* que es uno de los datos que proporciona cada evento de GDS) para calcular un posible tiempo de vuelo (y por tanto de impacto) y ver su relación con el tiempo de evento de IS.

En este punto, es necesario un experto en el estudio de los cometas para poner en contexto esta nueva información y aplicarla a los modelos teóricos de comportamiento del polvo. Así mismo, tendría que evaluar cómo se han de interpretar los eventos GDS de calidad baja.

7.3 Aplicación de las reglas de verificación de eventos GDS+IS a los datos de calibración

Una vez construidas las dos reglas para la verificación de eventos GDS+IS, es hora de ponerlas a prueba con los datos obtenidos en la calibración de G. La calibración del modelo de vuelo del instrumento se realizó desde el 10 al 14 de abril de 2002 en las instalaciones de Officine Galileo en Florencia, Italia. Existe un informe que describe el resultado de estas pruebas (RO-GIA-OACUPA-MN-xxx) y que de forma resumida consistió en lanzar un conjunto controlado de partículas con distinta composición química, tamaño y velocidad y observar la respuesta del instrumento.

Para concentrar el análisis, las reglas se aplicarán sobre los datos obtenidos el 13 de abril de 2002 durante una hora: desde 14h 02min hasta 14h 58min.

Utilizando el importador de GDB-GUI (6.2.5), es posible cargar los datos de esta franja temporal, obteniendo los siguientes resultados:

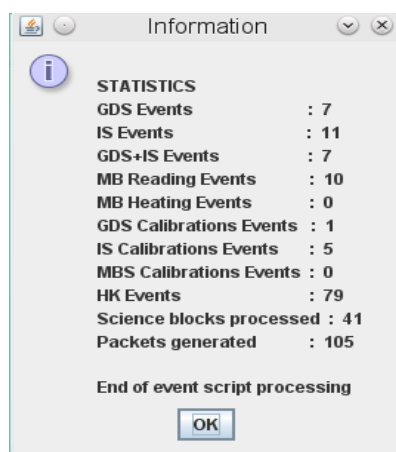


Ilustración 132. Datos de calibración de GIADA importados por GDB-GUI.

Los eventos relevantes para las reglas son los siguientes: 7 eventos individuales de GDS, 11 de IS, y 7 de GDS+IS. Utilizando la interfaz SQL de GDB-GUI (6.2), podemos conocer más detalles de ellos:

Tabla 47. Tabla GDB_GDSEvent con datos de calibración.

GDB_GDSEVENT	
GDS_EVENTTIME (cuentas)	Canal (izquierdo/derecho)
1410289408	izquierdo
1446813696	izquierdo
1475424512	izquierdo
1496523776	izquierdo
1510498304	izquierdo
1550211072	izquierdo
1571736064	izquierdo

Tabla 48. Tabla GDB_ISEvent con datos de calibración.

GDB_ISEVENT
IS_EVENTTIME (cuentas)
1392837632
1410298880
1410301952
1410305024
1410307328
1414914816
1510505728
1543109632
1550223616
1550226688
1550229504

Tabla 49. Tabla GDB_GDS_ISEvent con datos de calibración.

GDB_GDS_ISEvent			
GDS_IS_EVENTTIME (cuentas)	GDS_IS_CROSSINGTIME (cuentas/ms ⁻¹)	GDS_IS_FLIGHTTIME (cuentas/ms ⁻¹)	Canal (izquierdo/ derecho)
1410289408	230/1.30	4940/2.09	derecho
1446813696	186/1.61	6111/1.69	derecho
1475424512	241/1.24	5340/1.93	derecho
1496523776	3/100.00	5407/1.90	derecho
1510498304	226/1.33	5036/2.05	derecho
1550211072	224/1.34	5841/1.76	derecho

Atendiendo al documento con los resultados del impacto de las partículas durante la hora en estudio, se lanzó 6 veces una partícula de Nontronita de 200 μm - 500 μm . Las anotaciones manuales en el informe (que se corrigen a posteriori con los datos proporcionados por GIADA) indican que se detectaron 6 eventos GDS+IS, 6 eventos GDS y un solo evento de IS, como se muestra en la siguiente tabla:

Tabla 50. Anotaciones manuales en las sesiones calibración.

Sesiones de calibración				
Sesión (hora-minuto-segundo)	Número de partícula	GDS	IS	GDS+IS
14-02-01	13	1	1	1
14-27-57				
14-32-15	14	1	0	1
14-39-24	15	1	0	1
14-45-14	16	1	0	1
14-48-41				
14-55-39	17	1	0	1
14-58-53	18	1	0	1

Cada sesión corresponde a un fichero con datos en hexadecimal que serán importados por GDB-GUI. Existen dos sesiones dónde no se ha arrojado ninguna partícula (14-27-57 y 14-48-41).

Las anotaciones manuales indican que se han lanzado 6 partículas, pero analizando la marca de tiempo de la tabla de eventos de GDS (*Tabla 47*), comprobamos que realmente se lanzaron 7.

Un análisis de las tablas obtenidas con GDB-GUI: *Tabla 47*, *Tabla 48* y *Tabla 49*, indica que cada partícula fue detectada por ambos canales (izquierdo y derecho) y por el IS. Siguiendo el algoritmo de asociación de eventos (*7.1.3.1*), el GDS izquierdo se procesó como evento individual (como queda reflejado en la *Tabla 47*) y el derecho pasó a formar parte del evento GDS+IS (*Tabla 49*). Se puede apreciar la correlación de marcas de tiempo de ambas tablas (*Tabla 47* y *Tabla 49*), lo que demuestra la detección doble en el GDS.

Hasta este punto, la detección se ha realizado como indica el análisis de *7.1*. Los problemas aparecen a la hora de explicar el comportamiento del IS (*Tabla 48*). A priori, todos los eventos IS habían de ser procesados como eventos dobles GDS+IS, pero nos encontramos con 11 eventos inesperados.

El primer evento IS con marca de tiempo 1392837632, debe ser descartado ya que se ha producido antes que cualquier evento de GDS. Estos eventos espurios son inherentes al propio mecanismo que se utilizó para la generación de partículas: una pistola de aire comprimido. La pistola provocaba que el aire impactara en IS (generando una detección) antes que lo hiciera la partícula lanzada.

Para distinguir si una partícula ha generado un rebote, basta con agrupar los eventos de IS que se han producido antes de detectar la siguiente partícula GDS. Los cinco eventos de IS siguientes: 1410298880, 1410301952, 1410305024, 1410307328 y 1414914816, corresponden a rebotes de la partícula detectada por el GDS en 1410289408. Los eventos IS 1510505728 y 1543109632 son rebotes de la partícula detectada en 1510498304. Los restantes eventos IS 1550223616, 1550226688 y 1550229504 corresponden a rebotes de la partícula hallada en 1550211072.

Hay que recordar que la calibración se realizó en la Tierra, donde la gravedad juega un papel importante al atraer a las partículas una vez que han impactado, lo que provocó un conjunto de detecciones secundarias. En las regiones del espacio donde la gravedad es prácticamente nula, el comportamiento es diferente. Las partículas detectadas por el GDS en 1446813696, 1475424512, 1496523776 y 1571736064 no provocaron rebotes, pero sí lo hicieron 1410289408, 1510498304 y 1550211072.

Una vez analizado y comprendido el contenido de las tablas, se procede a ejecutar las dos reglas definidas para la verificación de eventos GDS+IS.

La primer regla, la de coherencia ([7.2.1](#)), arroja los siguientes resultados con una calidad mínima de 0.5:

Tabla 51. Resultado de la regla de coherencia aplicada a los datos de calibración con calidad mínima 0.5.

GDB_INT_COHERENCE_BY_TIME			
GDS_IS_ITEM (cuentas)	GDS_IS_CROSSINGTIME (ms ⁻¹)	GDS_IS_FLIGHTTIME (ms ⁻¹)	Calidad
1410289408	1	2	0.5
1446813696	1	1	1
1475424512	1	1	1
1510498304	1	2	0.5
1550211072	1	1	1
1571736064	1	1	1

El evento GDS+IS con tiempo 1496523776 ha sido descartado, dado que sus valores de velocidades entre *crossingtime* y *flighttime*: 100ms⁻¹ (3 cuentas) y 1.90ms⁻¹ (5407 cuentas), respectivamente, son demasiado diferentes entre sí. Esta regla asumía 1m⁻¹ como diferencia máxima para comparar ambas velocidades. En concreto, el valor extraño en este evento descartado es el del *crossingtime*. Una posible explicación es que la partícula haya penetrado una de las áreas de menos sensibilidad del GDS (ver [2.2.3.1.1.1](#)). La partícula ha sido detectada, pero sólo con el mínimo número pulsos necesario.

Aplicando el “rescate” de eventos descrito en [7.2.1.7](#), al evento GDS+IS descartado (con tiempo 1496523776), sería necesario calcular la población de eventos candidatos a sustituir al GDS defectuoso. En este caso es sencillo, su evento doble aparece como evento individual GDS con tiempo 1496523776 y además sin rebotes IS. En un caso más general, habría que calcular el momento en el que produjo el evento de IS a partir del *flighttime*, fijarlo y aplicar la regla de poblaciones para medir la idoneidad de los GDS candidatos respecto al intervalo definido por este tiempo de evento de IS.

La segunda regla, la de poblaciones ([7.2.2](#)) devuelve los siguientes resultados, asumiendo una calidad mínima de 0.1:

Tabla 52. Resultado de la regla de poblaciones aplicada a los datos de calibración con calidad mínima 0.1.

GDB_INT_POPULATION		
GDS_EVENTTIME (cuentas)	IS_EVENTTIME (cuentas)	Calidad
1410289408	1410301952	1
1410289408	1410305024	1
1410289408	1410307328	1
1410289408	1410298880	1
1510498304	1510505728	1
1550211072	1550223616	1
1550211072	1550226688	1
1550211072	1550229504	1

Es importante resaltar que la calidad es 1 en todas las tuplas, por lo que no tenemos ningún evento GDS “cerca” del intervalo definido por los eventos de IS, todos se encuentran dentro del intervalo.

Los eventos IS devueltos por GDB-GUI, coinciden con los rebotes detectados en el análisis de la tabla de IS, excepto en tres casos: 1392837632 (IS_A), 1414914816 (IS_B) y 1543109632 (IS_C). Recordemos que la semántica de la regla de poblaciones consiste en buscar conjuntos de eventos de IS y de GDS candidatos a formar un evento doble GDS+IS. Los rebotes se han comportado como un flujo de eventos IS, sobre los cuales la regla ha intentado buscar eventos GDS dentro del rango de detección. Como los GDS del canal derecho ya habían sido asociados a los eventos GDS+IS, los eventos IS restantes han sido los que encajan en el patrón de rebotes producidos por los GDS del canal izquierdo. Los eventos de IS espurios (IS_A) o demasiados alejados (IS_B y IS_C) han sido descartados directamente.

Por tanto, una funcionalidad adicional de la aplicación de la regla de poblaciones en los datos de calibración es la habilidad de detectar patrones de rebotes de partículas, eliminando IS espurios, además de los IS no relacionados con algún GDS.

7.4 Resumen y conclusiones

En este capítulo se ha puesto a prueba tanto el modelo de datos de GDB, como su interfaz GDB-GUI, creando dos reglas basadas en conocimiento experto, detallando su cálculo sobre datos sintéticos y verificando el resultado.

La primera regla creada (coherencia de eventos GDS+IS) ha sido capaz de identificar eventos GDS+IS que no corresponden a una detección legítima de partículas en G.

Mediante la segunda regla (cálculo poblaciones de eventos GDS e IS), se ha deducido nueva información que ha permitido ampliar los resultados que obtiene el instrumento ante una situación de estrés, como es un flujo continuo de partículas cometarias (jet).

La combinación de ambas reglas, ha permitido crear un método para recuperar eventos GDS+IS que inicialmente hubieran sido descartados, ya que no se ajustan a una detección válida.

Ambas reglas se han aplicado sobre los datos de calibración del modelo de vuelo de G. en Tierra, comprobando que son herramientas válidas para desechar falsos positivos en eventos GDS+IS, detectar rebotes en las mediciones de IS y recuperar eventos GDS+IS descartados.

Aunque las reglas creadas se han centrado en la detección y procesamiento de eventos GDS+IS, existen muchos otros tipos de conocimiento experto dentro de G. susceptibles de poder ser analizados aprovechando las posibilidades que ofrece GDB y GDB-GUI: relación entre el ángulo de entrada de la partícula y las posición del impacto en la membrana del IS, inferencia de características de las partículas a través de la luz dispersada en la cortina del GDS, análisis temporal de las prestaciones de los sensores, estudio de las acumulaciones de material en el sistema MBS respecto a la posición de Rosetta., etcétera.

8 Conclusiones y trabajos futuros

El objetivo principal de esta memoria era disponer de una herramienta capaz de representar y explotar los datos y el conocimiento experto procedentes del instrumento GIADA, con el fin de profundizar en el estudio de la dinámica del polvo cometario.

Se ha presentado GDB-GUI, un programa de computador multiusuario y multiplataforma que implementa una interfaz al modelo de base de datos GDB. Este modelo es una adaptación del modelo integrado de una base de datos relacional difusa deductiva, heredero de una evolución de modelos de bases de datos relacionales, lógicas y difusas.

Para verificar GDB-GUI y el propio modelo, se ha utilizado conocimiento experto, en concreto, el del sistema de detección de partículas implementado por el software de GIADA. Se ha construido un conjunto de reglas relativas a la detección de eventos GDS+IS (los tipos de eventos con mayor interés científico), se ha calculado el resultado y se ha contrastado con el obtenido por GDB-GUI.

Las reglas creadas han permitido identificar falsos positivos en la detección de eventos GDS+IS, revelar patrones de rebotes en el IS, así como descubrir y evaluar nueva información acerca de eventos GDS+IS no explícitamente identificados. Más aún, la combinación de ambas reglas ha establecido un procedimiento para reparar eventos GDS+IS detectados de forma defectuosa, que, inicialmente, deberían ser desechados. Se puede apreciar, por tanto, no sólo la capacidad de la herramienta para representar y manejar información, sino cómo su uso con conocimiento experto preciso ofrece resultados científicos relevantes.

Las aplicaciones de la herramienta alcanzan otros campos de trabajo. Se puede utilizar como repositorio del conocimiento experto en cualquier área de trabajo de GIADA, ya que lo centraliza, lo hace accesible a usuarios dispersos geográficamente y mantiene su trazabilidad. También ayuda a la interpretación de los datos reales del cometa, a los obtenidos en la fase de calibración del instrumento e incluso a los datos descartados. El simulador de eventos, inicialmente diseñado para la verificación de consultas, posee el potencial necesario para ser utilizado en la simulación de flujos reales de partículas y producir conjuntos de datos diseñados para validar otras herramientas (distintas a GDB-GUI) dentro de la cadena de datos del instrumento.

GDB-GUI ofrece a un usuario una interfaz capaz de realizar consultas clásicas a una base de datos relacional o bien crear reglas con componentes difusos, todo ello de forma visual y sin necesidad de conocer un lenguaje de consulta específico.

La adaptación de GDB a otro sistema/instrumento es sencilla y requiere redefinir cuatro ficheros de texto, siendo uno de ellos opcional.

El tratamiento de información difusa y deductiva que incorpora el modelo de base de datos, será especialmente útil en el análisis de los datos procedentes de GIADA. Las características propias del sistema de detección del instrumento, así como la degradación de los sensores durante una década de travesía espacial, implica que parte del conocimiento acerca del instrumento tienda hacia conceptos imprecisos: “eventos cercanos”, “partículas lentas”, “sensibilidad reducida”. Las capacidades deductivas del sistema presentado, permitirán aportar nueva información sobre los datos ya procesados, con los que se podrá reevaluar los resultados obtenidos.

Finalmente, se enumeran los posibles trabajos futuros a realizar:

- ❖ **Algoritmo de deducción.** El algoritmo de cálculo de reglas se realiza en un único paso, por lo que las relaciones intermedias hay que procesarlas cada vez, aunque la información no cambie. Por tanto, se hace necesario una optimización de este tipo de sub-consultas. Así mismo, sería recomendable crear un algoritmo de cálculo de

una secuencia de reglas que permitiera incorporar, de forma automática, el resultado de una regla, como predicado de la siguiente.

- ❖ **Mejoras en el formato de salida.** Los resultados de las consultas se pueden exportar, pero sólo de forma tabular. Formatos de salida como PDS (“Planetary Data System” usado por NASA y ESA) o específicos para otros tipos de bases de datos como GIPSI (“GIADA performance simulation”), serían ampliaciones convenientes de GDB-GUI.
- ❖ **Mejoras en el DFSQL.** Una mayor expresividad repercutiría en una mejor representación del conocimiento. Algunos puntos a tratar serían: incluir ordenación de atributos, añadir la gestión de grupos, ampliar la lista de operadores en la parte derecha de los comparadores y poder utilizar funciones polinómicas en la representación de los tipos de datos difusos.
- ❖ **Aplicación de nuevas técnicas de explotación de datos.** El camino natural de GDB-GUI sería la incorporación de técnicas novedosas de identificación de patrones recurrentes en grandes volúmenes de datos: “data mining” y “big data”. Este punto es especialmente interesante si se opta por combinar y contrastar los resultados obtenidos con los de otros instrumentos a bordo de Rosetta.
- ❖ **Inclusión de GDB-GUI en la cadena de análisis de datos de GIADA.** Una vez finaliza su construcción, GDB-GUI será presentado y evaluado por el grupo internacional de trabajo dedicado al estudio de los datos científicos procedentes del instrumento GIADA. Finalizado este proceso, se espera que GDB-GUI sea incorporado como un eslabón más en la cadena de análisis de datos del instrumento.

9 Bibliografía

[Aho86] A. V. Aho, R. Sethi and J. D. Ullman. "Compilers: principles, techniques and tools". Addison-Wesley, 1986.

[Aho88] A. V. Aho, J. E. Hopcroft and J. D. Ullman. "Estructuras de datos y algoritmos". Addison-Wesley, 1988.

[Alt15] K. Altwegg, H. Balsiger, A. Bar-Nun et al. "67P/Churyumov-Gerasimenko, a Jupiter family comet with a high D/H ratio". Science 23 January 2015. DOI:10.1126/science.1261952

[Bar93] A. Bar-Num et al. "Rosetta: comet rendezvous mission". ESA study report, Noordwijk, the Netherlands, 1993.

[Bar92] R. Barker. "CASE*METHOD: Entity relationship modeling". Addison Wesley, 1992.

[Bie02] J. Biele. "The experiments on board Rosetta Lander". Earth, Moon and Planets, vol 90, p 445-458 Addison Wesley, 2002.

[Bla00] I. Blanco, J.C. Cubero, O. Pons y M. A. Vila. "An implementation for fuzzy deductive relational databases". Recent research issues on the management of fuzziness in database. pp. 183-207. Studies in fuzziness and soft computing, Physica-Verlag, 2000.

[Bla0b] I. Blanco, N. Marín, O. Pons y M. A. Vila. "An extension of data description language (ddl) for fuzzy data handling". Flexible query answering systems, recent advances. Larsen, Kacprzyk, Zadrozny, Andreasen y Christiansen editors. Advances in soft computing. Physica-Verlag, 2000.

[Bla01] I. Blanco. "Deducción en bases de datos relacionales difusas". Tesis doctoral. Departamento de Ciencias de la Computación e Inteligencia Artificial. Universidad de Granada, España, 2001.

[Bow97] J.S. Bowman, S.L. Emerson and M. Darnovsky. "The practical SQL handbook". 3rd edition. Addison-Wesley, 1997.

[Boy71] R.S. Boyer. "Locking a restriction of resolution". Tesis doctoral, Universidad de Texas, Austin. E.E.U.U., 1971.

[Buc82a] B.P. Buckles and F. E. Petry. "A fuzzy representation of data for relational databases". Fuzzy sets and systems, 7, pp.213-226, 1982.

[Buc82b] B.P. Buckles and F. E. Petry. "Fuzzy databases and theirs applications". Fuzzy information and decision processes, Vol. 2, Eds. M. Gupta, E. Sánchez. North-Holland, Amsterdam, pp. 361-371, 1982.

[Buc84] B.P. Buckles and F. E. Petry. "Extending the fuzzy database with fuzzy numbers". Information sciences 34, pp. 45-55, 1984.

[Bus99] E. Bussoletti et al. "The GIADA experiment for Rosetta mission to comet 46P/Wirtanen: Design and performances". Advances in Space Research, 24, 1139-1148, 1999.

[Bus00] E. Bussoletti et al. "The scientific goals of the Grain Impact Analyser and Dust Accumulator". Astron. Soc. Pacific, 2000.

