

***Pg\_KDD - Entorno Gráfico para el Sistema de Descubrimiento de  
Conocimiento en Bases de Datos PostgresKDD***

**WILMER JEFFERSSON CUCHALA ARTEAGA  
FRANCISCO JAVIER URBANO ROSALES**

**UNIVERSIDAD DE NARIÑO  
FACULTAD DE INGENIERÍA  
PROGRAMA DE INGENIERÍA DE SISTEMAS  
SAN JUAN DE PASTO  
AGOSTO 2012**

***Pg\_KDD - Entorno Gráfico para el Sistema de Descubrimiento de  
Conocimiento en Bases de Datos PostgresKDD***

**WILMER JEFFERSSON CUCHALA ARTEAGA  
FRANCISCO JAVIER URBANO ROSALES**

**TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARCIAL  
PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS**

**PhD. Ricardo Timarán Pereira  
DIRECTOR**

**UNIVERSIDAD DE NARIÑO  
FACULTAD DE INGENIERÍA  
PROGRAMA DE INGENIERÍA DE SISTEMAS  
SAN JUAN DE PASTO  
AGOSTO 2012**

## **NOTA DE RESPONSABILIDAD**

***“las ideas y conclusiones aportadas en el presente trabajo de grado son responsabilidad del autor”***

***Artículo 1 del Acuerdo No. 324 del 11 de octubre de 1966, emanado del Honorable Concejo Directivo de la Universidad de Nariño***

Nota de aceptación

---

---

---

---

---

---

---

Firma del presidente del jurado

---

Firma del representante del jurado

---

Firma del jurado

---

Firma del director del proyecto

San Juan de Pasto, Agosto de 2012

## **AGRADECIMIENTOS**

Agradecemos a Dios, por ser quien ha estado a nuestro lado a cada momento, brindándonos energía y fortaleza constantes en cada paso de nuestra carrera.

A nuestros familiares por su apoyo, comprensión, colaboración incondicional y permanente.

Al Doctor Ricardo Timaran Pereira PhD, Asesor de nuestro proyecto. Por su tiempo, paciencia y continua asistencia durante el desarrollo del proyecto.

A nuestros amigos y compañeros, por ser partícipes en el proceso de esta etapa de nuestra vida.

A nuestros profesores por compartir con nosotros sus conocimientos y experiencias.

Y a todas aquellas personas que contribuyeron en la realización de este proyecto de investigación.

## DEDICATORIA

A Dios por que en su infinita sabiduría me doto de herramientas maravillosas para ir en busca de mi meta y de una u otra forma guiar mi camino.

A mi padre que cultivó en forma perseverante en mí el espíritu de lucha y superación.

A mi madre quien siempre tuvo una palabra de aliento, una voz de consuelo en los momentos difíciles y me enseñaron que no hay que enflaquecer nunca en los retos de la vida, la vida es para los valientes y ser valiente lo aprendí de ella.  
Alcanzar este peldaño en mi vida es merito también de ellos.

A mis hermanos que han creído en mí y me han apoyado incondicionalmente para alcanzar este logro.

A mis docentes por su sabiduría, conocimiento y ejemplo, por que me motivaron a continuar.

A todos quienes estuvieron guiándome para que este sueño se haga realidad.

Hoy les dedico este trabajo de grado porque en cada página hay algo de cada uno de ellos.

*Francisco Javier. Urbano Rosales.*

A mi madre, quien me enseñó el valor de la lectura y el continuo aprendizaje.

A mi padre, un ejemplo de superación y búsqueda continua de nuevos horizontes.

A ellos gracias por el gran apoyo que me brindaron desde niño, con sus enseñanzas me guiaron a la realización de este sueño.

A mi amado hijo Thomas, la luz de mi vida, mi inspiración y mi más grande logro, a ti que llegaste para cambiarlo todo, para llenarlo de luz y de alegría, para ti y por ti es este logro.

A mi querida Sandra, mi niña, gracias por todo, tú le diste nuevas fuerzas a mi vida, me regalaste lo más grande que puede una mujer dar, nuestro hijo, nuestra esperanza, nuestro futuro, gracias por creer en mí cuando ni siquiera yo lo hacía.  
Simplemente gracias.

A mis hermanos quienes siempre han estado allí listos para darme una mano en cualquier situación.

A todos mis maestros, por su dedicación a la formación de un mejor mañana.  
A todos mis amigos, compañeros y todos con quienes compartí en esta etapa de mi vida.

Hoy les dedico este trabajo de grado, su paciencia y enseñanzas lo hicieron posible.

*Wilmer Jefferesson Cuchala Arteaga.*

## RESUMEN

Este documento contiene el análisis y diseño del trabajo de grado: “***Pg\_KDD-entorno gráfico para el sistema de descubrimiento de conocimiento en bases de datos PostgresKDD***”

Pg\_KDD es un ambiente gráfico para el sistema de descubrimiento de conocimiento PostgresKDD, una herramienta computacional fuertemente acoplada con el SGBD PostgreSQL, que permite al usuario final, interactuar de una manera amigable con el motor KDD y extraer conocimiento de las bases de datos de una manera fácil.

Esta herramienta gráfica está dividida en tres módulos principales, los cuales son:

Módulo interfaz gráfica: este módulo administra la interfaz gráfica de Pg\_KDD.

*Módulo kernel Pg\_KDD:* en este módulo se administran todos los procesos referentes a conexiones al sistema, administración de bases de datos y demás procesos derivados a esta, como son administración de tablas, columnas, restricciones, reglas, disparadores, entre otros.

Este módulo contiene los siguientes submódulos:

Sentencias gráficas: en este submódulo se representa el resultado de sentencias SQL obtenidas del manejo gráfico de relaciones de una base de datos, en donde gráficamente y de manera manual se puede realizar selecciones, uniones, ordenamiento, aplicación de funciones de agregación y agrupamiento.

Minería de datos: en este submódulo se gestiona de manera gráfica los operadores, primitivas y algoritmos implementados en PostgresKDD, para las tareas de minería de datos asociación y clasificaciones, con el fin de descubrir conocimiento a partir de los datos seleccionados.

Utilidades: en este módulo se gestiona la conexión con el sistema de descubrimiento de conocimiento en base de datos PostgresKDD, al igual que el manejo de scripts SQL y la visualización de las tablas de las base de datos.

## **ABSTRACT**

This paper contains the analysis and the design of the Undergraduate thesis: "Pg\_KDD a graphical environment for the knowledge discovery database system PostgresKDD"

Pg\_KDD is a graphical environment for the knowledge discovery system PostgresKDD, a computational tool strongly coupled with the SGDB PostgreSQL, that allows end users to interact friendly with the KDD engine and extract knowledge from databases in an easy way.

This graphical tool is divided into three main modules, which are:

The graphic user interface module: this module is administered by Pg\_KDD gui.

The Pg\_KDD kernel module: in this module we can manage all the processes relating to connections to the system, database administration and other processes arising from this, such as administration of tables, columns, constraints, rules, triggers, among others.

This module contains the following submodules:

Case graphics: in this sub-module we can show the result of cases SQL obtained from the graphical management of databases relations, and where the user is able to make selections, unions, management, functions application aggregation and grouping in a graphically and manually way.

Data mining: in this sub-module we can manage in a graphical way the operators, primitives and algorithms implemented in PostgresKDD, for data mining tasks of association and rankings, in order to discover knowledge from the selected data.

Utilities: this module manages connection with the system of knowledge discovery database PostgresKDD, SQL scripts and tables display of database.

## CONTENIDO

|  |           |
|--|-----------|
| <b>INTRODUCCIÓN</b>  | <b>20</b> |
| <b>1. TEMA</b>   | <b>21</b> |
| <b>2. MARCO TEORICO</b>  | <b>26</b> |
| <b>2.1 ANTECEDENTES</b>  | <b>26</b> |
| 2.1.1 PostgresKDD  | 26        |
| 2.1.2 TariyKDD   | 27        |
| 2.1.3 MATE-KDD   | 28        |
| <b>2.2 EL PROCESO DE DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS- DCBD</b>      | <b>29</b> |
| 2.2.1 Minería de Datos   | 29        |
| 2.2.2 Tareas de Minería de Datos   | 30        |
| 2.2.3 Tarea de Asociación  | 31        |
| 2.2.4 Tarea de Clasificación   | 33        |
| 2.2.5 Arquitecturas de Integración de DCBD con un SGBD                               | 34        |
| <b>2.3 METODOLOGÍA PARA EL DESARROLLO DE PG_KDD</b>                                  | <b>36</b> |
| 2.3.1 Análisis y diseño de sistemas usando la metodología orientada a objetos (O.O). | 36        |
| 2.3.2 El lenguaje unificado de modelado (UML).                                       | 38        |
| 2.3.3 El modelo del proceso unificado (RUP).   | 44        |
| <b>2.4 HERRAMIENTAS DE DESARROLLO</b>  | <b>47</b> |
| 2.4.1 Lenguaje de Programación Java.   | 47        |
| 2.4.2 Entorno de Desarrollo NetBeans   | 50        |
| 2.4.3 Controlador JDBC   | 51        |
| 2.4.4 Manejador de Protocolo Nativo (tipo 4)   | 52        |
| 2.4.5 Biblioteca gráfica Swing de Java   | 52        |
| 2.4.6 Look and Feel (Skin para aplicaciones JAVA)                                    | 55        |
| 2.4.6.1 NimROD (Look & Feel)   | 56        |
| 2.4.6.2 InfoNode (Doking Windows – Tabbed Panel)                                     | 58        |

### **3. OPERADORES ALGEBRAICOS, PRIMITIVAS SQL Y ALGORITMOS PARA LAS TAREAS DE ASOCIACIÓN Y CLASIFICACIÓN** **62**

|  |           |
|--|-----------|
| <b>3.1 Operadores del Álgebra Relacional y Nuevas primitivas SQL para Asociación</b> | <b>62</b> |
| 3.1.1 Operador Associator y la primitiva Associator Range                            | 62        |
| 3.1.2 Operador Assorow y la primitiva Assorow Range                                  | 68        |
| 3.1.3 Operador EquiKeep y la primitiva EquiKep On                                    | 73        |
| 3.1.4 Operador Describe Associator y la primitiva Describe Association Rules         | 78        |
| 3.1.5 EquipAsso: un Algoritmo para el Descubrimiento de Reglas de Asociación         | 81        |
| <b>3.2 Operadores del Álgebra Relacional y Primitivas SQL para Clasificación</b>     | <b>83</b> |
| 3.2.1 Operador Mate y la primitiva Mate By   | 84        |
| 3.2.2 Función algebraica agregada Entro y Función agregada SQL Entro                 | 89        |
| 3.2.3 Función algebraica agregada Gain y Función agregada SQL Gain                   | 92        |
| 3.2.4 Operador Describe Classifier y SQL Describe Classification Rules               | 94        |
| 3.2.5 Mate-Tree: un Algoritmo de Clasificación por Árboles de Decisión               | 97        |

### **4. DESARROLLO DE Pg\_KDD** **104**

|   |            |
|---|------------|
| <b>4.1 ANÁLISIS UML</b>   | <b>104</b> |
| 4.1.1 Requerimientos  | 104        |
| 4.1.2 Casos de uso  | 109        |
| 4.1.3 Diagramas de casos de uso   | 113        |
| 4.1.3.1 Diagrama de Caso de Uso: Diagrama General                             | 113        |
| 4.1.3.2 Diagrama de Caso de Uso: Conectar                                     | 114        |
| 4.1.3.3 Diagrama Caso de Uso: Administrar Bases de Datos                      | 115        |
| 4.1.3.4 Diagrama de Caso de Uso: Realizar de Consultas a Bases de Datos       | 116        |
| 4.1.3.5 Diagrama de Caso de Uso: Administrar Consultas SQL Graficas.          | 117        |
| 4.1.3.6 Diagrama de Caso de Uso: Realizar de Consultas Graficas               | 118        |
| 4.1.3.7 Diagrama de Caso de Uso: Ordenar Resultados de las Consultas Graficas | 119        |
| 4.1.3.8 Diagrama de Caso de Uso: Gestionar Tablas                             | 120        |
| 4.1.3.9 Diagrama de Caso de Uso: Manipular el Contenido de una Tabla          | 121        |
| 4.1.3.10 Diagrama de Caso de Uso: Gestionar Columnas                          | 122        |
| 4.1.3.11 Diagrama de Caso de Uso: Gestionar Funciones                         | 123        |
| 4.1.3.12 Diagrama de Caso de Uso: Gestionar Vistas                            | 124        |
| 4.1.3.13 Diagrama de Caso de Uso: Gestionar Reglas                            | 125        |
| 4.1.3.14 Diagrama de Caso de Uso: Gestionar Triggers                          | 126        |
| 4.1.3.15 Diagrama de Caso de Uso: Aplicar Técnicas de Minería de Datos        | 127        |
| 4.1.3.16 Diagrama de Caso de Uso: Manipular scripts SQL                       | 128        |

|  |            |
|--|------------|
| <b>4.2 DIAGRAMA DE SECUENCIA DEL SISTEMA</b>                                 | <b>129</b> |
| <b>4.3 DISEÑO UML</b>  | <b>130</b> |
| 4.3.1 Diagramas de clases  | 130        |
| 4.3.1.1 Diagrama de Clases: Inicio de la Aplicación                          | 130        |
| 4.3.1.2 Diagrama de Clases: Administración Visual PG_KDD                     | 131        |
| 4.3.1.3 Diagrama de Clases: Árbol de Elementos                               | 133        |
| 4.3.1.4 Diagrama de Clases: Administración Lógica de las Sentencias Gráficas | 133        |
| 4.3.1.5 Diagrama de Clases: Consultas Graficas                               | 134        |
| 4.3.1.6 Diagrama de Clases: Administración Lógica KDD                        | 135        |
| 4.3.1.7 Diagrama de Clases: Administracion de servicios graficos KDD         | 136        |
| 4.3.1.8 Diagrama de Clases: Utilidades                                       | 137        |
| 4.3.2 Diagramas de paquetes  | 138        |
| 4.3.2.1 Diagrama: Paquete Principal  | 138        |
| 4.3.2.2 Diagrama: Paquete Class  | 139        |
| 4.3.2.3 Diagrama: Paquete Gui  | 140        |
| 4.3.2.4 Diagrama: Paquete GraphicSentences                                   | 140        |
| 4.3.2.5 Diagrama: Paquete KDD  | 142        |
| 4.3.2.6 Diagrama: Paquete Utilities  | 142        |
| <b>4.4 ARQUITECTURA DE Pg_KDD</b>  | <b>143</b> |
| 4.4.1 Modulo de Interfaz Grafica   | 144        |
| 4.4.2 Modulo Gestión: KERNEL Pg_KDD  | 144        |
| 4.4.2.1 Módulo Sentencias Graficas (Editor Grafico de Sentencias SQL)        | 145        |
| 4.4.2.2 Modulo Minería De Datos  | 145        |
| 4.4.3 Modulo Utilidades  | 146        |
| <b>4.5 ARQUITECTURA DE PAQUETES Y CLASES</b>                                 | <b>146</b> |
| 4.5.1 Utilidades   | 146        |
| 4.5.1 Kernel PgKDD: Sentencias graficas                                      | 151        |
| 4.5.1 Kernel PgKDD: Modulo Minería de Datos                                  | 154        |
| 4.5.1 Modulo Interfaz Grafica  | 159        |
| <b>5. NUEVAS IMPLEMENTACIONES EN POSTGRESKDD</b>                             | <b>165</b> |
| <b>6. PRUEBAS Y EVALUACION DE RESULTADOS</b>                                 | <b>168</b> |
| <b>6.1 CARACTERISTICAS DE LOS REPOSITORIOS</b>                               | <b>168</b> |
| 6.1.1 Base de datos Universitaria  | 168        |
| 6.1.2 Base de Datos Mercado  | 170        |
| 6.1.3 Base de Datos Guarderia  | 171        |
| 6.1.4 Base de datos JugarTenis   | 172        |
| 6.1.5 Base de datos Productos  | 172        |

|  |            |
|--|------------|
| <b>6.2 PRUEBA DE FUNCIONALIDAD DE Pg_KDD</b> | <b>173</b> |
| 6.2.1 Asociación                             | 173        |
| 6.2.2 Clasificación                          | 185        |
| <b>6.3 PRUEBAS DE RENDIMIENTO</b>            | <b>201</b> |
| 6.3.1 Asociación                             | 201        |
| 6.3.2 Clasificación                          | 232        |
| <b>7. CONCLUSIONES</b>                       | <b>251</b> |
| <b>8. RECOMENDACIONES</b>                    | <b>252</b> |
| <b>REFERENCIAS</b>                           | <b>253</b> |
| <b>ANEXOS</b>                                | <b>259</b> |

## LISTA DE TABLAS

|  | Pag |
|--|-----|
| Tabla 1: Relación R _____  | 63  |
| Tabla 2: Resultado Associator _____  | 64  |
| Tabla 3: Tabla Estudiantes _____   | 66  |
| Tabla 4: Resultado Associator Range. _____   | 67  |
| Tabla 5: Assostudent _____   | 67  |
| Tabla 6: temsets frecuentes tamaño 2 y 3 con soporte $\geq 2$ _____                        | 68  |
| Tabla 7: Resultado operación $R1=\alpha_{2,4}(R)$ _____                                    | 70  |
| Tabla 8: Tabla Transacciones _____   | 71  |
| Tabla 9: Assorow _____   | 72  |
| Tabla 10: Assotran _____   | 72  |
| Tabla 11: Ítemsets frecuentes tamaño 2 y 3 con soporte $\geq 2$ . _____                    | 73  |
| Tabla 12: Relación R _____   | 74  |
| Tabla 13: Resultado de la operación $R1=\chi A=a1 \vee B=b1 \vee C=c2 \vee D=d1 (R)$ _____ | 75  |
| Tabla 14: Resultado primitiva EquiKeep On _____  | 77  |
| Tabla 15: Resultado primitiva Assorow Range. _____   | 77  |
| Tabla 16: Soporte Ítemsets tamaño 2 y 3. _____   | 78  |
| Tabla 17: Tabla Equitran. _____  | 78  |
| Tabla 18: Relación R _____   | 79  |
| Tabla 19: Relación $R1=\beta\alpha_3(R)$ . _____   | 79  |
| Tabla 20: Relación R _____   | 85  |
| Tabla 21: Resultado operación $R1= \mu A, B; D(R)$ _____                                   | 85  |
| Tabla 22: Tabla Síntomas. _____  | 87  |
| Tabla 23: Resultado primitiva Mate _____   | 88  |
| Tabla 24: Clasesintomas _____  | 88  |
| Tabla 25: Resultado Parcial función Entro() _____  | 91  |
| Tabla 26: Entrosintomas _____  | 91  |
| Tabla 27: GanSintomas _____  | 94  |
| Tabla 28: Datos de entrenamiento para Mate-Tree _____                                      | 100 |
| Tabla 29: Resultado Primitiva Mate _____   | 100 |
| Tabla 30: Clase _____  | 101 |
| Tabla 31: Resultdo parcial función Entro() _____   | 102 |
| Tabla 32: Resultado Entropia. _____  | 102 |
| Tabla 33: Ganancia _____   | 103 |
| Tabla 34: Nodos _____  | 104 |
| Tabla 35: Requerimientos Ejecucion y Manejo de Pg_KDD _____                                | 105 |
| Tabla 36: Requerimientos administracion de conexiones _____                                | 106 |
| Tabla 37: Requerimientos Manipulacion de Archivos SQL _____                                | 107 |
| Tabla 38: Requerimientos Administración de bases de datos. _____                           | 107 |

|  |     |
|--|-----|
| Tabla 39: Requerimientos para la aplicación de minería de datos.                                 | 108 |
| Tabla 40: Requerimientos administración de consultas SQL gráficas.                               | 109 |
| Tabla 41: Repositorio Universitaria  | 169 |
| Tabla 42: Repositorio Universitaria en Ítems   | 170 |
| Tabla 43: Ítems  | 171 |
| Tabla 44: Valores transformados de Mercado   | 171 |
| Tabla 45: Repositorio Guardería  | 172 |
| Tabla 46: Repositorio JugarTenis   | 173 |
| Tabla 47: Repositorio Productos con transacciones  | 173 |
| Tabla 48: Valores Transformados para Productos   | 174 |
| Tabla 49: Ítems  | 174 |
| Tabla 50: Associator Range   | 175 |
| Tabla 51: fre_directos   | 176 |
| Tabla 52: Resultado de Describe_Associator_Rules   | 177 |
| Tabla 53: Prueba de rendimiento de Associator con Ítems variando N°Atributos                     | 202 |
| Tabla 54: Prueba de rendimiento de Associator con Ítems variando N°Transacciones                 | 203 |
| Tabla 55: Prueba de rendimiento de Associator con Ítems variando N°Transacciones y N°Atributos   | 205 |
| Tabla 56: Rendimiento EquipAsso con Ítems variando N°Atributos y N°Transacciones                 | 208 |
| Tabla 57: Prueba de rendimiento de Associator con mercado variando N°Atributos                   | 210 |
| Tabla 58: Prueba de rendimiento de Associator con mercado variando N°Transacciones               | 211 |
| Tabla 59: Prueba de rendimiento de Associator con mercado variando N°Transacciones y N°Atributos | 213 |
| Tabla 60: Rendimiento EquipAsso con mercado variando N°Atributos y N°Transacciones               | 216 |
| Tabla 61: Reglas de asociación tamaño 2, 3 y 4 generadas con mercado                             | 223 |
| Tabla 62: Reglas generadas con Weka para mercado   | 228 |
| Tabla 63: Comparación reglas generadas por Pg_KDD y Weka para mercado                            | 231 |
| Tabla 64: Regla transformada a valores reales  | 232 |
| Tabla 65: Rendimiento con Mate variando N°de trans acciones                                      | 233 |
| Tabla 66: Rendimiento con Mate variando N°de Atrib utos  | 234 |
| Tabla 67: Rendimiento con Mate variando N°de Atrib utos y N°Transacciones                        | 236 |
| Tabla 68: Rendimiento Mate-Tree variando N°Atribut os y N°Transacciones                          | 238 |
| Tabla 69: Reglas generadas por Pg_KDD para Guardería   | 243 |
| Tabla 70: Reglas generadas con Weka para Guardería   | 248 |
| Tabla 71: Comparación reglas generadas por Pg_KDD y Weka para mercado                            | 250 |

## LISTA DE FIGURAS

|  | Pag |
|--|-----|
| <i>Figura 1. Sistema de Minería de Datos Fuertemente Acoplada con un SGBD.</i> | 35  |
| <i>Figura 2: Representación gráfica de un Diagrama de Casos de Uso</i>         | 40  |
| <i>Figura 3: Representación gráfica de un Diagrama de clases</i>               | 40  |
| <i>Figura 4: Representación gráfica de un Diagrama de secuencia</i>            | 41  |
| <i>Figura 5: Representación de los diagramas que componen un Modelo</i>        | 41  |
| <i>Figura 6: Representación gráfica de un paquete</i>                          | 42  |
| <i>Figura 7: Representación gráfica de una relación tipo Dependencia</i>       | 42  |
| <i>Figura 8: Representación gráfica de una relación tipo Asociación</i>        | 43  |
| <i>Figura 9: Representación gráfica de una relación tipo Agregación</i>        | 43  |
| <i>Figura 10: Representación gráfica de una relación tipo Generalización</i>   | 44  |
| <i>Figura 11: Representación gráfica de una relación tipo Realización</i>      | 44  |
| <i>Figura 12: Etapas de Desarrollo</i>   | 46  |
| <i>Figura 13: Representación manejo protocolo nativo JAVA</i>                  | 52  |
| <i>Figura 14: Representación NimROD Tipical</i>                                | 57  |
| <i>Figura 15: Representación NimROD Blue</i>                                   | 58  |
| <i>Figura 16: Representación NimROD Red</i>                                    | 58  |
| <i>Figura 17: Representación InfoNode Doking Windows</i>                       | 60  |
| <i>Figura 18: Representación Look and Feel InfoNode Windows XP</i>             | 61  |
| <i>Figura 19 Algoritmo EquipAsso</i>   | 82  |
| <i>Figura 20: Algoritmo Mate-Tree</i>  | 98  |
| <i>Figura 21: Caso de Uso General</i>  | 115 |
| <i>Figura 22: Caso de Uso Conectar</i>   | 116 |
| <i>Figura 23: Caso de Uso Adm. Base de Datos</i>                               | 117 |
| <i>Figura 24: Caso de Uso Realizar Consultas</i>                               | 118 |
| <i>Figura 25: Caso de Uso Adm. Consultas Gráficas</i>                          | 119 |
| <i>Figura 26: Caso de Uso Realizar Consultas Gráficas</i>                      | 120 |
| <i>Figura 27: Caso de Uso Ordenar Resultado Consultas Gráficas</i>             | 121 |
| <i>Figura 28: Caso de Uso Gestionar Tablas</i>                                 | 122 |
| <i>Figura 29: Caso de Uso Manipular contenido de tablas</i>                    | 123 |
| <i>Figura 30: Caso de Uso Gestionar Columnas</i>                               | 124 |
| <i>Figura 31: Caso de Uso Gestionar Funciones</i>                              | 125 |
| <i>Figura 32: Caso de Uso Gestionar Vistas</i>                                 | 126 |
| <i>Figura 33: Caso de Uso Gestionar Reglas</i>                                 | 127 |
| <i>Figura 34: Caso de Uso Gestionar Disparadores</i>                           | 128 |
| <i>Figura 35: Caso de Uso Minería de Datos</i>                                 | 129 |
| <i>Figura 36: Caso de Uso Manipular Scripts SQL</i>                            | 130 |
| <i>Figura 37: Diagrama del Sistema</i>   | 131 |
| <i>Figura 38: Diagrama de Clases Inicio de Aplicación</i>                      | 132 |

|   |     |
|---|-----|
| <i>Figura 39: Diagrama de Clases Adm. Visual de Pg_KDD</i>                  | 133 |
| <i>Figura 40: Diagrama de Clases Elementos del Árbol</i>                    | 134 |
| <i>Figura 41: Diagrama de Clases Adm. Logica Sentencias Gráficas</i>        | 135 |
| <i>Figura 42: Diagrama de Clases Consultas Gráficas</i>                     | 136 |
| <i>Figura 43: Diagrama de Clases Adm. Logica KDD</i>                        | 137 |
| <i>Figura 44: Diagrama de Clases Adm. de Servicios KDD</i>                  | 138 |
| <i>Figura 45: Diagrama de Clases Utilidades</i>                             | 139 |
| <i>Figura 46: Diagrama de Paquetes Principal</i>                            | 140 |
| <i>Figura 47: Diagrama de Paquetes Class</i>                                | 140 |
| <i>Figura 48: Diagrama de Paquetes Gui</i>                                  | 141 |
| <i>Figura 49: Diagrama de Paquetes GraphicSentences</i>                     | 142 |
| <i>Figura 50: Diagrama de Paquetes KDD</i>                                  | 143 |
| <i>Figura 51: Diagrama de Paquetes Utilities</i>                            | 143 |
| <i>Figura 52: Arquitectura Pg_KDD</i>                                       | 145 |
| <i>Figura 53: Creación de la nueva database asociación</i>                  | 178 |
| <i>Figura 54: Script - Tabla item</i>                                       | 179 |
| <i>Figura 55: Cargar repositorio para asociación</i>                        | 180 |
| <i>Figura 56: Selección de los atributos</i>                                | 180 |
| <i>Figura 57: Configuración del operador Associator</i>                     | 181 |
| <i>Figura 58: Resultado de Associator Range</i>                             | 181 |
| <i>Figura 59: Configuración del operador Equipaso</i>                       | 182 |
| <i>Figura 60: Resultado de Equipaso</i>                                     | 183 |
| <i>Figura 61: Configuración del nodo Associator Rules</i>                   | 183 |
| <i>Figura 62: Reglas generadas por la función describe_associator_rules</i> | 184 |
| <i>Figura 63: Comparación de resultados</i>                                 | 185 |
| <i>Figura 64: jugartenis</i>  | 186 |
| <i>Figura 65: Árbol de decisión 1</i>                                       | 188 |
| <i>Figura 66: Valores para el estado nublado</i>                            | 188 |
| <i>Figura 67: Valores para el estado soleado</i>                            | 189 |
| <i>Figura 68: Árbol de Decisión 2</i>                                       | 190 |
| <i>Figura 69: Valores para estado lluvioso</i>                              | 191 |
| <i>Figura 70: Árbol de decisión final</i>                                   | 192 |
| <i>Figura 71: Reglas Generadas</i>  | 192 |
| <i>Figura 72: Creación de la nueva database clasificacion</i>               | 193 |
| <i>Figura 73: Creación de la table jugartenis</i>                           | 194 |
| <i>Figura 74: Carar repositorio para clasificación</i>                      | 195 |
| <i>Figura 75: Selección de los atributos</i>                                | 195 |
| <i>Figura 76: Configuración del operador Mate</i>                           | 196 |
| <i>Figura 77: Resultado del operador Mate By</i>                            | 197 |
| <i>Figura 78: Cálculo de la entropía</i>                                    | 198 |
| <i>Figura 79: Ganancias y Tabla con el árbol de decisión.</i>               | 199 |
| <i>Figura 80: Reglas generadas</i>  | 200 |

|   |     |
|---|-----|
| <i>Figura 81: Comparación de resultados</i>   | 200 |
| <i>Figura 82: Prueba de rendimiento de Associator con Ítems variando N°Atributos</i>  | 202 |
| <i>Figura 83: Prueba de rendimiento de Associator con Ítems variando N°Transacciones</i>  | 204 |
| <i>Figura 84: Prueba de rendimiento de Associator con Ítems variando N°Transacciones y N°Atributos- Tiempo vs N°de atributos</i>      | 206 |
| <i>Figura 85: Prueba de rendimiento de Associator con Ítems variando N°Transacciones y N°Atributos. Tiempo vs N°de transacciones.</i> | 207 |
| <i>Figura 86: Rendimiento de EquipAsso con Ítems variando N°de Atributos y N°de Transacciones. Tiempo vs N°de transacciones</i>       | 209 |
| <i>Figura 87: Prueba de rendimiento de Associator con mercado variando N°Atributos</i>  | 210 |
| <i>Figura 88: Prueba de rendimiento de Associator con mercado variando N°Transacciones</i>  | 212 |
| <i>Figura 89: Prueba de rendimiento de Associator con mercado variando N°Transacciones y N°Atributos. Tiempo vs N°Atributos</i>       | 214 |
| <i>Figura 90: Prueba de rendimiento de Associator con mercado variando N°Transacciones y N°Atributos. Tiempo vs N°Transacciones</i>   | 215 |
| <i>Figura 91: Rendimiento de EquipAsso con Ítems variando N°de Atributos y N°de Transacciones</i>                                     | 217 |
| <i>Figura 92: Resultados Associator con mercado</i>   | 218 |
| <i>Figura 93: Resultados EquipAsso con mercado</i>  | 219 |
| <i>Figura 94: Reglas de asociación generadas con mercado</i>  | 220 |
| <i>Figura 95: Reglas de asociación tamaño 2 generadas con mercado</i>   | 221 |
| <i>Figura 96: Reglas de asociación tamaño 3 generadas con mercado</i>   | 222 |
| <i>Figura 97: Reglas de asociación tamaño 4 generadas con mercado</i>   | 223 |
| <i>Figura 98: Selección de Repositorio Ítems en Weka</i>  | 226 |
| <i>Figura 99: Configuración algoritmo Apriori en Weka</i>   | 227 |
| <i>Figura 100: Generación de reglas con Weka</i>  | 228 |
| <i>Figura 101: Prueba de rendimiento con Guarderia variando N°transacciones</i>   | 234 |
| <i>Figura 102: Rendimiento con guarderia variando N°a tributos</i>  | 235 |
| <i>Figura 103: Rendimiento con Mate variando N°de Atributos y N°Transacciones</i>   | 237 |
| <i>Figura 104: Rendimiento con Mate variando N°de atributos y N°Transacciones</i>   | 239 |
| <i>Figura 105: Resultado Mate con Guarderia</i>   | 240 |
| <i>Figura 106: Resultado Entro con Guarderia</i>  | 241 |
| <i>Figura 107: Resultado Gain con Guarderia</i>   | 242 |
| <i>Figura 108: Resultado Reglas para Guarderia</i>  | 243 |
| <i>Figura 109: Selección de Repositorio Guardería en Weka</i>   | 245 |
| <i>Figura 110: Configuración algoritmo J48 en Weka</i>  | 246 |
| <i>Figura 111: Generación de reglas de clasificación con Weka</i>   | 247 |

## LISTA DE ANEXOS

|   |     |
|---|-----|
| ANEXO A. Nuevas implementaciones en postgresKDD _____ | 259 |
| ANEXO B. Manual de Instalación _____                  | 262 |
| ANEXO C. Manual de Usuario _____                      | 266 |

## INTRODUCCIÓN

El explosivo crecimiento en los volúmenes de los datos y las bases de datos que superan los métodos tradicionales de análisis basados en hojas de cálculo y consultas ad-hoc [6], ha generado una urgente necesidad de contar con nuevas técnicas y herramientas que puedan, inteligente y automáticamente, transformar los datos en información útil [3]. Estas técnicas y herramientas son el objeto del emergente campo de Descubrimiento de Conocimiento en Bases de Datos DCBD (KDD- Knowledge discovery in databases).