- [Cer90]** S. Ceri, G. Gottlob and L. Tanca. "Logic programming and databases". Surveys in computer science, Ed. Springer-Verlag, 1990.
- [Cha73]** C. Chang and R. C. Lee. "Symbolic logic and mechanical theorem proving". Semantic Resolution and lock resolution, pp. 100-129. Computer Science Classics, Academic Press, 1973.
- [Cha74]** D. D. Chamberlin. "SEQUEL: A structured english query language". Proc. ACM SIG-MOD workshop on data description, access and control, 1974.
- [Cha76]** D.D. Chamberlin et al. "SEQUEL 2: A unified approach to data definition, manipulation and control". IBM J. research and development, 20, num. 6, pp. 560-575, Nov 1976.
- [Clo81]** S. F. Clocksin and C.S. Mellish. "Programming in Prolog". Ed. Springer-Verlag, 1981.
- [Che76]** P.P-S. Chen. "The Entity-Relationship Model: Toward a Unified View of Data". ACM transactions on database systems, vol. 1, pp. 9-36, 1976.
- [Cod70]** E.F. Codd. "A relational model of data for large shared data banks". Communications of the ACM, 13, pp. 377-387, June 1970.
- [Cod82]** E.F. Codd. "Relational databases: a practical foundation for productivity". Communications of the ACM, 25(2), 1982.
- [Col04]** L. Colangeli et al. "The GIADA experiment for the Rosetta mission". Kluwer Academic Publishers, pp. 271-280, Dordrecht, Netherlands, 2004.
- [Col06]** L. Colangeli et al. "The grain impact analyzer and dust accumulator (GIADA) experiment for the Rosetta mission: design, performances and first results". Space Science Reviews, pp. 803-821, 2006.
- [Col07]** L. Colangeli et al. "GIADA: the grain impact analyser and dust accumulator for the Rosetta space mission". Advances in space research 39, pp. 446-450, 2007.
- [Cri97]** J. F. Crifo and A. V. Rodionov. "The Dependence of the Circumnuclear Coma Structure on the Properties of the Nucleus". Icarus 127, pp. 319-353, 1997.
- [Dat85]** C.J. Date. "An introduction to Database Systems". Vol. II, The Systems Programming series, Addison Wesley, 1985.
- [Dea74]** A. Deaño. "Introducción a la lógica formal". Ed. Alianza Universidad, 1974.
- [Dub80]** D. Dubois and H. Prade. "Fuzzy sets and systems: theory and applications". Academic Press, New York, 1980.
- [Dub85]** D. Dubois and H. Prade. "Fuzzy number, an overview of the analysis of fuzzy information". J.C. Bezdek, CRS press, Boca Raton F1, USA, 1985.
- [Deg86]** D. DeGroot and G. Lindstrom "Logic programming. Functions, relations and equations". Ed. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1986.
- [Eco13]** T. E. Economou, S. F. Green, D. E. Brownlee and B. C. Clark "Dust flux monitor instrument measurements during Stardust-NExT flyby of comet 9P/Tempel 1". Icarus 222, pp. 526-539, 2013.

[ESA01a] ESA. "Rosetta OBDH Interface requirements". RO-EST-RS-3009/EID B 2.7, Issue 2, 2001.

[ESA01b] ESA. "Rosetta Experiment software and autonomous functions". RO-EST-RS-3009/EID B 2.8, Issue 2, 2001.

[Esp02] F. Esposito, L. Colangeli, V. Della Corte, P. Palumbo and the international GIADA team. "Physical aspect of an 'impact sensor' for the detection of cometary dust momentum onboard the 'Rosetta' space mission". *Advances in space research* 8, pp. 1159-1163. 2002.

[Est80] F. Esteva y X. Domingo. "Sobre funciones de negación en $[0,1]$ ". *Stochastica*, IV(2):141-165, 1980.

[Fro86] R. A. Frost. "Introduction to knowledge base systems". Ed. Collins, 1986.

[Fuk79] S. Fukami, M. Umano, M. Muzimoto and H. Tanaka. "Fuzzy database retrieval and manipulation language". *IEICE technical reports*, vol. 8, nº 233, pp. 65-72, AL-78-85, 1979.

[Ful95] M. Fulle, L. Colangeli, V. Mennella, A. Rotundi and E. Bussoletti. "The sensitivity of the size distribution to the grain dynamics: simulation of the dust flux measured by GIOTTO at 1P/Halley". *Astronomy and astrophysics*, 304, 622, 1995.

[Ful99] M. Fulle, J. F. Crifo and A. V. Rodionov. "Numerical simulation of the dust flux on a spacecraft in orbit around an aspherical cometary nucleus". *Astronomy and astrophysics*, v.347, pp. 1009-1028, 1999.

[Ful10] M. Fulle et al. "Comet 67P/Churyumov-Gerasimenco: the GIADA dust environment model of the Rosetta mission target". *Astronomy and astrophysics*, 2010.

[Kow79] R. Kowalski. "Logic for problem solving". North Holland Publishing Co., 1979.

[Gal87] H. Gallaire, J. Minker and J. M. Nicolas "Logic and databases: a deductive approach". *ACM computing surveys*, vol. 16(2), pp. 153-185, 1987.

[Gal98a] J. Galindo, J.M. Medina, O. Pons, M.A. Vila and J.C. Cubero. "A prototype for a fuzzy relational database". Demo session in the 6th International Conference on Extending Database Technology, EDBT'98, Valencia (España), March 1998.

[Gal98b] J. Galindo, J.M. Medina, O. Pons and J.C. Cubero. "A server for fuzzy SQL queries". In "Flexible query answering systems", eds. T. Andreasen, H. Christiansen and H.L. Larsen. *Lecture notes in artificial intelligence (LNAI) 1495*, pp. 164-174. Ed. Springer, 1998. International conference on flexible query answering systems, FQAS'98, Roskilde (Dinamarca), May 1998.

[Gal98c] J. Galindo, J.M. Medina, A. Vila and O. Pons, "Fuzzy comparators for flexible queries in databases". *Iberoamerican conference on artificial intelligence, IBERAMIA'98*, Lisboa (Portugal), pp. 29-41, October 1998.

[Gal99] J. Galindo. "Tratamiento de la imprecisión en bases de datos relacionales: extensión del modelo y adaptación a los SGBD actuales". Tesis doctoral. Departamento de Ciencias de la Computación e Inteligencia Artificial. Universidad de Granada, España, 1999.

[Gia06] GIADA team. "GIADA FS experiment user manual RO-GIA-MA-007". Issue 4, 2006.

[Gra80] J. Grant. "Incomplete information in a relational database". *Fundamenta informatica*, 3, pp. 363-378, 1980.

[Gra84] P. M. D. Gray. "Representing programs by clauses". *Logic, algebra and databases*, pp. 73-96, Ed. Meek, B.L. Londres, Reino Unido 1984.

[Gre04] S. F. Green, J. A. M. McDonnell et al. "The dust mass distribution of comet 81P/Wild 2". *Journal of geophysical research*, 109, E12S04, 2004.

[Hec99] M. Hechler. "Rosetta mission design". *Advances in Space Research*, volume 19, pp. 127-136. 1997.

[Hog90] C. J. Hogger. "Essentials of logic programming". Eds. D.M. Gabay, C.L. Hanking and T.S. Maibaum. Clarendon press. Oxford, UK, 1990.

[Ioa88] Y. E. Ioannidis and R. Ramakrishnan. "Effective closure algorithms". *Proceedings of 14th VLDB Conference*, pp. 382-394, Bancilhon and D. J. DeWitt Eds., Morgan Kaufmann, 1988.

[Kow79] R. Kowalski. "Logic for problem solving". Ed. North Holland Publishing Co., 1979.

[Kro03] M. Krolikowska. "67P/Churyumov-Gerasimenko potential target for the Rosetta mission". *Acta astronautica*, vol 53, pp. 195-209, 2003.

[Li90] D. Li and D. Liu. "A fuzzy prolog database system". *Computing systems series*. John Wiley & Sons, 1990.

[Lop00] A. López, J. Rodríguez, J. Sánchez and M. Herranz. "DPU peripheral description, memory and I/O map". *RO-GIA-IAA-TN018*, issue 2, revision 1, IAA, 2000.

[Lop06] A. López. "Aplicación de dispositivos FPGA a la instrumentación espacial: los instrumentos GIADA y OSIRIS de la misión Rosetta". Tesis doctoral, Universidad de Granada, 2006.

[Lov70] D. W. Loveland. "A linear format of resolution". *Proceedings of IRIA symposium of automatic demonstration*, pp. 147-162, Springer-Verlag, New York, USA, 1970.

[Luc70] D. Luckham. "Refinements in resolution theory". *Proceedings of IRIA symposium of automatic demonstration*, pp. 163-190, Springer-Verlag, Versailles, Francia, 1970.

[Maz02] E. Mazzotta et al. "The grain detections system for the GIADA instrument: design and expected performances". *Adv. Space Res.*, 29:1165-1169, 2002.

[McD81] J. A. M. McDonnell et al. "A Dust Impact Detection System (DIDSY) for the Giotto Halley Mission". *Scientific and experimental aspects of the GIOTTO mission*, ESA-SP-169. pp. 61-75, 1981.

[Med94a] J.M. Medina, O. Pons and M.A. Vila. "GEFRED. A generalized model of fuzzy relational databases". *Information Sciences*, 76(1-2), 87-109, 1994.

[Med94b] J.M. Medina. "Bases de datos relacionales difusas. Modelo teórico y aspectos de su implementación". Tesis doctoral. Universidad de Granada, España, Mayo 1994.

[Med95a] J.M. Medina, J.C. Cubero, O. Pons and M.A. Vila. "Towards the implementation of a generalized fuzzy relational database model". *Fuzzy sets and systems*, 75, pp. 273-289, 1995.

[Med95b] J.M. Medina, O. Pons and M.A. Vila. "FIRST. A fuzzy interface for relational systems". I International fuzzy systems association world congress (IFSA 1995). Sao Paulo, Brasil, 1995.

[Med96] J.M. Medina, M.A. Vila, J.C. Cubero and M.A. Vila. "An architecture for a deductive fuzzy relational database". Lecture notes in artificial intelligence, Springer, tome 1079, pg. 491-500, 1996.

[Med97] J.M. Medina, O. Pons and M.A. Vila. "FREDDI. A fuzzy relational deductive database interface". International journal of intelligent systems, pp. 597-613, 1997.

[Min82] J. Minker. "On indefinite databases and the close world assumption". Proceedings of the sixth conference on automated deduction, Lecture notes in computer science, 38, pp. 292-308, 1982.

[Mor01a] R. Morales. "GIADA user and software requirements document". RO-GIA-IAA-DD-002, issue 2, IAA, 2001.

[Mor01b] R. Morales, I. Olivares. "GIADA software interface control document". RO-EST-RS-3009/EID B, issue 2, IAA, 2001.

[Mor02a] R. Morales. "GIADA software design document". RO-GIA-IAA-DD-002, issue 2, IAA, 2002.

[Mor02b] R. Morales. "GIADA software validation and verification plan". RO-GIA-IAA-DD-001, issue 2, IAA, 2002.

[Mor02c] R. Morales. "GIADA FS software validation test results", issue 2, IAA, 2002.

[Mor04] R. Morales. "GIADA-2 FS software user manual". RO-GIA-IAA-MA-009. issue 2, IAA, 2004.

[Mor08] R. Morales, I. Blanco, O.Pons and J. Rodríguez, The International GIADA Team and IDBIS Research Group "GDB: A Tool to build deductive rules using a fuzzy relational database with space scientific data". Fuzzy Sets and Systems, vol. 159/12, pp. 1577-1596, 2008.

[Nil87] N.J. Nilsson. "Principios de inteligencia artificial". Ed. Díaz Santos, 1987.

[Pet96] F.E. Petry. "Fuzzy Databases: Principles and Applications". International series in intelligent technologies". Ed. H.-J. Zimmermann, Kluwer academic publishers, 1996.

[Pon92] O. Pons y M. A. Vila. "Interface entre BD relacionales difusa y el entorno de Prolog". Proc. II Congreso español sobre tecnologías y lógica fuzzy, pp. 215-232, Madrid, España, 1992.

[Pon94a] O. Pons, M. A. Vila y J.M. Medina "Handling imprecise medical information in the framework of logic fuzzy databases". Fuzzy systems and artificial intelligence, 3(1), 5-25, Ed. Academiei Romane, 1994.

[Pon94b] O. Pons. "Representación lógica de bases de datos difusas. Fundamentos teóricos e implementación". Tesis doctoral. Departamento de Ciencias de la Computación e Inteligencia Artificial. Universidad de Granada, España, 1994.

[Pon96] O. Pons., J.M. Medina, J.C. Cubero and M. A. Vila "An architecture for a deductive fuzzy relational database". Foundations of Intelligent Systems, Springer, 1996.

[Pon97] O. Pons., J.M. Medina, J.C. Cubero and M. A. Vila “Flexible query answering systems”. A fuzzy deductive relational database, Kluwer academic publishers, 1997.

[Pra84a] H. Prade and C. Testemale. “Generalizing database relational algebra for the treatment of incomplete/uncertain information and vague queries”. Information Sciences, 34, pp. 115-143, 1984.

[Pra84b] H. Prade. “Lipski’s approach to incomplete information databases restated and generalized in the setting of Zadeh’s possibility theory”. Information Systems, 9, pp. 27-42, 1984.

[Pra87a] H. Prade and C. Testemale. “Fuzzy relational databases: representation issues and reduction using similarity measures”. J. Am. Soc. Information Sciences 38(2), pp. 118-126, 1987.

[Pra87b] H. Prade and C. Testemale. “Representation of soft constraints and fuzzy attribute values by means of possibility distributions in databases”. Analysis of fuzzy information, Vol. II: artificial intelligence and decision systems. Ed. J. Bezdek, CRC press, pp. 213-229, 1987.

[Rei78] R. Reiter. “On close world databases”. Logic and databases, H. Gallaire and J. Minker, Eds., Plenum press, pp. 55-76, New York, 1978.

[Rei80] R. Reiter. “Equality and domain closure in first-order databases”. Journal of the ACM, 27(2):235-249, 1980.

[Rei84] R. Reiter. “Towards a logical reconstruction of relational database theory”. Conceptual modeling, Eds., Brodie, Mylopoulos y Schmidt, pp. 193-238, 1984.

[Rob63] J. A. Robinson. “Theorem proving on the computer”. Journal of the ACM, 10(2), pp. 163-74, 1963.

[Rob65a] J. A. Robinson. “Automatic deduction with hyper-resolution”. International journal of the computers mathematics, 1(3), pp. 227-234, 1965.

[Rob65b] J. A. Robinson. “A machine oriented logic based on the resolution principle”. Journal of the ACM, 12(1), pp. 23-41, 1965.

[Rot12] A. Rotundi, V. Della Corte et al (including R. Morales). “State of health during the seven years of Rosetta cruise”. European Planetary Science Congress, Vol. 7 EPSC2012-907, Madrid, España, 2012.

[Rot15] A. Rotundi et al. “Dust measurements in the coma of comet 67P/Churyumov-Gerasimenko inbound to the sun”. Science magazine, Vol 347, Issue 6220.aaa3905-1, 2015.

[Sla67] J.R. Slage. “Automatic theorem proving with renamable and semantic resolution”. Journal of the ACM, 14(4):687-697, 1967.

[Sche98] D. J. Scheeres, F. Marzarib, L. Tomasellab and V. Vanzan. “Rosetta mission: satellite orbits around a cometary nucleus”. Planetary and Space Science, volume 46, Issues 6-7, pages 649-671, 1998.

[Sch99] G. Schwehm and R. Schulz. “Rosetta goes to comet Wirtanen”. Space Science Reviews, vol 90, p. 313-319, 1999.

[Sie15] H. Sierks. C. Barbieri. "On the nucleus structure and activity of comet 67P/Churyumov-Gerasimenko". Science aaa1044, 23 January 2015.

[Smu95] R. M. Smullyan. "First-order logic". Dover publications Inc. New York, USA, 1995.

[Ste86] C. Sterling and E. Shapiro. "The art of Prolog". Ed. MIT-press, 1986.

[Tri79] E. Trillas. "Sobre funciones de negación de la teoría de conjuntos difusos". Stochastica, Vol. 3 nº1, pp. 47-59, 1979.

[Tri80] E. Trillas. "Conjuntos borrosos". Ed. Vicens-Vives, 1980.

[Ull82] J.D. Ullman. "Principles of database systems". Computer Science Press, 2nd edition, 1982.

[Uma80] M. Umano, S. Fukami, M. Muzimoto and H. Tanaka. "Retrieval processing from fuzzy databases". Technical reports of IECE of Japan, Vol. 80, nº204, pp.45-54, L-80-50, 1980.

[Uma82] M. Umano. "Freedom-0: a fuzzy database system". Fuzzy information and decision processes. Eds. M. Gupta, E. Sánchez, North-Holland, Amsterdam, Pub. Compartícula, pp. 339-347, 1982.

[Uma83] M. Umano. "Retrieval from fuzzy database by fuzzy relational algebra". Fuzzy information, knowledge representation and decision analysis. Eds. M. Gupta, E. Sánchez, Pergamon press, New York, pp. 1-6, 1983.

[Urm98] S. Urman. "Oracle 8: Programación PL/SQL". Oracle press & Osborne McGraw-Hill, 1998.

[Vil92] M.A. Vila, J.C. Cubero, J.M. Medina and O. Pons. "A logical approach to fuzzy relational databases". Proc. IPMU, 1992.

[Vil93] M.A. Vila, J.C. Cubero, J.M. Medina and O. Pons. "On the use of logical definition of fuzzy relational database". Proc. 2º IEEE Int. Conf. on fuzzy systems, vol. I, pp. 489-495, San Francisco. E.E.U.U, 1993.

[Vil94] M.A. Vila, J.C. Cubero, J.M. Medina and O. Pons. "Logic and fuzzy relational databases: a new language and a new definition". Fuzzy sets and possibility theory in databases management systems, Bosc and J. Kacprzyk Eds. Physica-Verlag, Heidelberg, Alemania, 1994.

[Vil97] P. Villefranche, J. Evans and F. Faye. "The ESA comet rendezvous mission". Acta astronautica, volume 40, Issue 12, pages 871-877, 1997.

[Win84] P. H. Winston. "Artificial intelligence". Ed. Addison Wesley, 1984.

[Yag80] R.R. Yager. "On a general class of fuzzy connectives". Fuzzy sets and systems, 3, pp. 235-242, 1980.

[Yag82] R.R. Yager. "Fuzzy sets and possibility theory". Ed. R. Yager, Pergamon Press, 1982.

[Zad65] L.A. Zadeh. "Fuzzy sets". Information and Control 8, pp. 338-353, 1965.

[Zad71] L.A. Zadeh. "Similarity relations and fuzzy orderings". Information Sciences, vol. 3, pp. 177-200, 1971.

[Zad75a] L.A. Zadeh. "The concept of linguistic variable and its application to approximate reasoning I". Information sciences, vol. 8, pp. 199-249, 1975.

[Zad75b] L.A. Zadeh. "The concept of linguistic variable and its application to approximate reasoning II". Information sciences, vol. 8, pp. 301-357, 1975.

[Zad76] L.A. Zadeh. "The concept of linguistic variable and its application to approximate reasoning III". Information Sciences, vol. 9, pp. 43-80, 1976.

[Zad78] L.A. Zadeh. "Fuzzy sets as a basis for a theory of possibility". Fuzzy sets and systems 1, pp. 3-28, 1978.

[Zan84] C. Zaniolo "Prolog: a database query language for all seasons". Proc. First workshop on expert database systems, Kiawah, Islandia, 1984.

[Zem84] M. Zemankova-Leech and A. Kandel. "Fuzzy relational databases - a key to expert systems". Köln, Germany, Verlag TÜV Rheinland, 1984.

[Zem85] M. Zemankova-Leech and A. Kandel. "Implementing imprecision in information systems". Information sciences, 37, pp. 107-141, 1985.

10 Apéndices

El lector podrá encontrar a continuación una lista de apéndices, cuyo fin es el de ampliar ciertos puntos de esta memoria: 1), 2), o bien proporcionar una introducción a los cometas y en particular al cometa bajo estudio: 3) y 4).

- 1) Apéndice A. Declaración sintáctica y semántica de la base de datos relacional y difusa.
- 2) Apéndice B. Contenido del DVD.
- 3) Apéndice C. Introducción a los cometas.
- 4) Apéndice D. El cometa 67P/Churyumov-Gerasimenko.

10.1 Apéndice A. Declaración sintáctica y semántica de la base de datos relacional y difusa

Se declaran a continuación la sintaxis que define las tablas relacionales de G. utilizando el SQL de Oracle©. Los campos con componente difusa se resaltarán en color rojo. En esta memoria sólo se considerará el tratamiento difuso en la magnitud de “tiempo” mediante *datos difusos de tipo 1*.

Se incluye así mismo, la semántica de cada uno de los campos de las tablas mediante la siguiente estructura:

$$C (m, M) [t]. D$$

Donde C es el nombre del campo, m es el valor mínimo que puede tomar el campo, M el valor máximo, t es la curva de calibración (en caso de que exista) y D su descripción. El texto se encuentra en inglés para que sirva como manual de referencia para usuarios de GDB-GUI miembros del consorcio internacional de desarrollo de GIADA.

Las tablas y su composición aparecen un orden alfabético:

- 1) GDB_CALIBRATIONHK
- 2) GDB_GDS_ISEVENT
- 3) GDB_GDSCALIBRATION
- 4) GDB_GDSEVENT
- 5) GDB_HK
- 6) GDB_ISCALIBRATION
- 7) GDB_ISCALIBRATIONEVENT
- 8) GDB_ISEVENT
- 9) GDB_MBHEATING
- 10) GDB_MBHEATINGEVENT
- 11) GDB_MBREADINGEVENT
- 12) GDB_MBSCALIBRATION
- 13) GDB_PACKET
- 14) GDB_SCIENCEHK
- 15) GDB_WORKSESSION

GDB_CALIBRATIONHK

This table stores all the common calibration information of GDS, IS and MBS.

SQL definition

```
CREATE TABLE GDB_CalibrationHK(  
ID NUMBER(16) NOT NULL PRIMARY KEY,  
  
cal_hk_eLabel FTYPE1(8,9) NUMBER(5) NOT NULL Constraint cal_hk_HK_eLabel  
CHECK(cal_hk_eLabel=17223 or cal_hk_eLabel=17225 or cal_hk_eLabel=17229),  
  
cal_hk_eventTime FTYPE1(8,9) NUMBER(15) NOT NULL Constraint  
cal_hk_HK_eventTime CHECK(cal_hk_eventTime>=0 and  
cal_hk_eventTime<=281474976710655),  
  
cal_hk_adcTemperature NUMBER(5) NOT NULL Constraint cal_hk_HK_adcTemperature  
CHECK(cal_hk_adcTemperature>=4095 or cal_hk_adcTemperature<=4095),  
  
cal_hk_calibrationVoltage0 NUMBER(5) NOT NULL Constraint  
cal_hk_HK_calibrationVol0 CHECK(cal_hk_calibrationVoltage0>=4095 or  
cal_hk_calibrationVoltage0<=4095),  
  
cal_hk_calibrationVoltage1 NUMBER(5) NOT NULL Constraint  
cal_hk_HK_calibrationVol1 CHECK(cal_hk_calibrationVoltage1>=4095 or  
cal_hk_calibrationVoltage1<=4095),  
  
cal_hk_calibrationVoltage2 NUMBER(5) NOT NULL Constraint  
cal_hk_HK_calibrationVol2 CHECK(cal_hk_calibrationVoltage2>=4095 or  
cal_hk_calibrationVoltage2<=4095),  
  
cal_hk_calibrationVoltage3 NUMBER(5) NOT NULL Constraint  
cal_hk_HK_calibrationVol3 CHECK(cal_hk_calibrationVoltage3>=4095 or  
cal_hk_calibrationVoltage3<=4095),  
  
FOREIGN KEY (ID) REFERENCES GDB_PACKET(ID) ON DELETE CASCADE  
);
```

Semantic field

ID(0,9999999999999999). Unique Identifier. Milisecond (the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC) when the packet was inserted into the system.

cal_hk_eLabel (17223,17229). Calibration label: 0x4347 for GDS Calibration, 0x4349 for IS Calibration, 0x434D for MBS Calibration.

cal_eventTime (0,281474976710655). GIADA Time when the calibration was started.

cal_hk_adcTemperature (0,4095) [ADC Temperature]. Temperature of Analogue to Digital Converter in ADC counts.

cal_hk_calibrationVoltage0 (0,4095) [Calibration Voltage V0]. Calibration Voltage 1 in ADC counts.

cal_hk_calibrationVoltage1 (0,4095) [Calibration Voltage V1]. Calibration Voltage 2 in ADC counts.

cal_hk_calibrationVoltage2 (0,4095) [Calibration Voltage V2]. Calibration Voltage 3 in ADC counts.

cal_hk_calibrationVoltage3 (0,4095) [Calibration Voltage V3]. Calibration Voltage 4 in ADC counts.

GDB_GDS_ISEVENT

This table stores the events detected by the IS.

SQL definition

```
CREATE TABLE GDB_GDS_ISEvent (
  ID NUMBER(16) NOT NULL,
  GDS_IS_ITEM NUMBER(2) NOT NULL Constraint GDS_IS_ITEM CHECK(GDS_IS_ITEM>=0),
  GDS_IS_eLabel NUMBER(5) NOT NULL Constraint GDS_IS_eLabel
  CHECK(GDS_IS_eLabel=26468 or GDS_IS_eLabel=26483),
  GDS_IS_eventTime FTYPE1(8,9) NUMBER(15) NOT NULL Constraint
  GDS_IS_packetTime CHECK(GDS_IS_eventTime>=0 and
  GDS_IS_eventTime<=281474976710655),
  GDS_IS_scatteredLight NUMBER(5) NOT NULL Constraint GDS_IS_scatteredLight
  CHECK(GDS_IS_scatteredLight>=0 and GDS_IS_scatteredLight<=4095),
  GDS_IS_crossingTimeOverflow NUMBER(1) NOT NULL Constraint
  GDS_IS_crossingTimeOverflow CHECK(GDS_IS_crossingTimeOverflow=0 or
  GDS_IS_crossingTimeOverflow=1),
  GDS_IS_crossingTime FTYPE1(2,3) NUMBER(5) NOT NULL Constraint
  GDS_IS_GDS_IScrossingTime CHECK(GDS_IS_crossingTime>=0 and
  GDS_IS_crossingTime<=1023),
  GDS_IS_flightTimeOverflow NUMBER(1) NOT NULL Constraint
  GDS_IS_flightTimeOverflow CHECK(GDS_IS_flightTimeOverflow=0 or
  GDS_IS_flightTimeOverflow=1),
  GDS_IS_flightTime FTYPE1(2,3) NUMBER(5) NOT NULL Constraint
  GDS_IS_flightTime CHECK(GDS_IS_flightTime>=0 and GDS_IS_flightTime<=32767),
  GDS_IS_pzt_A_detected NUMBER(1) NOT NULL Constraint GDS_IS_pzt_A_detected
  CHECK(GDS_IS_pzt_A_detected=0 or GDS_IS_pzt_A_detected=1),
  GDS_IS_pzt_A_range NUMBER(1) NOT NULL Constraint GDS_IS_pzt_A_range
  CHECK(GDS_IS_pzt_A_range=0 or GDS_IS_pzt_A_range=1),
  GDS_IS_pzt_A_gain NUMBER(1) NOT NULL Constraint GDS_IS_pzt_A_gain
  CHECK(GDS_IS_pzt_A_gain=0 or GDS_IS_pzt_A_gain=1),
  GDS_IS_pzt_A_amplitude NUMBER(5) NOT NULL Constraint GDS_IS_pzt_A_amplitude
  CHECK(GDS_IS_pzt_A_amplitude>=0 and GDS_IS_pzt_A_amplitude<=4095),
  GDS_IS_pzt_A_timeOverflow NUMBER(1) NOT NULL Constraint
  GDS_IS_pzt_A_timeOverflow CHECK(GDS_IS_pzt_A_timeOverflow=0 or
  GDS_IS_pzt_A_timeOverflow=1),
```

```
GDS_IS_pzt_A_time FTYPE1(8,9) NUMBER(2) NOT NULL Constraint
GDS_IS_pzt_A_time CHECK(GDS_IS_pzt_A_time>=0 and GDS_IS_pzt_A_time<=31),

GDS_IS_pzt_B_detected NUMBER(1) NOT NULL Constraint GDS_IS_pzt_B_detected
CHECK(GDS_IS_pzt_B_detected=0 or GDS_IS_pzt_B_detected=1),

GDS_IS_pzt_B_range NUMBER(1) NOT NULL Constraint GDS_IS_pzt_B_range
CHECK(GDS_IS_pzt_B_range=0 or GDS_IS_pzt_B_range=1),

GDS_IS_pzt_B_gain NUMBER(1) NOT NULL Constraint GDS_IS_pzt_B_gain
CHECK(GDS_IS_pzt_B_gain=0 or GDS_IS_pzt_B_gain=1),

GDS_IS_pzt_B_amplitude NUMBER(5) NOT NULL Constraint GDS_IS_pzt_B_amplitude
CHECK(GDS_IS_pzt_B_amplitude>=0 and GDS_IS_pzt_B_amplitude<=4095),

GDS_IS_pzt_B_timeOverflow NUMBER(1) NOT NULL Constraint
GDS_IS_pzt_B_timeOverflow CHECK(GDS_IS_pzt_B_timeOverflow=0 or
GDS_IS_pzt_B_timeOverflow=1),

GDS_IS_pzt_B_time FTYPE1(8,9) NUMBER(2) NOT NULL Constraint
GDS_IS_pzt_B_time CHECK(GDS_IS_pzt_B_time>=0 and GDS_IS_pzt_B_time<=31),

GDS_IS_pzt_C_detected NUMBER(1) NOT NULL Constraint GDS_IS_pzt_C_detected
CHECK(GDS_IS_pzt_C_detected=0 or GDS_IS_pzt_C_detected=1),

GDS_IS_pzt_C_range NUMBER(1) NOT NULL Constraint GDS_IS_pzt_C_range
CHECK(GDS_IS_pzt_C_range=0 or GDS_IS_pzt_C_range=1),

GDS_IS_pzt_C_gain NUMBER(1) NOT NULL Constraint GDS_IS_pzt_C_gain
CHECK(GDS_IS_pzt_C_gain=0 or GDS_IS_pzt_C_gain=1),

GDS_IS_pzt_C_amplitude NUMBER(5) NOT NULL Constraint GDS_IS_pzt_C_amplitude
CHECK(GDS_IS_pzt_C_amplitude>=0 and GDS_IS_pzt_C_amplitude<=4095),

GDS_IS_pzt_C_timeOverflow NUMBER(1) NOT NULL Constraint
GDS_IS_pzt_C_timeOverflow CHECK(GDS_IS_pzt_C_timeOverflow=0 or
GDS_IS_pzt_C_timeOverflow=1),

GDS_IS_pzt_C_time FTYPE1(8,9) NUMBER(2) NOT NULL Constraint
GDS_IS_pzt_C_time CHECK(GDS_IS_pzt_C_time>=0 and GDS_IS_pzt_C_time<=31),

GDS_IS_pzt_D_detected NUMBER(1) NOT NULL Constraint GDS_IS_pzt_D_detected
CHECK(GDS_IS_pzt_D_detected=0 or GDS_IS_pzt_D_detected=1),

GDS_IS_pzt_D_range NUMBER(1) NOT NULL Constraint GDS_IS_pzt_D_range
CHECK(GDS_IS_pzt_D_range=0 or GDS_IS_pzt_D_range=1),

GDS_IS_pzt_D_gain NUMBER(1) NOT NULL Constraint GDS_IS_pzt_D_gain
CHECK(GDS_IS_pzt_D_gain=0 or GDS_IS_pzt_D_gain=1),

GDS_IS_pzt_D_amplitude NUMBER(5) NOT NULL Constraint GDS_IS_pzt_D_amplitude
CHECK(GDS_IS_pzt_D_amplitude>=0 and GDS_IS_pzt_D_amplitude<=4095),

GDS_IS_pzt_D_timeOverflow NUMBER(1) NOT NULL Constraint
GDS_IS_pzt_D_timeOverflow CHECK(GDS_IS_pzt_D_timeOverflow=0 or
GDS_IS_pzt_D_timeOverflow=1),

GDS_IS_pzt_D_time FTYPE1(8,9) NUMBER(2) NOT NULL Constraint
GDS_IS_pzt_D_time CHECK(GDS_IS_pzt_D_time>=0 and GDS_IS_pzt_D_time<=31),

GDS_IS_pzt_E_detected NUMBER(1) NOT NULL Constraint GDS_IS_pzt_E_detected
CHECK(GDS_IS_pzt_E_detected=0 or GDS_IS_pzt_E_detected=1),

GDS_IS_pzt_E_range NUMBER(1) NOT NULL Constraint GDS_IS_pzt_E_range
CHECK(GDS_IS_pzt_E_range=0 or GDS_IS_pzt_E_range=1),

GDS_IS_pzt_E_gain NUMBER(1) NOT NULL Constraint GDS_IS_pzt_E_gain
CHECK(GDS_IS_pzt_E_gain=0 or GDS_IS_pzt_E_gain=1),
```


GDS_IS_pzt_A_gain (0,1). If value='1' this magnitude multiply the total value of amplitude, else not modify this value This value can be set to '0' due to autogain function (RO-GIA-IAA-RS-001 Req 1400).

GDS_IS_pzt_A_amplitude (0,4095) [PZT#A Amplitude]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

GDS_IS_pzt_A_timeOverflow (0,1). If this value='0' means one of following situations: No signal threshold transition (signal under threshold after impact or signal remains over the threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

GDS_IS_pzt_A_time (0,31) [PZT#A Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation needs to be detected starting from first PZT detection. The max value without overflow is '30'.

GDS_IS_pzt_B_detected (0,1). If value='1' then this PZT has detected a particle (threshold has been crossed from below threshold to over threshold)If value='0' (there is not threshold transition) then all the information relates with this PZT is useless. Related field: "pzt_B_time-Overflow".

GDS_IS_pzt_B_range(0,1). If value='1' this magnitude divide the total value of amplitude.

GDS_IS_pzt_B_gain (0,0). Always 0.

GDS_IS_pzt_B_amplitude (0,4095) [PZT#B Amplitude]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

GDS_IS_pzt_B_timeOverflow (0,1). If this value='0' means one of following situations: No signal threshold transition (signals under threshold after impact or signal remains over the threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

GDS_IS_pzt_B_time (0,31) [PZT#B Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation needs to be detected starting from first PZT detection. The max value without overflow is '30'.

GDS_IS_pzt_C_detected (0,1). If value='1' then this PZT has detected a particle (threshold has been crossed from below threshold to over threshold)If value='0' (there is not threshold transition) then all the information relates with this PZT is useless. Related field: "pzt_C_time-Overflow".

GDS_IS_pzt_C_range (0,1). If value='1' this magnitude divide the total value of amplitude.

GDS_IS_pzt_C_gain (0,0). Always 0.

GDS_IS_pzt_C_amplitude (0,4095) [PZT#C Amplitude]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

GDS_IS_pzt_C_timeOverflow(0,1). If this value='0' means one of following situations: No signal threshold transition (signal under threshold after impact or signal remains over the

threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

GDS_IS_pzt_C_time (0,31) [PZT#C Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation needs to be detected starting from first PZT detection. The max value without overflow is '30'.

GDS_IS_pzt_D_detected (0,1). If value='1' then this PZT has detected a particle (threshold has been crossed from below threshold to over threshold) If value='0' (there is not threshold transition) then all the information relates with this PZT is useless. Related field: "pzt_D_timeOverflow".

GDS_IS_pzt_D_range (0,1). If value='1' this magnitude divide the total value of amplitude.

GDS_IS_pzt_D_gain (0,1). If value='1' this magnitude multiply the total value of amplitude, else not modify this value This value can be set to '0' due to autogain function (RO-GIA-IAA-RS-001 Req 1400).

GDS_IS_pzt_D_amplitude(0,4095) [PZT#D Amplitude]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

GDS_IS_pzt_D_timeOverflow (0,1). If this value='0' means one of following situations: No signal threshold transition (signal under threshold after impact or signal remains over the threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

GDS_IS_pzt_D_time (0,31) [PZT#D Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation needs to be detected starting from first PZT detection. The max value without overflow is '30'.

GDS_IS_pzt_E_detected (0,1). If value='1' then this PZT has detected a particle (threshold has been crossed from below threshold to over threshold) If value='0' (there is not threshold transition) then all the information relates with this PZT is useless. Related field: "pzt_E_timeOverflow".

GDS_IS_pzt_E_range(0,1). If value='1' this magnitude divide the total value of amplitude.

GDS_IS_pzt_E_gain(0,1). If value='1' this magnitude multiply the total value of amplitude, else not modify this value This value can be set to '0' due to autogain function (RO-GIA-IAA-RS-001 Req 1400).

GDS_IS_pzt_E_amplitude (0,4095) [PZT#E Amplitude]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

GDS_IS_pzt_E_timeOverflow(0,1). If this value='0' means one of following situations: No signal threshold transition (signal under threshold after impact or signal remains over the threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

GDS_IS_pzt_E_time(0,31) [PZT#E Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation

needs to be detected starting from first PZT detection. The max value without overflow is '30'.

GDB_GDSCALIBRATION

This table stores the information about the GDS calibration.

SQL definition

```
CREATE TABLE GDB_GDSCalibration(
  ID NUMBER(16) PRIMARY KEY,
  GDS_cal_laser_1_Temperature NUMBER(5) NOT NULL Constraint
  GDS_cal_laser_1_Temperature CHECK(GDS_cal_laser_1_Temperature>=0 and
  GDS_cal_laser_1_Temperature<=4095),

  GDS_cal_laser_2_Temperature NUMBER(5) NOT NULL Constraint
  GDS_cal_laser_2_Temperature CHECK(GDS_cal_laser_2_Temperature>=0 and
  GDS_cal_laser_2_Temperature<=4095),

  GDS_cal_laser_3_Temperature NUMBER(5) NOT NULL Constraint
  GDS_cal_laser_3_Temperature CHECK(GDS_cal_laser_3_Temperature>=0 and
  GDS_cal_laser_3_Temperature<=4095),

  GDS_cal_laser_4_Temperature NUMBER(5) NOT NULL Constraint
  GDS_cal_laser_4_Temperature CHECK(GDS_cal_laser_4_Temperature>=0 and
  GDS_cal_laser_4_Temperature<=4095),

  GDS_cal_laser_1_Light NUMBER(5) NOT NULL Constraint GDS_cal_laser_1_Light
  CHECK(GDS_cal_laser_1_Light>=0 and GDS_cal_laser_1_Light<=4095),

  GDS_cal_laser_2_Light NUMBER(5) NOT NULL Constraint GDS_cal_laser_2_Light
  CHECK(GDS_cal_laser_2_Light>=0 and GDS_cal_laser_2_Light<=4095),

  GDS_cal_laser_3_Light NUMBER(5) NOT NULL Constraint GDS_cal_laser_3_Light
  CHECK(GDS_cal_laser_3_Light>=0 and GDS_cal_laser_3_Light<=4095),

  GDS_cal_laser_4_Light NUMBER(5) NOT NULL Constraint GDS_cal_laser_4_Light
  CHECK(GDS_cal_laser_4_Light>=0 and GDS_cal_laser_4_Light<=4095),

  GDS_cal_sumLetft NUMBER(10) NOT NULL Constraint GDS_cal_sumLetft
  CHECK(GDS_cal_sumLetft>=0 and GDS_cal_sumLetft<=4294967295),

  GDS_cal_squareSumLeft NUMBER(10) NOT NULL Constraint
  GDS_cal_CAL_squareSumLeft CHECK(GDS_cal_squareSumLeft>=0 and
  GDS_cal_squareSumLeft<=4294967295),

  GDS_cal_sumRight NUMBER(10) NOT NULL Constraint GDS_cal_CAL_sumRight
  CHECK(GDS_cal_sumRight>=0 and GDS_cal_sumRight<=4294967295),

  GDS_cal_squareSumRight NUMBER(10) NOT NULL Constraint
  GDS_cal_CAL_squareSumRight CHECK(GDS_cal_squareSumRight>=0 and
  GDS_cal_squareSumRight<=4294967295),

  FOREIGN KEY (ID) REFERENCES GDB_CalibrationHK(ID) ON DELETE CASCADE
);
```

Semantic field

ID(0,999999999999). Unique Identifier. Milisecond (the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC) when the packet was inserted into the system.

GDS_cal_laser_1_Temperature(0,4095) [GDS Laser 1 Temperature]. Temperature of the Laser 1 Ignore this value if GDS is off.

GDS_cal_laser_2_Temperature (0,4095) [GDS Laser 2 Temperature]. Temperature of the Laser 2 Ignore this value if GDS is off.

GDS_cal_laser_3_Temperature(0,4095) [GDS Laser 3 Temperature]. Temperature of the Laser 3 Ignore this value if GDS is off.