A pesar del acelerado avance y del incremento de la investigación en el área de DCBD [3][7][9][17][18][23][28][29][1], relativamente son pocas las propuestas de integrar al lenguaje de consultas SQL nuevas primitivas que permitan descubrir eficientemente conocimiento en grandes bases de datos [4][14][21][24]. La mayor parte de los sistemas y herramientas DCBD existentes se han construido bajo el modelo de arquitectura débilmente acoplada con un SGBD. Además, no existe un consenso general sobre el conjunto de primitivas necesarias para soportar el descubrimiento de conocimiento en un SGBD.

En este proyecto se presenta la segunda etapa del proyecto PostgresKDD cuyo objetivo principal es el de construir un ambiente gráfico para el sistema de descubrimiento de conocimiento PostgresKDD, una herramienta computacional fuertemente acoplada con el SGBD PostgreSQL. En la primera etapa se implementaron algunos operadores y primitivas SQL propuestos por Timaran [32] para el Descubrimiento de Reglas de Asociación y Clasificación al interior del motor del SGBD PostgreSQL. El resultado de esa etapa fue la primera versión de la herramienta PostgresKDD con la capacidad de descubrir Reglas de Asociación y Clasificación. En esta etapa se pretende construir a PostgresKDD, un ambiente gráfico que permita de una manera amigable interactuar con el motor KDD y extraer conocimiento de las bases de datos de una manera fácil.

## **1. TEMA**

El desarrollo de un ambiente gráfico para el sistema de descubrimiento de conocimiento PostgresKDD, se presentó como propuesta para proyecto de grado.

En éste capítulo se presenta las generalidades del problema que se identificó como objeto del estudio: La línea de investigación realizada, la descripción del problema, los objetivos propuestos los cuales reflejan los alcances del proyecto y la organización del presente documento.

### **LINEA DE INVESTIGACION**

El presente trabajo, se encuentra inscrito bajo la línea de investigación arquitecturas de Integración de sistemas de descubrimiento de conocimiento y gestores de bases de datos.

### **PLANTEAMIENTO DEL PROBLEMA**

El proceso de extraer conocimiento a partir de los datos ha sido reconocido por muchos investigadores [3][7][9][17][18][23] como un tópico de investigación clave en los sistemas de bases de datos, y por muchas compañías industriales como una importante área y una oportunidad para obtener mayores ganancias.

El Descubrimiento de Conocimiento en Bases de Datos (DCBD) es el proceso no trivial de identificación de patrones válidos, novedosos, potencialmente útiles y fundamentalmente entendibles al usuario a partir de los datos [7].

A pesar del acelerado avance y del incremento de la investigación en el área de DCBD, relativamente son pocas las propuestas de integrar al lenguaje de consultas SQL nuevas primitivas que permitan descubrir eficientemente conocimiento en grandes bases de datos [14][21][24][4].

Timarán [32][33] propone extender el motor de un SGBD con nuevos operadores algebraicos y primitivas SQL que harían más eficiente el Proceso de Descubrir

Conocimiento bajo una arquitectura fuertemente acoplada con un Sistema Gestor de Bases de Datos (SGBD).

En la primera etapa del proyecto PostgresKDD se implementaron algunos operadores y primitivas SQL propuestos por Timaran [32] para el Descubrimiento de Reglas de Asociación y Clasificación al interior del motor del SGBD PostgreSQL, un Sistema Gestor de Bases de Datos robusto y de Licencia Pública.

Estos operadores y primitivas SQL se deben gestionar en modo texto en PostgresKDD para realizar un proyecto de minería de datos que tenga que ver con las tareas de asociación y clasificación. Esto dificulta el proceso de minería de datos, ya que el usuario final debe conocer el operador SQL que realiza la tarea específica y además conocer la sintaxis correcta, para poder ejecutar el comando.

En este proyecto se presenta Pg\_KDD, un ambiente gráfico para el sistema de descubrimiento de conocimiento PostgresKDD, enfocado a la administración gráfica de los operadores propuestos por Timaran [32][33] para el Descubrimiento de Conocimiento en Bases de Datos, que permita de una manera amigable interactuar con el motor KDD y extraer conocimiento de las bases de datos de una manera fácil.

Este proyecto de investigación responde a la pregunta ¿Cómo construir un ambiente gráfico para el sistema de descubrimiento de conocimiento PostgresKDD, que permita de una manera amigable interactuar con el motor KDD y extraer conocimiento de las bases de datos de una manera fácil?

## **ALCANCE Y DELIMITACIÓN**

Pg\_KDD es un ambiente gráfico para el sistema de descubrimiento de conocimiento PostgresKDD, una herramienta computacional fuertemente acoplada con el SGBD PostgreSQL, que permite al usuario final, interactuar de una manera amigable con el motor KDD y extraer conocimiento de las bases de datos de una manera fácil.

El proyecto se divide en tres módulos:

*MÓDULO GESTIÓN:* en este módulo se administran todos los procesos referentes a conexiones al sistema, administración de bases de datos y sus procedimientos como son manejo de tablas, columnas, restricciones, reglas, disparadores, entre otros.

*MÓDULO SENTENCIAS GRÁFICAS:* en este módulo se representa el resultado de sentencias SQL obtenidas a través del manejo gráfico de relaciones de una base de datos, en donde gráficamente y de manera manual se puede realizar selecciones, uniones, ordenamiento, aplicación de funciones de agregación y agrupamiento.

*MÓDULO MINERÍA DE DATOS:* en este módulo se gestiona de manera gráfica los operadores, primitivas y algoritmos implementados en PostgresKDD, para las tareas de minería de datos: asociación y clasificación, con el fin de descubrir conocimiento a partir de los datos seleccionados.

## **OBJETIVO GENERAL**

Construir un ambiente gráfico para el sistema de descubrimiento de conocimiento PostgresKDD, una herramienta computacional fuertemente acoplada con el SGBD PostgreSQL que permita de una manera amigable interactuar con el motor KDD y extraer conocimiento de las bases de datos de una manera fácil.

## **OBJETIVOS ESPECÍFICOS**

- ✓ Describir la Arquitectura del SGBD PostgreSQL.
- ✓ Apropiar la implementación de los operadores mediana y fuertemente acoplados en PostgresKDD, para asociación y clasificación.
- ✓ Desarrollar la Interfaz Gráfica de Usuario de PostgresKDD.
- ✓ Realizar y evaluar las pruebas con la herramienta desarrollada para PostgresKDD.

## **JUSTIFICACIÓN**

Todas las herramientas de DCBD o comúnmente conocidas como herramientas de minería de datos sirven en las organizaciones para apoyar la toma de decisiones de tipo semi-estructurado, única o que cambian rápidamente, y que no es fácil especificar por adelantado. Es evidente que por su diseño, estas herramientas tienen mayor capacidad analítica que otras. Están construidas explícitamente con diversos modelos para obtener patrones insospechados a partir de los datos. Apoyan la toma de decisiones al permitir a los usuarios extraer información útil que antes estaba enterrada en montañas de datos. Diversas herramientas de minería de datos disponibles en el mercado ofrecen diferentes tipos de arquitecturas que determinan, en alguna medida, su versatilidad y su costo. Por lo general todas estas son costosas, necesitan de licenciamiento para su uso y de software específico.

El desarrollo de Pg\_KDD como una herramienta DCBD bajo licencia pública de GNU, la convierte en un aporte significativo al mundo de los sistemas que abre las puertas a todo tipo de empresas u organizaciones para que puedan hacer uso de esta herramienta para el soporte a la toma de decisiones, sin costo alguno.

Por consiguiente, Pg\_KDD siendo una herramienta de DCBD, puede utilizarse en diferentes campos como la industria, la banca, la salud y la educación, entre otros.

Por otra parte, Pg\_KDD se convierte en otro aporte más en el área del Descubrimiento de Conocimiento en bases de datos que la Universidad de Nariño, a través de su programa de Ingeniería de Sistemas, hacia el mundo, contribuye a la investigación científica y al desarrollo de la región y del país.

## **ORGANIZACIÓN DEL DOCUMENTO**

El resto de este documento está organizado de la siguiente manera. En la siguiente sección se presenta el marco teórico en el área de integración de DCBD y SGBD, detallando previamente el planteamiento del problema, delimitándolo según sus módulos, describiendo el objetivo general y sus objetivos específicos. En la siguiente sección se describe la teoría de minería de datos relacionada con

las tareas de asociación y clasificación. Seguido a ello, se describe todo lo concerniente a la metodología de desarrollo, análisis y diseño, que se realizó para construir la herramienta, al igual que su arquitectura. Finalmente se presentan las pruebas de funcionalidad y rendimiento, sustentando todo el proceso de desarrollo en las conclusiones y recomendaciones.

## **2. MARCO TEORICO**

En este capítulo se describe el marco teórico de la investigación relacionado con la investigación. En la primera sección se da a conocer algunos antecedentes o trabajos relacionados con el proceso de DCBD, en la segunda sección se describe el formalmente el proceso de DCBD, en la tercera sección se describe la metodología de desarrollo del ambiente gráfico Pg\_KDD y finalmente en la última sección se describe las herramientas de desarrollo usadas en el proceso.

## 2.1 ANTECEDENTES

A continuación se describen algunos proyectos de investigación relacionados con el Descubrimiento de Conocimiento en Bases de datos con la aplicación de tareas de minería de datos.

**2.1.1 PostgresKDD:** Es un proyecto de investigación que se encuentra en el grupo de investigación GRIAS de la Universidad de Nariño, que se denota bajo la arquitectura fuertemente acoplada con el SGBD PostgreSQL, cuyo nombre es “Implantación de primitivas SQL para el descubrimiento de conocimiento de reglas de asociación y clasificación al interior del motor del sistema gestor de bases de datos PostgreSQL” [43].

El objetivo principal es dotar al SGBD PostgreSQL la capacidad de descubrir conocimiento en bases de datos al interior de su motor mediante la implementación de primitivas SQL para el descubrimiento de Reglas de asociación y clasificación.

Para el desarrollo de este proyecto se estudio los operadores de descubrimiento de conocimiento propuestos por Timaran [31], se analizo la arquitectura del SGBD PostgreSQL, su código fuente y finalmente se implementaron al interior del motor de PostgreSQL las primitivas SQL para que compile, transforme y ejecute eficientemente una consulta que involucre descubrimiento de reglas de Asociación y Clasificación, evaluando su rendimiento con bases de datos sintéticas con un gran volumen de datos.

Como conclusiones para este proyecto y de acuerdo a las pruebas de rendimiento, se recalca la mejora en eficiencia, en cuanto a tiempo de ejecución, de los nuevos

operadores de Asociación y Clasificación en comparación con otros proyectos débilmente acoplados como TariyKDD [51].

Al igual que PostgresKDD, Pg\_KDD esta desarrollado bajo los lineamientos de software libre los hace herramientas accesibles para para cualquier empresa u organización que requiera de un apoyo en la toma de decisiones. La diferencia de este proyecto con PgKDD, radica en la inexistencia de una herramienta computacional grafica para la fácil aplicación y visualización de los resultados obtenidos de las implementaciones que se encuentran dentro del motor de PostgreSQL

**2.1.2 TariyKDD:** Es un proyecto de investigación que se encuentra en el grupo de investigación GRIAS de la Universidad de Nariño, que se denota bajo la arquitectura débilmente acoplada con el SGBD PostgreSQL, cuyo nombre es *“Una herramienta genérica De Descubrimiento De Conocimiento En Bases de Datos débilmente acoplada con El SGBD PostgreSQL”* [51].

El objetivo principal es desarrollar una herramienta genérica para el Descubrimiento de Conocimiento en bases de datos débilmente acoplada con el sistema gestor de bases de datos PostgreSQL, bajo los lineamientos del Software Libre, que facilite la toma de decisiones a las pequeñas y medianas empresas u organizaciones de nuestro país y de cualquier parte del mundo.

Para el desarrollo de este proyecto se estudiaron las diferentes herramientas de Descubrimiento de conocimiento débilmente acopladas con un SGBD, de igual forma se desarrollaron diferentes módulos para suplir con todas las etapas del Descubrimiento de conocimiento, Selección, Reprocesamiento, Transformación, Minería de Datos, Visualización y Análisis de resultados.

Los algoritmos para la obtención de patrones y reglas de Asociación y Clasificación implementados en esta herramienta fueron EquipAsso, MateTree, Apriori Híbrido, FPGrowth y C4.5, de los cuales se presentaron las pruebas de rendimiento respectivas con grandes volúmenes de datos.

Como conclusiones de esta herramienta tenemos las mejoras en la implementación de los algoritmos EquipAsso, MateTree y FPGrowth, ausentes en otras herramientas débilmente acopladas, al igual que la obtención de buenos tiempos de respuesta según sus pruebas de rendimiento.

Al igual que TariyKDD, Pg\_KDD esta desarrollado bajo los lineamientos de software libre los hace herramientas accesibles para para cualquier empresa u organización que requiera de un apoyo en la toma de decisiones, de igual forma Pg\_KDD apoya en las tareas de selección, minería de datos y visualización del proceso de Descubrimiento de Conocimiento con Bases de Datos.

La diferencia de este proyecto con Pg\_KDD, radica en la implementación de módulos gráficos para la administración de bases de datos, como también en el uso y aplicación de algoritmos, operadores y primitivas que se encuentran dentro de un sistema de Descubrimiento de Conocimiento fuertemente acoplado con un SGBD como es PostgresKDD.

**2.1.3 MATE-KDD:** Es un proyecto de investigación que se encuentra en el grupo de investigación GRIAS de la Universidad de Nariño, que se denota bajo la arquitectura medianamente acoplada con el SGBD PostgreSQL, cuyo nombre es *“Una herramienta genérica para el Descubrimiento de reglas de Clasificación medianamente acoplada Al SGBD PostgreSQL”* [37].

Desarrollar una herramienta genérica para el Descubrimiento de reglas de Clasificación en bases de datos medianamente acoplada con el Sistema Gestor de Bases de Datos PostgreSQL, bajo los lineamientos del Software Libre.

Para el desarrollo de este proyecto se estudiaron las diferentes herramientas de Descubrimiento de conocimiento para Clasificación, se implementaron dentro de PostgresKDD, operadores y primitivas SQL para las tareas de Clasificación como funciones definidas por el usuario o medianamente acopladas. Se realizaron las respectivas pruebas de rendimiento del nuevo algoritmo propuesto MATE-TREE.

Al igual que MATE-KDD, Pg\_KDD trabaja en conexión con el Sistema de descubrimiento de conocimiento con Bases de Datos PostgresKDD y esta desarrollado bajo los lineamientos de software libre los hace herramientas accesibles para para cualquier empresa u organización que requiera de un apoyo en la toma de decisiones, de igual forma Pg\_KDD apoya en las tareas de selección, minería de datos y visualización del proceso de Descubrimiento de Conocimiento con Bases de Datos.

La diferencia de este proyecto con Pg\_KDD es que este no solo aplica tareas de minería de datos para el descubrimiento de reglas de clasificación si no también

tareas de minería de datos para el descubrimiento de reglas de asociación, haciendo uso de operadores y primitivas que se encuentran dentro del sistema PostgresKDD, además que Pg\_KDD contiene una interfaz adicional para la creación de sentencias graficas aplicando los conceptos generales como son selección, agrupamiento, ordenamiento, entre otros.

## **2.2 EL PROCESO DE DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS- DCBD**

El DCBD es el proceso que utiliza métodos de minería de datos (algoritmos) para extraer (identificar) patrones que evaluados e interpretados, de acuerdo a las especificaciones de medidas y umbrales, usando una base de datos con alguna selección, pre procesamiento, muestreo y transformación, se obtiene lo que se piensa es conocimiento [6][8].

El proceso de DCBD es interactivo e iterativo, involucra numerosos pasos con la intervención del usuario en la toma de muchas decisiones y se resumen en las siguientes etapas:

- ✓ Selección
- ✓ Pre procesamiento/Data cleaning
- ✓ Transformación/Reducción
- ✓ Minería de Datos/Data Mining (DM)
- ✓ Interpretación/Evaluación

**2.2.1 Minería de Datos.** La minería de datos consiste en la extracción no trivial de información que reside de manera implícita en los datos. Dicha información era previamente desconocida y podrá resultar útil para algún proceso. En otras palabras, la minería de datos prepara, sondea y explora los datos para sacar la información oculta en ellos [31].

Bajo el nombre de minería de datos se engloba todo un conjunto de técnicas encaminadas a la extracción de conocimiento procesable, implícito en las bases de datos. Está fuertemente ligado con la supervisión de procesos industriales ya que resulta muy útil para aprovechar los datos almacenados en las bases de datos [31].

Es una etapa dentro del proceso completo del descubrimiento del conocimiento, este intenta obtener patrones o modelos a partir de los datos recopilados. Decidir si los modelos obtenidos son útiles o no suele requerir una valoración subjetiva por parte del usuario. Según Fayyad, los algoritmos de DM suelen tener tres componentes [8]:

- ✓ El modelo, que contiene parámetros que han de fijarse en partir de los datos de entrada.
- ✓ El criterio de preferencia, que sirve para comparar modelos alternativos.
- ✓ El Algoritmo de búsqueda, que viene a ser como cualquier otro programa de inteligencia artificial (IA).

El criterio de preferencia suele ser algún tipo de heurística y los algoritmos de búsqueda empleados suelen ser los mismos que en otros programas de inteligencia artificial. Las principales diferencias entre los algoritmos de DM se hallan en el modelo de representación escogido y la función del mismo, es decir según el objetivo perseguido.

**2.2.2 Tareas de Minería de Datos.** Las técnicas DM empleadas en el proceso de KDD se pueden clasificar en dos grandes grupos:

- ✓ Técnicas de verificación, en las que el sistema se limita a comprobar hipótesis suministradas por el usuario.
- ✓ Métodos de descubrimiento, en los que se han de encontrar patrones potencialmente interesantes de forma automática, incluyendo en este grupo todas las técnicas de predicción.

El resultado obtenido con la aplicación de los algoritmos de DM al segundo grupo, el de técnicas de descubrimiento, pueden ser de carácter descriptivo o predictivo. Las predicciones sirven para prever el comportamiento futuro de algún tipo de entidad mientras que una descripción puede ayudar a su comprensión.

La aplicación de técnicas de DM en grandes bases de datos persiguen los siguientes resultados:

- ✓ **Clasificación:** Se trata de obtener un modelo que permita asignar un caso de clase desconocida a una clase concreta (seleccionada de un conjunto redefinido de clases), como son los árboles de clasificación (CART), cuyos resultados pueden expresarse mediante reglas ejecutables directamente del SQL o el método de Bayesiano.
- ✓ **Regresión:** Se persigue la obtención de un modelo que permita predecir el valor numérico de alguna variable (modelos de regresión logística).
- ✓ **Agrupamiento (clustering):** Hace corresponder cada caso de una clase, con la peculiaridad de que las clases se obtienen directamente de los datos de entrada utilizando medidas de similaridad. Es decir, agrupan a los datos bajo diferentes métodos y criterios. Las técnicas más usadas son las clásicas (distancia mínima) y las redes neuronales (método de Kohonen o método de Neural-Gas).
- ✓ **Resumen:** Se obtienen representaciones compactas para subconjuntos de los datos de entrada (análisis interactivo de datos, generación automática de informes, visualización de datos).
- ✓ **Modelado de Dependencias:** se obtienen descripciones de dependencias existentes entre variables. El análisis de relaciones (por ejemplo las reglas de asociación), en que se determinan relaciones existentes entre elementos de una base de datos, podría considerarse un caso particular de modelado de dependencias.
- ✓ **Análisis de Secuencias:** Se intenta modelar la evolución temporal de alguna variable, con los fines descriptivos o predictivos (redes neuronales multicapas).

**2.2.3 Tarea de Asociación.** La tarea de asociación se basa en los siguientes fundamentos [50]:

- Las entradas son un conjunto de ítems y una colección de transacciones (canastas)
  - Sea  $I = \{i_1, i_2, \dots, i_m\}$  un conjunto de ítems.
  - Sea  $D$  un conjunto de transacciones

- Cada transacción  $T$  es un conjunto de ítems tal que  $T \subseteq I$ .
- La tarea es generar reglas que cumplan un soporte y una confianza mínima expresados por el usuario.
  - Una regla de asociación es una implicación de la forma  $X \Rightarrow Y$ , donde  $X \subseteq I$ ,  $Y \subseteq I$  y  $X \cap Y = \emptyset$
  - La regla  $X \Rightarrow Y$  tiene soporte  $s$  en  $D$  si  $s\%$  de transacciones en  $D$  contienen  $X \cup Y$ .
  - La regla  $X \Rightarrow Y$  se cumple en  $D$  con confianza  $c$  si  $c\%$  de las transacciones en  $D$  que contienen  $X$  también contienen a  $Y$ .
  - $c$  denota la solidez de la implicación y  $s$  indica la frecuencia de ocurrencia del patrón en la regla.

Pasos para minar reglas de asociación [50]:

- El problema de minar reglas de asociación es encontrar todas las reglas de asociación que satisfagan un soporte mínimo especificado por el usuario y una mínima restricción de confianza.
- El problema se descompone en los siguientes pasos:
  - Descubrir los ítemsets frecuentes, i.e., el conjunto de ítemsets que tienen el soporte de transacciones por encima de un predeterminado soporte  $s$  mínimo.
  - Usar los ítemsets frecuentes para generar las reglas de asociación para la base de datos.
- Todo el rendimiento de minar las reglas de asociación es determinado por el primer paso.
- Después de que los ítemsets frecuentes son identificados, las correspondientes reglas de asociación pueden ser derivadas de una manera directa.

El conteo eficiente de los ítemsets frecuentes es el centro de atención de los algoritmos de Asociación.

**2.2.4 Tarea de Clasificación.** La clasificación es una tarea de minería de datos por medio de la cual se encuentra propiedades comunes entre un conjunto de

objetos de una base de datos y se los clasifica en diferentes clases, de acuerdo al modelo de clasificación [49].

### **Proceso de Clasificación**

La Clasificación de datos es un proceso de dos pasos [49]:

En el primero, se construye un modelo con base en un conjunto de datos denominado conjunto de entrenamiento. En el conjunto de entrenamiento, cada tupla se asume que pertenece a una clase predefinida, determinada por un atributo clase. El resto de atributos se denominan atributos condición.

En el segundo, el modelo inicialmente se prueba con otro conjunto de datos denominado conjunto de prueba. La exactitud del modelo se calcula de acuerdo al porcentaje de ejemplos de prueba que son correctamente clasificados.

### **Métodos de Clasificación**

Los métodos de clasificación se utilizan para descubrir clases en forma automática [49].

Algunos métodos de clasificación:

- Redes Neuronales
- Conjuntos aproximados (rough sets)
- Algoritmos genéticos
- Árboles de decisión

### **Clasificación por árboles de decisión**

Un árbol de decisión es una estructura compuesto por nodos internos, ramas y hojas. Cada nodo interno denota a un atributo de prueba. Cada rama representa una salida o regla. Cada hoja representa a una clase. El nodo inicial se denomina nodo raíz. [49]

El algoritmo básico para árboles de decisión es el C4.5, el cual construye el árbol de arriba hacia abajo recursivamente utilizando la manera de “divide y conquistaras”. El algoritmo usa una métrica basada en la entropía, conocida como ganancia de información para construir el árbol. [49]

## Ganancia de Información

Minimiza la medida de entropía aplicada a los ejemplos en un nodo. Esta heurística permite seleccionar el atributo que provee la mayor ganancia de información [49].

La ganancia de información de un atributo A, con respecto a un conjunto de ejemplos S es [49]:

$$Gain(S,A) = Entropía(S) - \sum_{v \in Valores(A)} \frac{|S_v|}{|S|} * Entropía(S_v)$$

Donde:

*Valores(A)* es el conjunto de todos los valores del atributo A  
*S<sub>v</sub>* es el subconjunto de S para el atributo A que toma valores *v*.

$Entropía(S) = - \sum p_i \log_2 p_i$ , donde  $p_i$  es la probabilidad que un ejemplo arbitrario pertenezca a la clase  $C_i$ .

**2.2.5 Arquitecturas de Integración de DCBD con un SGBD.** Muchos investigadores [2][30][13][20] han reconocido la necesidad de integrar los sistemas de Descubrimiento de Conocimiento con los Sistemas de Bases de Datos, haciendo de esta una área activa de investigación. Dentro de los enfoques de integración de KDD y SGBD reportados en la literatura se pueden ubicar en uno de tres tipos de arquitectura: sistemas débilmente acoplados, medianamente acoplados y sistemas fuertemente acoplados [31].

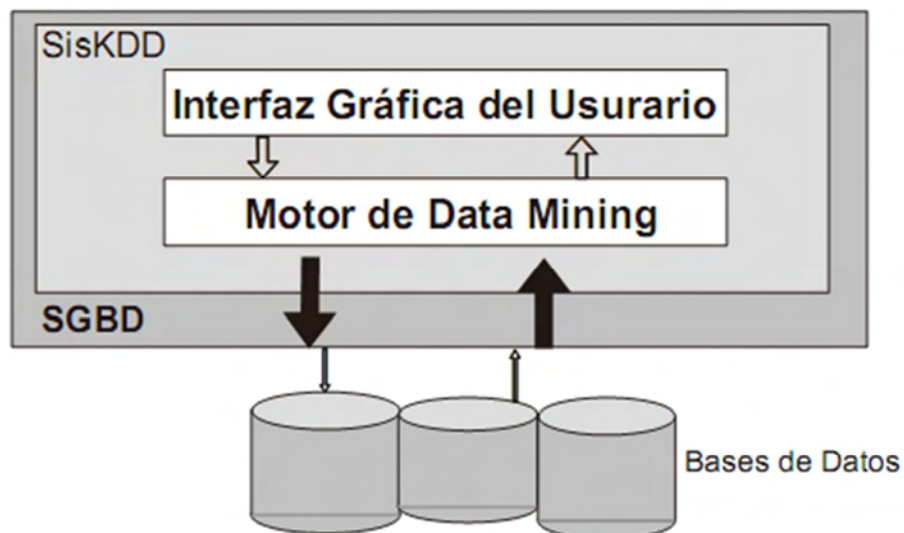
Una Herramienta para el DCBD debe integrar una variedad de componentes (técnicas de Minería de datos, consultas, métodos de visualización, interfaces, etc.), que juntos puedan eficientemente identificar y extraer patrones interesantes y útiles de los datos almacenados en las bases de datos.

Una arquitectura es débilmente acoplada cuando los algoritmos de Minería de Datos y demás componentes se encuentran en una capa externa al SGBD, por fuera del núcleo y su integración con este se hace a partir de una interfaz [31]

Una arquitectura es medianamente acoplada cuando ciertas tareas y algoritmos de descubrimiento de patrones se encuentran formando parte del SGBD mediante procedimientos almacenados o funciones definidas por el usuario [31].

Una arquitectura es fuertemente acoplada cuando la totalidad de las tareas y algoritmos de descubrimiento de patrones forman parte del SGBD como una operación primitiva, dotándolo de las capacidades de descubrimiento de conocimiento y posibilitándolo para desarrollar aplicaciones de este tipo (ver Figura 1).

Figura 1. Sistema de Minería de Datos Fuertemente Acoplada con un SGBD.



Fuente. Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. [31]

### 2.3 METODOLOGÍA PARA EL DESARROLLO DE Pg\_KDD

Con la finalidad de realizar el análisis y el diseño del proyecto Pg\_KDD se adoptó la metodología Proceso Unificado (RUP) que utiliza la notación del Lenguaje Unificado de Modelado (UML). De este modo, Pg\_KDD fue desarrollado siguiendo los estándares de modelado que plantea RUP. Para comprender sus lineamientos es necesario conocer, manejar y poder utilizar los diagramas del Lenguaje Unificado de Modelado (UML) que, a su vez, se basa en el paradigma de

programación Orientado a Objetos. Seguidamente se explicará en qué consisten cada una de estas herramientas del análisis, diseño y modelado de sistemas computacionales.

**2.3.1 Análisis y diseño de sistemas usando la metodología orientada a objetos (O.O).** Las metodologías de Ingeniería de Software tienen mecanismos para hacer análisis de necesidades y diseño de sistemas complejos, y han evolucionado con la tecnología en relación con el diseño computacional.

Así, por ejemplo, el análisis estructurado se basa en la descomposición en funciones o procesos. La metodología (Orientada a objetos), se centra en el análisis usando una descomposición del sistema por objetos o conceptos [33][38].

De acuerdo a Joseph Schmuller,

Los objetos concretos y virtuales, están a nuestro alrededor, ellos conforman nuestro mundo. El software actual simula al mundo (o un segmento de él), y los programas, por lo general, imitan los objetos del mundo. Si comprende algunas cuestiones básicas de los objetos, entenderá cómo se deben mostrar estos en las representaciones de software. Antes que nada, un objeto es la instancia de una clase (o categoría). Usted y yo, por ejemplo, somos instancias de la clase persona. Un objeto cuenta con una estructura, es decir atributos (propiedades) y acciones. Las acciones son todas las actividades que el objeto es capaz de realizar. Los atributos y acciones, en conjunto, se conocen como características o rasgos <sup>1</sup>.

Schmuller define a los objetos no solo como aquello que tiene masa, sino también habla de la posibilidad de encontrar objetos virtuales, que no pueden tocarse, como por ejemplo una dirección de correo electrónico.

La metodología O.O. trae una serie de ventajas. Por ejemplo: el suministro de modelos similares a los del mundo real, la facilidad para manejar sistemas complejos y la reutilización de los objetos, permite además el desarrollo iterativo de las aplicaciones y también facilita la interoperabilidad de las mismas.

1. Schmuller, Joseph. Aprendiendo UML en 24 horas. Prentice-Hall, 2001

Según lo que menciona Schmuller un objeto cuenta con atributos y acciones.

Dicho de otro modo, los **Atributos** pueden considerarse como un estado, o conjunto de propiedades inherentes a él. Las acciones pueden catalogarse como un conjunto de operaciones o **Métodos**. Pero, además, los objetos tienen también, una **Identidad**, es decir, un identificador que los hará únicos [33][38].

Desde un punto de vista enfocado a la programación puede decirse que un objeto es una parte de software que cumple con ciertas características como:

- ✓ **Herencia:** una clase (clase derivada) puede adoptar los atributos y métodos de una clase superior (superclase), de esta manera, cuando se crea un objeto como instancia de una clase, éste tendrá todas las características tanto de la clase de la cual es instancia, como de la superclase de la cual ésta hereda [38][39].
- ✓ **Polimorfismo:** hace referencia a como una operación puede tener el mismo nombre en una misma clase y, sin embargo, puede realizar diferentes procesos. A esto se le conoce como *Sobrecarga de Métodos*. La clave está en los valores de entrada que reciben dichas operaciones [38][39].

El polimorfismo también se presenta en el momento en que una superclase define una operación y sus clases derivadas redefinen los procesos que se ejecutan dentro de ella. Así, al llamar a determinada operación redefinida, se ejecuta la redefinición y no la definición original en la superclase. Esto recibe el nombre de *Redefinición de Métodos* [38][39].

- ✓ **Encapsulamiento:** esta característica de los objetos hace que se oculten sus funcionalidades internas, así solo se encargarán de realizar las acciones que le correspondan sin que el usuario se dé cuenta de qué es lo que pasa en su interior. esta característica hace que se protejan los datos de accesos indebidos, la forma de protegerlos es haciendo que los objetos puedan ser privados, protegidos o públicos [38][39].
- ✓ **Abstracción:** existen diferentes mecanismos de abstracción, como por ejemplo composición, clasificación, agrupación y especialización [38][39].

Una Clase es la unidad básica que encapsula toda la información de un objeto. A través de ellas se puede modelar el entorno, o la categoría de un objeto. Todos los objetos se crean por instancia de las clases.

**2.3.2 El lenguaje unificado de modelado (UML).** Un modelo puede considerarse como la abstracción de un sistema del mundo real, este modelo se encarga de describir completamente los aspectos del sistema que son relevantes para un nivel de abstracción dado, teniendo en cuenta los detalles importantes del sistema real [38][39].

La necesidad de nuevas formas de diseño ha dado pie a la creación de nuevas herramientas que permitan a los diseñadores de sistemas trabajar de forma que los demás comprendan las ideas de sus modelos.