GDS_cal_laser_4_Temperature(0,4095) [GDS Laser 4 Temperature]. Temperature of the Laser 4 Ignore this value if GDS is off.

GDS_cal_laser_1_Light(0,4095) [GDS Laser 1 Light Monitor]. Light of the Laser 1 Ignore this value if GDS is off .

GDS_cal_laser_2_Light (0,4095) [GDS Laser 2 Light Monitor]. Light of the Laser 2 Ignore this value if GDS is off.

GDS_cal_laser_3_Light (0,4095) [GDS Laser 3 Light Monitor]. Light of the Laser 3 Ignore this value if GDS is off.

GDS_cal_laser_4_Light (0,4095) [GDS Laser 4 Light Monitor]. Light of the Laser 4 Ignore this value if GDS is off.

GDS_cal_sumLeft (0,4294967295) [GDS Sum Left]. Sum of 8 means, each mean take 4 measures over GDS left channel. This channel previously has been disabled and enabled the peak detector. At the end of measures, the channel is enabled again and the laser is reset. The idea is to measure the noise in this channel.

GDS_cal_squareSumLeft (0,4294967295) [GDS Noise Left]. Sum of the squares of 8 means, each mean take 4 measures over GDS left channel. This channel previously has been disabled and enabled the peak detector. At the end of measures, the channel is enabled again and the laser is reset. The idea is to measure the noise in this channel.

GDS_cal_sumRight (0,4294967295) [GDS Sum Right]. Sum of 8 means, each mean take 4 measures over GDS right channel. This channel previously has been disabled and enabled the peak detector. At the end of measures, the channel is enabled again and the laser is reset. The idea is to measure the noise in this channel.

GDS_cal_squareRight (0,4294967295) [GDS Noise Right]. Sum of the squares of 8 means, each mean take 4 measures over GDS right channel. This channel previously has been disabled and enabled the peak detector. At the end of measures, the channel is enabled again and the laser is reset. The idea is to measure the noise in this channel.

GDB_GDSEVENT

This table stores the events detected by the GDS.

SQL definition

```
CREATE TABLE GDB_GDSEvent(
  ID NUMBER(16) NOT NULL,

  GDS_ITEM NUMBER(2) NOT NULL Constraint GDS_ITEM CHECK(GDS_ITEM>=0),

  GDS_eLabel NUMBER(5) NOT NULL Constraint GDS_eLabel CHECK(GDS_eLabel=26476
  or GDS_eLabel=26482),

  GDS_eventTime FTYPE1(8,9) NUMBER(15) NOT NULL Constraint GDS_packetTime
  CHECK(GDS_eventTime>=0 and GDS_eventTime<=281474976710655),

  GDS_scatteredLight NUMBER(5) NOT NULL Constraint GDS_scatteredLight
  CHECK(GDS_scatteredLight>=0 and GDS_scatteredLight<=4095),

  GDS_crossingTimeOverflow NUMBER(1) NOT NULL Constraint
  GDS_crossingTimeOverflow CHECK(GDS_crossingTimeOverflow=0 or
  GDS_crossingTimeOverflow=1),

  GDS_crossingTime FTYPE1(8,9)NUMBER(5) NOT NULL Constraint GDS_crossingTime
  CHECK(GDS_crossingTime>=0 and GDS_crossingTime<=1023),

  PRIMARY KEY (ID,GDS_ITEM),
  FOREIGN KEY (ID) REFERENCES GDB_ScienceHK(ID) ON DELETE CASCADE
);
```

Semantic field

ID(0,999999999999). Unique Identifier. Millisecond (the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC) when the packet was inserted into the system

GDS_ITEM (0,99). Event position inside Science TM.

GDS_eLabel (26476,26482). Identify the channel that has detected correctly (the detected signal is over the threshold) a particle. For Left channel value='0x676C', for Right channel value='0x6772'. The GDS must be on and Lasers must be armed and ON for GDS detection and GIADA must be in Normal mode.

GDS_eventTime (0,281474976710655). GDS event time stamp GIADA precision is 4ms and the GDS event detection routine speeds 1.62 ms (in mean), so it's possible to have two GDS events with the same eventTime. This is true for double GDS events (but not only): the particle can be detected by the two channels at the same time.

GDS_scatteredLight (0,4095) [Scattered Light]. GIADA measures the peak value of the Light fuzzy reflection in 90°.

GDS_crossingTimeOverflow (0,1). The particle is slower than 3 mm/10 ms = 3 m/s to cross the laser curtain.

GDS_crossingTime (0,1023) [Crossing Time]. Measure the velocity of the particle in the laser curtain. Max value without overflow '1022'. Each unit corresponds to 10 μs.

GDB_HK

This table stores all the information about the status of the instrument.

SQL definition

```
CREATE TABLE GDB_HK (
  ID NUMBER(16) PRIMARY KEY,

  HK_hkPad NUMBER(3) NOT NULL Constraint HK_hkPad CHECK(HK_hkPad=0),

  HK_sid NUMBER(3) NOT NULL Constraint HK_sid CHECK(HK_sid=1),
  HK_dustFlux NUMBER(5) NOT NULL Constraint HK_dustFlux CHECK(HK_dustFlux>=0
  and HK_dustFlux<=65535),

  HK_operationMode NUMBER(1) NOT NULL Constraint HK_operationMode
  CHECK(HK_operationMode>=0 and HK_operationMode<=3),

  HK_eventTMOverflow NUMBER(1) NOT NULL Constraint HK_eventTMOverflow
  CHECK(HK_eventTMOverflow=0 or HK_eventTMOverflow=1),

  HK_scienceTMOverflow NUMBER(1) NOT NULL Constraint HK_scienceTMOverflow
  CHECK(HK_scienceTMOverflow=0 or HK_scienceTMOverflow=1),

  HK_scienceTMEnabled NUMBER(1) NOT NULL Constraint HK_scienceTMEnabled
  CHECK(HK_scienceTMEnabled=0 or HK_scienceTMEnabled=1),

  HK_frangiboltArmed NUMBER(1) NOT NULL Constraint HK_frangiboltArmed
  CHECK(HK_frangiboltArmed=0 or HK_frangiboltArmed=1),

  HK_coverArmed NUMBER(1) NOT NULL Constraint HK_coverArmed
  CHECK(HK_coverArmed=0 or HK_coverArmed=1),

  HK_openReedSwitch NUMBER(1) NOT NULL Constraint HK_openReedSwitch
  CHECK(HK_openReedSwitch=0 or HK_openReedSwitch=1),

  HK_closedReedSwitch NUMBER(1) NOT NULL Constraint HK_closedReedSwitch
  CHECK(HK_closedReedSwitch=0 or HK_closedReedSwitch=1),

  HK_coverHeaterEnabled NUMBER(1) NOT NULL Constraint HK_coverHeaterEnabled
  CHECK(HK_coverHeaterEnabled=0 or HK_coverHeaterEnabled=1),

  HK_motorHeaterEnabled NUMBER(1) NOT NULL Constraint HK_motorHeaterEnabled
  CHECK(HK_motorHeaterEnabled=0 or HK_motorHeaterEnabled=1),

  HK_frangiboltHeaterEnabled NUMBER(1) NOT NULL Constraint HK_frangiboltHeater
  CHECK(HK_frangiboltHeaterEnabled=0 or HK_frangiboltHeaterEnabled=1),

  HK_frangiboltTemperature NUMBER(5) NOT NULL Constraint
  HK_frangiboltTemperature CHECK(HK_frangiboltTemperature>=0 and
  HK_frangiboltTemperature<=4095),

  HK_gdsON NUMBER(1) NOT NULL Constraint HK_gdsON CHECK(HK_gdsON=0 or
  HK_gdsON=1),

  HK_laserArmed NUMBER(1) NOT NULL Constraint HK_laserArmed
  CHECK(HK_laserArmed=0 or HK_laserArmed=1),

  HK_laserPower NUMBER(1) NOT NULL Constraint HK_laserPower
  CHECK(HK_laserPower>=0 and HK_laserPower<=3),

  HK_laserMode NUMBER(1) NOT NULL Constraint HK_laserMode
  CHECK(HK_laserMode>=0 and HK_laserMode<=3),
```

```
HK_receiverLeftEnabled NUMBER(1) NOT NULL Constraint HK_receiverLeftEnabled  
CHECK(HK_receiverLeftEnabled=0 or HK_receiverLeftEnabled=1),
```

```
HK_receiverRightEnabled NUMBER(1) NOT NULL Constraint  
HK_receiverRightEnabled CHECK(HK_receiverRightEnabled=0 or  
HK_receiverRightEnabled=1),
```

```
HK_laserON NUMBER(1) NOT NULL Constraint HK_laserON CHECK(HK_laserON=0 or  
HK_laserON=1),
```

```
HK_detections NUMBER(1) NOT NULL Constraint HK_detections  
CHECK(HK_detections>=0 and HK_detections<=3),
```

```
HK_laser_1_Temperature NUMBER(5) NOT NULL Constraint HK_laser_1_Temperature  
CHECK(HK_laser_1_Temperature>=0 and HK_laser_1_Temperature<=4095),
```

```
HK_laser_2_Temperature NUMBER(5) NOT NULL Constraint HK_laser_2_Temperature  
CHECK(HK_laser_2_Temperature>=0 and HK_laser_2_Temperature<=4095),
```

```
HK_laser_3_Temperature NUMBER(5) NOT NULL Constraint HK_laser_3_Temperature  
CHECK(HK_laser_3_Temperature>=0 and HK_laser_3_Temperature<=4095),
```

```
HK_laser_4_Temperature NUMBER(5) NOT NULL Constraint HK_laser_4_Temperature  
CHECK(HK_laser_4_Temperature>=0 and HK_laser_4_Temperature<=4095),
```

```
HK_laser_1_Light NUMBER(5) NOT NULL Constraint HK_laser_1_Light  
CHECK(HK_laser_1_Light>=0 and HK_laser_1_Light<=4095),
```

```
HK_laser_2_Light NUMBER(5) NOT NULL Constraint HK_laser_2_Light  
CHECK(HK_laser_2_Light>=0 and HK_laser_2_Light<=4095),
```

```
HK_laser_3_Light NUMBER(5) NOT NULL Constraint HK_laser_3_Light  
CHECK(HK_laser_3_Light>=0 and HK_laser_3_Light<=4095),
```

```
HK_laser_4_Light NUMBER(5) NOT NULL Constraint HK_laser_4_Light  
CHECK(HK_laser_4_Light>=0 and HK_laser_4_Light<=4095),
```

```
HK_gdsTheresholdLeft NUMBER(3) NOT NULL Constraint HK_gdsTheresholdLeft  
CHECK(HK_gdsTheresholdLeft>=0 and HK_gdsTheresholdLeft<=255),
```

```
HK_gdsTheresholdRight NUMBER(3) NOT NULL Constraint HK_gdsTheresholdRight  
CHECK(HK_gdsTheresholdRight>=0 and HK_gdsTheresholdRight<=255),
```

```
HK_isON NUMBER(1) NOT NULL Constraint HK_isON CHECK(HK_isON=0 or HK_isON=1),
```

```
HK_is_PZT_E_Enabled NUMBER(1) NOT NULL Constraint HK_is_PZT_E_Enabled  
CHECK(HK_is_PZT_E_Enabled=0 or HK_is_PZT_E_Enabled=1),
```

```
HK_is_PZT_D_Enabled NUMBER(1) NOT NULL Constraint HK_is_PZT_D_Enabled  
CHECK(HK_is_PZT_D_Enabled=0 or HK_is_PZT_D_Enabled=1),
```

```
HK_is_PZT_C_Enabled NUMBER(1) NOT NULL Constraint HK_is_PZT_C_Enabled  
CHECK(HK_is_PZT_C_Enabled=0 or HK_is_PZT_C_Enabled=1),
```

```
HK_is_PZT_B_Enabled NUMBER(1) NOT NULL Constraint HK_is_PZT_B_Enabled  
CHECK(HK_is_PZT_B_Enabled=0 or HK_is_PZT_B_Enabled=1),
```

```
HK_is_PZT_A_Enabled NUMBER(1) NOT NULL Constraint HK_is_PZT_A_Enabled  
CHECK(HK_is_PZT_A_Enabled=0 or HK_is_PZT_A_Enabled=1),
```

```
HK_isRange NUMBER(1) NOT NULL Constraint HK_isRange CHECK(HK_isRange=0 or  
HK_isRange=1),
```

```
HK_is_PZT_E_Gain NUMBER(1) NOT NULL Constraint HK_is_PZT_E_Gain  
CHECK(HK_is_PZT_E_Gain=0 or HK_is_PZT_E_Gain=1),
```

```
HK_is_PZT_D_Gain NUMBER(1) NOT NULL Constraint HK_is_PZT_D_Gain
CHECK(HK_is_PZT_D_Gain=0 or HK_is_PZT_D_Gain=1),

HK_is_PZT_A_Gain NUMBER(1) NOT NULL Constraint HK_is_PZT_A_Gain
CHECK(HK_is_PZT_A_Gain=0 or HK_is_PZT_A_Gain=1),

HK_isTemperature NUMBER(5) NOT NULL Constraint HK_isTemperature
CHECK(HK_isTemperature>=0 and HK_isTemperature<=4095),

HK_isThereshold_PZT_A NUMBER(3) NOT NULL Constraint HK_isThereshold_PZT_A
CHECK(HK_isThereshold_PZT_A>=0 and HK_isThereshold_PZT_A<=255),

HK_isThereshold_PZT_B NUMBER(3) NOT NULL Constraint HK_isThereshold_PZT_B
CHECK(HK_isThereshold_PZT_B>=0 and HK_isThereshold_PZT_B<=255),

HK_isThereshold_PZT_C NUMBER(3) NOT NULL Constraint HK_isThereshold_PZT_C
CHECK(HK_isThereshold_PZT_C>=0 and HK_isThereshold_PZT_C<=255),

HK_isThereshold_PZT_D NUMBER(3) NOT NULL Constraint HK_isThereshold_PZT_D
CHECK(HK_isThereshold_PZT_D>=0 and HK_isThereshold_PZT_D<=255),

HK_isThereshold_PZT_E NUMBER(3) NOT NULL Constraint HK_isThereshold_PZT_E
CHECK(HK_isThereshold_PZT_E>=0 and HK_isThereshold_PZT_E<=255),

HK_mbsON NUMBER(1) NOT NULL Constraint HK_mbsON CHECK(HK_mbsON=0 or
HK_mbsON=1),

HK_mb_5_Enabled NUMBER(1) NOT NULL Constraint HK_mb_5_Enabled
CHECK(HK_mb_5_Enabled=0 or HK_mb_5_Enabled=1),

HK_mb_4_Enabled NUMBER(1) NOT NULL Constraint HK_mb_4_Enabled
CHECK(HK_mb_4_Enabled=0 or HK_mb_4_Enabled=1),

HK_mb_3_Enabled NUMBER(1) NOT NULL Constraint HK_mb_3_Enabled
CHECK(HK_mb_3_Enabled=0 or HK_mb_3_Enabled=1),

HK_mb_2_Enabled NUMBER(1) NOT NULL Constraint HK_mb_2_Enabled
CHECK(HK_mb_2_Enabled=0 or HK_mb_2_Enabled=1),

HK_mb_1_Enabled NUMBER(1) NOT NULL Constraint HK_mb_1_Enabled
CHECK(HK_mb_1_Enabled=0 or HK_mb_1_Enabled=1),

HK_mbHeater_5_Enabled NUMBER(1) NOT NULL Constraint HK_mbHeater_5_Enabled
CHECK(HK_mbHeater_5_Enabled=0 or HK_mbHeater_5_Enabled=1),

HK_mbHeater_4_Enabled NUMBER(1) NOT NULL Constraint HK_mbHeater_4_Enabled
CHECK(HK_mbHeater_4_Enabled=0 or HK_mbHeater_4_Enabled=1),

HK_mbHeater_3_Enabled NUMBER(1) NOT NULL Constraint HK_mbHeater_3_Enabled
CHECK(HK_mbHeater_3_Enabled=0 or HK_mbHeater_3_Enabled=1),

HK_mbHeater_2_Enabled NUMBER(1) NOT NULL Constraint HK_mbHeater_2_Enabled
CHECK(HK_mbHeater_2_Enabled=0 or HK_mbHeater_2_Enabled=1),

HK_mbHeater_1_Enabled NUMBER(1) NOT NULL Constraint HK_mbHeater_1_Enabled
CHECK(HK_mbHeater_1_Enabled=0 or HK_mbHeater_1_Enabled=1),

HK_mb_1_Temperature NUMBER(5) NOT NULL Constraint HK_mb_1_Temperature
CHECK(HK_mb_1_Temperature>=0 and HK_mb_1_Temperature<=4095),

HK_mb_2_Temperature NUMBER(5) NOT NULL Constraint HK_mb_2_Temperature
CHECK(HK_mb_2_Temperature>=0 and HK_mb_2_Temperature<=4095),

HK_mb_3_Temperature NUMBER(5) NOT NULL Constraint HK_mb_3_Temperature
CHECK(HK_mb_3_Temperature>=0 and HK_mb_3_Temperature<=4095),

HK_mb_4_Temperature NUMBER(5) NOT NULL Constraint HK_mb_4_Temperature
CHECK(HK_mb_4_Temperature>=0 and HK_mb_4_Temperature<=4095),
```


HK_scienceTMOverflow(0,1). If value= '1' then number of GDS and/or IS events detected in one second is over the limit specified in the CF parameters "Max GDS Events Per second" and "Max IS Events Per second" respectively. This flag is reset every second.

HK_scienceTMEnabled (0,1). If value= '1' then GIADA can not add Science TM(20,3) to the current TM Block that will be sent to SC. If value='0' then GIADA can add it.

HK_frangiboltArmed(0,1). If value= '1' then the Frangibolt is armed. If value='0' then the Frangibolt is disarmed. This flag is set to '1' explicitly using "Arm Frangibolt TC(192,1)", set to '0' explicitly using "Disarm Frangibolt TC(192,2)". This flag is automatically set to '0' at the end of "Test Frangibolt TC(192,6)", "Activate Frangibolt TC(192,11)". Also is automatically set to '0' after the execution of TC "Arm Frangibolt TC(192,1)" and past time specified in CF parameter "Arm TCs Timeout" (30s by default) without the execution of "Test Frangibolt TC(192,6)" or "Activate Frangibolt TC(192,11)". When this flag is set to '0' due to timeout the following TM event is received: "'42200' Frangibolt to disarmed state".

HK_coverArmed (0,1). If value= '1' then the Cover is armed. If value='0' then the Cover is disarmed. This flag is set to '1' explicitly using "Arm Cover TC(192,16)", set to '0' explicitly using "Disarm Cover TC(192,17)". This flag is automatically set to '0' at the end of "Open Cover TC(192,21)" or "Close Cover TC(192,26)" with any valid parameter. This flag is automatically set to '0' at the end of "Test Frangibolt TC(192,6)", "Activate Frangibolt TC(192,11)". Also is automatically set to '0' after the execution of TC "Arm Cover TC(192,16)" and past time specified in CF parameter "Arm TCs Timeout" (30s by default) without the execution of "Open Cover TC(192,21)" or "Close Cover TC(192,26)". When this flag is set to '0' due to timeout the following TM event is received: "'42201' Motor to disarmed state".

HK_openReedSwitch (0,1). If value= '1' then the Open Reed Switch is open. If value='0' then the Open Reed Switch is closed. When the cover is completely opened this reed switch should be closed.

HK_closedReedSwitch (0,1). If value= '1' then the Closed Reed Switch is open. If value='0' then the Closed Reed Switch is closed. When the cover is completely closed this reed switch is opened and only is closed during the open and close operations.

HK_coverHeaterEnabled (0,1). If value= '1' then the Cover Heater is heating. If value='0' then the Cover Heater is off. This flag is On,Off (finally off) during the execution: "Test Frangibolt TC(192,6)", "Activate Frangibolt TC(192,11)", "Open Cover TC(192,21)", "Close Cover TC(192,26)" with any valid parameter, "Test Heater TC(192,31)" with any valid parameter. This flag is associated with the following events: "'42001' Heaters On", "'42002' Heaters Off".

HK_motorHeaterEnabled (0,1). If value= '1' then the Cover Motor Heater is heating. If value='0' then the Cover Motor Heater is off. This flag is On,Off (finally off) during the execution: "Open Cover TC(192,21)", "Close Cover TC(192,26)" with parameter '0xFFFF', "Test Heater TC(192,31)" with parameter '6'. This flag is associated with the following TM events: "'42001' Heaters On", "'42002' Heaters Off"

HK_frangiboltHeaterEnabled(0,1). If value= '1' then the Cover Frangibolt Heater is heating. If value='0' then the Cover Frangibolt Heater is off. This flag is On,Off (finally off) during the execution: "Test Frangibolt TC(192,6)", "Activate Frangibolt TC(192,11)", "Open Cover TC(192,21)", "Test Heater TC(192,31)" with parameter '5'. This flag is associated with the following TM events: "'42001' Heaters On", "'42002' Heaters Off".