UML es una de las herramientas más utilizadas para el modelado de sistemas. Nació en la empresa Rational Software Co, quien se puso en la tarea de crear una notación unificada en la que basar la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle e IBM, así como grupos de analistas y desarrolladores; y su desarrollo fue encabezado por: Grady Booch, Ivar Jacobson y Jim Rumbaugh [33][38][39].

Uno de los objetivos principales de la creación de UML era posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado.

Pero, específicamente ¿Qué es UML?

Según Grady Booch, James Rumbaugh e Ivar Jacobson, “El lenguaje unificado de modelado (Unified Modeling Language, UML) es un lenguaje estándar para escribir planos de software. UML puede utilizarse para visualizar, especificar, construir, y documentar los artefactos de un sistema con gran cantidad de Software” [33][38].

UML combina notaciones provenientes del modelo orientado a objetos, el modelo de datos, el modelo de componentes y modelo de flujos de trabajo, y la principal característica es que está dirigido por casos de uso y se centra en la arquitectura.

Esta combinación da origen a tres clases de bloques de construcción que son [33][38]:

- ✓ **Elementos:** los elementos son abstracciones de cosas reales o ficticias como objetos o acciones.
- ✓ **Relaciones:** formas de relacionar los *elementos* entre sí.
- ✓ **Diagramas:** son colecciones de *elementos* con sus *relaciones*.

Profundizando un poco más en los diagramas. Un diagrama es la representación gráfica de un conjunto de elementos con sus relaciones, ofrece una vista del sistema a modelar. Para poder representar correctamente un sistema, UML ofrece una amplia variedad de diagramas para visualizar el sistema desde varias perspectivas.

Los diferentes tipos de diagramas que se utilizan en UML son:

- ✓ *Diagrama de Casos de uso*
- ✓ *Diagrama de clases*
- ✓ *Diagrama de Objetos*

#### **Diagramas de Comportamiento**

- ✓ *Diagrama de Estados*
- ✓ *Diagrama de Actividad*

#### **Diagramas de Interacción**

- ✓ Diagrama de secuencia
- ✓ Diagrama de colaboración

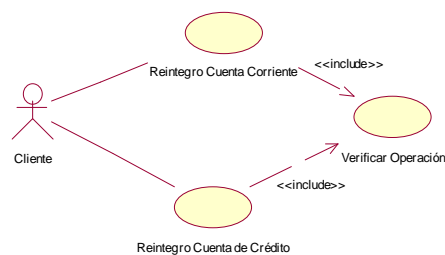
#### **Diagramas de Implementación**

- ✓ Diagrama de Componentes
- ✓ Diagrama de Despliegue

**Diagramas:** Los diagramas más utilizados en el modelamiento de sistemas como Pg\_KDD son:

- ✓ **Diagrama de Casos de Uso:** Muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa. Se define un caso de uso como cada interacción supuesta con el sistema a desarrollar, donde se representan los requisitos funcionales. Es decir, se está diciendo lo que tiene que hacer un sistema y cómo, (Figura 2) [33][38][39].

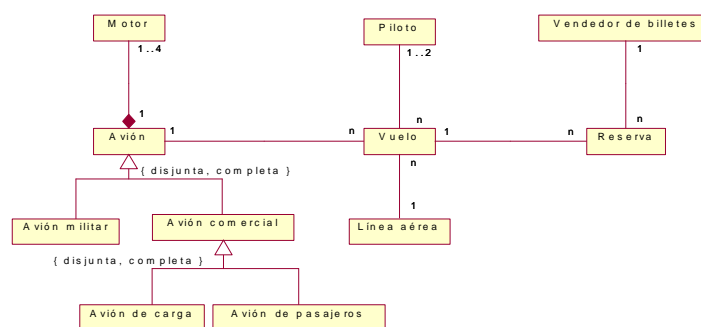
Figura 2: Representación gráfica de un Diagrama de Casos de Uso



Fuente. BOOCH, GRADY; RUMBAUGH, JAMES; JACOBSON, IVAR. El Lenguaje Unificado de Modelado. Madrid, 1999, Addison Wesley [38].

- ✓ **Diagrama de clases:** Muestra un conjunto de clases, interfaces y sus relaciones, es el diagrama más común a la hora de escribir un diseño de los sistemas orientados a objetos, (Figura 3) [38][39].

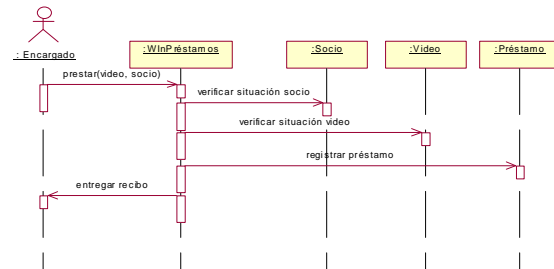
Figura 3: Representación gráfica de un Diagrama de clases



Fuente. BOOCH, GRADY; RUMBAUGH, JAMES; JACOBSON, IVAR. El Lenguaje Unificado de Modelado. Madrid, 1999, Addison Wesley [38].

- ✓ **Diagrama de secuencia:** Se muestra en la interacción entre los actores y los objetos que componen un caso de uso o una operación (Figura 4).

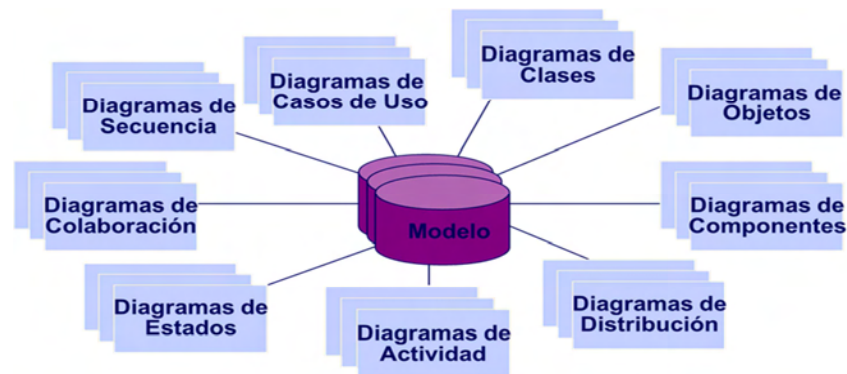
Figura 4: Representación gráfica de un Diagrama de secuencia



Fuente. BOOCH, GRADY; RUMBAUGH, JAMES; JACOBSON, IVAR. El Lenguaje Unificado de Modelado. Madrid, 1999, Addison Wesley [38].

Los diagramas expresan gráficamente las partes de un modelo (Figura 5). Un modelo es una abstracción del sistema centrada en una estructura semántica determinada. Los modelos están compuestos por un conjunto de diagramas según lo establezca la metodología de modelamiento seleccionada. Por ejemplo, en el Proceso Unificado, el Modelo de Análisis cuenta con diagramas de clases y de colaboración [38][39].

Figura 5: Representación de los diagramas que componen un Modelo

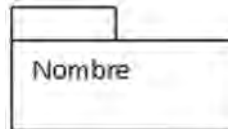


Fuente. BOOCH, GRADY; RUMBAUGH, JAMES; JACOBSON, IVAR. El Lenguaje Unificado de Modelado. Madrid, 1999, Addison Wesley. [38]

Todos estos diagramas deben organizarse en paquetes. Un paquete (Figura 6) es un mecanismo de propósito general para organizar elementos en grupos;

elementos estructurales, elementos de comportamiento, etc. Un paquete es puramente conceptual, es decir que solo existe en el tiempo de desarrollo. Gráficamente un paquete se visualiza como una carpeta, incluyendo normalmente su nombre y en ocasiones su contenido [38][39].

Figura 6: Representación gráfica de un paquete



Fuente. BOOCH, GRADY; RUMBAUGH, JAMES; JACOBSON, IVAR. El Lenguaje Unificado de Modelado. Madrid, 1999, Addison Wesley. [38]

**Relaciones:** En UML existen cuatro tipos de relaciones: dependencia, asociación, generalización y realización [38][39].

- ✓ **Dependencia:** es una relación semántica entre dos elementos. en la cual el cambio de un elemento puede afectar la semántica de otro, se representa gráficamente de la siguiente manera (Figura 7):

Figura 7: Representación gráfica de una relación tipo Dependencia



Fuente. BOOCH, GRADY; RUMBAUGH, JAMES; JACOBSON, IVAR. El Lenguaje Unificado de Modelado. Madrid, 1999, Addison Wesley. [38]

- ✓ **Asociación:** es una relación estructural que describe cómo una clase está asociada a otra al tener en sus atributos objetos de dicha clase (Figura 8). Existe una relación de asociación especial denominada agregación (Figura 9).

Figura 8: Representación gráfica de una relación tipo Asociación



Fuente. BOOCH, GRADY; RUMBAUGH, JAMES; JACOBSON, IVAR. El Lenguaje Unificado de Modelado. Madrid, 1999, Addison Wesley. [38]

- ✓ **Agregación:** es una relación de asociación que sirve para indicar que la asociación es de composición, es decir, que un objeto está compuesto por otro. La diferencia de la asociación y la agregación es prácticamente conceptual y se hace evidente cuando el objeto componente no puede existir o funcionar completamente sin el objeto agregado.

Por ejemplo, es posible que un objeto casa (de la clase Vivienda) esté compuesto por un objeto mueble (de la clase ComponentesDeLaVivienda) y por otro objeto paredes (de la clase MuroEstructural). En el primer caso la relación sería de *asociación* debido a que una casa sigue siendo casa si no posee sillas o muebles; pero, la segunda relación sería de *agregación* dado que una casa sin paredes no tendría aspecto de casa sino de kiosco.

Figura 9: Representación gráfica de una relación tipo Agregación



Fuente. BOOCH, GRADY; RUMBAUGH, JAMES; JACOBSON, IVAR. El Lenguaje Unificado de Modelado. Madrid, 1999, Addison Wesley. [38]

- ✓ **Generalización:** es una relación de especialización en la cual los objetos del elemento generalizado pueden sustituir los objetos del elemento general (hijo - padre). De esta forma el hijo comparte la estructura y el comportamiento del padre. (Figura 10)

Figura 10: Representación gráfica de una relación tipo Generalización



Fuente. BOOCH, GRADY; RUMBAUGH, JAMES; JACOBSON, IVAR. El Lenguaje Unificado de Modelado. Madrid, 1999, Addison Wesley. [38]

- ✓ **Realización:** es una relación semántica entre clasificadores, en donde un clasificador especifica un contrato que otro clasificador garantiza que cumplirá. Se pueden encontrar relaciones de realización entre interfaces y clases que las implementan, y entre casos de uso y colaboraciones que los realizan (Figura 11).

Figura 11: Representación gráfica de una relación tipo Realización



Fuente. BOOCH, GRADY; RUMBAUGH, JAMES; JACOBSON, IVAR. El Lenguaje Unificado de Modelado. Madrid, 1999, Addison Wesley. [38]

**2.3.3 El modelo del proceso unificado (RUP).** El Proceso Unificado (RUP) se basa en la metodología Orientada a Objetos (O.O.) y utiliza, para representar los elementos del sistema, los procesos y los flujos de datos, los diagramas del UML [39].

Aunque UML es independiente del proceso de desarrollo, los creadores (Grady Booch, Ivar Jacobson y Jim Rumbaugh [33][38][39]) de UML propusieron su propia metodología de desarrollo denominada Proceso Unificado de Desarrollo.

Las características principales de RUP son tres [33][38]:

- ✓ **Dirigido por casos de uso:** Un caso de uso representa una pieza de funcionalidad en el sistema que le devuelve al usuario un resultado de valor. Los casos de uso sirven para capturar requerimientos funcionales.
- ✓ **Centrado en la arquitectura:** Los casos de uso son desarrollados en conjunto con la arquitectura del sistema y se desarrollan a medida que lo hace el ciclo de vida.

- ✓ **Iterativo e incremental:** La vida de un sistema se encuentra dividida en ciclos. Cada ciclo termina con un lanzamiento de diferentes modelos del producto y consiste de cuatro fases.

El RUP se compone de 4 fases (ver Figura 12). Dichas fases, son una especie de mini proyectos en las que se repiten, iteración tras iteración, los pasos normales de un ciclo de vida de desarrollo. Es decir, el Análisis, el Diseño, la Implementación y las Pruebas, entre otras. Se realizan en cada una de las iteraciones, una y otra vez hasta completar la fase actual y pasar a la siguiente. Las iteraciones repetitivas van proporcionando resultados inmediatos comparables al desarrollo incremental basado en prototipos [33][39].

Las fases sirven para alcanzar hitos o puntos de toma de decisiones en cuanto a continuar o no con el desarrollo del proyecto dependiendo de los resultados obtenidos.

Las fases son [33][39]:

- ✓ *Inicio:* Se realizará una adecuada delimitación del alcance del software. Se establecerán los temas, subtemas y ejercicios que abordan tanto el módulo de información teórica como el módulo de ejercicios prácticos. Igualmente se establecerá el alcance del módulo de ejercicios planteados por el usuario.

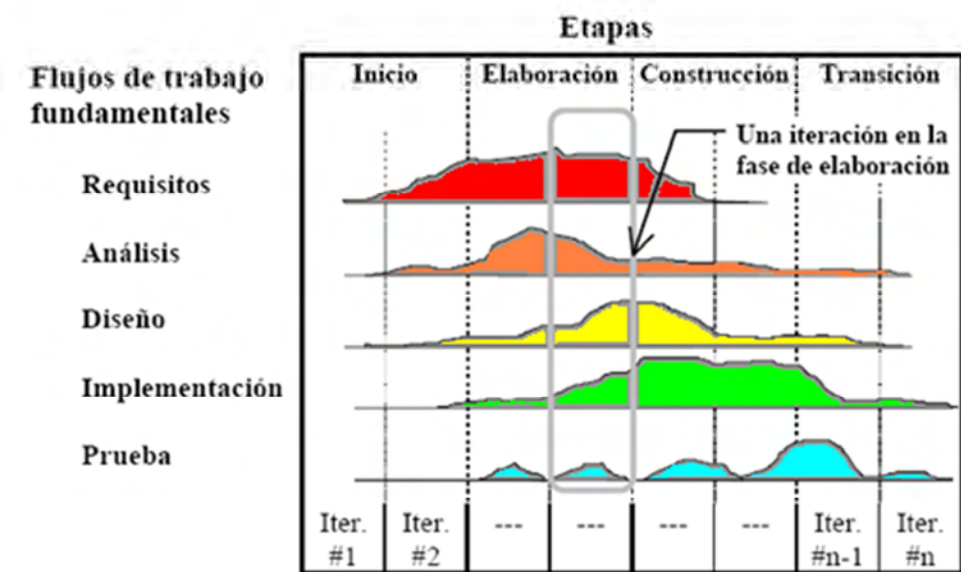
En cuanto a los criterios de éxito, se determinará el resultado esperado al terminar cada una de las iteraciones.

- ✓ *Elaboración:* Se determinará el mayor número de requisitos específicos del aplicativo y se realizará un prototipo con los casos de uso más significativos.

De esta manera se determinará de forma detallada el contenido de cada uno de los temas, subtemas y ejercicios, además de los respectivos requisitos del solucionador de ejercicios, entre ellos: la forma cómo el usuario se comunicará con él, el debido procedimiento que se llevará a cabo tras cada iteración en el desarrollo de un ejercicio y la forma en que el sistema proporcionará los resultados obtenidos.

- ✓ *Construcción*: Se desarrollará cada uno de los módulos diseñados para el sistema. El módulo de información teórica deberá explicar cualquiera de los temas, el módulo de ejercicios prácticos debe estar en capacidad de explicar cualquiera de los ejercicios del sistema y el solucionador de ejercicios debe poder realizar sin ningún problema cualquier ejercicio que el usuario le plantee. De cada una de las operaciones se realizarán las pruebas respectivas.
- ✓ *Transición*: Se realizarán las pruebas finales del aplicativo y se procederá a la utilización del sistema en una prueba piloto. En ella se observará cómo el grupo de estudiantes responde al uso del aplicativo y se determinará si es necesario implementar un requisito adicional. Al final de esta fase se tendrá el producto terminado: Pg\_KDD.

Figura 12: Etapas de Desarrollo



Fuente. BOOCH, GRADY; RUMBAUGH, JAMES; JACOBSON, IVAR. El Proceso Unificado de Desarrollo. Madrid, 2000, Addison Wesley. [39]

## 2.4 HERRAMIENTAS DE DESARROLLO

Entre las principales herramientas de desarrollo podemos encontrar al lenguaje de programación Java, el entorno de desarrollo NetBeans y la biblioteca gráfica Swing para Java.

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria [34].

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual (VM) y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre. Pero actualmente Oracle quien compro a Sun en el año 2009, es quien, junto con sus desarrolladores, lidera el desarrollo Java en el momento [34].

A continuación se explican brevemente los conceptos preliminares de estas herramientas para ser tenidas en cuenta en la posterior descripción de la implementación que se realiza en este capítulo.

**2.4.1 Lenguaje de Programación Java.** Java es un lenguaje de programación orientado a objetos desarrollado por James Gosling y sus compañeros de Sun Microsystems al inicio de la década de 1990. A diferencia de los lenguajes de programación convencionales, que generalmente están diseñados para ser compilados a código nativo, Java es compilado en un bytecode que es ejecutado

usando normalmente un compilador JIT (Just in Time), por una máquina virtual Java [34].

El lenguaje Java se crea con cinco objetivos principales [34]:

- ✓ Deberá usar la metodología de la programación orientada a objetos.
- ✓ Deberá permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- ✓ Deberá incluir por defecto soporte para trabajo en red.
- ✓ Deberá diseñarse para ejecutar código en sistemas remotos de forma segura.
- ✓ Deberá ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Entre sus principales características se encuentran:

**Orientado a Objetos [34].** La primera característica, orientado a objetos ("OO"), se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el software de forma que los distintos tipos de datos que use están unidos a sus operaciones. Así los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el "comportamiento" (el código) y el "estado" (datos) [34].

El principio es separar aquello que cambia de las cosas que permanecen inalterables. Frecuentemente, cambiar una estructura de datos implica un cambio en el código que opera sobre los mismos, o viceversa. Esta separación en objetos coherentes e independientes ofrece una base más estable para el diseño de un sistema software. El objetivo es hacer que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos. Otra de las grandes promesas de la programación orientada a objetos es la creación de entidades más Genéricas (objetos) que permitan la reutilización del software entre proyectos, una de las premisas fundamentales de la Ingeniería del Software. Un objeto genérico "cliente", por ejemplo, Deberá en teoría tener el mismo conjunto de comportamiento en diferentes proyectos, sobre todo cuando estos coinciden en cierta medida, algo que suele suceder en las grandes organizaciones. En este sentido, los objetos Podría verse como piezas reutilizables que pueden emplearse en múltiples proyectos distintos, posibilitando así a la industria del software a construir

proyectos de envergadura empleando componentes ya existentes y de comprobada calidad; conduciendo esto finalmente a una reducción drástica del tiempo de desarrollo. Podemos usar como ejemplo de objeto el aluminio. Una vez definidos datos (peso, maleabilidad, etc.), y su “comportamiento” (soldar dos piezas, etc.), el objeto “aluminio” puede ser reutilizado en el campo de la construcción, del automóvil, de la aviación, etc.

**Independencia de la plataforma [34].** La segunda característica, la independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Es lo que significa ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, “write once, run everywhere”. Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como “bytecode” (específicamente Java bytecode) que son instrucciones máquina simplificadas específicas de la plataforma Java. Esta pieza está “a medio camino” entre el código fuente y el código máquina que entiende el dispositivo destino. El bytecode es ejecutado entonces en la máquina virtual, un programa escrito en código nativo de la plataforma destino (que es el que entiende su hardware), que interpreta y ejecuta el código. Además, se suministran librerías adicionales para acceder a las características de cada dispositivo (como los gráficos, ejecución mediante hebras o threads, la interfaz de red) de forma unificada. Se debe tener presente que, aunque hay una etapa explícita de compilación, el bytecode generado es interpretado o convertido a instrucciones máquina del código nativo por el compilador JIT.

**El recolector de basura [34].** Un argumento en contra de lenguajes como C++ es que los programadores se encuentran con la carga añadida de tener que administrar la memoria de forma manual. En C++, el desarrollador debe asignar memoria en una zona conocida como heap (montículo) para crear cualquier objeto, y posteriormente desalojar el espacio asignado cuando desea borrarlo. Un olvido a la hora de desalojar memoria previamente solicitada, o si no lo hace en el instante oportuno, puede llevar a una fuga de memoria, ya que el sistema operativo piensa que esa zona de memoria está siendo usada por una aplicación cuando en realidad no es así así un programa mal diseñado Podría consumir una cantidad desproporcionada de memoria. Además, si una misma Región de memoria es desalojada dos veces el programa puede volverse inestable y llevar a un eventual cuelgue.

En Java, este problema potencial es evitado en gran medida por el recolector automático de basura (o automatic garbage collector). El programador determina cuándo se crean los objetos y el entorno en tiempo de ejecución de Java (Java runtime) es el responsable de gestionar el ciclo de vida de los objetos. El programa, u otros objetos pueden tener localizado un objeto mediante una referencia a éste (que, desde un punto de vista de bajo nivel es una dirección de memoria). Cuando no quedan referencias a un objeto, el recolector de basura de Java borra el objeto, liberando así la memoria que ocupaba previniendo posibles fugas (ejemplo: un objeto creado y únicamente usado dentro de un método sólo tiene entidad dentro de éste; al salir del método, el objeto es eliminado). Aun así es posible que se produzcan fugas de memoria si el código almacena referencias a objetos que ya no son necesarios es decir, pueden aún ocurrir, pero en un nivel conceptual superior. En definitiva, el recolector de basura de Java permite una fácil creación y eliminación de objetos, mayor seguridad y frecuentemente más rápida que en C++.

La recolección de basura de Java es un proceso prácticamente invisible al desarrollador. Es decir, el programador no tiene conciencia de cuándo la recolección de basura tendrá lugar, ya que ésta no tiene necesariamente que guardar relación con las acciones que realiza el código fuente. Debe tenerse en cuenta que la memoria es sólo uno de los muchos recursos que deben ser gestionados.

**2.4.2 Entorno de Desarrollo NetBeans.** NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un Entorno integrado de desarrollo (IDE) desarrollado usando la Plataforma NetBeans [44].

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de Java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software [44].

La Plataforma NetBeans es un framework reusable que simplifica el desarrollo de otras aplicaciones de escritorio. Cuando se ejecuta una aplicación basada en la

Plataforma NetBeans, la plataforma ejecuta la clase Main. Los módulos disponibles están localizados y puestos en un registro en memoria, y son ejecutadas las tareas de inicialización de los módulos. Generalmente el código de un módulo es cargado en memoria solo cuando se necesita. La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación [44].

Entre las características de la plataforma están [44]:

- ✓ Administración de las interfaces de usuario (ej. menús y barras de herramientas).
- ✓ Administración de las configuraciones del usuario.
- ✓ Administración del almacenamiento (guardando y cargando cualquier tipo de dato).
- ✓ Administración de ventanas.
- ✓ Framework basado en asistentes (diálogos paso a paso).

**2.4.3 Controlador JDBC.** JDBC es el acrónimo de Java Database Connectivity, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema de operación donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice [45].

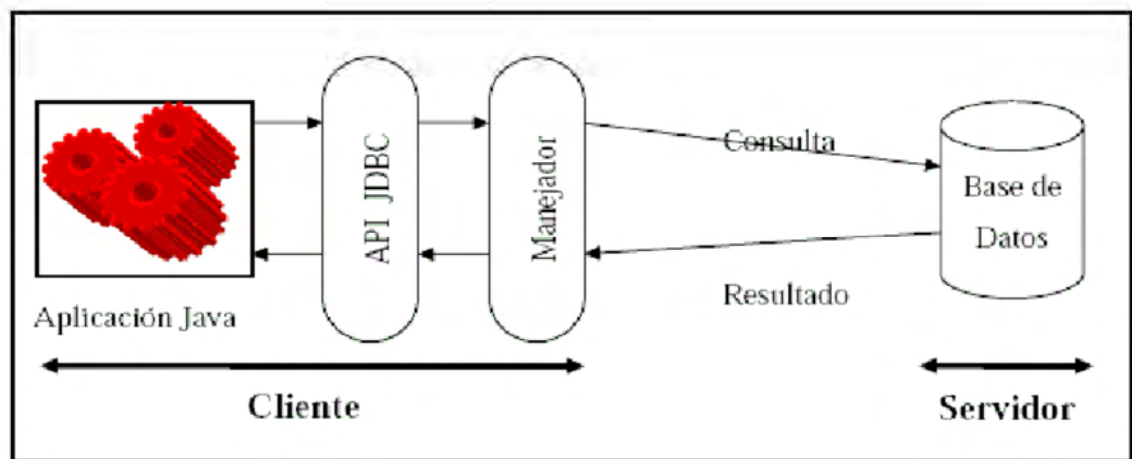
El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar. Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la librería de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión, para ello provee en localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar cualquier tipo de tareas con la base de datos a las que tenga permiso: consultas, actualizaciones, creado modificado y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, etc. [45].

Cada base de datos emplea un protocolo diferente de comunicación, protocolos que normalmente son propietarios. El uso de un manejador, una capa intermedia

entre el código del desarrollador y la base de datos, permite independizar el código Java que accede a la BD del sistema de BD concreto a la que estamos accediendo, ya que en nuestro código Java emplearemos comandos estándar, y estos comandos serían traducidos por el manejador a comandos propietarios de cada sistema de BD concreto. Si queremos cambiar el sistema de BD que empleamos lo único que deberemos hacer es reemplazar el antiguo manejador por el nuevo, y seremos capaces de conectarnos la nueva BD [45].

**2.4.4 Manejador de Protocolo Nativo (tipo 4).** El manejador traduce directamente las llamadas al API JDBC al protocolo nativo de la base de datos (ver Figura 13). Es el manejador que tiene mejor rendimiento, pero está más ligado a la base de datos que empleemos que el manejador tipo JDBC-Net, donde el uso del servidor intermedio nos da una gran flexibilidad a la hora de cambiar de base de datos. Este tipo de manejadores también emplea Tecnología 100 % Java [46].

Figura 13: Representación manejo protocolo nativo JAVA



Fuente. El ABC de Jdbc. <http://es.scribd.com/doc/84530902/EI-ABC-Jdbc2>.

**2.4.5 Biblioteca gráfica Swing de Java.** Swing es una biblioteca gráfica para Java que forma parte de las Java Foundation Classes (JFC). Incluye componentes para interfaz gráfica de usuario tales como cajas de texto, botones, listas desplegables y tablas [47].

Las Internet Foundation Classes (IFC) eran una biblioteca gráfica para el lenguaje de programación Java desarrollada originalmente por Netscape y que se publicó en 1996. En 1997, Sun Microsystems y Netscape Communications Corporation anunciaron su intención de combinar IFC con otras Tecnología de las Java Foundation Classes. Además de los componentes ligeros suministrados originalmente por la IFC, Swing introdujo un mecanismo que permite que el aspecto de cada componente de una aplicación pudiese cambiar sin introducir cambios sustanciales en el código de la aplicación. La introducción de soporte ensamblable para el aspecto permitió a Swing emular la apariencia de los componentes nativos manteniendo las ventajas de la independencia de la plataforma [47].

Algunos de los componentes utilizados en el desarrollo de este proyecto y pertenecientes a la biblioteca Swing se explican a continuación:

- ✓ JComponent. La clase base para todos los componentes de Swing exceptuando los contenedores de nivel superior. Para utilizar un componente que herede de JComponent, se debe poner el componente en una jerarquía de la contención cuya raíz sea un contenedor de nivel superior Swing tal como Ventanas o Paneles. Los contenedores de nivel superior son los componentes especializados que proporcionan un lugar para que otros componentes de Swing puedan pintarse.
- ✓ JFrame. Un JFrame es una ventana de nivel superior con un título y un borde. El tamaño del JFrame incluye cualquier área señalada por los bordes donde puede ser incluido uno o más componentes de Swing.
- ✓ JPanel. El JPanel es la clase más simple del contenedor. Un JPanel proporciona el espacio en el cual una aplicación puede unir cualquier otro componente, incluyendo otros paneles.
- ✓ JTabbedPane. Un componente que deja a usuario cambiar entre un grupo de componentes pulsando en una etiqueta con un título dado y/o un icono. Las etiquetas y los componentes son agregados a un objeto de TabbedPane usando los métodos del addTab e insertTab. Una etiqueta es representada por un índice que corresponde a la posición que fue agregado adentro, donde la primera etiqueta tiene un índice igual a 0 y la etiqueta pasada tiene un índice igual a la cuenta de la etiqueta menos 1.
- ✓ JSplitPane. JSplitPane se utiliza para dividir dos (y solamente dos) componentes. Los dos componentes se pueden entonces volver a clasificar según el tamaño recíprocamente por el usuario. La información sobre cómo usar JSplitPane está en cómo utilizar los paneles partidos en la clase

particular de Java. Se pueden incluir nuevos componentes en cada una de las divisiones de un JSplitPane. Los dos componentes en un panel partido se pueden alinear a la izquierda y a la derecha usando JSplitPane.HORIZONTAL\_SPLIT, o en la parte superior o inferior de la ventana con JSplitPane.VERTICAL\_SPLIT.

- ✓ JScrollPanne. JScrollPanne proporciona una vista deslizante de un componente ligero. Un JScrollPanne maneja un viewport, barras de desplazamiento vertical y horizontal opcional, y viewports opcionales del título de la fila y de columna. El viewport, o punto de vista, proporciona una ventana sobre una fuente de datos, por ejemplo, un archivo de texto o una imagen sobre la cual se va a deslizar.
- ✓ JFileChooser. Los seleccionadores de archivos proporcionan una interfaz gráfica de usuario para navegar el sistema de ficheros, y después elegir un archivo o un directorio de una lista o incorporar el nombre de un archivo o un directorio. Para exhibir un seleccionador de archivo, se utiliza generalmente el JFileChooser para mostrar un diálogo modal que contiene el seleccionador de archivo. Otra manera de presentar un seleccionador es agregar una instancia.
- ✓ JTable. Con la clase de JTable se puede exhibir una tabla de datos, permitiendo opcionalmente que el usuario edite el contenido de las celdas que contienen los datos. JTable no contiene ni deposita datos; es simplemente una vista de los datos.
- ✓ JTree. Con la clase JTree, puedes exhibir datos jerárquicos. Un objeto JTree no contiene realmente los datos sino que proporciona simplemente una vista de ellos. Como cualquier componente no trivial del Swing, el árbol consigue los datos conectándose a un modelo de los datos, un excelente componente para manejo de visualización de conexiones, bases de datos, tablas, campos, restricciones y demás en para este proyecto en específico y para proyectos similares o que requieran de manejo de este útil objeto de la Swing de Java.
- ✓ JTextArea. Un JTextArea es un área multilínea que exhibe el texto plano y que permite opcionalmente que el usuario corrija el texto. Este es un componente ligero que provee de compatibilidad a la hora de introducción o despejar pequeñas cantidades de información.
- ✓ JSpinner. Esta clase provee en una sola línea la posibilidad al usuario de entrar o seleccionar un valor de una secuencia pedida. Los JSpinner proporciona típicamente un par de los botones minúsculos de flechas para cambiar entre los elementos de la secuencia. Las teclas de flecha arriba/abajo del teclado también completan un ciclo a través de los

elementos. El usuario puede también mecanografiar el valor directamente en un JSpinner.

- ✓ JComboBox. Un componente que combina un botón o un campo editable y una lista desplegable. El usuario puede seleccionar un valor de la lista, que aparece a petición del usuario. Si la caja de texto es editable, entonces se incluye un campo en el cual el usuario pueda mecanografiar un valor.
- ✓ JRadioButton. Una puesta en práctica de un botón de radio. Este es un componente que puede ser seleccionado o deseleccionado, y que exhibe su estado al usuario. Utilizado con un objeto ButtonGroup crea un grupo de botones en los cuales solamente un botón puede ser seleccionado a la vez.

**2.4.6 Look and Feel (Skin para aplicaciones JAVA).** Normalmente las ventanas JAVA tienen su propio aspecto, su propio estilo de botones y demás. Con Java es muy fácil cambiar el aspecto (Look and Feel o skin) de nuestras ventanas para que tengan aspecto Java, aspecto Linux, Windows, Mac etc. Basta con tener la librería adecuada y una sola línea de código [35][36].

Para obtener el aspecto de ventanas propio del sistema operativo en el que estemos (Linux, Windows Xp, Solaris, etc.), basta con estas líneas de código antes de crear ninguna ventana [48]:

```
import javax.swing.UIManager;
...
try
{
    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
}
catch (Exception e)
{
    e.printStackTrace();
}
```

Con `UIManager.getSystemLookAndFeelClassName()` obtenemos el nombre del Look and Feel por defecto del sistema operativo en el que estemos. Con `UIManager.setLookAndFeel` decimos qué Look and Feel queremos.

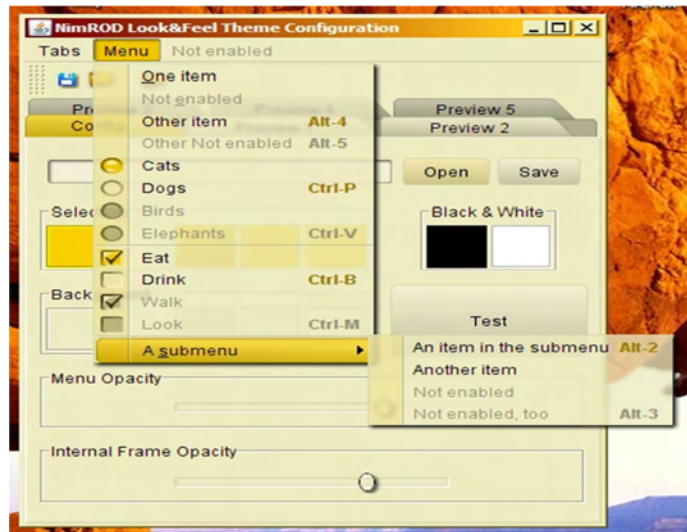
Hay muchos skins look and feel para las ventanas y solo es necesario descargar la librería y agregarla a la biblioteca en el proyecto de NetBeans, para Pg\_KDD se utilizó NimROD muy útil para aplicaciones de múltiples ventanas [35][36].

También, se utilizó una librería denominada InfoNode que contiene al igual que NimROD, skins para visualización agradable de las aplicaciones JAVA, pero este también contiene manejo de ventaneo, distribución de paneles (Tabbed Panel) y acoplamiento de ventanas (Docking Windows) que hace fácil el manejo de las ventanas para el usuario final.

**2.4.6.1 NimROD (Look & Feel).** NimROD es un Look & Feel para mejorar el aspecto de las aplicaciones de Java, de fácil implementación dentro del código fuente, NimROD Look & Feel se libera bajo licencia LGPL, de este modo se puede usar el binario de la librería sin ningún problema, tanto desde aplicaciones que tengan su código liberado como para los que tengas código restringido por derechos de autor. También es posible consultar el código fuente y modificarlo para uso personal, pero si hace público ese código modificado de cualquier manera está obligado a hacer públicas también las fuentes. [35][36]

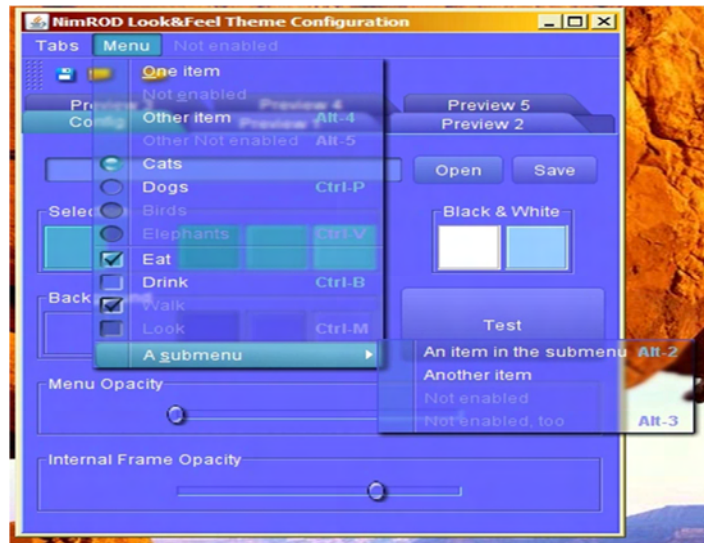
En la Figura 14, Figura 15 y Figura 16 se muestran algunos pantallazos de la última versión de NimROD Look & Feel:

Figura 14: Representación NimROD Typical



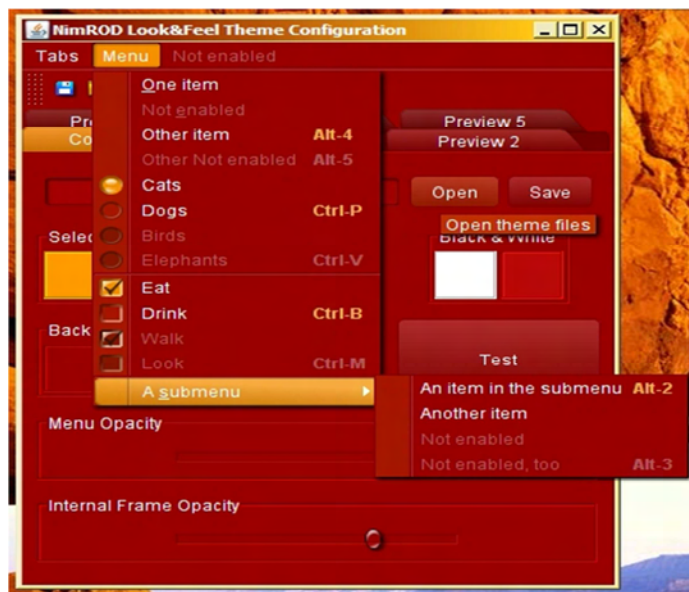
Fuente. Internet - Look & Feel - Nilo J. González, Madrid, <http://personales.ya.com/nimrod/index.html>

Figura 15: Representación NimROD Blue



Fuente. Internet - Look & Feel - Nilo J. González, Madrid, <http://personales.ya.com/nimrod/index.html>

Figura 16: Representación NimROD Red



Fuente. Internet - Look & Feel - Nilo J. González, Madrid, <http://personales.ya.com/nimrod/index.html>

**2.4.6.2 InfoNode (Doking Windows – Tabbed Panel).** InfoNode es una gama de productos de software de NNL Technology AB. Ofrece soluciones

basadas en Java Swing GUI. Sus productos permiten construir modernas y avanzadas interfaces gráficas de usuario para las aplicaciones Java y así reducir sus costos de desarrollo y acortar el tiempo de comercialización [36].

Los productos ofrecidos por NNL Technology AB con InfoNode son [36]:

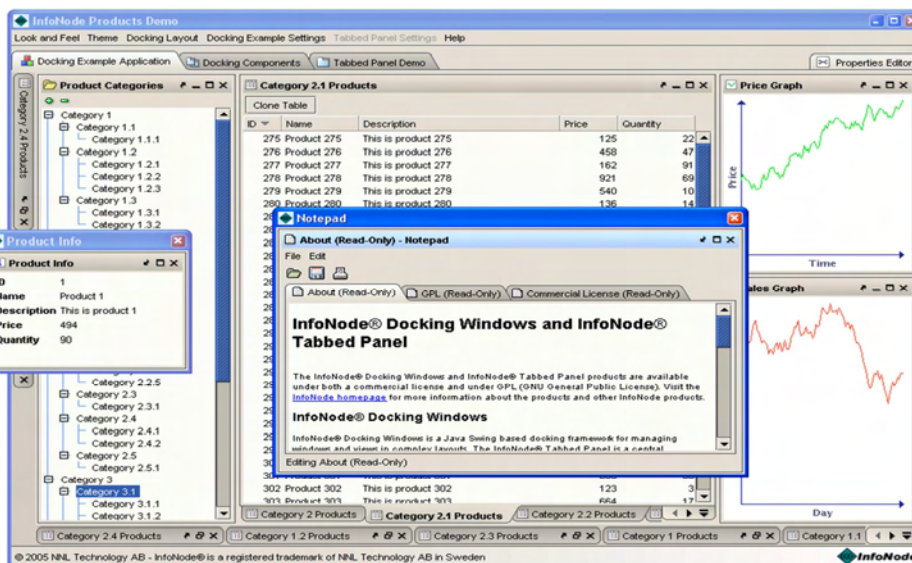
- ✓ InfoNode Docking Windows: Un componente Java basada en Swing que permite el acoplamiento de ventanas.

El Doking Windows InfoNode permite crear una GUI Swing de gran eficacia para su aplicación y con muy poco código. La aplicación sólo contendrá los componentes Swing normal con contenido de aplicación y los colocará dentro de las ventanas del marco. Las ventanas pueden estar dispersas u ordenadas en diseños avanzados utilizando Split, ventanas en pestañas y ventanas flotantes.

Una ventaja importante de InfoNode Docking Windows es que a través de un GUI Swing normal el usuario puede personalizar el diseño de la aplicación a su preferencia o gusto. El usuario puede minimizar, maximizar, cerrar, desacoplar, acoplar y moverse por las ventanas para crear un diseño de ventana personalizada. Usted puede incluso tomar ventaja de las pantallas múltiples desacoplando ventanas y pasar a otras pantallas. Cuando el usuario está satisfecho con el diseño puede ser fácilmente guardado y restaurado cada vez que se inicie la aplicación. Por supuesto no hay ninguna limitación en el número de diseños de ventana que se pueden guardar [36].

También puede utilizar InfoNode Docking Windows (Figura 17) para experimentar con prototipos y diseños de interfaz de usuario diferente para su aplicación. Se mueven alrededor de las ventanas hasta que esté satisfecho con su diseño, guardar el diseño y el uso está a disposición permanentemente como el diseño predeterminado que el usuario mismo diseño y así todo el diseño quede congelado según la preferencia del mismo.

Figura 17: Representación InfoNode Doking Windows



Fuente. InfoNode – Doking Windows – Tabbed Panel, NNL Technology AB. 2009, <http://www.infonode.net/index.html?idw>

- ✓ InfoNode Tabbed Panel: Un componente basado en Java Swing que permite crear un panel de pestañas.

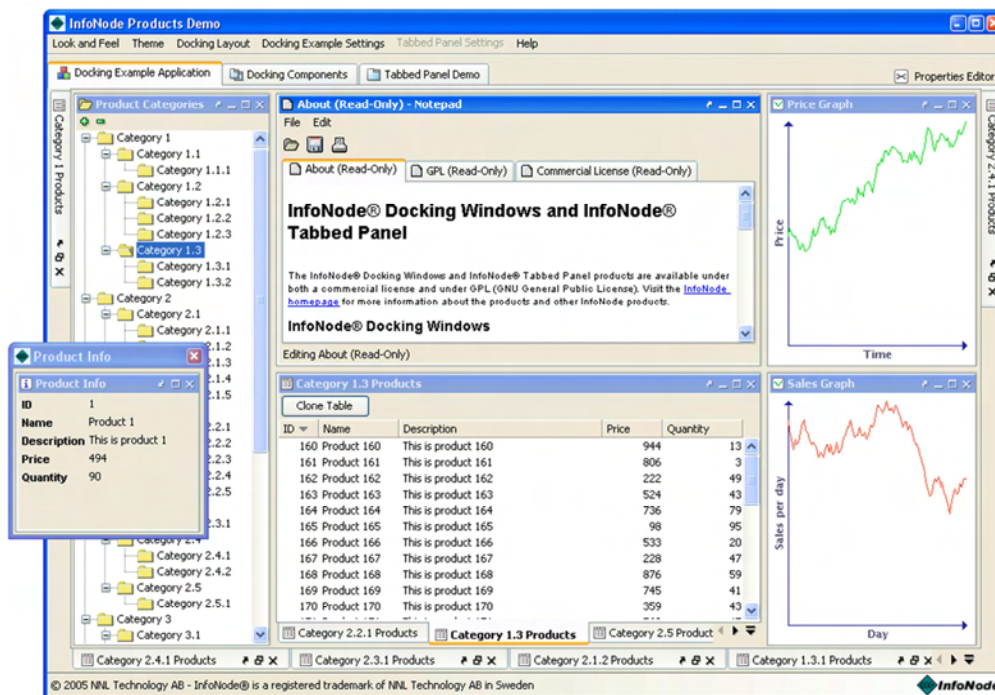
El Doking Windows utiliza el potente Tabbed Panel de InfoNode el cual se incluye en la distribución de las ventanas de la aplicación y se puede utilizar como un componente independiente para el manejo de pestañas que incluyen ventanas dentro de un panel. De acuerdo a ello InfoNode también incluye un estilo ordenado en la distribución de las aplicaciones JAVA que requieran del manejo de varias ventanas a la vez [36].

- ✓ InfoNode Look and Feel: Una variante de Java look and feel con un aspecto metálico.

Casi todos los aspectos de la apariencia del marco InfoNode Docking Windows son personalizables a través de un sistema de propiedades avanzadas. Hay más de 2.500 propiedades que se pueden cambiar. Se tiene la posibilidad de crear una apariencia única para su aplicación o elegir uno de los temas incluidos en la distribución. InfoNode Docking Windows le permite crear fichas de cualquier forma, crear fondos

degradados, crear interesantes efectos de la liberación del ratón y añadir sombras a las ventanas de la ficha (Figura 18).

Figura 18: Representación Look and Feel InfoNode Windows XP



Fuente. InfoNode – Doking Windows – Tabbed Panel, NNL Technology AB. 2009, <http://www.infonode.net/index.html?idw>

### 3. OPERADORES ALGEBRAICOS, PRIMITIVAS SQL Y ALGORITMOS PARA LAS TAREAS DE ASOCIACIÓN Y CLASIFICACIÓN

PostgresKDD es una herramienta computacional para el descubrimiento de conocimiento fuertemente acoplado con el SGBD PostgreSQL, al cual se la han implementado operadores y primitivas SQL propuestas por Timaran [31] para dotarlo de las capacidades de realizar minería de datos en las tareas de Asociación y Clasificación. Los operadores y primitivas SQL propuestos e implementados en PostgresKDD son:

#### 3.1 OPERADORES DEL ÁLGEBRA RELACIONAL Y NUEVAS PRIMITIVAS SQL PARA ASOCIACIÓN

Los siguientes son los nuevos operadores con los cuales se extiende el álgebra relacional para soportar eficientemente la tarea de Asociación propuestos por Timaran [31].

**3.1.1 Operador Associator y la primitiva Associator Range.** A continuación se describe el operador Associator y la primitiva Associator Range.

##### Operador Associator ( $\alpha$ ) [31]

*Associator* ( $\alpha$ ) es un operador algebraico unario que a diferencia del operador Selección o Restricción ( $\sigma$ ), aumenta la cardinalidad de una relación. *Associator* genera, a partir de cada tupla de una relación, todas las posibles combinaciones (de diferente tamaño) de los valores de sus atributos, como tuplas de una nueva relación, conservando el esquema de la relación inicial. Los atributos de la relación inicial pueden pertenecer a diferentes dominios [31].

Su sintaxis es la siguiente:

$\alpha_{\text{tamaño\_inicial}, \text{tamaño\_final}}(R)$

Donde  $\langle tamaño\_inicial \rangle$  y  $\langle tamaño\_final \rangle$  son dos parámetros de entrada que determinan el tamaño inicial y tamaño final de las combinaciones.

El operador *Associator* genera, por cada tupla de la relación  $R$ , todos sus posibles subconjuntos (conjuntos de ítems) de diferente tamaño. *Associator* toma cada tupla  $t$  de  $R$  y dos parámetros  $\langle tamaño\_inicial \rangle$  y  $\langle tamaño\_final \rangle$  como entrada, y retorna, por cada tupla  $t$ , las diferentes combinaciones de atributos  $X_i$ , de tamaño  $\langle tamaño\_inicial \rangle$  hasta tamaño  $\langle tamaño\_final \rangle$ , como tuplas en una nueva relación. El orden de los atributos en el esquema de  $R$  determina los atributos en los subconjuntos con valores, el resto se hacen nulos. El tamaño máximo de un ítemset y por consiguiente el tamaño final máximo ( $\langle tamaño\_final \rangle$ ) que se puede tomar como entrada es el correspondiente al valor del grado de la relación [31].

A pesar de que generar todas las posibles combinaciones de los valores, para cada tupla de una relación, es un proceso computacionalmente costoso, *Associator* lo hace en una sola pasada sobre la base de datos.

*Associator* facilita el cálculo de ítemsets frecuentes para el descubrimiento de reglas de asociación multidimensionales.

**Ejemplo. Timarán [31].** Sea la relación  $R(A, B, C, D)$  de la Tabla 1, Encontrar las diferentes combinaciones de tamaño 2 hasta tamaño 4, es decir  $R1 = \alpha_{2,4}(R)$ .

Tabla 1: Relación R

| A  | B  | C  | D  |
|----|----|----|----|
| a1 | b1 | c1 | d1 |
| a1 | b2 | c1 | d2 |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 5 [31]

El resultado del operador *Associator* se muestra en la Tabla 2.

Tabla 2: Resultado Associator

| A    | B    | C    | D    |
|------|------|------|------|
| a1   | b1   | null | null |
| a1   | null | c1   | null |
| a1   | null | null | d1   |
| null | b1   | c1   | null |
| null | b1   | null | d1   |
| null | null | c1   | d1   |
| a1   | b1   | c1   | null |
| a1   | b1   | null | d1   |
| a1   | null | c1   | d1   |
| null | b1   | c1   | d1   |
| a1   | b1   | c1   | d1   |
| a1   | b2   | null | null |
| a1   | null | c1   | null |
| a1   | null | null | d2   |
| null | b2   | c1   | null |
| null | b2   | null | d2   |
| null | null | c1   | d2   |
| a1   | b2   | c1   | null |
| a1   | b2   | null | d2   |
| a1   | null | c1   | d2   |
| null | b2   | c1   | d2   |
| a1   | b2   | c1   | d2   |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 5 [31]

### Primitiva Associator Range [31].

Esta primitiva implementa el operador algebraico *Associator* en la cláusula SQL SELECT. *Associator* permite obtener por cada tupla de una tabla, todos los posibles subconjuntos desde un tamaño inicial hasta un tamaño final determinado por la cláusula RANGE [31].

Dentro de la cláusula SELECT, la primitiva *Associator Range* tiene la siguiente sintaxis [31]:

```
SELECT <ListaAtributosTablaDatos> [INTO <NombreTablaAssociator>]
FROM <NombreTablaDatos>
WHERE <CláusulaWhere>
ASSOCIATOR RANGE <valor1> UNTIL <valor2>
GROUP BY <ListaAtributosTablaDatos>
```

```
<ListaAtributosTablaDatos>::=<Atributo>,
<ListaAtributos><ListaAtributos>::=<Atributo>,
<ListaAtributos><ListaAtributos>::=<Atributo>
```

```
<valor1>::= 1,2,3, 4 ...
<valor2>::= 1,2,3,4 ...
```

Donde:

La cláusula SELECT <ListaAtributosTablaDatos> permite seleccionar los atributos de la tabla de datos <NombreTablaDatos> cuyos valores formarán los diferentes subconjunto o ítemsets como tuplas de la nueva tabla <NombreTablaAssociator> [31].

La cláusula INTO <NombreTablaAssociator> define el nombre de la tabla <NombreTablaAssociator> donde se almacenarán los resultados de ASSOCIATOR RANGE con los atributos especificados en la cláusula SELECT <ListaAtributosTablaDatos> como esquema. Si no se especifica la cláusula INTO, el resultado se almacenará en una tabla temporal como sucede normalmente en el lenguaje SQL [31].

La cláusula FROM <NombreTablaDatos> WHERE <CláusulaWhere> determina el nombre de la tabla <NombreTablaDatos> de donde se extraerá el conjunto de tuplas que cumplan las restricciones de la <CláusulaWhere> para formar las diferentes combinaciones o ítemsets [31].

La cláusula ASSOCIATOR RANGE <número1> UNTIL <número2> determina el tamaño inicial hasta el final de los diferentes subconjuntos o ítemsets que formará el operador *Associator* [31].

La cláusula GROUP BY <ListaAtributosTablaDatos> agrupa la tabla <NombreTablaDatos> por los atributos especificados en <ListaAtributosTablaDatos>

La primitiva ASSOCIATOR RANGE facilita el cálculo de los ítemsets frecuentes para el descubrimiento de Reglas de Asociación en tablas multicolumna [31].

**Ejemplo. Timarán [31].** Sea la siguiente tabla, Estudiantes(PROGRAMA, EDAD, SEXO, ESTRATO, PROMEDIO), cuya descripción se observa en la Tabla 3. Para la tabla Estudiantes, calcular el soporte para las ítemsets de tamaño 2 y 3 formados por lo atributos PROGRAMA, SEXO, ESTRATO y almacenar los resultados en la tabla Assostudent.

Tabla 3: Tabla Estudiantes

| PROGRAMA   | EDAD   | SEXO | ESTRATO | PROMEDIO   |
|------------|--------|------|---------|------------|
| Sistemas   | 16..20 | M    | 2       | Medio alto |
| Idiomas    | 21..25 | F    | 3       | Regular    |
| Sistemas   | 16..20 | F    | 3       | Regular    |
| Física     | 21..25 | M    | 2       | Bajo       |
| Psicología | 21..25 | F    | 4       | Alto       |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 20 [31]

Utilizando la nueva primitiva *Associator Range*, esta consulta se realiza con la siguiente sentencia SQL [31]:

```
SELECT programa, sexo, estrato, count(*) AS soporte INTO assostudent
FROM estudiantes
ASSOCIATOR RANGE 2 UNTIL 3
GROUP BY programa, sexo, estrato
```

La anterior sentencia SQL se ejecuta de la siguiente manera [31]:

Inicialmente se seleccionan de la tabla *Estudiantes*, los atributos *programa*, *sexo* y *estrato*. Luego se ejecuta la cláusula ASSOCIATOR RANGE, es decir, se procede a generar los ítemsets de tamaño 2 y 3 como se muestra en la Tabla 4.

Tabla 4: Resultado Associator Range.

| PROGRAMA | SEXO | ESTRATO |
|----------|------|---------|
| Sistemas | M    | Null    |
| Sistemas | Null | 2       |
| Null     | M    | 2       |
| Sistemas | M    | 2       |
| Idiomas  | F    | Null    |
| Idiomas  | Null | 3       |
| Null     | F    | 3       |
| Idiomas  | F    | 3       |
| Sistemas | F    | Null    |
| Sistemas | Null | 3       |
| Null     | F    | 3       |
| Sistemas | F    | 3       |
| Idiomas  | M    | Null    |
| Idiomas  | Null | 3       |
| Null     | M    | 3       |
| Idiomas  | M    | 3       |
| Sistemas | F    | Null    |
| Sistemas | Null | 2       |
| Null     | F    | 2       |
| Sistemas | F    | 2       |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 21 [31]

Finalmente se agrupa por *programa*, *sexo* y *estrato* de acuerdo a la cláusula GROUP BY, se calcula la función agregada *count()* y el resultado se almacena en la tabla *Assostudent* como lo muestra en la Tabla 5 [31].

Tabla 5: Assostudent

| PROGRAMA | SEXO | ESTRATO | SOPORTE |
|----------|------|---------|---------|
| Sistemas | F    |         | 2       |
| Sistemas | M    |         | 1       |
| Sistemas | F    | 2       | 1       |
| Sistemas | M    | 2       | 1       |
| Sistemas |      | 2       | 2       |
| Sistemas | F    | 3       | 1       |
| Sistemas |      | 3       | 1       |
| Idiomas  | F    |         | 1       |
| Idiomas  | M    |         | 1       |

| PROGRAMA | SEXO | ESTRATO | SOPORTE |
|----------|------|---------|---------|
| Idiomas  | F    | 3       | 1       |
| Idiomas  | M    | 3       | 1       |
| Idiomas  |      | 3       | 2       |
|          | F    | 2       | 1       |
|          | M    | 2       | 1       |
|          | F    | 3       | 2       |
|          | M    | 3       | 1       |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 21 [31]

**Ejemplo. Timarán [31].** Obtener de la tabla *Estudiantes* del ejemplo anterior, los ítemsets frecuentes de tamaño 2 y 3 que cumplan con un soporte mínimo mayor o igual a 2.

Los ítemsets frecuentes se obtienen con la siguiente sentencia SQL [31]:

```
SELECT programa, sexo, estrato, count(*) AS soporte INTO assostudent
FROM estudiantes
ASSOCIATOR RANGE 2 UNTIL 3
GROUP BY programa, sexo, estrato HAVING count(*) >= 2
```

El resultado final se muestra en la Tabla 6.

Tabla 6: ítemsets frecuentes tamaño 2 y 3 con soporte >= 2

| PROGRAMA | SEXO | ESTRATO | SOPORTE |
|----------|------|---------|---------|
| Sistemas | F    |         | 2       |
| Sistemas |      | 2       | 2       |
| Idiomas  |      | 3       | 2       |
|          | F    | 3       | 2       |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 21 [31]

**3.1.2 Operador Assorow y la primitiva Assorow Range.** A continuación se describe el operador Assorow y la primitiva Assorow Range.

### Operador Assorow ( $\rho$ ) [31].

*Assorow*, al igual que *Associator*, es un operador unario que genera, por cada tupla de la relación  $R$ , todos sus posibles subconjuntos (ítemsets) de diferente tamaño, pero sin hacer nulos los atributos que no forman el ítemset. Los valores de los atributos que se hacen nulos siempre serán los atributos de la parte derecha de cada tupla, que por el tamaño del ítemset, no se combinan. *Assorow* es un operador que requiere que todos los atributos de la relación  $R$  a la cual se aplica, sean del mismo dominio (i.e. del mismo tipo) [31].

La sintaxis de *Assorow* es la siguiente [31]:

$$\alpha\rho_{\text{tamaño\_inicial,tamaño\_final}}(R)$$

Donde  $\langle \text{tamaño\_inicial} \rangle$  y  $\langle \text{tamaño\_final} \rangle$  son dos parámetros de entrada que determinan el tamaño inicial y tamaño final de las combinaciones.

*Assorow* toma cada tupla  $t$  de  $R$  y los tamaños inicial y final ( $\langle \text{tamaño\_inicial} \rangle$ ,  $\langle \text{tamaño\_final} \rangle$ ) de los ítemsets como entrada, y retorna, una nueva relación con el mismo esquema de  $R$ , y con las diferentes combinaciones de atributos  $X_i$  de tamaño  $\langle \text{tamaño\_inicial} \rangle$  hasta tamaño  $\langle \text{tamaño\_final} \rangle$ , como tuplas. En cada combinación  $X_i$ , los atributos con valores serán los del tamaño generado, iniciando siempre del primer atributo del esquema de  $R$  (por tener  $R$  atributos de un mismo dominio), el resto de atributos se hacen nulos [31].

El tamaño máximo de un ítemset y por consiguiente el tamaño final máximo ( $\langle \text{tamaño\_final} \rangle$ ) que se puede tomar como entrada es el correspondiente al valor del grado de la relación [31].

*Assorow* facilita el cálculo de ítemsets frecuentes para el descubrimiento de reglas de asociación unidimensionales [31].

**Ejemplo. Timarón [31].** Sea la relación  $R(A,B,C,D)$  de la Tabla 1, en la cual  $\text{dom}(A)=\text{dom}(B)=\text{dom}(C)=\text{dom}(D)$ , obtener el resultado de aplicar *Assorow* a la relación  $R$ :  $R1 = \alpha\rho_{2,4}(R)$ .

El resultado de *Assorow* se muestra en la Tabla 7.

Tabla 7: Resultado operación  $R1=ap_{2,4}(R)$

| A  | B  | C    | D    |
|----|----|------|------|
| a1 | b1 | null | null |
| a1 | c1 | null | null |
| a1 | d1 | null | null |
| b1 | c1 | null | null |
| b1 | d1 | null | null |
| c1 | d1 | null | null |
| a1 | b1 | c1   | null |
| a1 | b1 | d1   | null |
| a1 | c1 | d1   | null |
| b1 | c1 | d1   | null |
| a1 | b1 | c1   | d1   |
| a1 | b2 | null | null |
| a1 | c1 | null | null |
| a1 | d2 | null | null |
| b2 | c1 | null | null |
| b2 | d2 | null | null |
| c1 | d2 | null | null |
| a1 | b2 | c1   | null |
| a1 | b2 | d2   | null |
| a1 | c1 | d2   | null |
| b2 | c1 | d2   | null |
| a1 | b2 | c1   | d2   |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 5 [31]

### Primitiva Assorow Range [31]

Esta primitiva implementa el operador algebraico Assorow en la cláusula SQL SELECT. Assorow permite obtener por cada tupla de una tabla, todos los posibles subconjuntos desde un tamaño inicial hasta un tamaño final determinado por la cláusula RANGE siempre y cuando todos los atributos de la tabla de datos sean del mismo tipo [31].

Dentro de la cláusula SELECT, Assorow tiene una sintaxis equivalente a Associator [31]:

```

SELECT <ListaAtributosTablaDatos> [INTO <NombreTablaAssorow>]
FROM <NombreTablaDatos> WHERE <CláusulaWhere>
ASSOROW RANGE <valor1> UNTIL <valor2>
GROUP BY <ListaAtributosTablaDatos>

```

```

<ListaAtributosTablaDatos> ::= <Atributo>, <ListaAtributos>
<ListaAtributos> ::= <Atributo>, <ListaAtributos>
<ListaAtributos> ::= <Atributo>
<valor1> ::= 1,2,3, 4 ...
<valor2> ::= 1,2,3,4 ...

```

La primitiva ASSOROW facilita el cálculo de los ítemsets frecuentes para el descubrimiento de Reglas de Asociación en tablas multicolumna especialmente en análisis de canasta de mercado [31].

**Ejemplo. Timarán [31].** Sea la tabla Transaccion (TID, ID\_ITEM1, ID\_ITEM2, ID\_ITEM3) de la Tabla 8. Calcular el soporte para las ítemsets de tamaño 2 y 3 formados por los atributos ID\_ITEM1, ID\_ITEM2, ID\_ITEM3 y almacenar los resultados en la tabla Assotran.

Tabla 8: Tabla Transacciones

| TID | ID_ITEM1 | ID_ITEM2 | ID_ITEM3 |
|-----|----------|----------|----------|
| 100 | i1       | i2       |          |
| 200 | i2       | i3       | i4       |
| 300 | i1       | i2       | i3       |
| 400 | i2       | i3       | i4       |
| 500 | i3       | i4       |          |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 22 [31]

La sentencia SQL que resuelve esta consulta es la siguiente [31]:

```

SELECT id_item1, id_item2, id_item3, count(*) AS soporte INTO assotran
FROM transaccion
ASSOROW RANGE 2 UNTIL 3
GROUP BY id_item1, id_item2, id_item3

```

Inicialmente se selecciona los atributos `id_item1`, `id_item2`, `id_item3` de la tabla transacción. Luego *ASSOROW* genera todos los ítemsets de tamaño 2 y 3 (ver Tabla 9) y finalmente se agrupa por los atributos especificados en la cláusula `GROUP BY`, se cuenta la frecuencia de los ítemsets y se genera la tabla *assotran* (ver Tabla 10) [31].

**Tabla 9: Assorow**

| ID_ITEM1 | ID_ITEM2 | ID_ITEM3 |
|----------|----------|----------|
| i1       | i2       | null     |
| i2       | i3       | null     |
| i2       | i4       | null     |
| i3       | i4       | null     |
| i2       | i3       | i4       |
| i1       | i2       | null     |
| i1       | i3       | null     |
| i2       | i3       | null     |
| i1       | i2       | i3       |
| i2       | i3       | null     |
| i2       | i4       | null     |
| i3       | i4       | null     |
| i2       | i3       | i4       |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 23 [31]

**Tabla 10: Assotran**

| ID_ITEM1 | ID_ITEM2 | ID_ITEM3 | SOPORTE |
|----------|----------|----------|---------|
| i1       | i2       |          | 2       |
| i1       | i2       | i3       | 1       |
| i1       | i3       |          | 1       |
| i2       | i3       |          | 4       |
| i2       | i3       | i4       | 2       |
| i2       | i4       |          | 2       |
| i3       | i4       |          | 3       |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 23 [31]

**Ejemplo. Timarán [31].** Obtener de la tabla Transacciones del ejemplo anterior, los ítemssets frecuentes de tamaño 2 y 3 que cumplan con un soporte mínimo mayor o igual a 2.

Los ítemssets frecuentes se obtienen con la siguiente sentencia SQL [31]:

```
SELECT id_item1, id_item2, id_item3, count(*) AS soporte INTO assotran
FROM transaccion
ASSOROW RANGE 2 UNTIL 3
GROUP BY id_item1, id_item2, id_item3 HAVING count(*)>=2
```

El resultado final se muestra en la Tabla 11.

Tabla 11: Ítemssets frecuentes tamaño 2 y 3 con soporte  $\geq 2$ .

| ID_ITEM1 | ID_ITEM2 | ID_ITEM3 | SOPORTE |
|----------|----------|----------|---------|
| i1       | i2       |          | 2       |
| i2       | i3       |          | 4       |
| i2       | i3       | i4       | 2       |
| i2       | i4       |          | 2       |
| i3       | i4       |          | 3       |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 23 [31]

**3.1.3 Operador EquiKeep y la primitiva EquiKeep On.** A continuación se describe el operador EquiKeep y la primitiva EquiKeep On.