HK_frangiboltTemperature (0,4095) [Frangibolt Temperature]. Frangibolt device temperature in ADC counts.

HK_gdsON(0,1). If value= '1' then GDS is On .If value='0' then GDS is off. This flag is set to '1' explicitly using "Set GDS On/Off TC(193,11)" with parameter '0xFFFF', set to '0' explicitly using "Set GDS On/Off TC(193,11)" with parameter '0'. This flag is automatically set to '0' at the end of: "Go to Safe Mode TC(196,1)". This flag is automatically set (according with CF parameter "GDS Status") at the end of "Go to Normal Mode TC(196,11)". When this flag is '1' the following HK fields are updated according with CF parameter "GDS Status": gdsThresholdLeft, gdsThresholdRight.

HK_laserArmed (0,1). If value= '1' then GDS is armed .If value='0' then GDS is disarmed. This flag is explicitly set to '1' using "Arm Laser TC(193,1)", set to '0' explicitly using "Disarm Laser TC(193,2)". This flag is automatically set to '0' at the end of "Switch Laser On/Off TC(193,6)" with parameter '0' or "Set GDS On/Off TC(193,11)" with parameter '0'. Ignore this value if GDS is off. Also is automatically set to '0' after the execution of TC "Arm Laser TC(193,1)" and past time specified in CF parameter "Arm TCs Timeout" (30s by default) without the execution of "Switch Laser On/Off TC(193,6)" with parameter '0xFFFF'. When this flag is set to '0' due to timeout the following TM event is received: "'42210' Laser To Disarmed State".

HK_laserPower (0,3). Laser Consumption: value= '0' (laser is physically off but not logically) Laser Off, value= '1' Low Consumption, value= '2' Medium Consumption, value= '3' High Consumption. By default in every "Switch Laser On/Off TC(193,6)" with parameter '0xFFFF', Laser Consumption is set to Low during 30s and after this time is set to the Laser Consumption configuration established in the CF parameter "GDS Status". The value of this parameter can be changed using: "Set GDS Operation Mode TC(193,16). Ignore this value if the GDS is off.

HK_laserMode(1,3). Laser Operation Mode: value= '1' DC Couple 1 only, value= '2' DC Couple 2 only, value= '3' AC Both Couples. This value is set autonomously (according CF parameter "GDS Status") in every "Switch Laser On/Off TC(193,6)" with parameter '0xFFFF' and .The value of this parameter can be changed explicitly using: "Set GDS Operation Mode TC(193,16)". This flag is automatically set (according with CF parameter "GDS Status") at the end of "Go to Normal Mode TC(196,11)". Ignore this value if GDS is off.

HK_receiverLeftEnabled(0,1). If value= '1' then Left Receiver is enabled, and the particle detections are allowed by this channel. If value='0' then Left Receiver is disabled and no particle will be detected by this channel. This value is set autonomously according CF parameter "GDS Status" in every "Switch Laser On/Off TC(193,6)" with parameter '0xFFFF'. The value of this parameter can be changed using: "Set GDS Operation Mode TC(193,16)". Ignore this value if GDS is off.

HK_receiverRightEnabled(0,1). If value= '1' then Right Receiver is enabled, and the particle detections are allowed by this channel. If value='0' then Right Receiver is disabled and no particle will be detected by this channel. This value is set autonomously according CF parameter "GDS Status" in every "Switch Laser On/Off TC(193,6)" with parameter '0xFFFF'. The value of this parameter can be changed using: "Set GDS Operation Mode TC(193,16)". Ignore this value if GDS is off.

HK_laserON (0,1). If value= '1' then Laser is On and GDS Events are not allowed .If value='0' then Laser is off and GDS Events are not allowed. This flag is set to '1' explicitly using "Switch Laser On/Off TC(193,6)" with parameter '0xFFFF' and set to '0' explicitly using "Switch Laser

On/Off TC(193,6)" with parameter '0'. This flag is automatically set to '0' at the end of: "Go to Safe Mode TC(196,1)". Ignore this value if GDS is off.

HK_detections (1,3). Number of clock pulses needed to detect a particle by the GDS in Laser Mode '3', in the other modes this parameter is useless. This value is set autonomously according CF parameter "GDS Status" in every "Switch Laser On/Off TC(193,6)" with parameter '0xFFFF'. The value of this parameter can be changed using: "Set GDS Operation Mode TC(193,16)". Ignore this value if GDS is off.

HK_laser_1_Temperature (0,4095) [GDS Laser 1 Temperature]. Temperature of the Laser 1. Ignore this value if GDS is off.

HK_laser_2_Temperature (0,4095) [GDS Laser 2 Temperature]. Temperature of the Laser 2. Ignore this value if GDS is off.

HK_laser_3_Temperature (0,4095) [GDS Laser 3 Temperature]. Temperature of the Laser 3. Ignore this value if GDS is off.

HK_laser_4_Temperature (0,4095) [GDS Laser 4 Temperature]. Temperature of the Laser 4. Ignore this value if GDS is off.

HK_laser_1_Light(0,4095) [GDS Laser 1 Light Monitor]. Light of the Laser 1. Ignore this value if GDS is off.

HK_laser_2_Light (0,4095) [GDS Laser 2 Light Monitor]. Light of the Laser 2. Ignore this value if GDS is off.

HK_laser_3_Light (0,4095) [GDS Laser 3 Light Monitor]. Light of the Laser 3. Ignore this value if GDS is off.

HK_laser_4_Light (0,4095) [GDS Laser 4 Light Monitor]. Light of the Laser 4. Ignore this value if GDS is off.

HK_gdsThresholdLeft (0,255) [GDS Threshold Left]. Threshold of the left Receiver. Min signal value necessary to detect a particle by this channel. This value can be changed using "Set Photodiode Threshold TC(193,26)". This value is set to CF parameter "GDS Status" with "Set GDS On/Off TC(193,11)" with parameter '0xFFFF'. Ignore this value if GDS is off.

HK_gdsThresholdRight (0,255) [GDS Threshold Right]. Threshold of the right Receiver. Min signal value necessary to detect a particle by this channel. This value can be changed using "Set Photodiode Threshold TC(193,26)". This value is set to CF parameter "GDS Status" with "Set GDS On/Off TC(193,11)" with parameter '0xFFFF'. Ignore this value if GDS is off.

HK_isON (0,1). If value= '1' then IS is On and IS Events are allowed .If value='0' then IS is off and IS Events are not allowed. This flag is set to '1' explicitly using "Set IS On/Off TC(194,1)" with parameter '0xFFFF', set to '0' explicitly using "Set IS On/Off TC(194,1)" with parameter '0'. This flag is automatically set to '0' at the end of: "Go to Safe Mode TC(196,1)". This flag is automatically set to '1' or '0' due to IS thermal contingency actions (see "isTemperature" field). With "Go to Normal Mode TC(196,11)" the value is this flag is set according with CF parameter "IS Status". When this flag is '1' the following HK fields are updated according with CF parameter "IS Status":isPZT_E_Enabled,isPZT_D_Enabled,isPZT_C_Enabled,isPZT_B_Enabled,isPZT_A_Enabled,isRange,isPZT_E_Gain,isPZT_D_Gain,isPZT_A_Gain,isThreshold_PZT_A,isThreshold_PZT_B,isThreshold_PZT_C,isThreshold_PZT_D,isThreshold_PZT_E.

HK_is_PZT_E_Enabled(0,1). If value= '1' this PZT is enabled and can detect perturbations. If value='0' then this PZT is disabled and no perturbations can be detected by this channel (PZT amplitude and PZT time always 0 for this channel).This value is set autonomously according CF parameter "IS Status" in every "Set IS On/Off TC(194,1)" with parameter '0xFFFF' and "Go to Normal Mode TC(196,11)". The value of this parameter can be changed using: "Set IS Operation Mode TC(194,6). Ignore this value if IS is off.

HK_is_PZT_D_Enabled(0,1). If value= '1' this PZT is enabled and can detect perturbations. If value='0' then this PZT is disabled and no perturbations can be detected by this channel (PZT amplitude and PZT time always 0 for this channel).This value is set autonomously according CF parameter "IS Status" in every "Set IS On/Off TC(194,1)" with parameter '0xFFFF' and "Go to Normal Mode TC(196,11)". The value of this parameter can be changed using: "Set IS Operation Mode TC(194,6). Ignore this value if IS is off.

HK_is_PZT_C_Enabled(0,1). If value= '1' this PZT is enabled and can detect perturbations. If value='0' then this PZT is disabled and no perturbations can be detected by this channel (PZT amplitude and PZT time always 0 for this channel).This value is set autonomously according CF parameter "IS Status" in every "Set IS On/Off TC(194,1)" with parameter '0xFFFF' and "Go to Normal Mode TC(196,11)". The value of this parameter can be changed using: "Set IS Operation Mode TC(194,6). Ignore this value if IS is off.

HK_is_PZT_B_Enabled(0,1). If value= '1' this PZT is enabled and can detect perturbations. If value='0' then this PZT is disabled and no perturbations can be detected by this channel (PZT amplitude and PZT time always 0 for this channel).This value is set autonomously according CF parameter "IS Status" in every "Set IS On/Off TC(194,1)" with parameter '0xFFFF' and "Go to Normal Mode TC(196,11)". The value of this parameter can be changed using: "Set IS Operation Mode TC(194,6). Ignore this value if IS is off.

HK_is_PZT_A_Enabled(0,1). If value= '1' this PZT is enabled and can detect perturbations. If value='0' then this PZT is disabled and no perturbations can be detected by this channel (PZT amplitude and PZT time always 0 for this channel).This value is set autonomously according CF parameter "IS Status" in every "Set IS On/Off TC(194,1)" with parameter '0xFFFF' and "Go to Normal Mode TC(196,11)". The value of this parameter can be changed using: "Set IS Operation Mode TC(194,6). Ignore this value if IS is off.

HK_isRange(0,1). If value= '1' the IS Range is high of all channels. If value='0' the IS Range is low. This value is set autonomously according CF parameter "IS Status" in every "Set IS On/Off TC(194,1)" with parameter '0xFFFF' and "Go to Normal Mode TC(196,11)". The value of this parameter can be changed using: "Set IS Operation Mode TC(194,6). Ignore this value if IS is off.

HK_is_PZT_E_Gain (0,1). If value= '1' the PZT gain is high of this channel. If value='0' the PZT gain is low. This value is set autonomously according CF parameter "IS Status" in every "Set IS On/Off TC(194,1)" with parameter '0xFFFF' and "Go to Normal Mode TC(196,11)". The value of this parameter can be changed using: "Set IS Operation Mode TC(194,6). Ignore this value if IS is off.

HK_is_PZT_D_Gain (0,1). If value= '1' the PZT gain is high of this channel. If value='0' the PZT gain is low. This value is set autonomously according CF parameter "IS Status" in every "Set IS On/Off TC(194,1)" with parameter '0xFFFF' and "Go to Normal Mode TC(196,11)". The

value of this parameter can be changed using: "Set IS Operation Mode TC(194,6). Ignore this value if IS is off.

HK_is_PZT_A_Gain (0,1). If value= '1' the PZT gain is high of this channel. If value='0' the PZT gain is low. This value is set autonomously according CF parameter "IS Status" in every "Set IS On/Off TC(194,1)" with parameter '0xFFFF' and "Go to Normal Mode TC(196,11)". The value of this parameter can be changed using: "Set IS Operation Mode TC(194,6). Ignore this value if IS is off.

HK_isTemperature (0,4095) [IS Plate Temperature]. Temperature of the IS.

HK_isThreshold_PZT_A (0,255) [IS Threshold PZT#A]. Threshold of the PZTA. Min signal value necessary to detect a particle by this PZT. This value is set autonomously according CF parameter "IS Status" in every "Set IS On/Off TC(194,1)" with parameter '0xFFFF' and "Go to Normal Mode TC(196,11)". The value of this parameter can be changed using: "Set IS Operation Mode TC(194,6). Ignore this value if IS is off.

HK_isThreshold_PZT_B(0,255) [IS Threshold PZT#B]. Threshold of the PZTB. Min signal value necessary to detect a particle by this PZT. This value is set autonomously according CF parameter "IS Status" in every "Set IS On/Off TC(194,1)" with parameter '0xFFFF' and "Go to Normal Mode TC(196,11)". The value of this parameter can be changed using: "Set IS Operation Mode TC(194,6). Ignore this value if IS is off.

HK_isThreshold_PZT_C(0,255) [IS Threshold PZT#C]. Threshold of the PZTC. Min signal value necessary to detect a particle by this PZT. This value is set autonomously according CF parameter "IS Status" in every "Set IS On/Off TC(194,1)" with parameter '0xFFFF' and "Go to Normal Mode TC(196,11)". The value of this parameter can be changed using: "Set IS Operation Mode TC(194,6). Ignore this value if IS is off.

HK_isThreshold_PZT_D(0,255) [IS Threshold PZT#D]. Threshold of the PZTD. Min signal value necessary to detect a particle by this PZT. This value is set autonomously according CF parameter "IS Status" in every "Set IS On/Off TC(194,1)" with parameter '0xFFFF' and "Go to Normal Mode TC(196,11)". The value of this parameter can be changed using: "Set IS Operation Mode TC(194,6). Ignore this value if IS is off.

HK_isThreshold_PZT_E(0,255) [IS Threshold PZT#E]. Threshold of the PZTE. Min signal value necessary to detect a particle by this PZT. This value is set autonomously according CF parameter "IS Status" in every "Set IS On/Off TC(194,1)" with parameter '0xFFFF' and "Go to Normal Mode TC(196,11)". The value of this parameter can be changed using: "Set IS Operation Mode TC(194,6). Ignore this value if IS is off.

HK_mbsON(0,1). If value= '1' then MBS is On and MBS Events are allowed .If value='0' then MBS is off and MBS Events are not allowed. This flag is set to '1' explicitly using "Set MBS On/Off TC(195,1)" with parameter '0xFFFF', set to '0' explicitly using "Set MBS On/Off TC(195,1)" with parameter '0'. This flag is automatically set to '0' at the end of: "Go to Safe Mode TC(196,1)". This flag is automatically set according with CF parameter "MBS Status" at the end of: "Go to Flux Mode TC(196,16)" and "Go to Normal Mode TC(196,11)". When this flag is set to '1', the following HK fields are updated according with CF parameter "MBS Status":mb_5_Enabled,mb_4_Enabled,mb_3_Enabled,mb_2_Enabled,mb_1_Enabled.

HK_mb_5_Enabled (0,1). If value= '1' this MBS Channel is enabled and MBS events can be generated by this channel. If value='0' then this MBS channel is disabled and no MBS event

can be generated by this channel (Frequency and Temperature always 0). This value is set autonomously according CF parameter "MBS Status" in every "Set MBS On/Off TC(195,1)" with parameter '0xFFFF'. The value of this parameter can be changed using: "Set MBS Operation Mode TC(195,6). This value is set autonomously according CF parameter "MBS Status" in every: "Go to Flux Mode TC(196,16)" and "Go to Normal Mode TC(196,11)". Ignore this value if MBS is off.

HK_mb_4_Enabled (0,1). If value= '1' this MBS Channel is enabled and MBS events can be generated by this channel. If value='0' then this MBS channel is disabled and no MBS event can be generated by this channel (Frequency and Temperature always 0). This value is set autonomously according CF parameter "MBS Status" in every "Set MBS On/Off TC(195,1)" with parameter '0xFFFF'. The value of this parameter can be changed using: "Set MBS Operation Mode TC(195,6). This value is set autonomously according CF parameter "MBS Status" in every: "Go to Flux Mode TC(196,16)" and "Go to Normal Mode TC(196,11)". Ignore this value if MBS is off.

HK_mb_3_Enabled (0,1). If value= '1' this MBS Channel is enabled and MBS events can be generated by this channel. If value='0' then this MBS channel is disabled and no MBS event can be generated by this channel (Frequency and Temperature always 0). This value is set autonomously according CF parameter "MBS Status" in every "Set MBS On/Off TC(195,1)" with parameter '0xFFFF'. The value of this parameter can be changed using: "Set MBS Operation Mode TC(195,6). This value is set autonomously according CF parameter "MBS Status" in every: "Go to Flux Mode TC(196,16)" and "Go to Normal Mode TC(196,11)". Ignore this value if MBS is off.

HK_mb_2_Enabled (0,1). If value= '1' this MBS Channel is enabled and MBS events can be generated by this channel. If value='0' then this MBS channel is disabled and no MBS event can be generated by this channel (Frequency and Temperature always 0). This value is set autonomously according CF parameter "MBS Status" in every "Set MBS On/Off TC(195,1)" with parameter '0xFFFF'. The value of this parameter can be changed using: "Set MBS Operation Mode TC(195,6). This value is set autonomously according CF parameter "MBS Status" in every: "Go to Flux Mode TC(196,16)" and "Go to Normal Mode TC(196,11)". Ignore this value if MBS is off.

HK_mb_1_Enabled (0,1). If value= '1' this MBS Channel is enabled and MBS events can be generated by this channel. If value='0' then this MBS channel is disabled and no MBS event can be generated by this channel (Frequency and Temperature always 0). This value is set autonomously according CF parameter "MBS Status" in every "Set MBS On/Off TC(195,1)" with parameter '0xFFFF'. The value of this parameter can be changed using: "Set MBS Operation Mode TC(195,6). This value is set autonomously according CF parameter "MBS Status" in every: "Go to Flux Mode TC(196,16)" and "Go to Normal Mode TC(196,11)". Ignore this value if MBS is off.

HK_mbHeater_5_Enabled(0,1). If value= '1' then MB heater is heating up to the temperature specified in CF parameter "MBS Maximum Temperature during Heating" or during the time specified in CF parameter "Heating Timeout". If value='0' then the MB heater is off. If this MB is disabled, and a TC "Heat MBS TC(195,26)" with parameter '5' is executed the MB science data generated (frequency and temperature) is always '0', but physically the MB has being heated. Ignore this value if MBS is off.

HK_mbHeater_4_Enabled(0,1). If value= '1' then MB heater is heating up to the temperature specified in CF parameter "MBS Maximum Temperature during Heating" or during the time specified in CF parameter "Heating Timeout". If value='0' then the MB heater is off. If this MB is disabled, and a TC "Heat MBS TC(195,26)" with parameter '4' is executed the MB science data generated (frequency and temperature) is always '0', but physically the MB has being heated. Ignore this value if MBS is off.

HK_mbHeater_3_Enabled(0,1). If value= '1' then MB heater is heating up to the temperature specified in CF parameter "MBS Maximum Temperature during Heating" or during the time specified in CF parameter "Heating Timeout". If value='0' then the MB heater is off. If this MB is disabled, and a TC "Heat MBS TC(195,26)" with parameter '3' is executed the MB science data generated (frequency and temperature) is always '0', but physically the MB has being heated. Ignore this value if MBS is off.

HK_mbHeater_2_Enabled(0,1). If value= '1' then MB heater is heating up to the temperature specified in CF parameter "MBS Maximum Temperature during Heating" or during the time specified in CF parameter "Heating Timeout". If value='0' then the MB heater is off. If this MB is disabled, and a TC "Heat MBS TC(195,26)" with parameter '2' is executed the MB science data generated (frequency and temperature) is always '0', but physically the MB has being heated. Ignore this value if MBS is off.

HK_mbHeater_1_Enabled(0,4095). If value= '1' then MB heater is heating up to the temperature specified in CF parameter "MBS Maximum Temperature during Heating" or during the time specified in CF parameter "Heating Timeout". If value='0' then the MB heater is off. If this MB is disabled, and a TC "Heat MBS TC(195,26)" with parameter '1' is executed the MB science data generated (frequency and temperature) is always '0', but physically the MB has being heated. Ignore this value if MBS is off.

HK_mb_1_Temperature(0,4095) [MBS 1 Temperature]. Temperature of the MB 1. Ignore this value if MBS is off.

HK_mb_2_Temperature(0,4095) [MBS 2 Temperature]. Temperature of the MB 2. Ignore this value if MBS is off.

HK_mb_3_Temperature(0,4095) [MBS 3 Temperature]. Temperature of the MB 3. Ignore this value if MBS is off.

HK_mb_4_Temperature(0,4095) [MBS 4 Temperature]. Temperature of the MB 4. Ignore this value if MBS is off.

HK_mb_5_Temperature(0,4095) [MBS 5 Temperature]. Temperature of the MB 5. Ignore this value if MBS is off.