### Operador EquiKeep ( $\chi$ ) [31]

*EquiKeep* ( $\chi$ ) es un operador unario, que se asemeja a la Selección o Restricción por tener una expresión lógica que evaluar sobre una relación  $R$ , y conserva su esquema. Se diferencia de la Restricción ( $\sigma$ ) en que en lugar de aplicar la condición a las filas (tuplas) de la relación, *EquiKeep* aplica la expresión lógica a las columnas (atributos) de  $R$ , es decir restringe los valores de los atributos de cada una de las tuplas de la relación  $R$ , a únicamente aquellos que satisfacen

una condición determinada, haciendo nulos al resto de valores y conserva el esquema de la relación [31].

Su sintaxis es la siguiente [31]:

$\chi_{\text{expresión\_lógica}}(R)$

donde *<expresión lógica>* es la condición que deben cumplir los valores de los atributos de la relación R para no hacerse nulos.

El operador *EquiKeep*, restringe los valores de los atributos de cada una de las tuplas de la relación R a únicamente los valores de los atributos que satisfacen una expresión lógica *<expresión lógica>*, la cual esta formada por un conjunto de cláusulas de la forma *Atributo=Valor*, conectivos lógicos AND, OR y NOT. En cada tupla, los valores de los atributos que no cumplen la condición *<expresión lógica>* se hacen nulos. *EquiKeep* elimina las tuplas vacías, es decir, aquellas tuplas en las cuales los valores de todos sus atributos son nulos [31].

*EquiKeep* optimiza el trabajo de los operadores *Associator* o *Assorow* en el cálculo de ítemsets frecuentes en la tarea de Asociación, al disminuir el número de combinaciones [31].

**Ejemplo. Timarán [31].** Sea la relación R(A,B,C,D) de la Tabla 12. Restringir los valores de los atributos A=a1, B=b1, C=c2 y D=d1, es decir, R1= A=a1 v B=b1 v C=c2 v D=d1 (R). El resultado del operador *EquiKeep* se muestra en la Tabla 13.

Tabla 12: Relación R

| A  | B  | C  | D  |
|----|----|----|----|
| a1 | b2 | c1 | d2 |
| a2 | b2 | c2 | d2 |
| a2 | b1 | c1 | d1 |
| a2 | b2 | c1 | d2 |
| a1 | b2 | c2 | d1 |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 10 [31]

Tabla 13: Resultado de la operación  $R1 = X \ A=a1 \vee B=b1 \vee C=c2 \vee D=d1 \ (R)$

| A    | B    | C    | D    |
|------|------|------|------|
| a1   | b1   | null | d1   |
| a1   | null | null | Null |
| Null | null | c2   | Null |
| Null | b1   | null | d1   |
| Null | null | null | Null |
| a1   | null | c2   | d1   |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 10 [31]

En este ejemplo, la tupla  $\{a2, b2, c1, d2\}$  es eliminada por resultar todos sus valores nulos.

### Primitiva *EquiKeep On* [31]

Esta primitiva implementa el operador algebraico *EquiKeep* en la cláusula SQL SELECT. *EquiKeep On* conserva en cada registro de una tabla los valores de los atributos que cumplen una condición determinada. El resto de valores de los atributos se hacen nulos [31].

Dentro de la cláusula SELECT, *EquiKeep On* tiene la siguiente sintaxis [31]:

```
SELECT <ListaAtributosTablaDatos> [INTO]
<NombreTablaEquiKeep>] FROM <NombreTablaDatos>
WHERE <CláusulaWhere>
EQUIKEEP ON <CondiciónValoresAtributos >
```

<ListaAtributosTablaDatos> ::= <Atributo>, <ListaAtributos>

<ListaAtributos> ::= <Atributo>, <ListaAtributos>

<ListaAtributos> ::= <Atributo>

<CondiciónValoresAtributos> ::= <Atributo=Valor><operador><ListaCondición>  
| <Atributo><operador><(Listavalores)>,  
<ListaCondición>

<ListaCondición> ::= <Atributo=Valor>| <Atributo><operador><( Listavalores)>

<operador> ::= AND, OR, NOT, IN

<Listavalores> ::= 1,2,3, 4...

<Valor> ::= 1,2,3, 4...

donde:

La cláusula SELECT <ListaAtributosTablaDatos> permite seleccionar los atributos de la tabla de datos <NombreTablaDatos> cuyos valores se conservarán si cumplen la condición <CondiciónValoresAtributos>, especificada en la primitiva *EquiKeep On* [31].

La cláusula INTO <NombreTablaEquiKeep> define el nombre de la tabla <NombreTablaEquiKeep> donde se almacenarán los resultados de EQUIKEEP ON con los atributos <ListaAtributosTablaDatos> como esquema. Si no se especifica la cláusula INTO, el resultado se almacenará en una tabla temporal [31].

La cláusula FROM <NombreTablaDatos> WHERE <CláusulaWhere> determina el nombre de la tabla de datos <NombreTablaDatos> con el conjunto de registros que cumplan las restricciones de la <CláusulaWhere> [31].

La primitiva EQUIKEEP ON <CondiciónValoresAtributos> conserva los valores de los atributos de <ListaAtributosTablaDatos> que cumplan la condición especificada en <CondiciónValoresAtributos>. El resto de valores de los atributos se hacen nulos [31].

La primitiva EQUIKEEP ON facilita la generación de los ítemsets frecuentes en el descubrimiento de Reglas de Asociación, al permitir conservar en cada registro de una tabla únicamente los valores de los atributos frecuentes o ítemsets frecuentes [31].

**Ejemplo Timarán [31].** Sea la tabla Transaccion (TID, ID\_ITEM1, ID\_ITEM2, ID\_ITEM3) de la Tabla 8. Encontrar los ítemsets frecuentes de tamaño 2 y 3 formados por los atributos ID\_ITEM1, ID\_ITEM2, ID\_ITEM3 que cumplan un soporte mínimo mayor o igual a 3. Almacenar los resultados en la tabla Equitran.

Suponiendo que ya se conocen los ítemsets frecuentes de tamaño 1: {i2:4}, {i3:4}, {i4:3}, la sentencia SQL que permite obtener esta consulta utilizando las primitivas *EquiKeep On* y *Assorow Range* es la siguiente [31]:

```

SELECT id_item1, id_item2, id_item3, count(*) AS soporte INTO equitran
FROM transaccion
EQUIKEEP ON id_item1 IN (i2,i3,i4) , id_item2 IN (i2,i3,i4), id_item3
IN(i2,i3,i4) ASSOROW RANGE 2 UNTIL 3
GROUP BY id_item1, id_item2, id_item3 HAVING count(*)>=3

```

En esta sentencia SQL, inicialmente se ejecuta la cláusula SELECT, en este caso se seleccionan los atributos *id\_item1*, *id\_item2*, *id\_item3* de la tabla *transacción*. Luego sobre la tabla resultante, se ejecuta la primitiva EQUIKEEP ON que conserva los valores de los atributos que cumplan la condición *id\_item1 IN(i2,i3,i4), id\_item2 IN(i2,i3,i4), id\_item3 IN(i2,i3,i4)* (Tabla 14) [31].

Tabla 14: Resultado primitiva EquiKeep On

| ID_ITEM1 | ID_ITEM2 | ID_ITEM3 |
|----------|----------|----------|
| Null     | i2       |          |
| i2       | i3       | i4       |
| Null     | i2       | i3       |
| i2       | i3       | i4       |
| i3       | i4       |          |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 28 [31]

Posteriormente se generan los ítems de tamaño 2 y 3 con la primitiva ASSOROW RANGE 2 UNTIL 3 (Tabla 15), se agrupa, se cuenta la frecuencia de los ítems (Tabla 16) y finalmente se hallan los ítems frecuentes cuyo soporte mínimo es mayor o igual a tres y se almacenan en la tabla *Equitran* (Tabla 17) [31].

Tabla 15: Resultado primitiva Assorow Range.

| ID_ITEM1 | ID_ITEM2 | ID_ITEM3 |
|----------|----------|----------|
| i2       | i3       | Null     |
| i2       | i4       | Null     |
| i3       | i4       | Null     |
| i2       | i3       | i4       |
| i2       | i3       | Null     |

| ID_ITEM1 | ID_ITEM2 | ID_ITEM3 |
|----------|----------|----------|
| i2       | i3       | Null     |
| i2       | i4       | Null     |
| i3       | i4       | Null     |
| i2       | i3       | i4       |
| i3       | i4       | Null     |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 28 [31]

Tabla 16: Soporte Ítemsets tamaño 2 y 3.

| ID_ITEM1 | ID_ITEM2 | ID_ITEM3 | SOPORTE |
|----------|----------|----------|---------|
| i2       | i3       |          | 4       |
| i2       | i3       | i4       | 2       |
| i2       | i4       |          | 2       |
| i3       | i4       |          | 3       |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 28 [31]

Tabla 17: Tabla Equitran.

| ID_ITEM1 | ID_ITEM2 | ID_ITEM3 | SOPORTE |
|----------|----------|----------|---------|
| i2       | i3       |          | 4       |
| i3       | i4       |          | 3       |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 28 [31]

**3.1.4 Operador Describe Associator y la primitiva Describe Association Rules.** A continuación se describe el Describe Associator y la primitiva Describe Association Rules.

### Operador Describe Associator ( $\beta\alpha$ ) [31]

*Describe Associator* es un operador unario que toma como entrada la relación

resultante de los operadores *Associator*, *Assorow* o *Assocol* y por cada tupla de esta relación, genera, a partir de los atributos  $l$  no nulos de la tupla, todos los diferentes subconjuntos de un tamaño específico de la forma  $\{\{a\}, \{l-a\}, s\}$ , donde  $\{a\}$  se denomina subconjunto antecedente y  $\{l-a\}$  subconjunto consecuente.  $\{a\}$  y  $\{l-a\}$ , son subconjuntos de atributos de  $l$  y  $s$  es el tamaño del subconjunto antecedente  $\{a\}$  [31].

La sintaxis del operador *Describe Associator* es la siguiente [31]:

$$\beta\alpha_{longitud\_regla}(R)$$

donde  $\langle longitud\_regla \rangle$  es la longitud máxima de atributos no nulos por tupla.

El operador *Describe Associator* toma cada tupla  $t$  de  $R$  y el parámetro  $\langle longitud\_regla \rangle$  como entrada, y retorna, por cada tupla  $t$ , los diferentes subconjuntos de atributos  $X_i$ , de tamaño  $\langle longitud\_regla \rangle$ , como tuplas en una nueva relación [31].

Cada tupla  $X_i$  se forma siguiendo la regla  $(\{a\} \Rightarrow \{l-a\}, s)$ , donde  $l$  es el conjunto de atributos no nulos de la tupla  $t$ ,  $\{a\}$  es un subconjunto de  $l$  denominado antecedente,  $\{l-a\}$  se denomina subconjunto consecuente y  $s$  es el tamaño del subconjunto  $\{a\}$  [31].

**Ejemplo. Timarán [31].** Sea la relación  $R(A,B,C)$  de la Tabla 18. Obtener los diferentes subconjuntos de tamaño 3 con el operador *Describe Associator*, es decir  $R1 = \beta\alpha_3(R)$ . El resultado se muestra en la Tabla 19.

Tabla 18: Relación R

| A  | B  | C  |
|----|----|----|
| a1 | b1 | c1 |
| a2 | b2 | c2 |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 12 [31]

Tabla 19: Relación  $R1 = \beta\alpha_3(R)$ .

| A1 | A2 | A3 | S |
|----|----|----|---|
| a1 | b1 | c1 | 1 |

| A1 | A2 | A3 | S |
|----|----|----|---|
| b1 | a1 | c1 | 1 |
| c1 | a1 | b1 | 1 |
| a1 | b1 | c1 | 2 |
| a1 | c1 | b1 | 2 |
| b1 | c1 | a1 | 2 |
| a2 | b2 | c2 | 1 |
| b2 | a2 | c2 | 1 |
| c2 | a2 | b2 | 1 |
| a2 | b2 | c2 | 2 |
| a2 | c2 | b2 | 2 |
| c2 | b2 | a2 | 2 |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 12 [31]

### Operador SQL Describe Association Rules [31]

El operador SQL Describe Association Rules implementa el operador algebraico Describe Associator en una nueva cláusula SQL. Describe Association Rules permite generar, una vez calculados los ítemsets frecuentes, todas las reglas de asociación unidimensionales o multidimensionales de una longitud específica, que cumplen con un soporte y una confianza mayor o igual que unos determinados umbrales especificados por el usuario [31].

Describe Association Rules tiene la siguiente sintaxis [31]:

```

RULES
DESCRIBE ASSOCIATION
[INTO <TablaReglasAsociación>]
FROM <NombreTablaÍtemsetsFrecuentes> WITH CONFIDENCE <valor1>
LENGTH <valor2 >
[DO <SubconsultaÍtemsetsFrecuentes>]
<valor 1> ::=1,2,3,4,...
<valor 2> ::=1,2,3,4,...

```

donde:

< SubconsultaÍtemsetsFrecuentes > ::= <SFWEAG> | <SFWEARG> | <SFWEACG>  
<SFWEAG> ::= <SELECT FROM WHERE EQUIKEEP ASSOCIATOR GROUP BY>  
<SFWEARG> ::= <SELECT FROM WHERE EQUIKEEP ASSOROW GROUP BY>  
<SFWEACG> ::= <SELECT FROM WHERE EQUIKEEP ASSOCOL GROUP BY>

La cláusula *DESCRIBE ASSOCIATION RULES* especifica el tipo de reglas de asociación a extraer.

La cláusula [INTO <TablaReglasAsociación>] permite almacenar en una tabla <TablaReglasAsociación> las reglas de asociación que se generen, para posteriores consultas.

La cláusula FROM <NombreTablaÍtemsetsFrecuentes> especifica el nombre de la tabla de datos <NombreTablaÍtemsetsFrecuentes> donde se encuentran los ítemsets frecuentes para la extracción de reglas.

La cláusula WITH CONFIDENCE <valor 1> especifica el valor de la confianza mínima <valor 1> que permite la generación de reglas de asociación fuertes.

La cláusula LENGTH <valor 2> determina la longitud de las reglas de asociación a generar.

La cláusula opcional [DO <SubconsultaCalculoÍtemsetsFrecuentes>] permite especificar conjuntamente con la cláusula DESCRIBE, la subconsulta <SubconsultaCalculoÍtemsetsFrecuentes> que halla los ítemsets frecuentes utilizando cualesquiera de las nuevas primitivas SQL para asociación. En el caso de que esta cláusula no se especifique, los procesos de cálculo de los ítemsets frecuentes y el de generación de reglas se realizarían en órdenes SQL independientes [31].

**3.1.5 EquipAsso: un Algoritmo para el Descubrimiento de Reglas de Asociación.** EquipAsso es un algoritmo que se basa en los nuevos operadores del álgebra relacional *Associator*, *Assorow* y *EquiKeep* para el cálculo de conjuntos de ítems frecuentes [31]. Este hecho facilita su integración al interior de cualquier SGBD, acoplando la tarea de Asociación de una manera fuerte y favorece la aplicación de técnicas de optimización de consultas para

mejorar su rendimiento. [31],

El algoritmo es el siguiente [31]:

El primer paso del algoritmo cuenta el número de ocurrencias de cada ítem para determinar los 1-conjuntos de ítems frecuentes L1. En el subsiguiente paso, se aplica el operador *EquipKeep* para extraer de todas las transacciones en D, los conjuntos de ítems frecuentes tamaño 1, haciendo nulos el resto de valores. Luego, a la relación resultante R, se aplica el operador *Associator* (para reglas de asociación multidimensionales) o el operador *Assorow* (para reglas de asociación unidimensionales) para generar todos los conjuntos de ítems tamaño 2 ( $l_s=2$ ) hasta máximo tamaño n, donde n es el grado de R. Finalmente, se calculan todos los conjuntos de ítems frecuentes L, contando el soporte de las diferentes combinaciones generadas por *Associator* (o *Assorow*) en la relación R1. En la Figura 19 se muestra el algoritmo *EquipAsso* [31].

Figura 19 Algoritmo *EquipAsso*

```
L1={1-conjuntos de ítems frecuentes};  
Forall transacciones  $t \in D$  do begin  
  // se aplica el operador EquipKeep  
   $R=\chi_{l,1}(D)$   
   $k=2$   
   $g=\text{grado}(R)$   
  //genera todos los conjuntos de ítems posibles  
   $R1=\alpha_{k,g}(R) = \{\cup_{\text{all}} X_i \mid X_i \subseteq t_i\}$   
End  
  // conjuntos de ítems frecuentes  
   $L = \{\text{count}(R1) \mid \text{count} \geq \text{minsup}\}$ 
```

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 43 [31]

El algoritmo *EquipAsso* se implementa en el lenguaje SQL, utilizando la nuevas primitivas SQL *EquipKeep On* conjuntamente con *Associator Range*, para reglas de asociación multidimensionales y *EquipKeep On* con *Assorow Range* para reglas de asociación unidimensionales. El resultado se muestra en al siguiente

sentencia [31].

```
SELECT <ListaAtributosTablaDatos> , count(*) AS soporte INTO R1  
FROM D  
EQUIKEEP ON L1  
ASSOCIATOR RANGE k UNTIL g  
GROUP BY <ListaAtributosTablaDatos> HAVING count(*)>=minsup
```

### **EquipAsso mejorado [31]**

Genera los ítemsets frecuentes tamaño 1 hasta el grado de la tabla si es posible sin el uso del operador *Equikeep*.

```
SELECT <ListaAtributosTablaDatos> , count(*) AS soporte INTO R1  
FROM D  
ASSOCIATOR RANGE k UNTIL g  
GROUP BY <ListaAtributosTablaDatos> HAVING count(*)>=minsup
```

## **3.2 OPERADORES DEL ÁLGEBRA RELACIONAL Y PRIMITIVAS SQL PARA CLASIFICACIÓN**

La clasificación por árboles de decisión es probablemente el modelo más utilizado y popular por su simplicidad y facilidad para su entendimiento.

Durante la etapa de construcción del árbol, en forma recursiva, cada conjunto de datos se particiona en subconjuntos de acuerdo a un criterio de particionamiento, con el fin de escoger el atributo que mejor separe los ejemplos restantes en clases individuales. Seleccionar el mejor punto de particionamiento es la parte de la construcción del árbol que mayor tiempo consume [31].

Se han propuesto varias métricas para este proceso. El cálculo del valor de la métrica que permite seleccionar, en cada nodo, el atributo que tenga una mayor potencia para clasificar sobre el conjunto de valores del atributo clase, es la

parte más costosa del algoritmo utilizado [WalS98]. Los algoritmos ID3 [40] y C4.5 [41] utilizan como métrica, para seleccionar el atributo candidato en cada nodo del árbol, la reducción de la entropía denominada *Ganancia de Información*, mientras que SLIQ [42] y SPRINT [31], usan el índice *Gini*. Para el cálculo de estas métricas, no se necesitan los datos en si, sino las estadísticas acerca del número de registros en los cuales se combinan los atributos condición con el atributo clase.

Un nuevo operador algebraico para clasificación basado en árboles de decisión debe facilitar estas combinaciones, que conjuntamente con una función agregada, permita el cálculo de estas métricas[31].

El nuevo operador algebraico que realiza las diferentes combinaciones de los atributos condición con el atributo clase y las funciones agregadas específicas que facilitan el cálculo de la Entropía y Ganancia de Información (de igual manera se puede definir las funciones agregadas para la métrica *índice Gini*) son [31]:

**3.2.1 Operador Mate y la primitiva Mate By.** A continuación se describe el Operador Mate y la primitiva Mate By.

### **Operador Mate ( $\mu$ ) [31]**

El operador *Mate* genera, por cada una de las tuplas de una relación, todas los posibles combinaciones formadas por los valores no nulos de los atributos pertenecientes a una lista de atributos denominados *Atributos Condición*, y el valor no nulo del atributo denominado *Atributo Clase* [31].

Tiene la siguiente sintaxis [31]:

$\mu_{\langle lista\_atributos\_condición; atributo\_clase \rangle}(R)$

donde  $\langle lista\_atributos\_condición \rangle$  es el conjunto de atributos de la relación  $R$  a combinar con el atributo clase y  $\langle atributo\_clase \rangle$  es el atributo de  $R$  definido como clase.

*Mate* toma como entrada cada tupla de  $R$  y produce una nueva relación cuyo esquema esta formado por los atributos condición,  $\langle lista\_atributos\_condición \rangle$  y el atributo clase,  $\langle atributo\_clase \rangle$  con tuplas formadas por todas las posibles combinaciones de cada uno de los atributos condición con el atributo clase, los demás valores de los atributos se hacen nulos [31].

*Mate* empareja en cada partición todos los atributos condición con el atributo clase, lo que facilita el conteo y el posterior cálculo de las medidas de entropía y ganancia de información. El operador *Mate* genera estas combinaciones, en una sola pasada sobre la tabla de entrenamiento (lo que redunda en la eficiencia del proceso de construcción del árbol de decisión) [31].

**Ejemplo. Timarán [31].** Sea la relación  $R(A,B,C,D)$  de la Tabla 20 obtener las diferentes combinaciones de los atributos  $A,B$  con el atributo  $D$ , es decir,  $R1 = \mu_{A,B;D}(R)$ .

El resultado de la operación  $R1 = \mu_{A,B;D}(R)$ , se muestra en la Tabla 21.

Tabla 20: Relación R

| A  | B  | C  | D  |
|----|----|----|----|
| a1 | b1 | c1 | d1 |
| a1 | b2 | c1 | d2 |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 14 [31]

Tabla 21: Resultado operación  $R1 = \mu_{A,B;D}(R)$

| A    | B    | D  |
|------|------|----|
| a1   | null | d1 |
| null | b1   | d1 |
| a1   | b1   | d1 |
| a1   | null | d2 |
| null | b2   | d2 |
| a1   | b2   | d2 |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 15 [31]

### Primitiva Mate By [31]

Esta primitiva implementa el operador algebraico *Mate* en la cláusula SQL SELECT. *Mate By* toma los valores de los atributos de una tabla denominados *atributos condición* y por cada registro forma todas las posibles combinaciones de estos atributos con otro atributo de la misma tabla denominado *atributo clase* [31].

Este proceso lo realiza en una sola pasada sobre la tabla [31].

Dentro de la cláusula SELECT, *Mate By* tiene la siguiente sintaxis [31]:

```
SELECT <ListaAtributosTablaDatos> [INTO
<NombreTablaMate>] FROM <NombreTablaDatos>
WHERE <CláusulaWhere>
MATE BY<ListaAtributosCondicion> WITH
<AtributoClase> GROUP BY <ListaAtributosTablaDatos>
```

Donde

```
<ListaAtributosTablaDatos> ::= <Atributo>, <ListaAtributos>
<ListaAtributos> ::= <Atributo>, <ListaAtributos>
<ListaAtributos> ::= <Atributo>
<ListaAtributosCondicion> ::= <Atributo>, <ListaAtributos>
<AtributoClase> ::= <Atributo>
```

La cláusula SELECT <ListaAtributosTablaDatos> permite seleccionar de la tabla <NombreTablaDatos> tanto el conjunto de atributos que serán combinados <ListaAtributosCondicion> como el atributo con el cual se combinará el <AtributoClase>.

La cláusula INTO <NombreTablaMate> define el nombre de la tabla <NombreTablaMate> donde se almacenarán los resultados de la primitiva MATE cuyo esquema estará determinado por los atributos de <ListaAtributosTablaDatos>. Si no se especifica la cláusula INTO, el resultado se almacenará en una tabla temporal [31].

La cláusula FROM <NombreTablaDatos> WHERE<CláusulaWhere> determina el nombre de la tabla de datos de entrada <NombreTablaDatos> con las restricciones <CláusulaWhere> que los registros deben cumplir [31].

La cláusula MATE BY <ListaAtributosCondicion> WITH <AtributoClase> determina el conjunto de atributos <ListaAtributosCondicion> con los cuales el atributo clase <AtributoClase> se combinará [31].

La cláusula GROUP BY <ListaAtributosTablaDatos> agrupa la tabla de entrada <NombreTablaDatos> por los atributos <ListaAtributosTablaDatos>

La primitiva MATE BY facilita la tarea de clasificación y la construcción de un árbol de decisión, al calcular conjuntamente con las funciones agregadas *Gain()* y *Entro()*, en cada partición y para cada atributo, la ganancia de información y la entropía, respectivamente [31].

**Ejemplo. Timarón [31].** Sea la tabla Síntomas (SID, D\_MUSCULAR, TEMPERATURA, GRIPA) de la Tabla 22. Obtener las ocurrencias de las diferentes combinaciones de los atributos d\_muscular, temperatura con el atributo gripa y almacenar el resultado en la tabla clasesintomas. La orden SQL que permite obtener esta consulta es la siguiente [31]:

```
SELECT d_muscular,tempeartura,gripa, count(*) INTO clasesintomas
FROM sintomas
MATE BY d_muscular,tempertura WITH gripa
GROUP BY d_muscular,tempertura,gripa
```

**Tabla 22: Tabla Síntomas.**

| SID | D_MUSCULAR | TEMPERATURA | GRIPA |
|-----|------------|-------------|-------|
| 1   | Si         | Alta        | Si    |
| 2   | No         | Alta        | Si    |
| 3   | Si         | Media       | No    |
| 4   | No         | Media       | Si    |
| 5   | Si         | Normal      | Si    |
| 6   | No         | Normal      | No    |

Fuente Timarón, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 30 [31]

Una vez seleccionados los atributos especificados en la cláusula SELECT, la primitiva MATE BY forma todas las posibles combinaciones con los atributos *d\_muscular*, *temperatura* y el atributo clase *gripa* (Tabla 23). El resultado se agrupa con la cláusula GROUP BY, se cuenta la frecuencia de las parejas y finalmente se almacena en la tabla *clasesintomas* (Tabla 24) [31].

Tabla 23: Resultado primitiva Mate

| D_MUSCULAR | TEMPERATURA | GRIPA |
|------------|-------------|-------|
| Si         | Null        | si    |
| Null       | alta        | si    |
| Si         | Alta        | si    |
| No         | Null        | si    |
| Null       | Alta        | si    |
| No         | Alta        | si    |
| Si         | null        | no    |
| Null       | media       | no    |
| Si         | media       | no    |
| No         | null        | si    |
| Null       | media       | si    |
| No         | media       | si    |
| Si         | Null        | si    |
| Null       | Normal      | si    |
| Si         | Normal      | si    |
| No         | Null        | no    |
| Null       | Normal      | no    |
| No         | Normal      | no    |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 30 [31]

Tabla 24: Clasesintomas

| D_MUSCULAR | TEMPERATURA | GRIPA | COUNT |
|------------|-------------|-------|-------|
| si         | null        | si    | 2     |
| si         | null        | no    | 1     |
| si         | normal      | si    | 1     |
| si         | media       | no    | 1     |
| si         | alta        | si    | 1     |
| no         | null        | si    | 2     |
| no         | null        | no    | 1     |

| D_MUSCULAR | TEMPERATURA | GRIPA | COUNT |
|------------|-------------|-------|-------|
| no         | normal      | no    | 1     |
| no         | media       | si    | 1     |
| no         | alta        | si    | 1     |
| null       | normal      | si    | 1     |
| null       | normal      | no    | 1     |
| null       | media       | si    | 1     |
| null       | media       | no    | 1     |
| null       | alta        | si    | 2     |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 30 [31]

**3.2.2 Función algebraica agregada Entro y Función agregada SQL Entro.** A continuación se describe la función algebraica agregada Entro y función agregada SQL Entro.

### Función algebraica agregada Entro() [31]

La función *Entro()* permite calcular la entropía de una relación *R* con respecto a un atributo denominado atributo condición y un atributo clase [31].

Tiene la siguiente sintaxis [31]:

*Entro(atributo; atributo\_clase;R)*

donde *<atributo>* es el atributo condición de la relación *R* y *<atributo\_clase>* es el atributo con el que se combina el atributo *<atributo>* [31].

La función *Entro(A<sub>k</sub>; A<sub>c</sub>; R)*, retorna la entropía de *R* con respecto al atributo *A<sub>k</sub>*, que se obtiene de la siguiente manera :

$$Entro(A_k; A_c; R) = \{y \mid y = - \sum p_{ij} \log_2(p_{ij}), i=1..t, j=1..q, p_{ij} = s_{ij} / |S_j| \}$$

donde  $p_{ij} = s_{ij} / |S_j|$  es la probabilidad que una tupla en *S<sub>j</sub>* pertenezca a la clase *C<sub>j</sub>*. La entropía de *R* con respecto al atributo clase *A<sub>c</sub>* es:

$$Entro(Ac;Ac; R)=\{y \mid y = - \sum p_i \log_2(p_i), i=1 ..t, p_i = r_i / m\}$$

donde  $p_i$  es la probabilidad que un tupla cualquier pertenezca a la clase  $C_i$  y  $r_i$  el número de tuplas de  $r(A)$  que pertenecen a la clase  $C_i$  [31].

### **Función agregada SQL Entro() [31]**

Esta función agregada SQL implementa la función algebraica agregada *Entro()* en la cláusula SQL SELECT SQL *Entro()* permite calcular, conjuntamente con la primitiva *Mate By* la entropía de una tabla con respecto a cada una de las combinaciones de los *atributos condición* con el *atributo clase*. SQL *Entro()* se debe ejecutar conjuntamente con la función agregada *Count()* [31].

Dentro de la cláusula SELECT, SQL *Entro()* tiene la siguiente sintaxis [31]:

```
SELECT <ListaAtributosTablaDatos>, Count(*), Entro(*)
[INTO <NombreTablaMate>]
FROM <NombreTablaDatos>
WHERE <CláusulaWhere>
MATE BY<ListaAtributosCondicion> WITH <AtributoClase>
GROUP BY <ListaAtributosTablaDatos>
```

**Ejemplo. Timarán [31].** Sea la tabla Sintomas de la Tabla 22, calcular la entropía de las diferentes combinaciones de los atributos *d\_muscular*, *temperatura* con el atributo *gripa* y almacenar el resultado en la tabla *entosintomas*.

Esta consulta se obtiene mediante la siguiente sentencia SQL [31]:

```
SELECT d_muscular,temperatura,gripa, count(*), Entro(*) INTO
entosintomas
FROM sintomas
MATE BY d_muscular,tempertura WITH gripa
GROUP BY d_muscular,tempertura,gripa
```

Una vez se obtiene el resultado de ejecutar la primitiva MATE BY, el agrupamiento y el count(\*) (Tabla 24), se ejecuta la función *Entro()*, la cual,

inicialmente, por cada valor igual, no nulo, de cada atributo condicional o combinación de atributos condicionales, calcula su parte de la entropía, teniendo en cuenta los diferentes valores del atributo clase (Tabla 25). Posteriormente suma estos valores parte, obtiene la entropía y genera la tabla *entrosintomas* concatenando los diferentes valores del atributo clase en un solo valor, para cada atributo condicional o combinación de atributos condicionales, con respecto a los cuales, se calcula la entropía (Tabla 26) [31].

Tabla 25: Resultado Parcial función Entro()

| D_MUSCULAR | TEMPERATURA | GRIPA | COUNT() | ENTRO()          |
|------------|-------------|-------|---------|------------------|
| Si         | Null        | Si    | 2       | $2/3\log_2(2/3)$ |
| Si         | Null        | No    | 1       | $1/3\log_2(1/3)$ |
| Si         | Normal      | Si    | 1       | $1/1\log_2(1/1)$ |
| Si         | Media       | No    | 1       | $1/1\log_2(1/1)$ |
| Si         | Alta        | Si    | 1       | $1/1\log_2(1/1)$ |
| No         | Null        | Si    | 2       | $2/3\log_2(2/3)$ |
| No         | Null        | No    | 1       | $1/3\log_2(1/3)$ |
| No         | Normal      | No    | 1       | $1/1\log_2(1/1)$ |
| No         | Media       | Si    | 1       | $1/1\log_2(1/1)$ |
| No         | Alta        | Si    | 1       | $1/1\log_2(1/1)$ |
| Null       | normal      | Si    | 1       | $1/2\log_2(1/2)$ |
| Null       | normal      | No    | 1       | $1/2\log_2(1/1)$ |
| Null       | media       | Si    | 1       | $1/2\log_2(1/2)$ |
| Null       | media       | No    | 1       | $1/2\log_2(1/2)$ |
| Null       | alta        | Si    | 2       | $2/2\log_2(2/2)$ |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 31 [31]

Tabla 26: Entrosintomas

| D MUSCULAR | TEMPERATURA | GRIPA | COUNT() | ENTRO()                         |
|------------|-------------|-------|---------|---------------------------------|
| Si         | Null        | sino  | 3       | $2/3\log_2(2/3)+1/3\log_2(1/3)$ |
| Si         | Normal      | Si    | 1       | $1/1\log_2(1/1)$                |
| Si         | Media       | No    | 1       | $1/1\log_2(1/1)$                |
| Si         | Alta        | Si    | 1       | $1/1\log_2(1/1)$                |
| No         | Null        | sino  | 3       | $2/3\log_2(2/3)+1/3\log_2(1/3)$ |
| No         | Normal      | No    | 1       | $1/1\log_2(1/1)$                |
| No         | Media       | Si    | 1       | $1/1\log_2(1/1)$                |
| No         | Alta        | Si    | 1       | $1/1\log_2(1/1)$                |
| Null       | normal      | sino  | 2       | $1/2\log_2(1/2)+1/2\log_2(1/2)$ |

| D MUSCULAR | TEMPERATURA | GRIPA | COUNT() | ENTRO()                         |
|------------|-------------|-------|---------|---------------------------------|
| Null       | media       | sino  | 2       | $1/2\log_2(1/2)+1/2\log_2(1/2)$ |
| Null       | alta        | Si    | 2       | $2/2\log_2(2/2)$                |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 32 [31]

**3.2.3 Función algebraica agregada Gain y Función agregada SQL Gain.** A continuación se describe la función algebraica agregada Gain y función agregada SQL Gain.