HK_powerSupplyTemperature(0,4095) [PS Temperature]. Power Supply Temperature in ADC counts. If this value is greater than CF parameter "ME MAX-OP Temp" then an event report " '42242 'Main Electronic Temperature Above MAX-OP Temp" is generated and if the GIADA Operation mode is "Normal" or "Flux" an additional OBCP "Emergency Close Cover" is generated. This check can be enabled or disabled according with CF parameter "ME MAX-OP Temp" subparameter "Temp. Checking".

HK_plus_5v_PowerConsumption (0,4095) [+5V Power Consumption]. +5V Power Consumption in ADC counts.

HK_plus_15v_PowerConsumption(0,4095) [+15V Power Consumption]. +15V Power Consumption in ADC counts.

HK_minus_15v_PowerConsumption (0,4095) [-15V Power Consumption]. -15V Power Consumption in ADC counts.

HK_patchesStatus_1(0,65535). Patches status from 48 to 63 patch .Every bit means one patch applied. Greater patch value is related with the most significative bit.

HK_patchesStatus_2 (0,65535). Patches status from 32 to 47 patch .Every bit means one patch applied. Greater patch value is related with the most significative bit.

HK_patchesStatus_3 (0,65535). Patches status from 16 to 31 patch .Every bit means one patch applied. Greater patch value is related with the most significative bit.

HK_patchesStatus_4 (0,65535). Patches status from 0 to 15 patch .Every bit means one patch applied. Greater patch value is related with the most significative bit.

GDB ISCALIBRATION

This table stores the information about the IS calibration.

SQL definition

```
CREATE TABLE GDB_ISCalibration(
ID NUMBER(16) PRIMARY KEY,

IS_cal_isTemperature NUMBER(5) NOT NULL Constraint IS_CAL_isTemperature
CHECK(IS_cal_isTemperature>=0 and IS_cal_isTemperature<=4095),

IS_cal_sumPZT_A NUMBER(10) NOT NULL Constraint IS_CAL_sumPZT_A
CHECK(IS_cal_sumPZT_A>=0 and IS_cal_sumPZT_A<=4294967295),

IS_cal_squareSumPZT_A NUMBER(10) NOT NULL Constraint IS_CAL_squareSumPZT_A
CHECK(IS_cal_squareSumPZT_A>=0 and IS_cal_squareSumPZT_A<=4294967295),

IS_cal_sumPZT_B NUMBER(10) NOT NULL Constraint IS_CAL_sumPZT_B
CHECK(IS_cal_sumPZT_B>=0 and IS_cal_sumPZT_B<=4294967295),

IS_cal_squareSumPZT_B NUMBER(10) NOT NULL Constraint IS_CAL_squareSumPZT_B
CHECK(IS_cal_squareSumPZT_B>=0 and IS_cal_squareSumPZT_B<=4294967295),

IS_cal_sumPZT_C NUMBER(10) NOT NULL Constraint IS_CAL_sumPZT_C
CHECK(IS_cal_sumPZT_C>=0 and IS_cal_sumPZT_C<=4294967295),

IS_cal_squareSumPZT_C NUMBER(10) NOT NULL Constraint IS_CAL_squareSumPZT_C
CHECK(IS_cal_squareSumPZT_C>=0 and IS_cal_squareSumPZT_C<=4294967295),

IS_cal_sumPZT_D NUMBER(10) NOT NULL Constraint IS_CAL_sumPZT_D
CHECK(IS_cal_sumPZT_D>=0 and IS_cal_sumPZT_D<=4294967295),

IS_cal_squareSumPZT_D NUMBER(10) NOT NULL Constraint IS_CAL_squareSumPZT_D
CHECK(IS_cal_squareSumPZT_D>=0 and IS_cal_squareSumPZT_D<=4294967295),

IS_cal_sumPZT_E NUMBER(10) NOT NULL Constraint IS_CAL_sumPZT_E
CHECK(IS_cal_sumPZT_E>=0 and IS_cal_sumPZT_E<=4294967295),

IS_cal_squareSumPZT_E NUMBER(10) NOT NULL Constraint IS_CAL_squareSumPZT_E
CHECK(IS_cal_squareSumPZT_E>=0 and IS_cal_squareSumPZT_E<=4294967295),
```


and processed. At this point, the IS is enabled again and stimulated if it's necessary At the end, the IS stimulator is disabled and IS is enabled.

IS_cal_sumPZT_C(0,4294967295) [IS Sum PZT#C]. Sum of 8 means, each mean take 4 measures over this PZT The idea of calibration is use the internal calibrator in order to generate a simulate impact "numberOfStimuli" times with a "calibrationLevel" signal power and see which is the signal that is received by this channel if IS event not arrives after 2 second timeout, it's amplitude and time (science data) will be set to 0. Initially, IS is enabled, in each IS simulated detected, the IS is disabled until the science data can be read and processed. At this point, the IS is enabled again and stimulated if it's necessary At the end, the IS stimulator is disabled and IS is enabled.

IS_cal_squaresumPZT_C(0,4294967295) [IS Noise PZT#C]. Sum of the square of 8 means, each mean take 4 measures over this PZT The idea of calibration is use the internal calibrator in order to generate a simulate impact "numberOfStimuli" times with a "calibrationLevel" signal power and see which is the signal that is received by this channel if IS event not arrives after 2 second timeout, it's amplitude and time (science data) will be set to 0. Initially, IS is enabled, in each IS simulated detected, the IS is disabled until the science data can be read and processed. At this point, the IS is enabled again and stimulated if it's necessary At the end, the IS stimulator is disabled and IS is enabled.

IS_cal_sumPZT_D(0,4294967295) [IS Sum PZT#D]. Sum of 8 means, each mean take 4 measures over this PZT The idea of calibration is use the internal calibrator in order to generate a simulate impact "numberOfStimuli" times with a "calibrationLevel" signal power and see which is the signal that is received by this channel if IS event not arrives after 2 second timeout, it's amplitude and time (science data) will be set to 0. Initially, IS is enabled, in each IS simulated detected, the IS is disabled until the science data can be read and processed. At this point, the IS is enabled again and stimulated if it's necessary At the end, the IS stimulator is disabled and IS is enabled.

IS_cal_squaresumPZT_D(0,4294967295) [IS Noise PZT#D]. Sum of the square of 8 means, each mean take 4 measures over this PZT The idea of calibration is use the internal calibrator in order to generate a simulate impact "numberOfStimuli" times with a "calibrationLevel" signal power and see which is the signal that is received by this channel if IS event not arrives after 2 second timeout, it's amplitude and time (science data) will be set to 0. Initially, IS is enabled, in each IS simulated detected, the IS is disabled until the science data can be read and processed. At this point, the IS is enabled again and stimulated if it's necessary At the end, the IS stimulator is disabled and IS is enabled.

IS_cal_sumPZT_E(0,4294967295) [IS Sum PZT#E]. Sum of 8 means, each mean take 4 measures over this PZT The idea of calibration is use the internal calibrator in order to generate a simulate impact "numberOfStimuli" times with a "calibrationLevel" signal power and see which is the signal that is received by this channel if IS event not arrives after 2 second timeout, it's amplitude and time (science data) will be set to 0. Initially, IS is enabled, in each IS simulated detected, the IS is disabled until the science data can be read and processed. At this point, the IS is enabled again and stimulated if it's necessary At the end, the IS stimulator is disabled and IS is enabled.

IS_cal_squaresumPZT_E (0,4294967295) [IS Noise PZT#E]. Sum of the square of 8 means, each mean take 4 measures over this PZT The idea of calibration is use the internal calibrator in order to generate a simulate impact "numberOfStimuli" times with a "calibrationLevel"

signal power and see which is the signal that is received by this channel if IS event not arrives after 2 second timeout, it's amplitude and time (science data) will be set to 0. Initially, IS is enabled, in each IS simulated detected, the IS is disabled until the science data can be read and processed. At this point, the IS is enabled again and stimulated if it's necessary At the end, the IS stimulator is disabled and IS is enabled.

IS_cal_calibrationLevel(0,255). Signal Level used in the IS calibration simulation The idea of calibration is use the internal calibrator in order to generate a simulate impact "numberOfStimuli" times with a "calibrationLevel" signal power and see which is the signal that is received by this channel if IS event not arrives after 2 second timeout, it's amplitude and time (science data) will be set to 0. Initially, IS is enabled, in each IS simulated detected, the IS is disabled until the science data can be read and processed. At this point, the IS is enabled again and stimulated if it's necessary At the end, the IS stimulator is disabled and IS is enabled.

IS_cal_numberOfStimuli(0,8). Number of times that the IS simulated Event must be generated Also, the number of IS events detected expected to put inside IS calibration Packet.

GDB_ISCALIBRATIONEVENT

This table stores the events generated to calibrate the IS.

SQL definition

```
CREATE TABLE GDB_ISCalibrationEvent(
  ID NUMBER(16) NOT NULL,

  IS_cal_eve_ITEM NUMBER(2) NOT NULL Constraint IS_CAL_ITEM
  CHECK(IS_cal_eve_ITEM>=0),

  IS_cal_eve_pzt_A_detected NUMBER(1) NOT NULL Constraint
  IS_CAL_pzt_A_detected CHECK(IS_cal_eve_pzt_A_detected=0 or
  IS_cal_eve_pzt_A_detected=1),

  IS_cal_eve_pzt_A_range NUMBER(1) NOT NULL Constraint IS_CAL_pzt_A_range
  CHECK(IS_cal_eve_pzt_A_range=0 or IS_cal_eve_pzt_A_range=1),

  IS_cal_eve_pzt_A_gain NUMBER(1) NOT NULL Constraint IS_CAL_pzt_A_gain
  CHECK(IS_cal_eve_pzt_A_gain=0 or IS_cal_eve_pzt_A_gain=1),

  IS_cal_eve_pzt_A_amplitude NUMBER(5) NOT NULL Constraint
  IS_CAL_pzt_A_amplitude CHECK(IS_cal_eve_pzt_A_amplitude>=0 and
  IS_cal_eve_pzt_A_amplitude<=4095),

  IS_cal_eve_pzt_A_timeOverflow NUMBER(1) NOT NULL Constraint
  IS_CAL_pzt_A_timeOverflow CHECK(IS_cal_eve_pzt_A_timeOverflow=0 or
  IS_cal_eve_pzt_A_timeOverflow=1),

  IS_cal_eve_pzt_A_time FTYPE1(8,9) NUMBER(2) NOT NULL Constraint
  IS_CAL_pzt_A_time CHECK(IS_cal_eve_pzt_A_time>=0 and
  IS_cal_eve_pzt_A_time<=31),
```

```
IS_cal_eve_pzt_B_detected NUMBER(1) NOT NULL Constraint
IS_CAL_pzt_B_detected CHECK(IS_cal_eve_pzt_B_detected=0 or
IS_cal_eve_pzt_B_detected=1),

IS_cal_eve_pzt_B_range NUMBER(1) NOT NULL Constraint IS_CAL_pzt_B_range
CHECK(IS_cal_eve_pzt_B_range=0 or IS_cal_eve_pzt_B_range=1),

IS_cal_eve_pzt_B_gain NUMBER(1) NOT NULL Constraint IS_CAL_pzt_B_gain
CHECK(IS_cal_eve_pzt_B_gain=0 or IS_cal_eve_pzt_B_gain=1),

IS_cal_eve_pzt_B_amplitude NUMBER(5) NOT NULL Constraint
IS_CAL_pzt_B_amplitude CHECK(IS_cal_eve_pzt_B_amplitude>=0 and
IS_cal_eve_pzt_B_amplitude<=4095),

IS_cal_eve_pzt_B_timeOverflow NUMBER(1) NOT NULL Constraint
IS_CAL_pzt_B_timeOverflow CHECK(IS_cal_eve_pzt_B_timeOverflow=0 or
IS_cal_eve_pzt_B_timeOverflow=1),

IS_cal_eve_pzt_B_time FTYPE1(8,9) NUMBER(2) NOT NULL Constraint
IS_CAL_pzt_B_time CHECK(IS_cal_eve_pzt_B_time>=0 and
IS_cal_eve_pzt_B_time<=31),

IS_cal_eve_pzt_C_detected NUMBER(1) NOT NULL Constraint
IS_CAL_pzt_C_detected CHECK(IS_cal_eve_pzt_C_detected=0 or
IS_cal_eve_pzt_C_detected=1),

IS_cal_eve_pzt_C_range NUMBER(1) NOT NULL Constraint IS_CAL_pzt_C_range
CHECK(IS_cal_eve_pzt_C_range=0 or IS_cal_eve_pzt_C_range=1),

IS_cal_eve_pzt_C_gain NUMBER(1) NOT NULL Constraint IS_CAL_pzt_C_gain
CHECK(IS_cal_eve_pzt_C_gain=0 or IS_cal_eve_pzt_C_gain=1),

IS_cal_eve_pzt_C_amplitude NUMBER(5) NOT NULL Constraint
IS_CAL_pzt_C_amplitude CHECK(IS_cal_eve_pzt_C_amplitude>=0 and
IS_cal_eve_pzt_C_amplitude<=4095),

IS_cal_eve_pzt_C_timeOverflow NUMBER(1) NOT NULL Constraint
IS_CAL_pzt_C_timeOverflow CHECK(IS_cal_eve_pzt_C_timeOverflow=0 or
IS_cal_eve_pzt_C_timeOverflow=1),

IS_cal_eve_pzt_C_time FTYPE1(8,9) NUMBER(2) NOT NULL Constraint
IS_CAL_pzt_C_time CHECK(IS_cal_eve_pzt_C_time>=0 and
IS_cal_eve_pzt_C_time<=31),

IS_cal_eve_pzt_D_detected NUMBER(1) NOT NULL Constraint
IS_CAL_pzt_D_detected CHECK(IS_cal_eve_pzt_D_detected=0 or
IS_cal_eve_pzt_D_detected=1),

IS_cal_eve_pzt_D_range NUMBER(1) NOT NULL Constraint IS_CAL_pzt_D_range
CHECK(IS_cal_eve_pzt_D_range=0 or IS_cal_eve_pzt_D_range=1),

IS_cal_eve_pzt_D_gain NUMBER(1) NOT NULL Constraint IS_CAL_pzt_D_gain
CHECK(IS_cal_eve_pzt_D_gain=0 or IS_cal_eve_pzt_D_gain=1),

IS_cal_eve_pzt_D_amplitude NUMBER(5) NOT NULL Constraint
IS_CAL_pzt_D_amplitude CHECK(IS_cal_eve_pzt_D_amplitude>=0 and
IS_cal_eve_pzt_D_amplitude<=4095),

IS_cal_eve_pzt_D_timeOverflow NUMBER(1) NOT NULL Constraint
IS_CAL_pzt_D_timeOverflow CHECK(IS_cal_eve_pzt_D_timeOverflow=0 or
IS_cal_eve_pzt_D_timeOverflow=1),

IS_cal_eve_pzt_D_time FTYPE1(8,9) NUMBER(2) NOT NULL Constraint
IS_CAL_pzt_D_time CHECK(IS_cal_eve_pzt_D_time>=0 and
IS_cal_eve_pzt_D_time<=31),
```

```

IS_cal_eve_pzt_E_detected NUMBER(1) NOT NULL Constraint
IS_CAL_pzt_E_detected CHECK(IS_cal_eve_pzt_E_detected=0 or
IS_cal_eve_pzt_E_detected=1),

IS_cal_eve_pzt_E_range NUMBER(1) NOT NULL Constraint IS_CAL_pzt_E_range
CHECK(IS_cal_eve_pzt_E_range=0 or IS_cal_eve_pzt_E_range=1),

IS_cal_eve_pzt_E_gain NUMBER(1) NOT NULL Constraint IS_CAL_pzt_E_gain
CHECK(IS_cal_eve_pzt_E_gain=0 or IS_cal_eve_pzt_E_gain=1),

IS_cal_eve_pzt_E_amplitude NUMBER(5) NOT NULL Constraint
IS_CAL_pzt_E_amplitude CHECK(IS_cal_eve_pzt_E_amplitude>=0 and
IS_cal_eve_pzt_E_amplitude<=4095),

IS_cal_eve_pzt_E_timeOverflow NUMBER(1) NOT NULL Constraint
IS_CAL_pzt_E_timeOverflow CHECK(IS_cal_eve_pzt_E_timeOverflow=0 or
IS_cal_eve_pzt_E_timeOverflow=1),

IS_cal_eve_pzt_E_time FTYPE1(8,9) NUMBER(2) NOT NULL Constraint
IS_CAL_pzt_E_time CHECK(IS_cal_eve_pzt_E_time>=0 and
IS_cal_eve_pzt_E_time<=31),

PRIMARY KEY (ID,IS_cal_eve_ITEM),
FOREIGN KEY (ID) REFERENCES GDB_ISCalibration(ID) ON DELETE CASCADE
);

```

Semantic field

ID (0,999999999999). Unique Identifier. Millisecond (the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC) when the packet was inserted into the system.

IS_cal_eve_item(0,4095). Event position inside Science TM.

IS_cal_eve_pzt_A_detected(0,1). If value='1' then this PZT has detected a particle (threshold has been crossed from below threshold to over threshold) If value='0' (there is not threshold transition) then all the information relates with this PZT is useless. Related field: "pzt_A_timeOverflow".

IS_cal_eve_pzt_A_range(0,1). If value='1' this magnitude divide the total value of amplitude.

IS_cal_eve_pzt_A_gain(0,1). If value='1' this magnitude multiply the total value of amplitude, else not modify this value This value can be set to '0' due to autogain function (RO-GIA-IAA-RS-001 Req 1400).

IS_cal_eve_pzt_A_amplitude (0,4095) [PZT#A Amplitude]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

IS_cal_eve_pzt_A_timeOverflow (0,1). If this value='0' means one of following situations: No signal threshold transition (signal under threshold after impact or signal remains over the threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

IS_cal_eve_pzt_A_time(0,31) [PZT#A Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation needs to be detected starting from first PZT detection. The max value without overflow is '30'.

IS_cal_eve_pzt_B_detected(0,1). If value='1' then this PZT has detected a particle (threshold has been crossed from below threshold to over threshold)If value='0' (there is not threshold transition) then all the information relates with this PZT is useless. Related field: "pzt_B_timeOverflow".

IS_cal_eve_pzt_B_range (0,1). If value='1' this magnitude divide the total value of amplitude.

IS_cal_eve_pzt_B_gain (0,0). Always 0.

IS_cal_eve_pzt_B_amplitude (0,4095) [PZT#B Amplitude]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

IS_cal_eve_pzt_B_timeOverflow(0,1). If this value='0' means one of following situations: No signal threshold transition (signal under threshold after impact or signal remains over the threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

IS_cal_eve_pzt_B_time(0,31) [PZT#B Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation needs to be detected starting from first PZT detection. The max value without overflow is '30'.

IS_cal_eve_pzt_C_detected(0,1). If value='1' then this PZT has detected a particle (threshold has been crossed from below threshold to over threshold)If value='0' (there is not threshold transition) then all the information relates with this PZT is useless. Related field: "pzt_C_timeOverflow".

IS_cal_eve_pzt_C_range(0,1). If value='1' this magnitude divide the total value of amplitude.

IS_cal_eve_pzt_C_gain (0,0). Always 0.

IS_cal_eve_pzt_C_amplitude (0,4095) [PZT#C Amplitude]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

IS_cal_eve_pzt_C_timeOverflow(0,1). If this value='0' means one of following situations: No signal threshold transition (signal under threshold after impact or signal remains over the threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

IS_cal_eve_pzt_C_time(0,31) [PZT#C Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation needs to be detected starting from first PZT detection. The max value without overflow is '30'.

IS_cal_eve_pzt_D_detected(0,1). If value='1' then this PZT has detected a particle (threshold has been crossed from below threshold to over threshold)If value='0' (there is not threshold transition) then all the information relates with this PZT is useless. Related field: "pzt_D_timeOverflow".

IS_cal_eve_pzt_D_range(0,1). If value='1' this magnitude divide the total value of amplitude.

IS_cal_eve_pzt_D_gain(0,1). If value='1' this magnitude multiply the total value of amplitude, else not modify this value This value can be set to '0' due to autogain function (RO-GIA-IAA-RS-001 Req 1400).

IS_cal_eve_pzt_D_amplitude(0,4095) [PZT#D Amplitude]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

IS_cal_eve_pzt_D_timeOverflow (0,1). If this value='0' means one of following situations: No signal threshold transition (signal under threshold after impact or signal remains over the threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

IS_cal_eve_pzt_D_time (0,31) [PZT#D Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation needs to be detected starting from first PZT detection. The max value without overflow is '30'.

IS_cal_eve_pzt_E_detected(0,1). If value='1' then this PZT has detected a particle (threshold has been crossed from below threshold to over threshold)If value='0' (there is not threshold transition) then all the information relates with this PZT is useless. Related field: "pzt_E_time-Overflow".

IS_cal_eve_pzt_E_range(0,1). If value='1' this magnitude divide the total value of amplitude.

IS_cal_eve_pzt_E_gain(0,1). If value='1' this magnitude multiply the total value of amplitude, else not modify this value This value can be set to '0' due to autogain function (RO-GIA-IAA-RS-001 Req 1400).

IS_cal_eve_pzt_E_amplitude(0,4095) [PZT#E Amplitude]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

IS_cal_eve_pzt_E_timeOverflow(0,1). If this value='0' means one of following situations: No signal threshold transition (signal under threshold after impact or signal remains over the threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

IS_cal_eve_pzt_E_time(0,31) [PZT#E Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation needs to be detected starting from first PZT detection. The max value without overflow is '30'.

GDB_ISEVENT

This table stores the events detected by the IS.

SQL definition

```
CREATE TABLE GDB_ISEvent (
  ID NUMBER(16) NOT NULL,
  IS_ITEM NUMBER(2) NOT NULL Constraint IS_ITEM CHECK(IS_ITEM>=0),
  IS_eLabel NUMBER(5) NOT NULL Constraint IS_eLabel CHECK(IS_eLabel=26985),
```

```
IS_eventTime FTYPE1(8,9) NUMBER(15) NOT NULL Constraint IS_packetTime
CHECK(IS_eventTime>=0 and IS_eventTime<=281474976710655),

IS_pzt_A_detected NUMBER(1) NOT NULL Constraint IS_pzt_A_detected
CHECK(IS_pzt_A_detected=0 or IS_pzt_A_detected=1),

IS_pzt_A_range NUMBER(1) NOT NULL Constraint IS_pzt_A_range
CHECK(IS_pzt_A_range=0 or IS_pzt_A_range=1),

IS_pzt_A_gain NUMBER(1) NOT NULL Constraint IS_pzt_A_gain
CHECK(IS_pzt_A_gain=0 or IS_pzt_A_gain=1),

IS_pzt_A_amplitude NUMBER(5) NOT NULL Constraint IS_pzt_A_amplitude
CHECK(IS_pzt_A_amplitude>=0 and IS_pzt_A_amplitude<=4095),

IS_pzt_A_timeOverflow NUMBER(1) NOT NULL Constraint IS_pzt_A_timeOverflow
CHECK(IS_pzt_A_timeOverflow=0 or IS_pzt_A_timeOverflow=1),

IS_pzt_A_time FTYPE1(8,9) NUMBER(2) NOT NULL Constraint IS_pzt_A_time
CHECK(IS_pzt_A_time>=0 and IS_pzt_A_time<=31),

IS_pzt_B_detected NUMBER(1) NOT NULL Constraint IS_pzt_B_detected
CHECK(IS_pzt_B_detected=0 or IS_pzt_B_detected=1),

IS_pzt_B_range NUMBER(1) NOT NULL Constraint IS_pzt_B_range
CHECK(IS_pzt_B_range=0 or IS_pzt_B_range=1),

IS_pzt_B_gain NUMBER(1) NOT NULL Constraint IS_pzt_B_gain
CHECK(IS_pzt_B_gain=0 or IS_pzt_B_gain=1),

IS_pzt_B_amplitude NUMBER(5) NOT NULL Constraint IS_pzt_B_amplitude
CHECK(IS_pzt_B_amplitude>=0 and IS_pzt_B_amplitude<=4095),

IS_pzt_B_timeOverflow NUMBER(1) NOT NULL Constraint IS_pzt_B_timeOverflow
CHECK(IS_pzt_B_timeOverflow=0 or IS_pzt_B_timeOverflow=1),

IS_pzt_B_time FTYPE1(8,9) NUMBER(2) NOT NULL Constraint IS_pzt_B_time
CHECK(IS_pzt_B_time>=0 and IS_pzt_B_time<=31),

IS_pzt_C_detected NUMBER(1) NOT NULL Constraint IS_pzt_C_detected
CHECK(IS_pzt_C_detected=0 or IS_pzt_C_detected=1),

IS_pzt_C_range NUMBER(1) NOT NULL Constraint IS_pzt_C_range
CHECK(IS_pzt_C_range=0 or IS_pzt_C_range=1),

IS_pzt_C_gain NUMBER(1) NOT NULL Constraint IS_pzt_C_gain
CHECK(IS_pzt_C_gain=0 or IS_pzt_C_gain=1),
IS_pzt_C_amplitude NUMBER(5) NOT NULL Constraint IS_pzt_C_amplitude
CHECK(IS_pzt_C_amplitude>=0 and IS_pzt_C_amplitude<=4095),

IS_pzt_C_timeOverflow NUMBER(1) NOT NULL Constraint IS_pzt_C_timeOverflow
CHECK(IS_pzt_C_timeOverflow=0 or IS_pzt_C_timeOverflow=1),

IS_pzt_C_time FTYPE1(8,9) NUMBER(2) NOT NULL Constraint IS_pzt_C_time
CHECK(IS_pzt_C_time>=0 and IS_pzt_C_time<=31),

IS_pzt_D_detected NUMBER(1) NOT NULL Constraint IS_pzt_D_detected
CHECK(IS_pzt_D_detected=0 or IS_pzt_D_detected=1),

IS_pzt_D_range NUMBER(1) NOT NULL Constraint IS_pzt_D_range
CHECK(IS_pzt_D_range=0 or IS_pzt_D_range=1),

IS_pzt_D_gain NUMBER(1) NOT NULL Constraint IS_pzt_D_gain
CHECK(IS_pzt_D_gain=0 or IS_pzt_D_gain=1),

IS_pzt_D_amplitude NUMBER(5) NOT NULL Constraint IS_pzt_D_amplitude
CHECK(IS_pzt_D_amplitude>=0 and IS_pzt_D_amplitude<=4095),
```


IS_pzt_A_gain(0,1). If value='1' this magnitude multiply the total value of amplitude, else not modify this value This value can be set to '0' due to autogain function (RO-GIA-IAA-RS-001 Req 1400).

IS_pzt_A_amplitude(0,4095) [PZT#A Amplitude]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

IS_pzt_A_timeOverflow(0,1). If this value='0' means one of following situations: No signal threshold transition (signal under threshold after impact or signal remains over the threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

IS_pzt_A_time(0,31) [PZT#A Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation needs to be detected starting from first PZT detection. The max value without overflow is '30'.

IS_pzt_B_detected(0,1). If value='1' then this PZT has detected a particle (threshold has been crossed from below threshold to over threshold)If value='0' (there is not threshold transition) then all the information relates with this PZT is useless. Related field: "pzt_B_timeOverflow".

IS_pzt_B_range(0,1). If value='1' this magnitude divide the total value of amplitude.

IS_pzt_B_gain(0,0). Always 0.

IS_pzt_B_amplitude(0,4095) [PZT#B Time]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

IS_pzt_B_timeOverflow(0,1). If this value='0' means one of following situations: No signal threshold transition (signal under threshold after impact or signal remains over the threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

IS_pzt_B_time(0,31) [PZT#B Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation needs to be detected starting from first PZT detection. The max value without overflow is '30'.

IS_pzt_C_detected(0,1). If value='1' then this PZT has detected a particle (threshold has been crossed from below threshold to over threshold)If value='0' (there is not threshold transition) then all the information relates with this PZT is useless. Related field: "pzt_C_timeOverflow".

IS_pzt_C_range(0,1). If value='1' this magnitude divide the total value of amplitude.

IS_pzt_C_gain(0,0). Always 0.

IS_pzt_C_amplitude(0,4095) [PZT#C Amplitude]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

IS_pzt_C_timeOverflow(0,1). If this value='0' means one of following situations: No signal threshold transition (signal under threshold after impact or signal remains over the threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

IS_pzt_C_time(0,31) [PZT#C Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation needs to be detected starting from first PZT detection. The max value without overflow is '30'.

IS_pzt_D_detected(0,1). If value='1' then this PZT has detected a particle (threshold has been crossed from below threshold to over threshold)If value='0' (there is not threshold transition) then all the information relates with this PZT is useless. Related field: "pzt_D_timeOverflow".

IS_pzt_D_range(0,1). If value='1' this magnitude divide the total value of amplitude.

IS_pzt_D_gain(0,1). If value='1' this magnitude multiply the total value of amplitude, else not modify this value This value can be set to '0' due to autogain function (RO-GIA-IAA-RS-001 Req 1400).

IS_pzt_D_amplitude(0,4095) [PZT#D Amplitude]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

IS_pzt_D_timeOverflow(0,1). If this value='0' means one of following situations: No signal threshold transition (signal under threshold after impact or signal remains over the threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

IS_pzt_D_time(0,31) [PZT#D Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation needs to be detected starting from first PZT detection. The max value without overflow is '30'.

IS_pzt_E_detected(0,1). If value='1' then this PZT has detected a particle (threshold has been crossed from below threshold to over threshold)If value='0' (there is not threshold transition) then all the information relates with this PZT is useless. Related field: "pzt_E_timeOverflow".

IS_pzt_E_range(0,1). If value='1' this magnitude divide the total value of amplitude.

IS_pzt_E_gain(0,1). If value='1' this magnitude multiply the total value of amplitude, else not modify this value This value can be set to '0' due to autogain function (RO-GIA-IAA-RS-001 Req 1400).

IS_pzt_E_amplitude(0,4095) [PZT#E Amplitude]. This is the peak value of perturbation generated by the particle impact in IS Plate Max value without saturation '4094'.

IS_pzt_E_timeOverflow(0,1). If this value='0' means one of following situations: No signal threshold transition (signal under threshold after impact or signal remains over the threshold after last IS rest), secondary signal detected (if "detected"='1') If value='0' then the IS impact is valid If value='1' then all the information relates with this PZT is useless.

IS_pzt_E_time(0,31) [PZT#E Time]. This value='0' for the first PZT that detected the signal, in all the rest this value is the time (multiply by 3 microsecond) that the propagation needs to be detected starting from first PZT detection. The max value without overflow is '30'.

GDB_MBHEATING

This table stores all the information about the heating MBS process.

SQL definition

```

CREATE TABLE GDB_MBHeating(
ID NUMBER(16) NOT NULL PRIMARY KEY,
MB_Heat_eLabel NUMBER(5) NOT NULL Constraint MB_Heat_eLabel
CHECK(MB_Heat_eLabel>=26625 and MB_Heat_eLabel<=26629),
MB_Heat_eventTime FTYPE1(8,9) NUMBER(15) NOT NULL Constraint
MB_Heat_packetTime CHECK(MB_Heat_eventTime>=0 and
MB_Heat_eventTime<=281474976710655),
FOREIGN KEY (ID) REFERENCES GDB_PACKET(ID) ON DELETE CASCADE
);

```

Semantic field

ID(0,999999999999999999). Unique Identifier. Millisecond (the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC) when the packet was inserted into the system

MB_Heat_eLabel(26625,2709). Value ='0x6801' for Heating MB 1, Value ='0x6802' for Heating MB 2, Value ='0x6803' for Heating MB 3, Value ='0x6804' for Heating MB 4, Value ='0x6805' for Heating MB 5 If the MB that will be read or heated is disabled, then "frequencyOverflow" and "frequency" are always '0'.

MB_Heat_eventTime(0,281474976710655). MBS event time stamp GIADA precision is 4ms.

GDB_MBHEATINGEVENT

This table stores the events generated to heat the MBS.

SQL definition

```

CREATE TABLE GDB_MBHeatingEvent(
ID NUMBER(16) NOT NULL,
MB_Heat_eve_ITEM NUMBER(2) NOT NULL Constraint MB_Heat_Event_ITEM
CHECK(MB_Heat_eve_ITEM>=0),
MB_Heat_eve_frequencyOverflow NUMBER(1) NOT NULL Constraint
MB_HeatEvent_frequencyOverflow CHECK(MB_Heat_eve_frequencyOverflow=0 or
MB_Heat_eve_frequencyOverflow=1),
MB_Heat_eve_frequency NUMBER(8) NOT NULL Constraint MB_HeatEvent_frequency
CHECK(MB_Heat_eve_frequency>=0 and MB_Heat_eve_frequency<=16777215),
MB_Heat_eve_temperature NUMBER(5) NOT NULL Constraint
MB_HeatEvent_temperature CHECK(MB_Heat_eve_temperature>=0 and
MB_Heat_eve_temperature<=4095),
PRIMARY KEY (ID,MB_Heat_eve_ITEM),
FOREIGN KEY (ID) REFERENCES GDB_MBHeating(ID) ON DELETE CASCADE
);

```

Semantic field

ID(0,9999999999999999). Unique Identifier. Milisecond (the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC) when the packet was inserted into the system.

MB_Heat_eve_item(0,4095). Event position inside Science TM.

MB_Heat_eve_frequencyOverflow(0,1). If Value='1' then the field "frequency" is not valid, with value='0' it's valid. By HW limitations, a frequency overflow can not be reached.

MB_Heat_eve_frequency(0,16777215). This value is the number of cycles in 100352ms of beating signal of MB This magnitude is proportional to the mass deposited in MB and is function of MB Temperature Max value='16777215' without overflow

MB_Heat_eve_temperature (0,4095). This value is the MB temperature when the frequency was acquired.

GDB_MBREADINGEVENT

This table stores all the information about the MBS events

SQL definition

```
CREATE TABLE GDB_MBReadingEvent (
  ID NUMBER(16) NOT NULL,
  MB_Read_eve_ITEM NUMBER(2) NOT NULL Constraint MB_Read_Event_ITEM
  CHECK (MB_Read_eve_ITEM >= 0),
  MB_Read_eve_eLabel NUMBER(5) NOT NULL Constraint MB_Read_Event_eLabel
  CHECK (MB_Read_eve_eLabel >= 27905 and MB_Read_eve_eLabel <= 27909),
  MB_Read_eve_eventTime FTYPE1(8,9) NUMBER(15) NOT NULL Constraint
  MB_Read_Event_packetTime CHECK (MB_Read_eve_eventTime >= 0 and
  MB_Read_eve_eventTime <= 281474976710655),
  MB_Read_eve_frequencyOverflow NUMBER(1) NOT NULL Constraint
  MB_Read_Event_freqOverflow CHECK (MB_Read_eve_frequencyOverflow = 0 or
  MB_Read_eve_frequencyOverflow = 1),
  MB_Read_eve_frequency NUMBER(8) NOT NULL Constraint MB_Read_Event_frequency
  CHECK (MB_Read_eve_frequency >= 0 and MB_Read_eve_frequency <= 16777215),
  MB_Read_eve_temperature NUMBER(5) NOT NULL Constraint
  MB_Read_Event_temperature CHECK (MB_Read_eve_temperature >= 0 and
  MB_Read_eve_temperature <= 4095),
  PRIMARY KEY (ID, MB_Read_eve_ITEM),
  FOREIGN KEY (ID) REFERENCES GDB_ScienceHK(ID) ON DELETE CASCADE
);
```

Semantic field

ID(0,999999999999999999). Unique Identifier. Millisecond (the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC) when the packet was inserted into the system.

MB_Read_eve_item(0,4095). Event position inside Science TM.

MB_Read_eve_eLabel(26625,2709). Value = '0x6861' for Heating MB 1, Value = '0x6D01' for Reading MB 1, Value = '0x6D02' for Reading MB 2, Value = '0x6D03' for Reading MB 3, Value = '0x6D04' for Reading MB 4, Value = '0x6D05' for Reading MB 5. For MB event detection (Heating and Reading), the MB must be on and GIADA in Normal or Flux modes and besides only for MB Heating the GDS and IS must be off if the MB that will be read or heated is disabled, then "frequencyOverflow" and "frequency" are always '0'.

MB_Read_eve_eventTime(0,281474976710655). MBS event time stamp GIADA precision is 4ms.

MB_Read_eve_frequencyOverflow(0,1). If Value='1' then the field "frequency" is not valid, with value='0' it's valid. By HW limitations, a frequency overflow can not be reached.

MB_Read_eve_frequency(0,16777215). This value is the number of cycles in 100352ms of beating signal of MB. This magnitude is proportional to the mass deposited in MB and is function of MB Temperature. Max value='16777215' without overflow.

MB_Read_eve_temperature(0,4095). This value is the MB temperature when the frequency was acquired.

GDB_MBSCALIBRATION

This table stores the information about the IS calibration.

SQL definition

```
CREATE TABLE GDB_MBSCalibration(
  ID NUMBER(16) NOT NULL,

  MBS_cal_mb_1_Temperature NUMBER(5) NOT NULL Constraint
  MB_CAL_mb_1_Temperature CHECK (MBS_cal_mb_1_Temperature >= 0 and
  MBS_cal_mb_1_Temperature <= 4095),

  MBS_cal_mb_2_Temperature NUMBER(5) NOT NULL Constraint
  MB_CAL_mb_2_Temperature CHECK (MBS_cal_mb_2_Temperature >= 0 and
  MBS_cal_mb_2_Temperature <= 4095),

  MBS_cal_mb_3_Temperature NUMBER(5) NOT NULL Constraint
  MB_CAL_mb_3_Temperature CHECK (MBS_cal_mb_3_Temperature >= 0 and
  MBS_cal_mb_3_Temperature <= 4095),

  MBS_cal_mb_4_Temperature NUMBER(5) NOT NULL Constraint
  MB_CAL_mb_4_Temperature CHECK (MBS_cal_mb_4_Temperature >= 0 and
  MBS_cal_mb_4_Temperature <= 4095),
```



```

packet_sourceSequenceCount NUMBER(5) NOT NULL Constraint
Packet_sourceSequenceCount CHECK(packet_sourceSequenceCount>=0 and
packet_sourceSequenceCount<=16383),

packet_packetLength NUMBER(5) NOT NULL Constraint Packet_packetLength
CHECK(packet_packetLength>=14 and packet_packetLength<=4105),

packet_packetTime FTYP1(2,4) NUMBER(15) NOT NULL Constraint
Packet_packetTime CHECK(packet_packetTime>=0 and
packet_packetTime<=281474976710655),

packet_pusVersion NUMBER(1) NOT NULL Constraint Packet_pusVersion
CHECK(packet_pusVersion=0 or packet_pusVersion=2),

packet_checksumFlag NUMBER(1) NOT NULL Constraint Packet_sumFlag
CHECK(packet_checksumFlag=0),

packet_spare NUMBER(2) NOT NULL Constraint Packet_spare
CHECK(packet_spare=0),

packet_service NUMBER(2) NOT NULL Constraint Packet_service
CHECK(packet_service=3 or packet_service=20),

packet_subService NUMBER(2) NOT NULL Constraint Packet_subService
CHECK(packet_subService=3 or packet_subService=25),

packet_pad NUMBER(2) NOT NULL Constraint Packet_pad CHECK(packet_pad>=0 and
packet_pad<=255),

packet_crc NUMBER(5) NOT NULL Constraint Packet_crc CHECK(packet_crc>=0 and
packet_crc<=65535),

PRIMARY KEY (ID)
);

```

Semantic field

ID(0,999999999999). Unique Identifier. Millisecond (the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC) when the packet was inserted into the system.

Packet_versionNumber(0,0). Number of Version Packet Must be always '0'.

type (0,1). The type shall be set to '0' for all TM, 1 for all TC.

Packet_dataFieldHeaderFlag(1,1). Indicates the presence of the Data Field Header and must be always '1'.

Packet_apid(1444,1452). The Application Process ID field shall uniquely identify the on-board source of the packet (experiment) Each Application Process ID is logically associated with the Source Sequence Count sub-field of the Packet Sequence Control field Implying that a separate counter must be maintained for each different application ID used GIADA process ID='90' Category=4 (HK) for HK packets (value after shifts='1444' Category=12 (Private) for Science packets (value after shifts='1452').

Packet_segmentationFlags(3,3). In the Source packet, the Segmentation Flags shall always be set to '3' indicating that the source packet is not segmented by the data source.

Packet_sourceSequenceCount(0,16383). The Source Sequence Count field shall represent the actual sequence count (incremented per packet of a given APID) That is : a separate

source sequence count shall be maintained for each APID and shall be incremented by 1 whenever the source (APID) releases a packet Ideally, this counter shall never re-initialise, however under no circumstances shall it “short-cycle” (ie. have a discontinuity other than to a value zero) The counter shall wrap around from 214-1 to zero, and shall start at zero at power on of the unit The counter corresponds to the order of release of packets by the source and enables the ground to detect missing packets of a given APID.