### Función algebraica agregada *Gain()* [31]

La función *Gain()* permite calcular la reducción de la entropía causada por el conocimiento del valor de un atributo de una relación [31].

Su sintaxis es la siguiente[31]:

*Gain(atributo;atrib\_clase;R)*

donde *<atributo>* es el atributo condición de la relación *R* y *<atributo\_clase>* es el atributo con el que se combina el atributo *<atributo>*.

La función *Gain()* permite calcular la ganancia de información obtenida por el particionamiento de la relación *R* de acuerdo con el atributo *<atributo>* y se define [31]:

$$Gain(A_k; A_c; R) = \{y \mid y = Entro(A_c; A_c; R) - Entro(A_k; A_c; R)\}$$

donde *Entro (A<sub>c</sub>; A<sub>c</sub>; R)* es la entropía de la relación *R* con respecto al atributo clase *A<sub>c</sub>* y *Entro(A<sub>k</sub>;A<sub>c</sub>; R)* es la entropía de la relación *R* con respecto al atributo *A<sub>k</sub>* .

## Función agregada SQL Gain() [31]

Esta función agregada SQL implementa la función algebraica Gain() en la cláusula SQL SELECT. SQL Gain() permite calcular, conjuntamente con la primitiva Mate by, la ganancia de información de una tabla con respecto a cada una de las combinaciones de los atributos condición con el atributo clase. Internamente SQL Gain() calcula la entropía del atributo clase, por ello se debe ejecutar conjuntamente con las funciones agregadas Count() y Entro() [31].

### Sintaxis:

```
SELECT <ListaAtributosTablaDatos>, Count (*), Entro (*), Gain (*)  
[INTO <NombreTablaMate>]  
FROM <NombreTablaDatos>  
WHERE <CláusulaWhere>  
MATE BY <ListaAtributosCondicion> WITH <AtributoClase>  
GROUP BY <ListaAtributosTablaDatos>
```

**Ejemplo. Timarán [31].** Sea la tabla Sintomas de la Tabla 22. Calcular la ganancia de información de las diferentes combinaciones de los atributos d\_muscular, temperatura con el atributo gripa y almacenar el resultado en la tabla GanSintomas.

Esta consulta se obtiene mediante la siguiente sentencia SQL [31]:

```
SELECT d_muscular,temperatura,gripa, count(*), Entro(*), Gain(*)  
INTO gansintomas  
FROM sintomas  
MATE BY d_muscular,tempertura WITH gripa  
GROUP BY d_muscular,tempertura,gripa
```

El proceso para el cálculo de la ganancia es parecido a la de entropía. Una vez calculada la entropía (Tabla 26) se ejecuta la función Gain(), la cual, inicialmente, concatena los diferentes valores de los atributos condición y clase en un solo valor, así mismo suma las ocurrencias del resultado de la función count() y los valores de la función entro() [31].

Posteriormente se calcula la entropía del conjunto clase y la de los posibles nodos y finalmente, con estos datos, calcula la ganancia de información y la almacena en la tabla gansintomas (Tabla 27).

Tabla 27: GanSintomas

| D_MUSCULAR | TEMPERATURA          | GRIPA              | COUNT() | ENTRO()  | GAIN()  |
|------------|----------------------|--------------------|---------|--|---|
| Sino       | Null                 | sinosino           | 6       | $(\frac{2}{3}\log_2(\frac{2}{3}) + \frac{1}{3}\log_2(\frac{1}{3})) + (\frac{2}{3}\log_2(\frac{2}{3}) + \frac{1}{3}\log_2(\frac{1}{3}))$                              | $(\frac{4}{6}\log_2(\frac{4}{6}) + \frac{2}{6}\log_2(\frac{2}{6})) - \frac{3}{6}(\frac{2}{3}\log_2(\frac{2}{3}) + \frac{1}{3}\log_2(\frac{1}{3})) - \frac{3}{6}(\frac{2}{3}\log_2(\frac{2}{3}) + \frac{1}{3}\log_2(\frac{1}{3}))$ |
| Sino       | Normal               | sino               | 2       | $\frac{1}{1}\log_2(\frac{1}{1}) + \frac{1}{1}\log_2(\frac{1}{1})$  | $(\frac{1}{2}\log_2(\frac{1}{2}) + \frac{1}{2}\log_2(\frac{1}{2})) - \frac{1}{2}(\frac{1}{1}\log_2(\frac{1}{1})) - \frac{1}{2}(\frac{1}{1}\log_2(\frac{1}{1}))$   |
| Sino       | Media                | nosi               | 2       | $\frac{1}{1}\log_2(\frac{1}{1}) + \frac{1}{1}\log_2(\frac{1}{1})$  | $(\frac{1}{2}\log_2(\frac{1}{2}) + \frac{1}{2}\log_2(\frac{1}{2})) - \frac{1}{2}(\frac{1}{1}\log_2(\frac{1}{1})) - \frac{1}{2}(\frac{1}{1}\log_2(\frac{1}{1}))$   |
| Sino       | Alta                 | Sisi               | 2       | $\frac{1}{1}\log_2(\frac{1}{1}) + \frac{1}{1}\log_2(\frac{1}{1})$  | $\frac{2}{2}\log_2(\frac{2}{2}) - \frac{1}{2}(\frac{1}{1}\log_2(\frac{1}{1})) - \frac{1}{2}(\frac{1}{1}\log_2(\frac{1}{1}))$  |
| null       | normal<br>media alta | sino<br>sino<br>Si | 6       | $\frac{1}{2}\log_2(\frac{1}{2}) + \frac{1}{2}\log_2(\frac{1}{2}) + \frac{1}{2}\log_2(\frac{1}{2}) + \frac{1}{2}\log_2(\frac{1}{2}) + \frac{2}{2}\log_2(\frac{2}{2})$ | $(\frac{4}{6}\log_2(\frac{4}{6}) + \frac{2}{6}\log_2(\frac{2}{6})) - \frac{2}{6}(\frac{1}{2}\log_2(\frac{1}{2}) + \frac{1}{2}\log_2(\frac{1}{2})) - \frac{2}{6}(\frac{1}{2}\log_2(\frac{1}{2}) + \frac{1}{2}\log_2(\frac{1}{2}))$ |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 33 [31]

### 3.2.4 Operador Describe Classifier y SQL Describe Classification Rules. A continuación se describe Operador Describe Classifier y SQL Describe Classification Rules.

#### Operador Describe Classifier ( $\beta\mu$ ) [31]

Describe Classifier ( $\beta\mu$ ) es un operador unario que toma como entrada la relación

resultante de los operadores *Mate By*, *Entro()* y *Gain()* y produce una nueva relación donde se almacenan los valores de los atributos que formarán los diferentes nodos del árbol de decisión [31].

La sintaxis del operador Describe Classifier es la siguiente [31]:

$\beta\mu(R)$

Formalmente, sea  $A=\{A1, \dots, An, E, G\}$  el conjunto de atributos de la relación  $R$  de grado  $n+2$  y cardinalidad  $m$ . El operador  $\beta\mu$  aplicado a  $R$ :

$\beta\mu(R) = \{ t_i(Y) \mid Y=\{N, P, A, V, C\}, \text{ si } t_i=\langle \text{val}(N), \text{null}, \text{val}(A), \text{null}, \text{null} \rangle \in \text{raiz},$

$\text{si } t_i=\langle \text{val}(N), \text{val}(P), \text{val}(A), \text{val}(V), \text{val}(C) \rangle \in \text{hoja}, \text{ si}$

$t_i=\langle \text{val}(N), \text{val}(P), \text{val}(A), \text{val}(V), \text{null} \rangle \in \text{nodo interno} \}$

produce una nueva relación con esquema  $R(Y)$ ,  $Y=\{N, P, A, V, C\}$  donde  $N$  es el atributo que identifica el número de nodo,  $P$  identifica el nodo padre,  $A$  identifica el nombre del atributo asociado a ese nodo,  $V$  es el valor del atributo  $A$  y  $C$  es el atributo clase. Su extensión  $r(Y)$ , está formada por un conjunto de tuplas en las cuales si los valores de los atributos son:

$N\langle \rangle \text{null}$ ,  $P=\text{null}$ ,  $A\langle \rangle \text{null}$ ,  $V=\text{null}$  y  $C=\text{null}$  corresponde a un nodo raiz; si  $N\langle \rangle \text{null}$ ,  $P\langle \rangle \text{null}$ ,  $A\langle \rangle \text{null}$ ,  $V\langle \rangle \text{null}$  y  $C\langle \rangle \text{null}$  corresponde a una hoja o nodo terminal y si  $N\langle \rangle \text{null}$ ,  $P\langle \rangle \text{null}$ ,  $A\langle \rangle \text{null}$ ,  $V\langle \rangle \text{null}$  y  $C=\text{null}$  corresponde a un nodo interno.

Describe Classifier facilita la construcción del árbol de decisión y por consiguiente la generación de reglas de clasificación [31].

### **Operador SQL Describe Classification Rules [31]**

El operador SQL *Describe Classification Rules* implementa el operador algebraico *Describe Classifier* en una nueva cláusula SQL. Este operador construye el árbol de decisión y genera las reglas de clasificación. Permite la construcción del árbol de decisión de manera unificada con el cálculo de la métrica de particionamiento con la primitiva *Mate By* y las funciones agregadas *Entro()* y *Gain()* en una sola

instrucción SQL o en forma separada con sentencias SQL independientes, para luego generar las reglas [31].

Sintaxis:

```
DESCRIBE CLASSIFICATION RULES
[INTO <TablaReglasClasificación>] FROM <NombreTablaArbol>
USING <NombreTablaMétrica>
[DO <SubconsultaCálculoMétrica>]

<SubconsultaCálculoMétrica> ::= <SFWMG>
<SFWMG> ::= <SELECT FROM WHERE MATE BY GROUP BY>
```

Donde:

La cláusula *DESCRIBE CLASSIFICATION RULES* especifica la obtención de reglas de clasificación.

La cláusula opcional *[INTO <TablaReglasClasificación>]* permite almacenar en una tabla *<TablaReglasClasificación>* las reglas de clasificación que se generen, a partir de la tabla de nodos del árbol *<NombreTablaArbol>*, para posteriores consultas [31].

La cláusula *FROM <NombreTablaArbol>* especifica la tabla *<NombreTablaArbol>* donde se encuentran los datos necesarios para la construcción del árbol de decisión, que servirá de base para la generación de reglas de clasificación. La tabla *<NombreTablaArbol>*, la construye el operador *Describe*, en tiempo de ejecución, utilizando la tabla de métricas *<NombreTablaMétrica>*.

La cláusula *USING <NombreTablaMétrica>* determina la tabla *<NombreTablaMétrica>* donde se encuentran los valores de los atributos y las métricas utilizadas para construir el árbol [31].

La cláusula opcional *[DO <SubconsultaCálculoMétrica>]* permite especificar conjuntamente con la cláusula *DESCRIBE*, la subconsulta *<SubconsultaCálculoMétrica>* que permite calcular, para los atributos condición y atributo clase, las métricas Entropía y Ganancia de Información, utilizando las nuevas funciones agregadas *Entro()* y *Gain()* junto con la primitiva *MATE BY*. En

el caso de que esta cláusula no se especifique, los procesos de cálculo de la métrica y el de construcción y generación de reglas se realizarían de manera independiente. Primero se calculará la métrica de particionamiento en cada nodo del árbol mediante una sentencia <SELECT FROM WHERE MATE BY GROUP BY> y posteriormente se construirá el árbol y se generarán las correspondientes reglas de Clasificación [31]. Los siguientes ejemplos ilustran estos procesos:

**Ejemplo. Timarán [31].** Sea la tabla Síntomas (Tabla 22), encontrar las reglas de clasificación y almacenarlas en la tabla *ReglasClasificación*.

La sentencia SQL unificada que genera las reglas de clasificación es [31]:

```
DESCRIBE CLASSIFICATION RULES
INTO reglasclasificación
FROM nodosarbol
USING métricaspartición
DO
SELECT sid,d_muscular,temperatura,gripa, count(*), Entro(*), Gain(*)
INTO gansintomas
FROM sintomas
MATE BY sid,d_muscular,tempertura WITH gripa
GROUP BY d_muscular,tempertura,gripa
```

De igual manera se procede con sentencias SQL independientes [31]:

```
SELECT sid,d_cabeza,d_muscular,temperatura,gripa, count(*), Entro(*),
Gain(*)
INTO gansintomas
FROM sintomas
MATE BY sid,d.cabeza,d_muscular,tempertura WITH gripa
GROUP BY d_cabeza,d_muscular,tempertura,gripa
DESCRIBE CLASSIFICATION RULES
INTO reglasclasificación
FROM nodosarbol
USING métricaspartición
```

En este ejemplo a partir de la tabla *métricaspartición*, el operador Describe Classification Rules construye la tabla *nodosarbol* y con ella genera las reglas, almacenándolas en la tabla *reglasclasificación* [31].

### 3.2.5 Mate-Tree: un Algoritmo de Clasificación por Árboles de Decisión.

*Mate-Tree* es un algoritmo que se basa en los operadores del álgebra relacional *Mate*, *Describe Classifier*, y las funciones agregadas *Entro()* y *Gain()* para la generación de reglas de clasificación. Este hecho facilita su integración al interior de cualquier SGBD, acoplando la tarea de Clasificación de una manera fuerte y favorece la aplicación de técnicas de optimización de consultas para mejorar su rendimiento [31].

El algoritmo se muestra en la Figura 20 [31].

Figura 20: Algoritmo Mate-Tree

```

//Calcula entropía atributo clase Ac
Entro(Ac:Ac, R);
Forall t | R do begin
// Se aplica el operador Mate
    R1=m(LC,Ac)(R)
// cuenta las ocurrencias de las diferentes combinaciones
// de los atributos condición LC y atributo clase Ac
    Count(R1)
//calcula entropía
    Entro(LC,R1)
//calcula ganancia
    R2=Gain(LC,R1)
End
//construye el árbol de decisión con operador Describe Classifier
Arbol=bm(R2)

```

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 44 [31]

En el primer paso el algoritmo calcula la Entropía de la relación R con respecto al atributo clase. En el subsiguiente paso, se aplica el operador *Mate* con el fin de generar todas las posibles combinaciones de los atributos condición con el atributo clase. Se calcula el número de ocurrencias de cada combinación y a la relación resultante R1, se aplica la función agregada *Entro()* que calcula para cada combinación de atributos condición y clase la entropía de la relación R1 con respecto a estos atributos. Posteriormente, se calcula con la función agregada *Gain()*, la Ganancia de Información obtenida por el particionamiento de la relación R1 por las diferentes combinaciones de atributos condición y clase. Finalmente,

con la relación resultante R2, el operador *describe classifier* construye el árbol de decisión [31].

**Ejemplo. Timarán [31].** Sea un conjunto de datos de entrenamiento, el cual es representado en la Tabla 28.

Tabla 28: Datos de entrenamiento para Mate-Tree

| SID | D_MUSCULAR | TEMPERATURA | GRIPA |
|-----|------------|-------------|-------|
| 1   | Si         | Alta        | Si    |
| 2   | No         | Alta        | Si    |
| 3   | Si         | Media       | No    |
| 4   | No         | Media       | Si    |
| 5   | Si         | Normal      | Si    |
| 6   | No         | Normal      | No    |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 45 [31]

Si se aplica a esta tabla la primitiva *MATE BY*, se forman todas las posibles combinaciones con los atributos D\_MUSCULAR, TEMPERATURA y el atributo clase GRIPA (Tabla 29). Se cuenta la frecuencia de las parejas y finalmente se almacena en una nueva tabla (Tabla 30) [31].

Tabla 29: Resultado Primitiva Mate

| D_MUSCULAR | TEMPERATURA | GRIPA |
|------------|-------------|-------|
| Si         | Null        | Si    |
| Null       | Alta        | Si    |
| Si         | Alta        | Si    |
| No         | Null        | Si    |
| Null       | Alta        | Si    |
| No         | Alta        | Si    |
| Si         | Null        | No    |
| Null       | Media       | No    |
| Si         | Media       | No    |
| No         | Null        | Si    |
| Null       | Media       | Si    |
| No         | Media       | Si    |
| Si         | Null        | Si    |
| Null       | Normal      | Si    |
| Si         | Normal      | Si    |
| No         | Null        | No    |
| Null       | Normal      | No    |
| No         | Normal      | No    |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 45 [31]

Tabla 30: Clase

| D_MUSCULAR | TEMPERATURA | GRIPA | COUNT |
|------------|-------------|-------|-------|
| Si         | Null        | Si    | 2     |
| Si         | Null        | No    | 1     |
| Si         | Normal      | Si    | 1     |
| Si         | Media       | No    | 1     |
| Si         | Alta        | Si    | 1     |
| No         | Null        | Si    | 2     |
| No         | Null        | No    | 1     |
| No         | Normal      | No    | 1     |
| No         | Media       | Si    | 1     |
| No         | Alta        | Si    | 1     |
| Null       | Normal      | Si    | 1     |
| Null       | Normal      | No    | 1     |
| Null       | Media       | Si    | 1     |
| Null       | Media       | No    | 1     |
| Null       | Alta        | Si    | 2     |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 46 [31]

A la Tabla 30 se le aplica la función agregada *Entro()* que calcula la entropía de una tabla con respecto a cada una de las combinaciones de los atributos condición con el atributo clase [31]. Se obtiene la tabla de entropía (ver Tabla 31,Tabla 32).

Tabla 31: Resultdo parcial función Entro()

| D MUSCULAR | TEMPERATURA | GRIPA | COUNT | ENTRO()           |
|------------|-------------|-------|-------|-------------------|
| Si         | Null        | Si    | 2     | $2/3 \log_2(2/3)$ |
| Si         | Null        | No    | 1     | $1/3 \log_2(1/3)$ |
| Si         | Normal      | Si    | 1     | $1/1 \log_2(1/1)$ |
| Si         | Media       | No    | 1     | $1/1 \log_2(1/1)$ |
| Si         | Alta        | Si    | 1     | $1/1 \log_2(1/1)$ |
| No         | Null        | Si    | 2     | $2/3 \log_2(2/3)$ |
| No         | Null        | No    | 1     | $1/3 \log_2(1/3)$ |
| No         | Normal      | No    | 1     | $1/1 \log_2(1/1)$ |
| No         | Media       | Si    | 1     | $1/1 \log_2(1/1)$ |
| No         | Alta        | Si    | 1     | $1/1 \log_2(1/1)$ |
| Null       | Normal      | Si    | 1     | $1/2 \log_2(1/2)$ |
| Null       | Normal      | No    | 1     | $1/2 \log_2(1/2)$ |
| Null       | Media       | Si    | 1     | $1/2 \log_2(1/2)$ |
| Null       | Media       | No    | 1     | $1/2 \log_2(1/2)$ |
| Null       | Alta        | Si    | 2     | $2/2 \log_2(2/2)$ |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 33 [31]

Tabla 32: Resultado Entropia.

| D MUSCULAR | TEMPERATURA | GRIPA | COUNT | ENTRO()                             |
|------------|-------------|-------|-------|-------------------------------------|
| Si         | Null        | SiNo  | 3     | $2/3 \log_2(2/3) + 1/3 \log_2(1/3)$ |
| Si         | Normal      | Si    | 1     | $1/1 \log_2(1/1)$                   |
| Si         | Media       | No    | 1     | $1/1 \log_2(1/1)$                   |
| Si         | Alta        | Si    | 1     | $1/1 \log_2(1/1)$                   |
| No         | Null        | SiNo  | 3     | $2/3 \log_2(2/3) + 1/3 \log_2(1/3)$ |
| No         | Normal      | No    | 1     | $1/1 \log_2(1/1)$                   |
| No         | Media       | Si    | 1     | $1/1 \log_2(1/1)$                   |
| No         | Alta        | Si    | 1     | $1/1 \log_2(1/1)$                   |
| Null       | Normal      | SiNo  | 2     | $1/2 \log_2(1/2) + 1/2 \log_2(1/2)$ |
| Null       | Media       | SiNo  | 2     | $1/2 \log_2(1/2) + 1/2 \log_2(1/2)$ |
| Null       | Alta        | Si    | 2     | $2/2 \log_2(2/2)$                   |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 46 [31]

Una vez calculada la entropía se aplica la función agregada *gain()*, la cual, inicialmente, concatena los diferentes valores de los atributos condición y clase en un solo valor, así mismo suma las ocurrencias y los valores de las entropías.

Posteriormente, se calcula la entropía del conjunto clase y la de los posibles nodos y finalmente, con estos datos, calcula la ganancia de información y la almacena en la tabla de ganancia (ver Tabla 33), y se genera la tabla contenedora de nodos (ver Tabla 34), a la cual se le aplica el operador describe classifier y genera el árbol de decisión.

Tabla 33: Ganancia

| D_MUS-CULAR | TEMPE-RATURA            | GRIPA              | COUNT | ENTRO()   | GAIN()  |
|-------------|-------------------------|--------------------|-------|---|---|
| Si No       | Null                    | SiNo<br>SiNo       | 6     | $(2/3 \log_2(2/3) + 1/3 \log_2(1/3)) + (2/3 \log_2(2/3) + 1/3 \log_2(1/3))$                     | $(4/6 \log_2(4/6) + 2/6 \log_2(2/6)) - 3/6 (2/3 \log_2(2/3) + 1/3 \log_2(1/3)) - 3/6 (2/3 \log_2(2/3) + 1/3 \log_2(1/3))$                         |
| Si No       | Normal                  | Si No              | 2     | $(1/1 \log_2(1/1)) + (1/1 \log_2(1/1))$   | $(1/2 \log_2(1/2) + 1/2 \log_2(1/2)) - 1/2 (1/1 \log_2(1/1)) - 1/2 (1/1 \log_2(1/1))$   |
| SiNo        | Media                   | SiNo               | 2     | $(1/1 \log_2(1/1)) + (1/1 \log_2(1/1))$   | $2/2 \log_2(2/2) - 1/2 (1/1 \log_2(1/1)) - 1/2 (1/1 \log_2(1/1))$   |
| SiNo        | Alta                    | SiSi               | 1     | $(1/1 \log_2(1/1)) + (1/1 \log_2(1/1))$   | $2/2 \log_2(2/2) - 1/2 (1/1 \log_2(1/1)) - 1/2 (1/1 \log_2(1/1))$   |
| Null        | Normal<br>Media<br>Alta | SiNo<br>SiNo<br>Si | 6     | $(1/2 \log_2(1/2) + 1/2 \log_2(1/2)) + (1/2 \log_2(1/2) + 1/2 \log_2(1/2)) + (2/2 \log_2(2/2))$ | $(4/6 \log_2(4/6) + 2/6 \log_2(2/6)) - 2/6 (1/2 \log_2(1/2) + 1/2 \log_2(1/2)) - 2/6 (1/2 \log_2(1/2) + 1/2 \log_2(1/2)) - 2/6 (2/2 \log_2(2/2))$ |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 46 [31]

Tabla 34: Nodos

| NODO | PADRE | ATRIBUTO    | VALOR  | CLASE |
|------|-------|-------------|--------|-------|
| N0   | Null  | Temperatura | Null   | Null  |
| N1   | N0    | Temperatura | Alta   | Si    |
| N2   | N0    | Temperatura | Media  | No    |
| N3   | N0    | Temperatura | Normal | Null  |
| N4   | N3    | D_Muscular  | Si     | Si    |
| N5   | N3    | D_Muscular  | No     | No    |

Fuente Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002. Pág. 47 [31]

## 4. DESARROLLO DE Pg\_KDD

Con la finalidad de realizar el análisis y el diseño del proyecto Pg\_KDD se adoptó la metodología Proceso Unificado (RUP) que utiliza la notación del Lenguaje Unificado de Modelado (UML). En los siguientes ítems, se detallan algunos de los artefactos arrojados tras las iteraciones efectuadas.

### 4.1 ANÁLISIS UML

En esta etapa se definen los requerimientos del sistema, los casos de uso de la herramienta con sus respectivos diagramas y el diagrama de secuencia del sistema.

**4.1.1 Requerimientos.** Los requerimientos para el desarrollo de la herramienta PG\_KDD, están distribuidos por las funciones principales que componen la herramienta; de este modo se describirán cada una de las funciones y sub funciones:

Los requerimientos del manejo de la ejecución y manejo de la herramienta se describen en la Tabla 35.

Tabla 35: Requerimientos Ejecucion y Manejo de Pg\_KDD

| # Ref. | Función  | Categoría | Atributo | Observaciones                       | Prioridad   |
|--------|--|-----------|----------|-------------------------------------|-------------|
| R1     | Ejecutar la aplicación                                       | Evidente  | Interfaz |                                     | Obligatorio |
| R2     | Mostrar interfaz gráfica de la herramienta                   | Evidente  | Interfaz |                                     | Obligatorio |
| R2     | Mostrar mensajes de ayuda, respuesta o error si es necesario | Evidente  | Interfaz | Cuadros de dialogo, barra de estado | Deseable    |
| R3     | Modificar interfaz visual del aplicativo                     | Evidente  | Interfaz |                                     | Opcional    |
| R4     | Permitir ocultar la Barra de Estados                         | Evidente  | Interfaz |                                     | Obligatorio |
| R5     | Mostrar u ocultar el área de Propiedades                     | Evidente  | Interfaz |                                     | Obligatorio |

| # Ref. | Función                                       | Categoría | Atributo | Observaciones | Prioridad   |
|--------|---|-----------|----------|---------------|-------------|
| R6     | Mostrar u ocultar el árbol de Conexiones      | Evidente  | Interfaz |               | Obligatorio |
| R7     | Facilitar el cambio de Temas de la interfaz   | Evidente  | Interfaz |               | Deseable    |
| R8     | Refrescar toda la interfaz visual del sistema | Evidente  | Interfaz |               | Obligatorio |

Los requerimientos de la administración de conexiones a la herramienta se describen en la Tabla 36.

**Tabla 36: Requerimientos administración de conexiones**

| # Ref. | Función   | Categoría | Atributo    | Observaciones                                       | Prioridad   |
|--------|---|-----------|-------------|---|-------------|
| R1     | Permitir conexiones con el SGBD PostgreSQL                  | Evidente  | Interfaz    | El SGBD debe estar instalado en el equipo           | Obligatorio |
| R2     | Mostrar listado de conexiones que hayan sido guardadas      | Oculto    | Información |   | Obligatorio |
| R3     | Abrir el archivo de las conexiones que hayan sido guardadas | Oculto    | Información | El archivo lo debe haber generado por la aplicación | Obligatorio |
| R4     | Extraer datos del archivo de conexiones                     | Oculto    | Información | El archivo debe tener el formato correcto           | Obligatorio |
| R5     | Autorizar creación de Nuevas conexiones                     | Evidente  | Interfaz    |   | Obligatorio |
| R6     | Facilitar el que se establezcan los atributos de conexión   | Evidente  | Interfaz    |   | Obligatorio |
| R7     | Realizar la prueba del estado de la conexión                | Evidente  | Interfaz    |   | Obligatorio |
| R8     | Permitir el guardado de los datos de la conexión            | Evidente  | Interfaz    |   | Obligatorio |
| R9     | Autorizar apertura de conexiones existentes                 | Evidente  | Interfaz    | De las mostradas en el listado                      | Opcional    |
| R10    | Extraer los atributos de la conexión que se seleccione      | Oculto    | Información |   | Obligatorio |
| R11    | Mostrar el listado de bases de datos                        | Evidente  | Interfaz    |   | Obligatorio |
| R12    | Cargar las tablas y atributos de cada base                  | Oculto    | Información | Organizar la información en                         | Obligatorio |

| # Ref. | Función  | Categoría | Atributo | Observaciones                                | Prioridad |
|--------|----------|-----------|----------|--|-----------|
|        | de datos |           |          | listados en forma de árbol visual (TreeView) |           |

Los requerimientos de la manipulación de archivos SQL de la herramienta se describen en la Tabla 37.

**Tabla 37: Requerimientos Manipulación de Archivos SQL**

| # Ref.        | Función  | Categoría | Atributo    | Observaciones | Prioridad   |
|---------------|--|-----------|-------------|---------------|-------------|
| <b>R1</b>     | Manipular archivos planos SQL  | Evidente  | Interfaz    |               | Obligatorio |
| <b>R2</b>     | Crear archivos SQL   | Evidente  | Interfaz    |               | Obligatorio |
| <b>R3</b>     | Mostrar área de edición para el archivo SQL                                | Evidente  | Interfaz    |               | Obligatorio |
| <b>R4</b>     | Abrir archivos SQL que hayan sido guardados previamente                    | Evidente  | Interfaz    |               | Opcional    |
| <b>R5</b>     | Guardar archivos SQL   | Evidente  | Interfaz    |               | Opcional    |
| <b>R6</b>     | Ejecutar las sentencias SQL  | Evidente  | Interfaz    |               | Obligatorio |
| <b>R4.4.1</b> | Indicar al SGBD qué instrucciones debe ejecutar                            | Oculto    | Información |               | Obligatorio |
| <b>R7</b>     | Recibir del SGBD, y mostrar, el resultado de la ejecución de las consultas | Oculto    | Información |               | Obligatorio |
| <b>R8</b>     | Mostrar los errores proporcionados por el SGBD                             | Oculto    | Información |               | Obligatorio |

Los requerimientos de la administración de base de datos de la herramienta se describen en la Tabla 38.

**Tabla 38: Requerimientos Administración de bases de datos.**

| # Ref.    | Función                             | Categoría | Atributo | Observaciones | Prioridad   |
|-----------|-------------------------------------|-----------|----------|---------------|-------------|
| <b>R1</b> | Administrar las bases de datos (DB) | Evidente  | Interfaz |               | Obligatorio |
| <b>R2</b> | Permitir la eliminación de          | Evidente  | Interfaz |               | Obligatorio |

| # Ref. | Función                                       | Categoría | Atributo | Observaciones | Prioridad   |
|--------|---|-----------|----------|---------------|-------------|
|        | cualquier DB                                  |           |          |               |             |
| R3     | Permitir la creación de bases de datos        | Evidente  | Interfaz |               | Obligatorio |
| R4     | Editar cualquier objeto funcional de Postgres | Evidente  | Interfaz |               | Opcional    |
| R5     | Permitir la creación de Tablas de DB          | Evidente  | Interfaz |               | Obligatorio |
| R6     | Permitir la eliminación de Tablas de DB       | Evidente  | Interfaz |               | Obligatorio |
| R7     | Permitir la creación de Funciones             | Evidente  | Interfaz |               | Obligatorio |
| R8     | Permitir la eliminación de Funciones          | Evidente  | Interfaz |               | Obligatorio |
| R9     | Permitir la creación de Triggers              | Evidente  | Interfaz |               | Obligatorio |
| R10    | Permitir la eliminación de Triggers           | Evidente  | Interfaz |               | Obligatorio |
| R11    | Permitir la creación de Reglas                | Evidente  | Interfaz |               | Obligatorio |
| R12    | Permitir la eliminación de Reglas             | Evidente  | Interfaz |               | Obligatorio |
| R13    | Permitir la creación de Vistas                | Evidente  | Interfaz |               | Obligatorio |
| R14    | Permitir la eliminación de Vistas             | Evidente  | Interfaz |               | Obligatorio |
| R15    | Editar cualquier Tabla de la base de datos    | Evidente  | Interfaz |               | Opcional    |

Los requerimientos de la aplicación de tareas de minería de datos para el descubrimiento de conocimiento con bases de datos de la herramienta se describen en la Tabla 39.

**Tabla 39: Requerimientos para la aplicación de minería de datos.**

| # Ref. | Función                              | Categoría | Atributo | Observaciones | Prioridad   |
|--------|--------------------------------------|-----------|----------|---------------|-------------|
| R1     | Aplicar técnicas de Minería de Datos | Evidente  | Interfaz |               | Obligatorio |
| R2     | Algoritmo de Asociacion              | Evidente  | Interfaz |               | Obligatorio |
| R3     | Aplicación de Algoritmo EquipAsso    | Evidente  | Interfaz |               | Obligatorio |
| R4     | Algoritmo de Clasificación           | Evidente  | Interfaz |               | Obligatorio |
| R5     | Aplicación de Algoritmo Mate-Tree    | Evidente  | Interfaz |               | Obligatorio |

Los requerimientos de la administración de consultas SQL gráficas en la herramienta se describen en la Tabla 40.

**Tabla 40: Requerimientos administración de consultas SQL gráficas.**

| # Ref. | Función   | Categoría | Atributo | Observaciones   | Prioridad   |
|--------|---|-----------|----------|---|-------------|
| R1     | Realizar Consultas a la base de datos                                     | Evidente  | Interfaz |   | Obligatorio |
| R2     | Manipular el contenido de las tablas de la DB                             | Evidente  | Interfaz |   | Opcional    |
| R3     | Mostrar todos los datos de la Tabla seleccionada                          | Evidente  | Interfaz |   | Opcional    |
| R4     | Permitir la adición de nuevas Tuplas a las tablas de la DB                | Evidente  | Interfaz |   | Opcional    |
| R5     | Permitir la eliminación de Tuplas   | Evidente  | Interfaz |   | Opcional    |
| R6     | Permitir la modificación de los datos de las Tuplas                       | Evidente  | Interfaz |   | Opcional    |
| R7     | Permitir digitar y ejecutar código SQL                                    | Evidente  | Interfaz |   | Obligatorio |
| R8     | Facilitar la administración de consultas SQL usando un diseñador          | Evidente  | Interfaz |   | Opcional    |
| R9     | Permitir establecer las tablas que intervendrán en la consulta            | Evidente  | Interfaz |   | Obligatorio |
| R10    | Permitir establecer las columnas que se mostrarán al ejecutar la consulta | Evidente  | Interfaz |   | Obligatorio |
| R11    | Permitir establecer el orden en el que aparecerán las columnas            | Evidente  | Interfaz |   | Obligatorio |
| R12    | Permitir la creación gráfica y automática de condiciones para la consulta | Evidente  | Interfaz |   | Obligatorio |
| R13    | Permitir la creación gráfica y manual de condiciones para la consulta     | Evidente  | Interfaz | Evitando la forma convencional de los Joins (con signo "=") | Obligatorio |
| R14    | Permitir el uso de Agrupaciones para los resultados de la consulta        | Evidente  | Interfaz |   | Obligatorio |
| R15    | Permitir el Ordenamiento de los resultados de la consulta                 | Evidente  | Interfaz |   | Obligatorio |
| R16    | Visualizar los resultados de la consulta diseñada                         | Evidente  | Interfaz |   | Obligatorio |
| R17    | Permitir guardar en un archivo SQL el diseño de la consulta               | Evidente  | Interfaz |   | Obligatorio |

| # Ref. | Función  | Categoría | Atributo | Observaciones | Prioridad   |
|--------|--|-----------|----------|---------------|-------------|
|        | realizada  |           |          |               |             |
| R18    | Permitir la edición manual de las consultas que se estén diseñando | Evidente  | Interfaz |               | Obligatorio |
| R19    | Borrar la consulta actual si se cambia de base de datos            | Evidente  | Interfaz |               | Obligatorio |
| R20    | Preguntar si se desea guardar la consulta al salir del diseñador   | Evidente  | Interfaz |               | Obligatorio |

**4.1.2 Casos de uso.** Mediante este modelo se presentan los diferentes escenarios (o caminos de uso) del sistema. Partiendo de ellos, el analista de sistemas puede identificar de forma más clara, la mayoría de los objetos involucrados en realizar determinada tarea en comunicación con los diferentes roles que adquieren los usuarios del sistema.

A continuación se describen algunos de los casos de uso más representativos de la aplicación.

### Caso de Uso: Conexión

| Caso de uso  | Conexion  |
|--|---|
| <b>Actor principal</b>   | <b>Usuario</b>  |
| <b>Precondiciones</b>  | ⌘ <b>Se debe haber ejecutado el aplicativo.</b>                 |
| <b>Poscondiciones</b>  | ⌘ <b>Se ejecuta la opción que el Usuario haya seleccionado.</b> |
| <b>Flujo básico</b>  |   |
| <ol style="list-style-type: none"> <li>1. Se visualiza la interfaz de conexión de la aplicación.</li> <li>2. Se procede a ingresar los datos respectivos para la conexión.</li> <li>3. Se presiona el botón de testeo para verificar si los datos de conexión son validos.</li> <li>4. El sistema evalúa internamente los datos ingresados e indica el resultado del testeo mediante un mensaje dentro de la interfaz.</li> <li>5. Si el proceso anterior es exitoso, se procede a presionar el botón conectar.</li> <li>6. Se crea la nueva conexión y se cierra la interfaz de conexión.</li> <li>7. Se cargan todos los controles gráficos (Objetos SQL que contenga el usuario) en el árbol de conexiones (Tree Connections).</li> </ol> |   |
| <b>Flujos alternativos</b>   |   |
| <ol style="list-style-type: none"> <li>4a. En caso de que el usuario haya ingresado mal los datos de conexión: <ol style="list-style-type: none"> <li>1. El sistema muestra un mensaje, indicando que la conexión ha fallado debido a un mal ingreso en los datos de la conexión.</li> </ol> </li> </ol>   |   |

| Caso de uso | Conexion   |
|-------------|--|
|             | <ol style="list-style-type: none"> <li>2. Se reinicia el caso de uso.</li> </ol>   |
| 4.b         | <p>En caso de que los datos estén ingresados correctamente, y el usuario presiona el botón guardar.</p> <ol style="list-style-type: none"> <li>1. El sistema guarda la nueva conexión, visualizándola en la tabla de conexiones.</li> <li>2. Se procede a presionar conectar.</li> </ol>   |
|             | <p>5. En caso de que los datos estén bien ingresados, pero exista una conexión con el mismo nombre o para el mismo usuario:</p> <ol style="list-style-type: none"> <li>1. El sistema muestra mensajes de alerta describiendo que existe un dato repetido dentro de la aplicación.</li> <li>2. Se reinicia el caso de uso.</li> </ol> |

## Caso de Uso: Administración Base de Datos

| Caso de uso   | Administración Base de Datos   |
|---|--|
| <b>Actor principal</b>  | <b>Usuario</b>   |
| <b>Precondiciones</b>   | <ol style="list-style-type: none"> <li>✗ Se debe haber ejecutado el aplicativo.</li> <li>✗ Se debe haber el caso de uso conectar.</li> </ol> |
| <b>Poscondiciones</b>   | ✗ Se ejecuta la opción que el Usuario haya seleccionado.   |
| <b>Flujo básico</b>   |  |
| <ol style="list-style-type: none"> <li>1. Se visualiza la interfaz principal de la aplicación.</li> <li>2. Si ya se ha realizado el camino del caso de uso conectar, el aplicativo carga automáticamente en el árbol de conexiones aquellos elementos pertenecientes al usuario de la conexión entre ellos sus bases de datos.</li> <li>3. El usuario procede a desplegar la conexión del árbol, visualizando las bases de datos con sus derivados de ellas (Tablas, Columnas, llaves, reglas, disparadores, funciones).</li> <li>4. Con una base de datos seleccionada, el sistema activa todas las opciones de manejo básicas (creación, modificación, eliminación) módulos de sentencias gráficas, módulo KDD.</li> <li>5. El sistema espera a que el usuario seleccione una de las opciones.</li> </ol> |  |
| <b>Flujos alternativos</b>  |  |
| <p>2a. En caso de que el usuario aun no tenga bases de datos registradas dentro del gestor:</p> <ol style="list-style-type: none"> <li>1. El usuario puede crear una nueva base de datos.</li> <li>2. Al ser creada la base de datos, el aplicativo crea automáticamente los elementos correspondientes a tablas, funciones, en otros. Los cuales también tienen las opciones de manejo básicas.</li> <li>3. Se reinicia el caso de uso.</li> </ol> <p>4.b En caso de que el usuario no haya seleccionado ningún elemento partiendo de un objeto de base de datos:</p> <ol style="list-style-type: none"> <li>3. El aplicativo mantiene inactivas las opciones de manejo básicas, módulo de sentencias gráficas y módulo KDD.</li> <li>4. Se reinicia el caso de uso.</li> </ol>                            |  |

## Caso de Uso: Aplicación técnicas de minería de datos

| Caso de uso   | Apelación técnicas de minería de datos   |
|---|--|
| <b>Actor principal</b>  | <b>Usuario</b>   |
| <b>Precondiciones</b>   | <ul style="list-style-type: none"> <li>✗ Se debe haber ejecutado el aplicativo.</li> <li>✗ Se debe haber ejecutado el caso de uso conectar</li> <li>✗ Se debe haber ejecutado el caso de uso administrar bases de datos</li> </ul> |
| <b>Poscondiciones</b>   | ✗ Se ejecuta la opción que el Usuario haya seleccionado.   |
| <b>Flujo básico</b>   |  |
| <ol style="list-style-type: none"> <li>1. Se visualiza la interfaz de minería de datos de la aplicación.</li> <li>2. Se procede a aplicar el algoritmo de asociación.</li> <li>3. El usuario procede insertar nodo repositorio, configurarlo, ejecutarlo y conectarlo a un nodo operador associator/assocol.</li> <li>4. El usuario procede a insertar el nodo operador associator/assocol, configurarlo y ejecutarlo y lo conecta a un nodo tabla de visualización.</li> <li>5. El aplicativo visualiza la sentencia sql que se procederá a ejecutar en un marco de texto.</li> <li>6. El usuario procede correr el nodo tabla de visualización anterior y el aplicativo muestra su resultado en un marco de visualización.</li> <li>7. El usuario conecta el nodo operador associator/assocol a un nodo algoritmo EquipAsso.</li> <li>8. El usuario procede a insertar el nodo algoritmo EquipAsso, configurarlo y ejecutarlo y se conecta a un nodo tabla de visualización.</li> <li>9. El aplicativo visualiza la sentencia sql que se procederá a ejecutar en un marco de texto.</li> <li>10. El usuario procede correr el nodo tabla de visualización anterior y el aplicativo muestra su resultado en un marco de visualización.</li> <li>11. El usuario conecta el nodo algoritmo equipaso a un nodo describe association rules.</li> <li>12. El usuario procede a insertar el nodo describe rules, configurarlo y ejecutarlo y se conecta a un nodo reglas de visualización.</li> <li>13. El aplicativo visualiza la sentencia sql que se procederá a ejecutar en un marco de texto.</li> <li>14. El usuario procede correr el nodo reglas de visualización anterior y el aplicativo muestra las reglas de asociación en un marco de visualizacion.</li> <li>15. Termina el algoritmo de asoccioacion.</li> <li>16. Termina el caso de uso.</li> </ol> |  |
| <b>Flujos alternativos</b>  |  |
| <ol style="list-style-type: none"> <li>2a. En caso de que el usuario opte por aplicar el algoritmo de clasificacion:               <ol style="list-style-type: none"> <li>1. El usuario procede insertar nodo repositorio,configurarlo, ejecutarlo y conectarlo a un nodo operador mate.</li> <li>2. El usuario procede a insertar el nodo operador mate, configurarlo y ejecutarlo y lo conecta a un nodo tabla de visualización.</li> <li>3. El aplicativo visualiza la sentencia sql que se procederá a ejecutar en un marco de texto.</li> <li>4. El usuario procede correr el nodo tabla de visualización anterior y el aplicativo muestra su resultado en un marco de visualización.</li> <li>5. El usuario conecta el nodo operador mate a un nodo algoritmo matetree.</li> <li>6. El usuario procede a insertar el nodo algoritmo matetree, configurarlo y ejecutarlo y se conecta a un nodo tabla de visualización.</li> <li>7. El aplicativo visualiza la sentencia sql que se procederá a ejecutar en un marco de texto</li> <li>8. El usuario procede correr el nodo tabla de visualización anterior y el aplicativo muestra su resultado en un marco de visualización.</li> <li>9. El usuario conecta el nodo operador matetree a un nodo función agregada entro.</li> <li>10. El usuario procede a insertar el nodo función agregada entro, configurarlo y ejecutarlo y se</li> </ol> </li> </ol>   |  |

| Caso de uso   | Apelación técnicas de minería de datos  |
|---|---|
|   | <p>conecta a un nodo tabla de visualización.</p> <ol style="list-style-type: none"> <li>11. El aplicativo visualiza la sentencia sql que se procederá a ejecutar en un marco de texto</li> <li>12. El usuario procede correr el nodo tabla de visualización anterior y el aplicativo muestra su resultado en un marco de visualización.</li> <li>13. El usuario conecta el nodo función agregada entro a un nodo función agregada gain.</li> <li>14. El usuario procede a insertar el nodo función agregada gain, configurarlo y ejecutarlo y se conecta a un nodo tabla de visualización.</li> <li>15. El aplicativo visualiza la sentencia sql que se procederá a ejecutar en un marco de texto</li> <li>16. El usuario procede correr el nodo tabla de visualización anterior y el aplicativo muestra su resultado en un marco de visualización.</li> <li>17. El usuario conecta el nodo función agregada gain a un nodo describe clasification rules.</li> <li>18. El usuario inserta el nodo describe clasification rules, configurarlo y ejecutarlo y se conecta a un nodo reglas de visualización.</li> <li>19. El aplicativo visualiza la sentencia sql que se procederá a ejecutar en un marco de texto</li> <li>20. El usuario procede correr el nodo reglas de visualización anterior y el aplicativo muestra las reglas de clasificacion en un marco de visualizacion.</li> <li>21. El usuario inserta un nodo de visualizacion TreeViewer , conecta el nodo describe clasification rules este nodo y lo ejecuta.</li> <li>22. El aplicativo visualiza el árbol de nodos procedentes a las reglas de clasificación.</li> <li>23. Termina el algoritmo de clasificación.</li> <li>24. Termina el caso de uso.</li> </ol> |
| Observaciones   |   |
| <p>Tanto para el camino básico como el alternativo, el resultado final contendrá errores, si la configuración del repositorio, los operadores, operadores de descripción es incorrecta, por ende el aplicativo informara estos errores por medio de mensajes emergentes.</p> <p>Si el usuario intenta realizar conexiones incoherentes de nodos, el aplicativo informa mediante mensajes emergentes el error del intento de conexión.</p> |   |

## Caso de Uso: Manipulación de scripts SQL

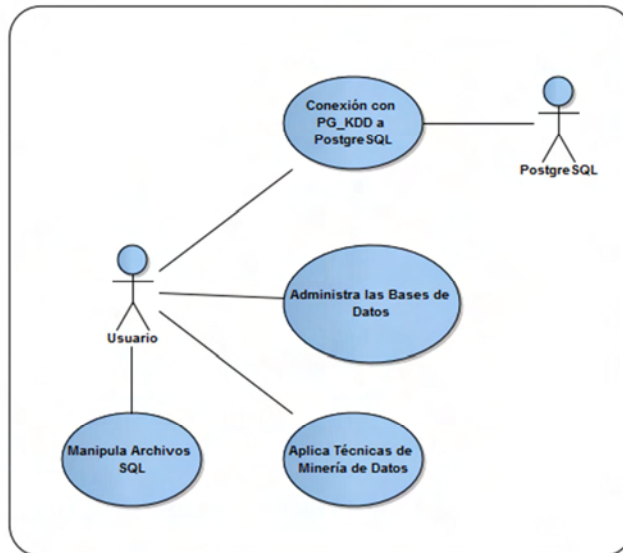
| Caso de uso  | Manipular de scripts SQL  |
|--|---|
| <b>Actor principal</b>   | <b>Usuario</b>  |
| <b>Precondiciones</b>  | <ul style="list-style-type: none"> <li>⌘ Se debe haber ejecutado el aplicativo.</li> <li>⌘ Se debe haber ejecutado el caso de uso conectar.</li> <li>⌘ Se debe haber ejecutado el caso de uso administrar bases de datos</li> </ul> |
| <b>Poscondiciones</b>  | ⌘ Se ejecuta la opción que el Usuario haya seleccionado.  |
| Flujo básico   |   |
| <ol style="list-style-type: none"> <li>1. Se visualiza la interfaz principal de la aplicación.</li> <li>2. Si ya se a realizado el camino del caso de uso conectar, el aplicativo carga automaticamente en el árbol de conexiones aquellos elementos pertenecientes al usuario de la conexión entre ellos sus bases de datos.</li> <li>3. El usuario elije una base de datos automáticamente se activan los controles de gestión de archivos al igual que otros.</li> <li>4. El usuario selecciona la opción de nueva hoja SQL de la barra de herramientas.</li> <li>5. El aplicativo despliega una interfaz para manejo de scripts SQL como pestaña en el área vistas de</li> </ol> |   |

| Caso de uso         | Manipular de scripts SQL  |
|---------------------|---|
|                     | <p>la interfaz principal.</p> <ol style="list-style-type: none"> <li>6. El usuario tiene la posibilidad de elegir entre las opciones de abrir, crear, editar y guardar un script SQL.</li> <li>7. El usuario digita código SQL dentro del marco de texto de la interfaz y selecciona la opción guardar.</li> <li>8. El aplicativo despliega una interfaz de carga de archivos (FileUpload) y espera que el usuario elija la ruta donde desee guardar el nuevo script.</li> <li>9. El usuario presiona guardar. La interfaz guarda el archivo en la ruta establecida.</li> <li>10. Fin del caso de uso.</li> </ol>   |
| Flujos alternativos |   |
|                     | <ol style="list-style-type: none"> <li>7a. En caso que el usuario seleccione la opción abrir script: <ol style="list-style-type: none"> <li>1. La Interfaz despliega una interfaz de carga de archivos (FileUpload) y espera que el usuario elija un script de extencion SQL.</li> <li>2. El usuario elije el script y selecciona la opción abrir.</li> <li>3. La aplicacion cierra la interfaz de carga de archivos y automáticamente visualiza el contenido del script en el marco de texto.</li> <li>4. La aplicación espera a que el usuario edite el script o intente cargar uno nuevo.</li> <li>5. Se reinicia el caso de uso.</li> </ol> </li> </ol> |

**4.1.3 Diagramas de casos de uso.** En esta sección se presentan los diagramas de casos de uso de Pg\_KDD.

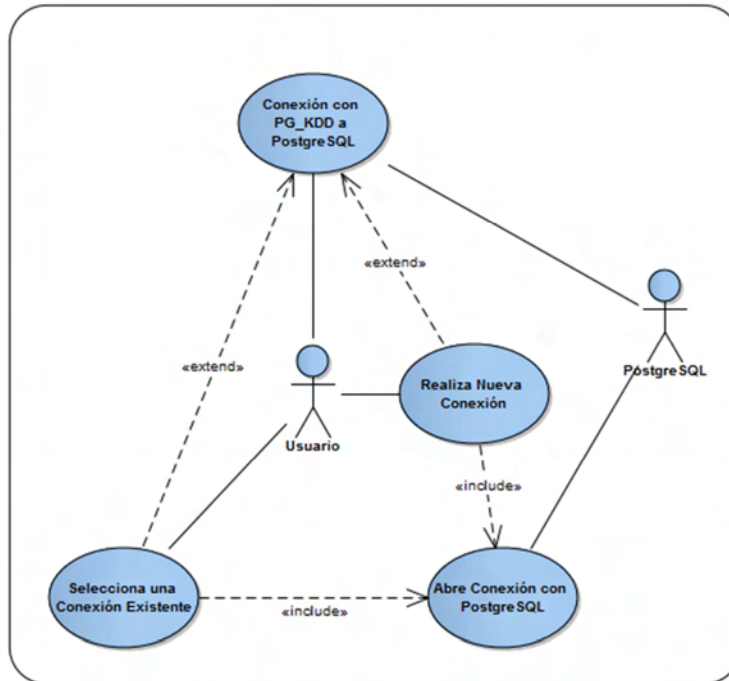
**4.1.3.1 Diagrama de Caso de Uso: Diagrama General.** En la Figura 21, se muestra el diagrama de casos de uso general de Pg\_KDD.

Figura 21: Caso de Uso General



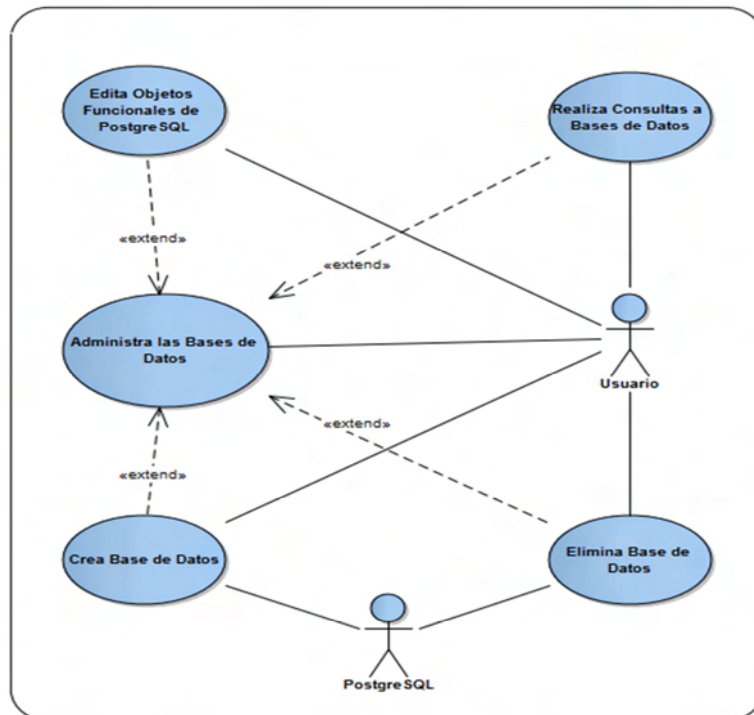
**4.1.3.2 Diagrama de Caso de Uso: Conectar.** En la Figura 22, se muestra el diagrama de caso de uso que permite la conexión con el sistema PostgresKDD.

Figura 22: Caso de Uso Conectar



**4.1.3.3 Diagrama Caso de Uso: Administrar Bases de Datos.** En la Figura 23, se presenta el caso de uso que permite la administración de base de datos.

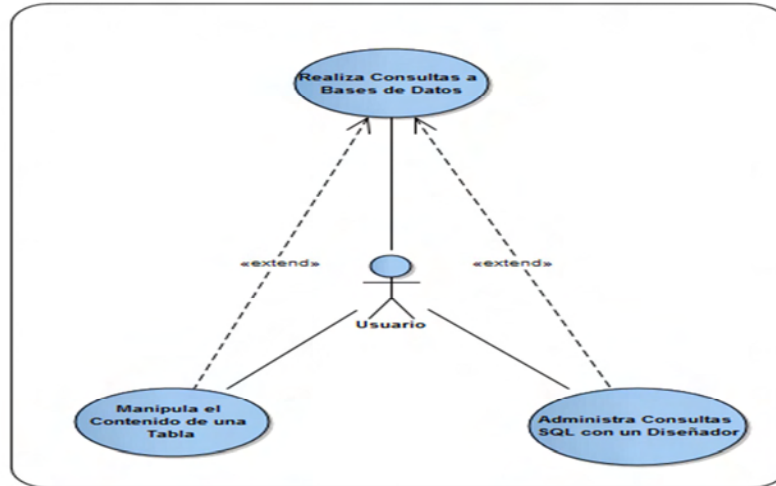
Figura 23: Caso de Uso Adm. Base de Datos



#### 4.1.3.4 Diagrama de Caso de Uso: Realizar de Consultas a Bases de Datos.

En la Figura 24, se describe el caso de uso que permite realizar las consultas desde Pg\_KDD con las bases de datos.

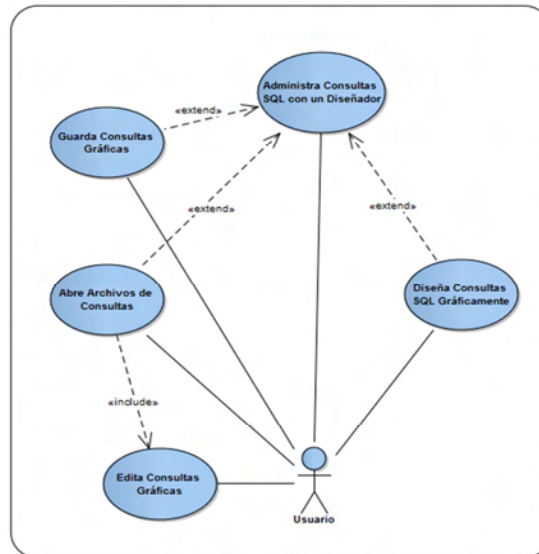
Figura 24: Caso de Uso Realizar Consultas



#### 4.1.3.5 Diagrama de Caso de Uso: Administrar Consultas SQL Gráficas.

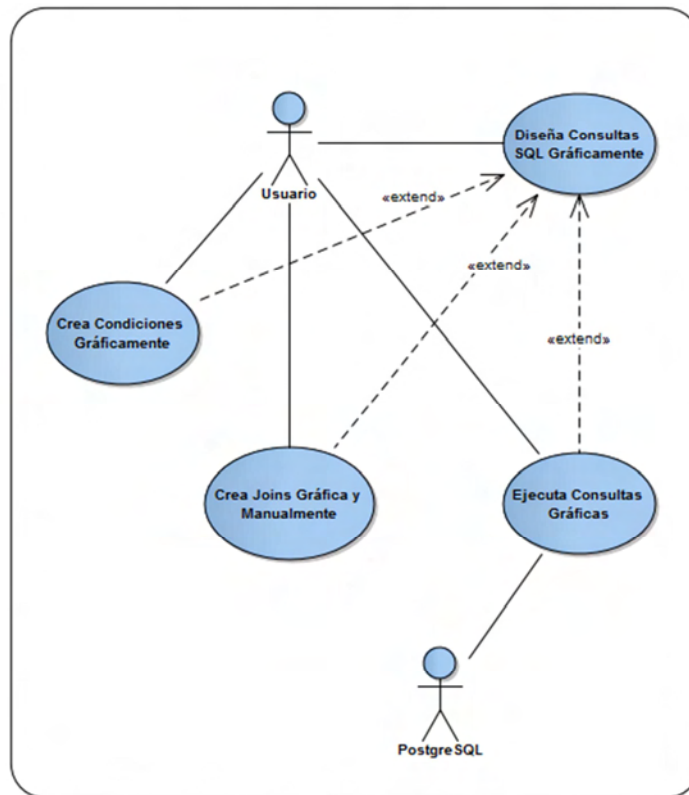
En la Figura 25, se describe el caso de uso que permite la administración de consultas SQL de manera gráfica.

Figura 25: Caso de Uso Adm. Consultas Gráficas



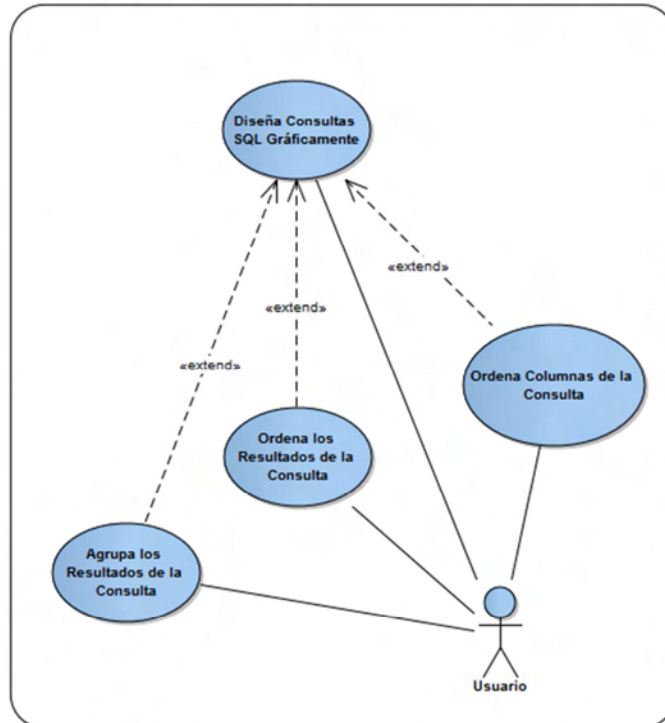
**4.1.3.6 Diagrama de Caso de Uso: Realizar de Consultas Gráficas.** En la Figura 26, se describe el caso de uso que permite realizar gráficamente una consulta SQL en Pg\_KDD.

Figura 26: Caso de Uso Realizar Consultas Gráficas



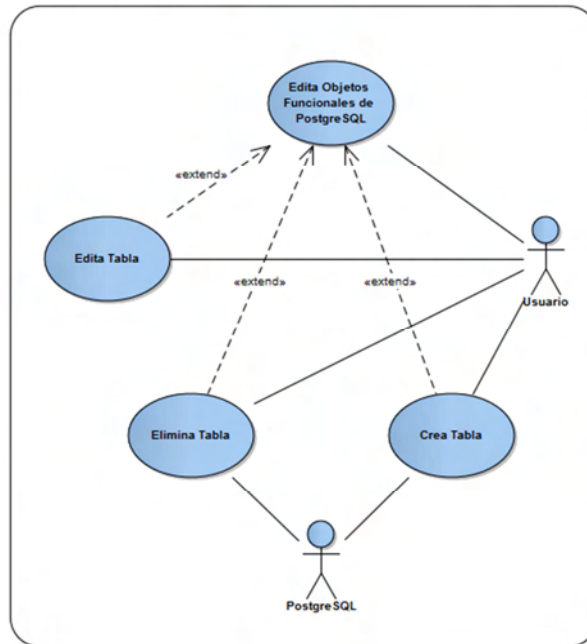
**4.1.3.7 Diagrama de Caso de Uso: Ordenar Resultados de las Consultas Gráficas.** En la Figura 27, se describe el caso de uso que permite ordenar los resultados de la consulta gráfica realizada previamente.

Figura 27: Caso de Uso Ordenar Resultado Consultas Gráficas



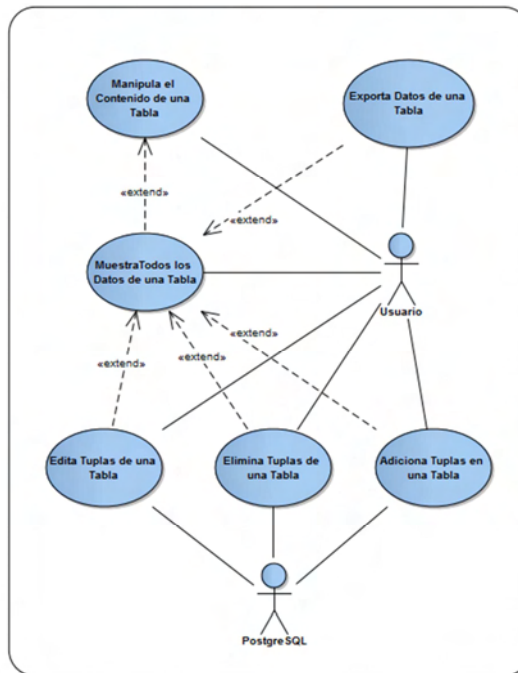
**Diagrama de Caso de Uso: Gestionar Tablas.** En la Figura 28, se describe el caso de uso que permite gestionar las tablas de una base de datos.

Figura 28: Caso de Uso Gestionar Tablas



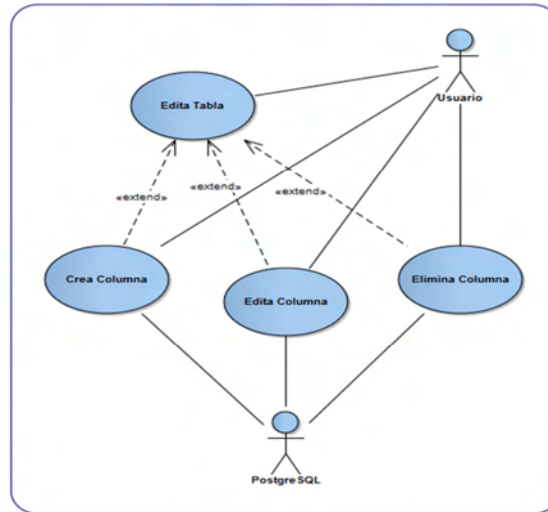
**4.1.3.8 Diagrama de Caso de Uso: Manipular el Contenido de una Tabla.**  
En la Figura 29 se describe el caso de uso que permite manipular los registros de una tabla de una base de datos.