Packet_packetLength(14,4105). The Packet Length field shall specify the number of octets (minus one) contained within the Packet Data Field as an unsigned binary value expressing (Number of octets in Data Field - 1) For Rosetta, the maximum length of a Telemetry Source Packet Data Field is 4106 octets, including 4096 octets of source data and 10 data field header octets (there being no error control on source TM packets) Min value includes size of Data Field Header(10 bytes), at least 2 bytes of data and CRC (2 bytes) The max value have incorporated the minus 1 count.

Packet_packetTime(0,281474976710655). Defines the time that the acquisition of the data within the packet was finished The time code format is provided 4 octets of unit seconds followed by 2 octets of fractional seconds.

Packet_pusVersion(0,2). Refers to the associated ESA Packet Utilisation Standard (PUS) This field shall be used to provide the DMS with additional routing information, as follows: value = '0' for Science data (service 20,3 and 20,13) Value = '2' for all other TM reports.

Packet_checksumFlag(0,0). Indicates if there is a Packet Error Control Field at the end of the Packet Data Field There is no error control for TM source packets, therefore this field shall be set to value '0'.

Packet_spare(0,0). Shall be set to all zeros.

Packet_service(3,20). Indicates the Packet Service type to which the telemetry source packet relates HK Service value='3', Science Service value='20'

Packet_subService(3,25). Together with the Type field, the Subtype shall uniquely identify the nature of the telemetry contained within the telemetry source packet. HK subservice value='25', Science subservice value='3'.

Packet_pad(0,255). This field is placed into the header to ensure that the data contained in the packet source data field begins on a 16-bit boundary This field shall be used to provide the DMS with additional routing information, as follows: the value shall be zero for all TM reports, except where the TM report is a direct response to a TC packet as specified in EID-A section 28 table 282-2 Such TM packets are referred to a ‘solicited’ telemetry, in this case the Pad field of the acknowledged TC shall be copied into the Pad field of TM report.

Packet_crc(0,65535). The Packet Error Control field provides an error detection code (checksum) in the packet, allowing the receiving application to verify the integrity of the telecommand packet data The checksum shall be calculated over the complete packet less the final 16 bits (ie. excluding the Packet Error Control field itself) The specification of the checksum method selected (CRC checksum) is given in EID-C part 04 This field is not necessary in TM, but all Science and HK TM have it.

GDB_SCIENCEHK

This table stores all the common information for GDS, IS and MBS events.

SQL definition

```
CREATE TABLE GDB_ScienceHK(
  ID NUMBER(16) PRIMARY KEY,

  sci_hk_eLabel NUMBER(5) NOT NULL Constraint SciHK_eLabel
  CHECK(sci_hk_eLabel=18507),

  sci_hk_laser_1_Temperature NUMBER(5) NOT NULL Constraint
  SciHK_laser_1_Temperature CHECK(sci_hk_laser_1_Temperature>=0 and
  sci_hk_laser_1_Temperature<=4095),

  sci_hk_laser_2_Temperature NUMBER(5) NOT NULL Constraint
  SciHK_laser_2_Temperature CHECK(sci_hk_laser_2_Temperature>=0 and
  sci_hk_laser_2_Temperature<=4095),

  sci_hk_laser_3_Temperature NUMBER(5) NOT NULL Constraint
  SciHK_laser_3_Temperature CHECK(sci_hk_laser_3_Temperature>=0 and
  sci_hk_laser_3_Temperature<=4095),

  sci_hk_laser_4_Temperature NUMBER(5) NOT NULL Constraint
  SciHK_laser_4_Temperature CHECK(sci_hk_laser_4_Temperature>=0 and
  sci_hk_laser_4_Temperature<=4095),

  sci_hk_laser_1_Light NUMBER(5) NOT NULL Constraint SciHK_laser_1_Light
  CHECK(sci_hk_laser_1_Light>=0 and sci_hk_laser_1_Light<=4095),

  sci_hk_laser_2_Light NUMBER(5) NOT NULL Constraint SciHK_laser_2_Light
  CHECK(sci_hk_laser_2_Light>=0 and sci_hk_laser_2_Light<=4095),

  sci_hk_laser_3_Light NUMBER(5) NOT NULL Constraint SciHK_laser_3_Light
  CHECK(sci_hk_laser_3_Light>=0 and sci_hk_laser_3_Light<=4095),

  sci_hk_laser_4_Light NUMBER(5) NOT NULL Constraint SciHK_laser_4_Light
  CHECK(sci_hk_laser_4_Light>=0 and sci_hk_laser_4_Light<=4095),

  sci_hk_isTemperature NUMBER(5) NOT NULL Constraint SciHK_isTemperature
  CHECK(sci_hk_isTemperature>=0 and sci_hk_isTemperature<=4095),

  FOREIGN KEY (ID) REFERENCES GDB_PACKET(ID) ON DELETE CASCADE
);
```

Semantic field

ID(0,9999999999999999). Unique Identifier. Millisecond (the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC) when the packet was inserted into the system.

Sci_hk_eLabel(18507,18507). Science HK label. Ignore this value if GDS is off.

Sci_hk_laser_1_Temperature(0,4095) [GDS Laser 1 Temperature]. Temperature of the Laser 1 Ignore this value if GDS is off.

Sci_hk_laser_2_Temperature(0,4095) [GDS Laser 2 Temperature]. Temperature of the Laser 2 Ignore this value if GDS is off.

Sci_hk_laser_3_Temperature(0,4095) [GDS Laser 3 Temperature]. Temperature of the Laser 3 Ignore this value if GDS is off.

Sci_hk_laser_4_Temperature(0,4095) [GDS Laser 4 Temperature]. Temperature of the Laser 4 Ignore this value if GDS is off.

Sci_hk_laser_1_Light(0,4095) [GDS Laser 1 Light Monitor]. Light of the Laser 1 Ignore this value if GDS is off.

Sci_hk_laser_2_Light(0,4095) [GDS Laser 2 Light Monitor]. Light of the Laser 2 Ignore this value if GDS is off.

Sci_hk_laser_3_Light(0,4095) [GDS Laser 3 Light Monitor]. Light of the Laser 3 Ignore this value if GDS is off.

Sci_hk_laser_4_Light(0,4095) [GDS Laser 4 Light Monitor]. Light of the Laser 4 Ignore this value if GDS is off.

Sci_hk_isTemperature(0,4095) [IS Plate Temperature]. Temperature of the IS.

GDB_WORKSESSION

This table stores the initial and end packet of an user session.

SQL definition

```
CREATE TABLE GDB_WorkSession(
  ID NUMBER(16) PRIMARY KEY,

  work_ID_End NUMBER(16) NOT NULL,

  work_mainBoard NUMBER(1) NOT NULL Constraint Session_Board
  CHECK(work_mainBoard=0 or work_mainBoard=1),

  work_name CHAR(128) NOT NULL UNIQUE,

  FOREIGN KEY (ID) REFERENCES GDB_PACKET(ID) ON DELETE CASCADE,
  FOREIGN KEY (work_ID_End) REFERENCES GDB_PACKET(ID) ON DELETE CASCADE
);
```

Semantic field

ID(0, 9999999999999999). Packet ID of the first packet of the session.

Work_ID_END(0, 9999999999999999). Packet ID of the last packet of the session.

Work_mainBoard(0,1). GIADA data board that generated the stored data. Value 0=main board, value=1 redundant board.

Work_name(0,0). Session name.

10.2 Apéndice B. Contenido del DVD

En el DVD anexo a esta memoria se podrán encontrar los siguientes directorios:

- 1) **Tesis doctoral.** Se incluye en formato PDF la tesis doctoral y la documentación aneja.
- 2) **GDB-GUI.** Código fuente y compilado de la interfaz del modelo de datos GDB.
- 3) **Gramática del simulador de eventos.** Ver [6.5](#).
- 4) **Instalación y configuración del SGBDR.** Documentos de instalación y configuración del servidor Oracle© ([6.2.10](#)).

10.3 Apéndice C. Introducción a los cometas

Podemos considerar a los cometas como unos pequeños, frágiles e irregulares cuerpos compuestos por una mezcla de partículas sólidas y gases helados. Sólo son visibles cuando se encuentran lo suficientemente cerca del sol, debido a los gases fluorescentes del *coma* y a luz reflejada por el propio cometa. El *coma* (o cabellera) es la atmósfera que rodea a un cometa y se compone de gases y pequeños trozos de material expulsados por la acción de la radiación solar.

El núcleo del cometa es sólido y a veces es llamado conglomerado de hielo o bola sucia. Un núcleo de cometa puede tener pequeñas partículas de silicatos semejantes a las rocas más comunes de la Tierra. La mayor parte de los núcleos son negros debido a la alta concentración de compuestos del carbono. En la juventud de los cometas, aparecen en su composición gases helados semejantes a los que componen el agua. Se piensa que el agua constituye el 75-80% del material volátil de un cometa, y que debido a las condiciones de frío y baja presión del espacio profundo, pasa directamente de sólido a gas (sublimación). En un cometa pueden aparecer también otros gases como dióxido de carbono, metano o amoníaco. Conforme el cometa desarrolla su órbita alrededor del Sol, va perdiendo parte del conglomerado de gases y partículas debido a la radiación solar.

Un cometa es poco masivo, por lo que su campo gravitatorio es pequeño. Se calcula que la velocidad de escape para salir del campo del cometa es de 1 m/s, lejos de los 11 Km/s de la Tierra. Esto provoca que los gases y las pequeñas partículas que arrastra, nunca aterricen en la superficie del cometa, pero sí que se mantengan entorno al núcleo.

La presión de la radiación solar, fuerza a que las partículas creen colas de polvo que pueden extenderse millones de kilómetros. Además, las moléculas de los gases son atacadas por la luz ultravioleta proveniente del Sol, lo que provoca que pierdan electrones, convirtiéndolos en iones. Estos iones interactúan con las partículas solares cargadas eléctricamente, construyendo una segunda cola, llamada cola de iones. Esta cola de iones al contacto con la luz del Sol se hace fluorescente y puede ser vista desde la Tierra. Alrededor del cometa también se desarrolla una tenue envoltura de hidrógeno debida a los procesos químicos producidos por la absorción de luz ultravioleta.

Los componentes principales de un cometa se muestran en la siguiente ilustración:

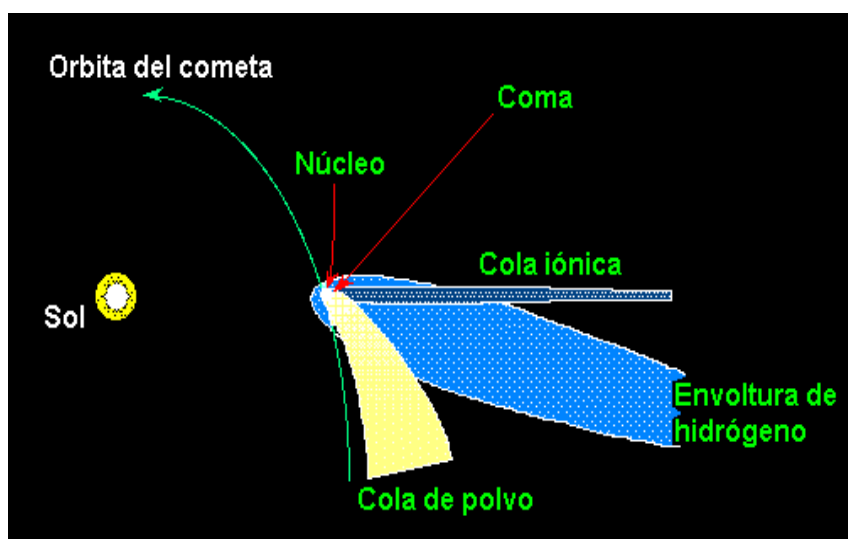


Ilustración 133. Componentes de un cometa.

Los cometas están ligados al sistema gravitatorio del Sol y normalmente siguen recorridos extremadamente largos.

Los cometas son muy pequeños en relación a los planetas. Su diámetro medio se encuentra en el rango de los 750 metros a los 20 kilómetros. Recientemente, se ha podido comprobar que existen cometas de más de 300 Km. Además, se supone que son muy frágiles. Debido a su reducido tamaño, es bastante difícil estudiarlos desde la Tierra. Siempre aparecen como un punto luminoso, incluso cuando se utiliza un gran telescopio.

El movimiento de los cometas

Las leyes de movimiento universal explican las órbitas tan extensas de los cometas. En una órbita, se denomina “perihelio” a la posición de la cometa más cercana al Sol y “aphelio” a la más alejada.

Si se consideran dos cuerpos, sea el Sol y un cometa, el cuerpo de menor masa, en este caso el cometa, describe una órbita elíptica entorno al cuerpo de mayor masa (el Sol) que se sitúa en uno de los focos de la elipse. La excentricidad es una medida matemática que parte de la circularidad. Un círculo tiene excentricidad cero. Normalmente, la órbita de los planetas tiene una excentricidad próxima al cero, a diferencia de los cometas que tiende a 1.

Júpiter, debido a sus enormes dimensiones, posee un más que apreciable campo gravitatorio que afecta directamente a la órbita de los cometas y del resto de cuerpos del Sistema Solar. Los cometas pueden caer presa de su campo gravitatorio, convirtiéndose en satélites, estrellándose en su superficie (como pasó con el cometa Shoemaker-Levy 9), o bien ser acelerados con la suficiente energía para superar la velocidad de escape del Sistema Solar.

Una de las leyes del movimiento de Newton, indica que, para cada acción existe una reacción igual en magnitud y de sentido opuesto. Los cometas expulsan polvo y gas desde regiones localizadas dentro de la superficie de su núcleo. Esto provoca un aceleramiento o deceleramiento del núcleo a modo de motor de cohete. Estos efectos se denominan fuerzas no gravitacionales y afectan al tamaño y a la forma de la órbita del cometa. Para hacer cálculos fiables de predicción, es necesario contar con estas fuerzas no gravitacionales.

10.4 Apéndice D. El cometa 67P/Churyumov-Gerasimenko

El cometa 67P/Churyumov-Gerasimenko (6CG) fue descubierto en septiembre de 1969 cuando unos astrónomos de Kiev visitaban el Instituto de Astrofísica de Alma-Ata en Kazajistán. Klim Churyumov examinaba unas placas fotográficas del cometa 32P/Comas-Solá tomadas por Svetlana Gerasimenko, cuando descubrió a 6CG.

6GC orbita entorno al Sol cada 6.57 años, oscilando entre Júpiter y la Tierra. Ha sido seleccionado por ESA como objetivo de la misión Rosetta, tras descartar al cometa 46/P Wirtanen.

Gracias a Rosetta y los instrumentos a bordo, se está realizando un estudio detallado del 6GC. Algunas de sus características principales se muestran a continuación:

Tabla 53. Características principales del cometa 67P/Churyumov-Gerasimenko.

Año de descubrimiento	1969
Descubridores	K. Churyumov, University of Kiev, Ukraine S. Gerasimenko. Institute of Astrophysics, Dushanbe, Tajikistan
Período de rotación	12 horas
Período de su órbita	6.57 años
Perihelio	1.29 UA
Aphelio	5.74 UA
Excentricidad de la órbita	0.632
Inclinación de la órbita	7.12 grados
Volumen	21.4 km ³
Masa	10 ¹³ kg
Densidad	470 kg/m ³
Porosidad	70% - 80%
Temperatura de la superficie	de -93°C a -43°C
Temperatura de la bajo la superficie	de -243°C a -113°C

Para finalizar esta memoria, se ofrecen imágenes del cometa 6CG en orden cronológico. Inicialmente, desde los telescopios, aparecía como un punto borroso en el espacio profundo y gracias a Rosetta ya se pueden adquirir imágenes desde sólo decenas de kilómetros.



Ilustración 134. Cometa 6CG visto desde el VLT (2013), ESO, Cerro Paranal, Chile. Créditos: ESO.

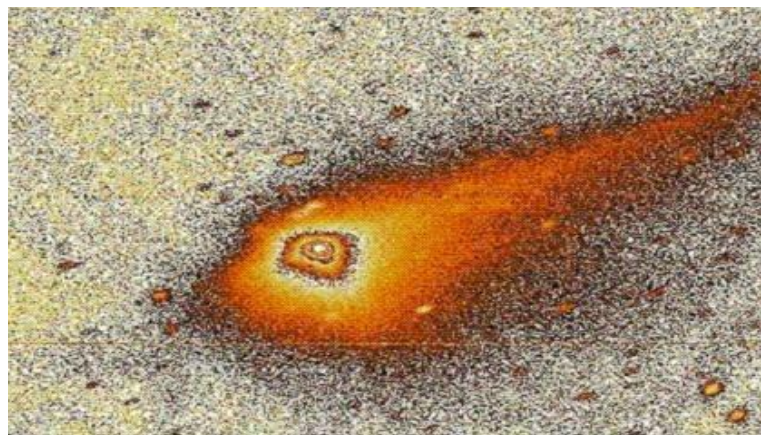


Ilustración 135. Cometa 6CG visto con el instrumento FORS2 (2014). Telescopio Antu de 8.2m, VLT, ESO, Cerro Paranal, Chile. Créditos: ESO.



Ilustración 136. Núcleo, coma y cola de polvo de 6GC (2015). Créditos: ESA, Rosetta spacecraft, NAVCAM.

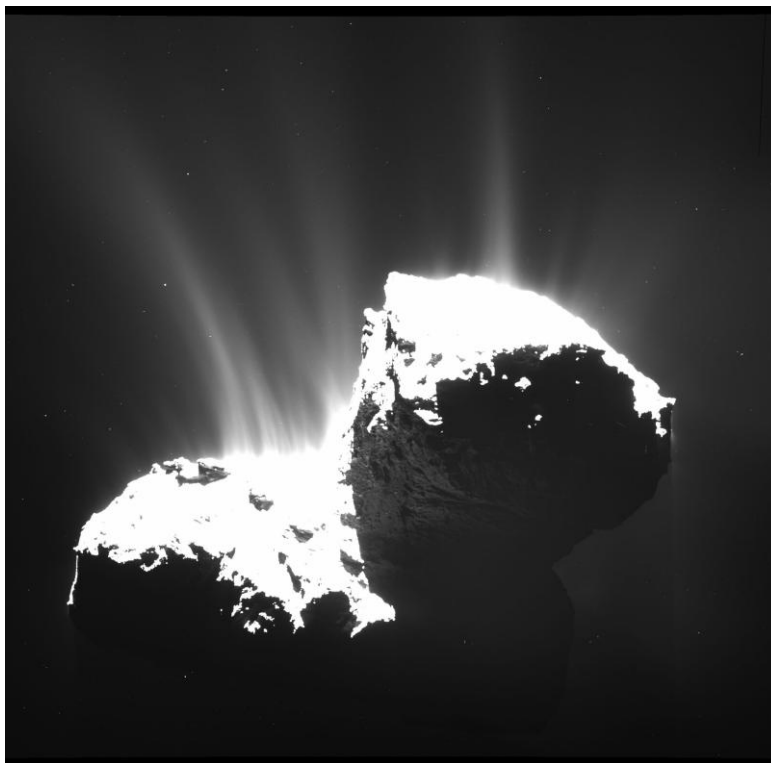


Ilustración 137. Coma y sublimación en 6CG observado desde Rosetta (2015). Créditos: ESA/Rosetta/MPS for OSIRIS Team MPS/UPD/LAM/IAA/SSO/INTA/UPM/DASP/IDA.

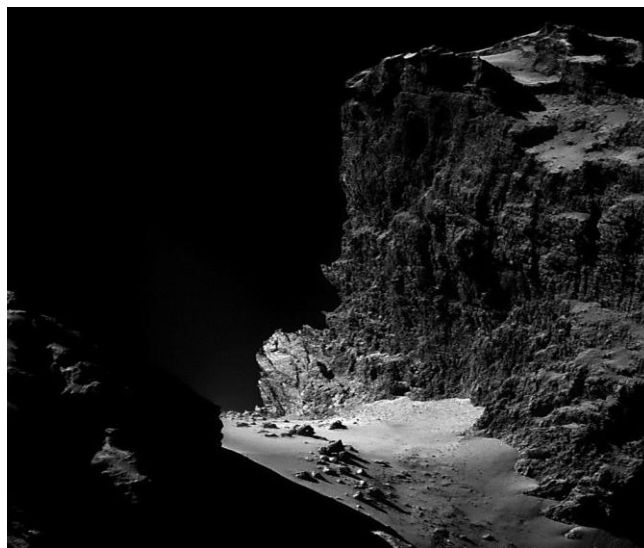


Ilustración 138. 6CG observado a decenas de kilómetros desde Rosetta (2015). Créditos: ESA, Rosetta spacecraft, NAVCAM.



Ilustración 139. 6CG observado a 2 metros desde la cámara 3 del instrumento CIVA de Philae en el lugar de aterrizaje de Philae (Julio 2015). Créditos: ESA/Rosetta/Philae/CIVA.