Figura 29: Caso de Uso Manipular contenido de tablas



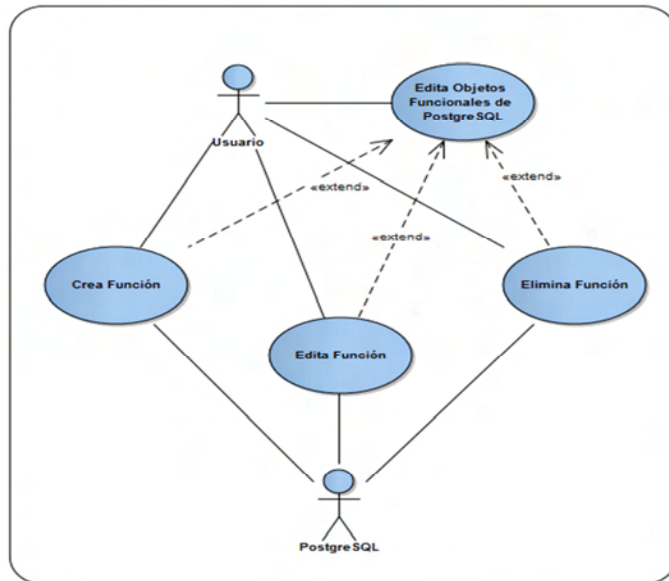
**4.1.3.9 Diagrama de Caso de Uso: Gestionar Columnas.** En la Figura 30, se describe el caso de uso que permite gestionar la columna de una tabla de una base de datos.

Figura 30: Caso de Uso Gestionar Columnas



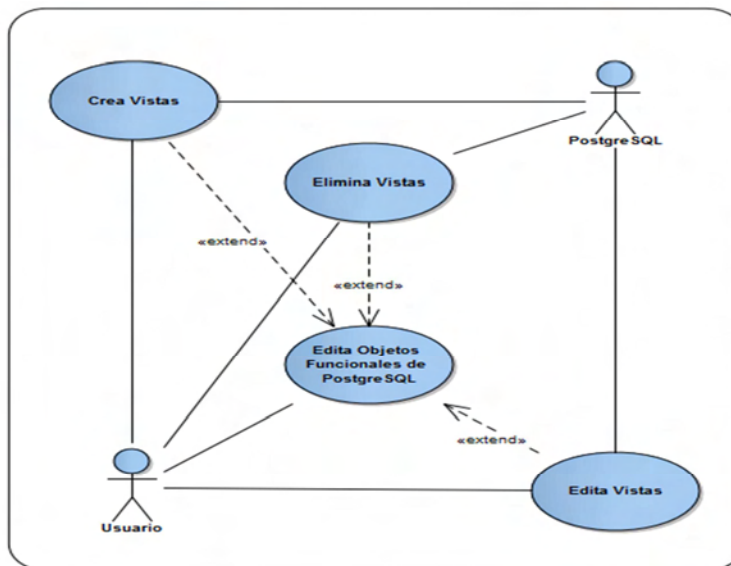
**4.1.3.10 Diagrama de Caso de Uso: Gestionar Funciones.** En la Figura 31, se describe el caso de uso que permite gestionar las funciones de una base de datos.

Figura 31: Caso de Uso Gestionar Funciones



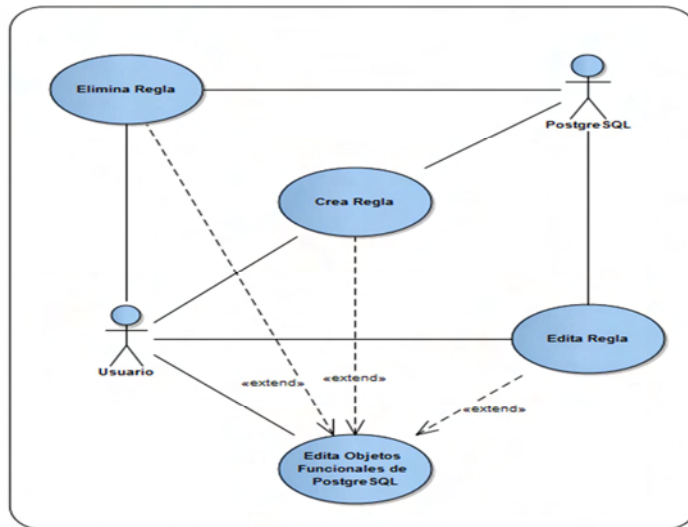
**4.1.3.11 Diagrama de Caso de Uso: Gestionar Vistas.** En la Figura 32, se describe el caso de uso que permite gestionar las vistas de una base de datos.

Figura 32: Caso de Uso Gestionar Vistas



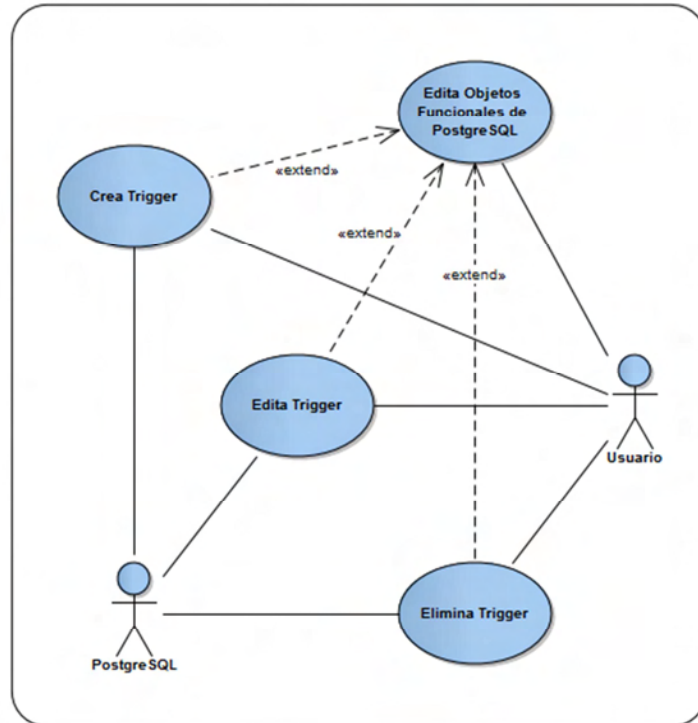
**4.1.3.12 Diagrama de Caso de Uso: Gestionar Reglas.** En la Figura 33, se describe el caso de uso que permite gestionar las reglas de una base de datos.

Figura 33: Caso de Uso Gestionar Reglas



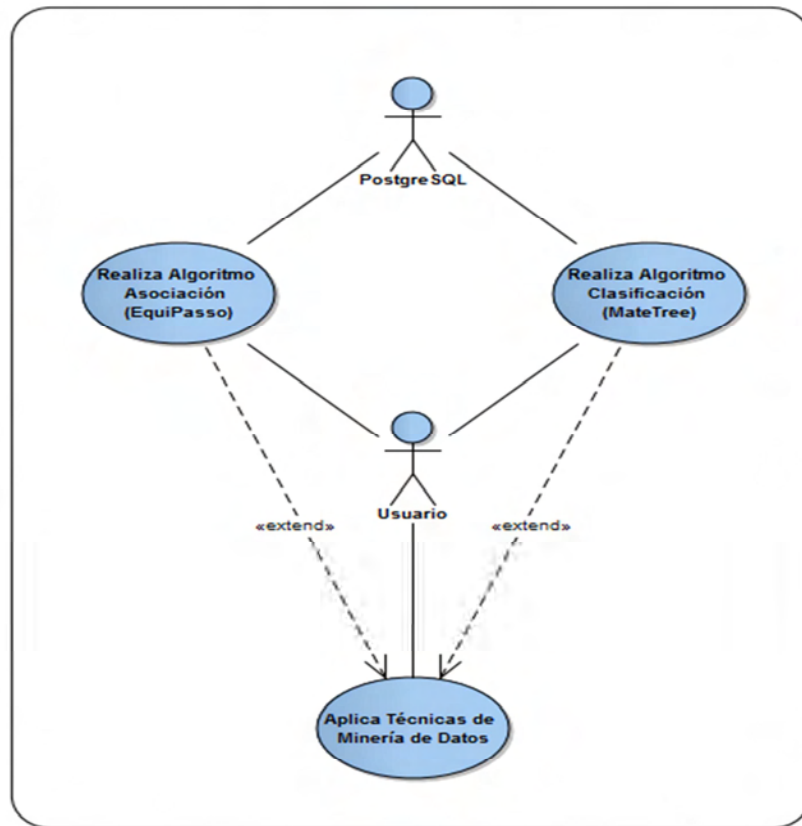
**4.1.3.13 Diagrama de Caso de Uso: Gestionar Triggers.** En la Figura 34, se describe el caso de uso que permite gestionar los disparadores de una base de datos.

Figura 34: Caso de Uso Gestionar Disparadores



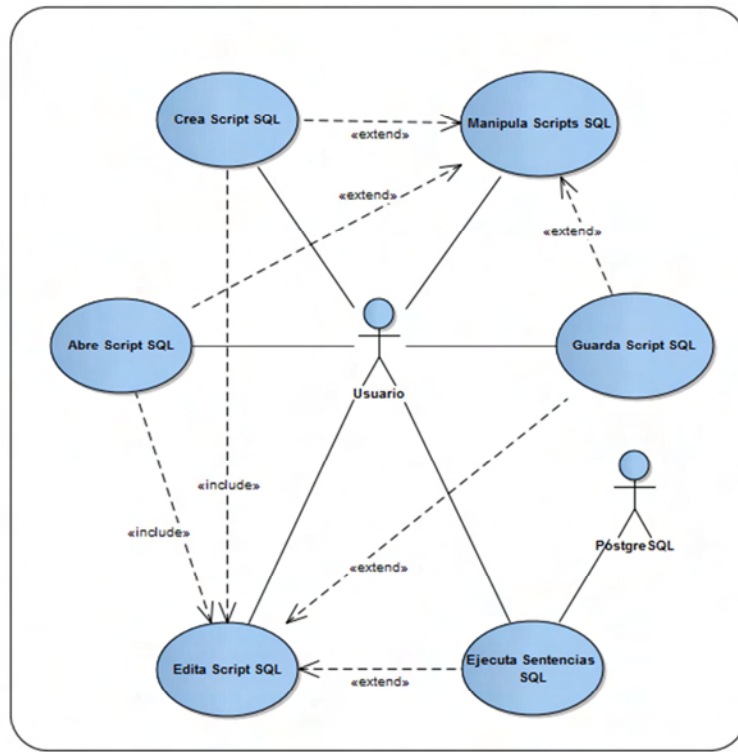
**4.1.3.14 Diagrama de Caso de Uso: Aplicar Técnicas de Minería de Datos.**  
En la Figura 35, se describe el caso de uso que permite realizar los algoritmos respectivos a las tareas de descubrimiento de conocimiento para asociación y clasificación.

Figura 35: Caso de Uso Minería de Datos



**4.1.3.15 Diagrama de Caso de Uso: Manipular scripts SQL.** En la Figura 36, se describe el caso de uso que permite manipular los scripts SQL dentro de la aplicación.

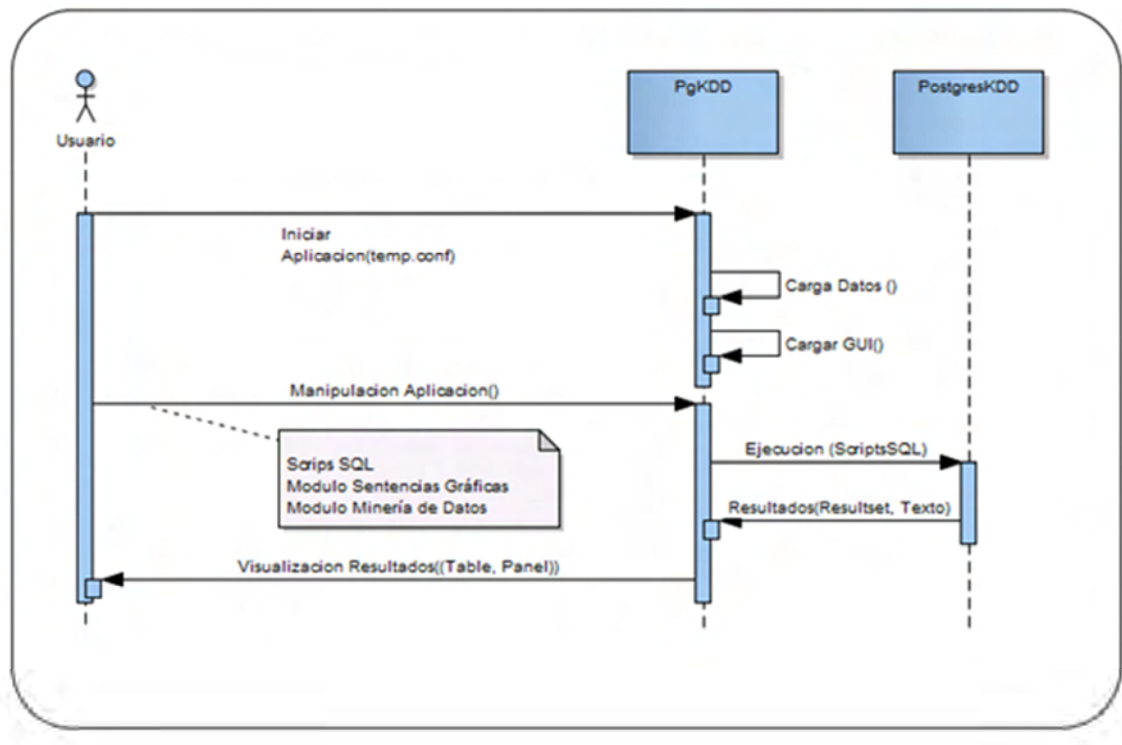
Figura 36: Caso de Uso Manipular Scripts SQL



## 4.2 DIAGRAMA DE SECUENCIA DEL SISTEMA.

En la Figura 37, se describe el diagrama de secuencia de Pg\_KDD, ilustrando la interacción entre el usuario con la herramienta gráfica, y este con el sistema de descubrimiento de conocimiento PostgresKDD.

Figura 37: Diagrama del Sistema



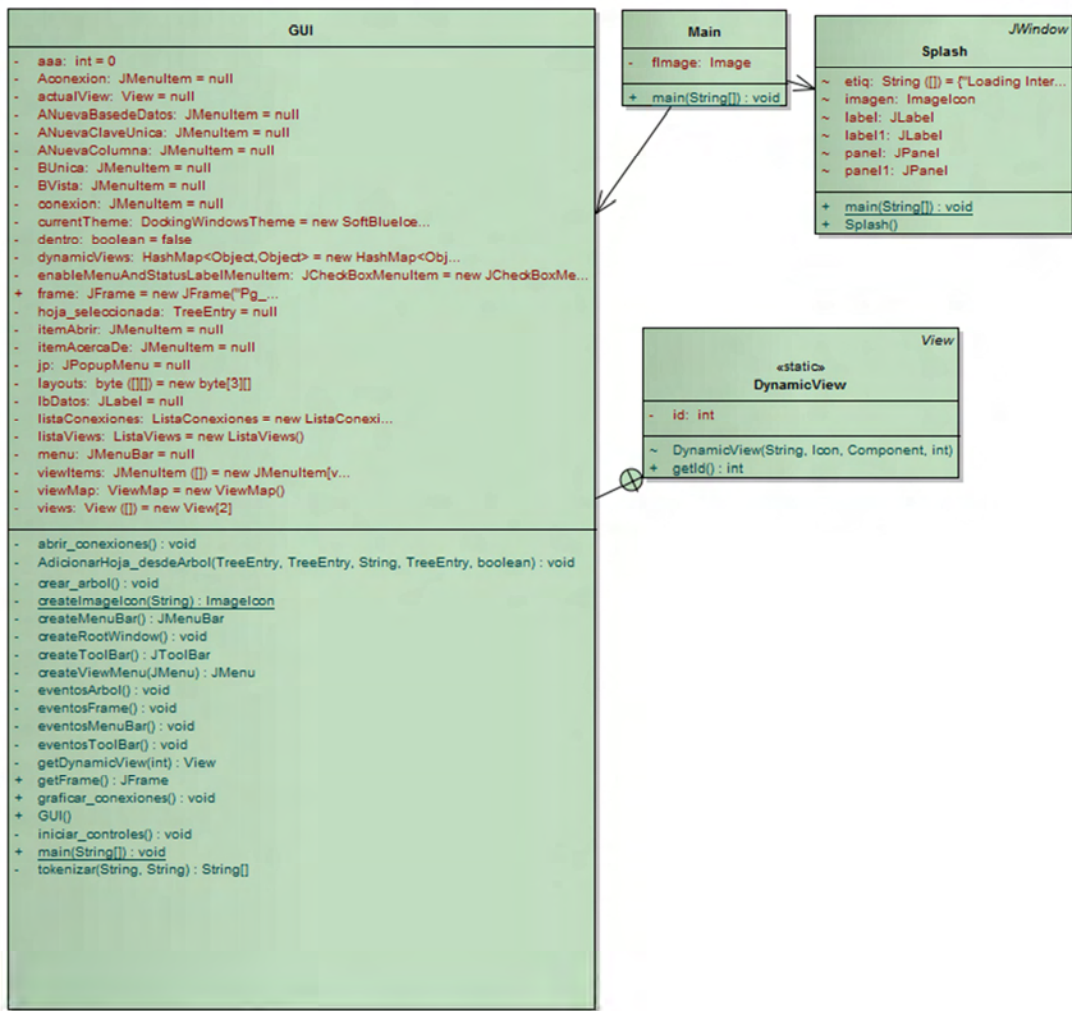
### 4.3 DISEÑO UML

En esta etapa se define el diseño de la herramienta y se describen los diagramas de clase y los diagramas de paquetes.

**4.3.1 Diagramas de clases.** A continuación se describen los diagramas de clase del sistema.

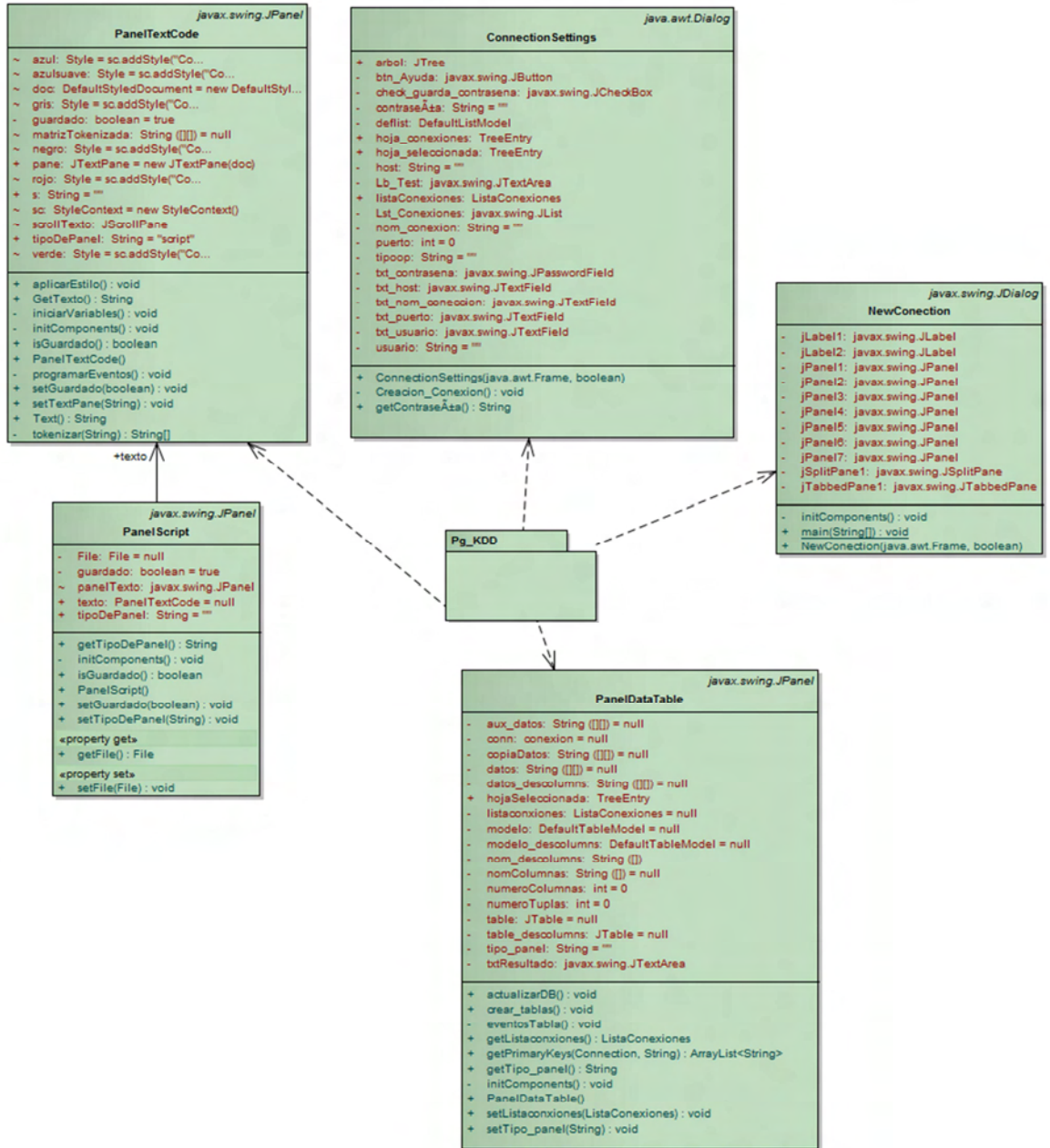
**4.3.1.1 Diagrama de Clases: Inicio de la Aplicación.** En la Figura 38, se muestra el diagrama de clases del inicio de la aplicación Pg\_KDD.

Figura 38: Diagrama de Clases Inicio de Aplicación



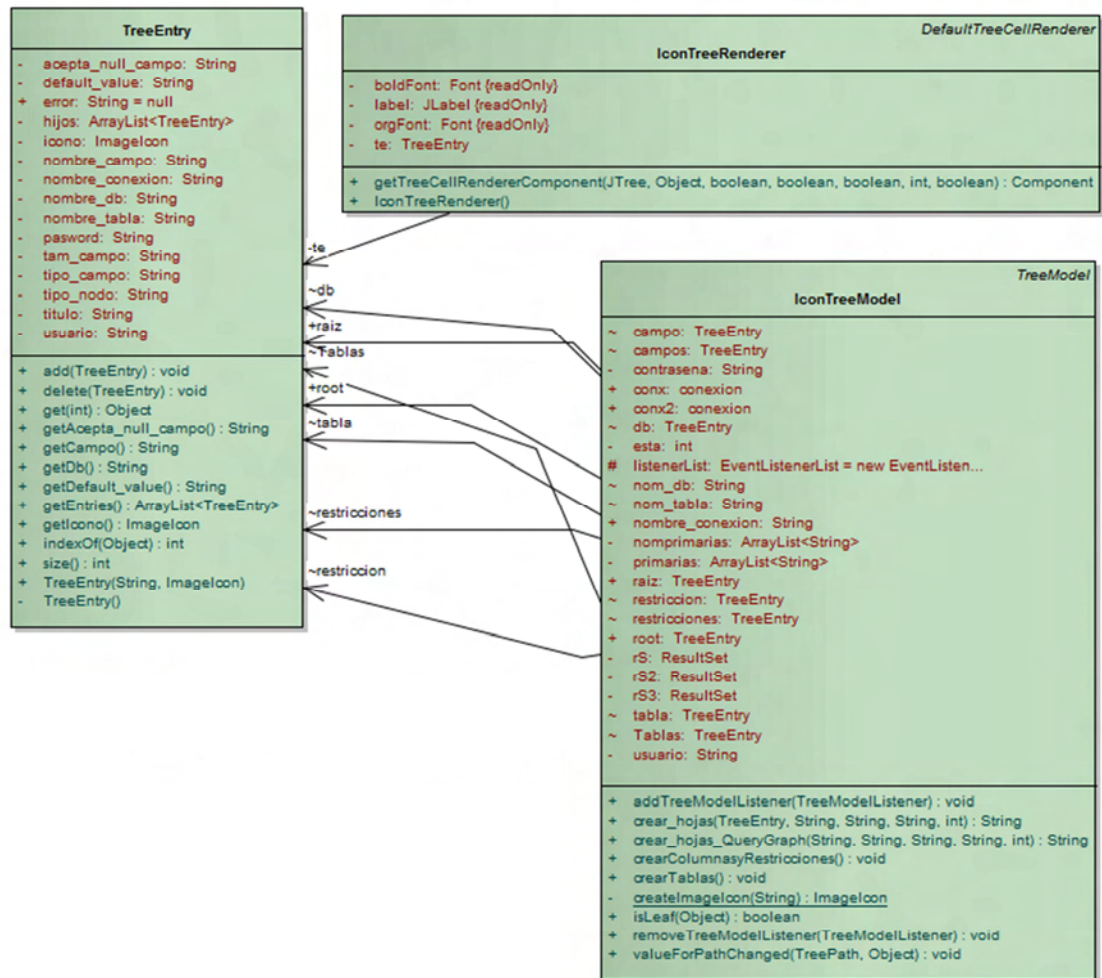
4.3.1.2 Diagrama de Clases: Administración Visual Pg\_KDD. En la Figura 39, se muestra el diagrama de clases que permite la administración visual de Pg\_KDD.

Figura 39: Diagrama de Clases Adm. Visual de Pg\_KDD



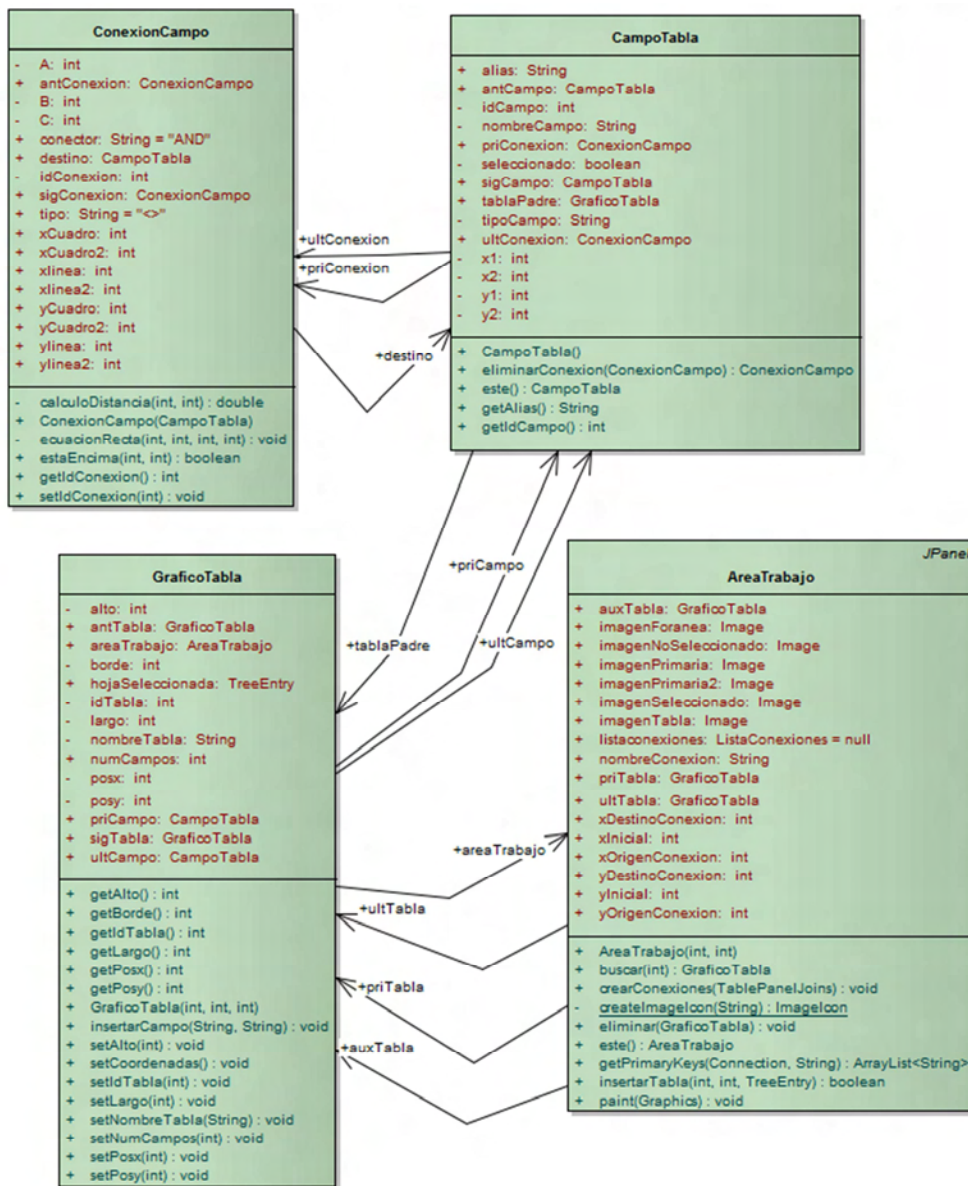
**4.3.1.3 Diagrama de Clases: Árbol de Elementos.** En la Figura 40, se muestra el diagrama de clases que permite manejar los elementos del árbol de conexiones de Pg\_KDD.

Figura 40: Diagrama de Clases Elementos del Árbol



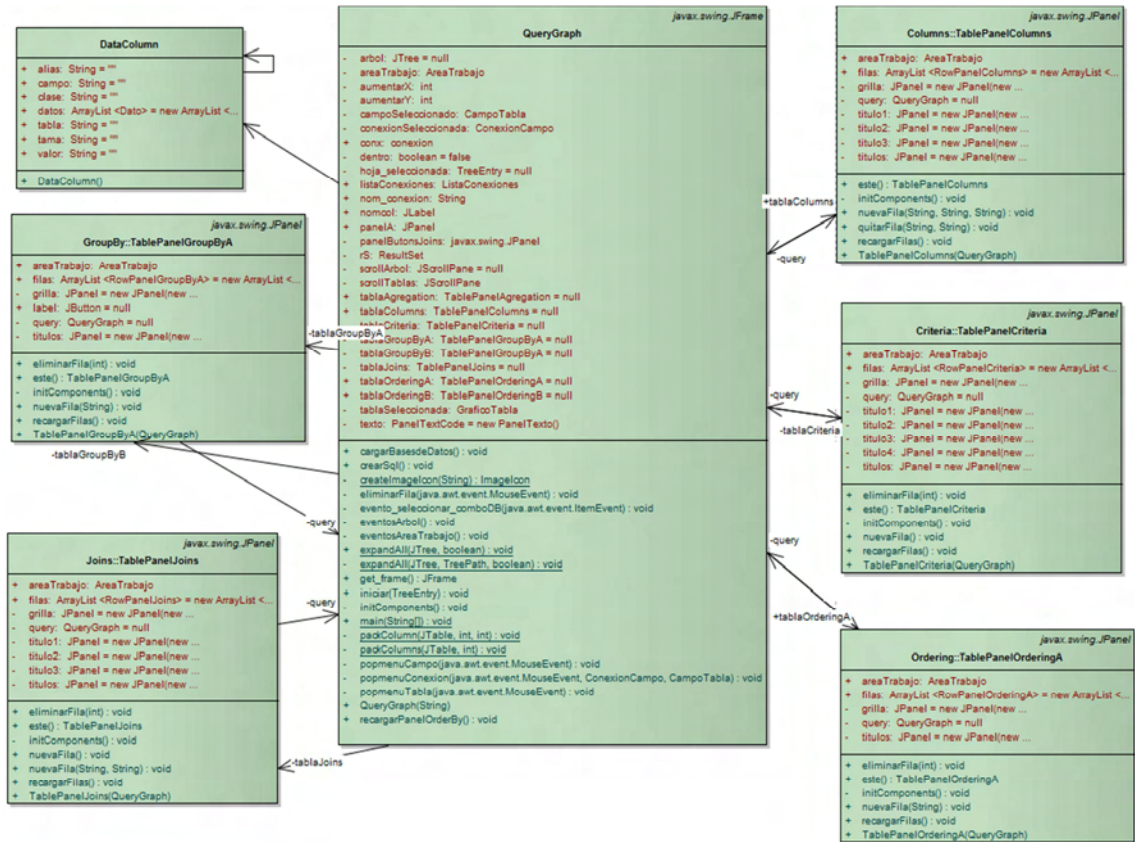
**4.3.1.4 Diagrama de Clases: Administración Lógica de las Sentencias Gráficas.** En la Figura 41, se muestra el diagrama de clases que permite la administración lógica del submódulo sentencias gráficas de Pg\_KDD.

Figura 41: Diagrama de Clases Adm. Logica Sentencias Gráficas



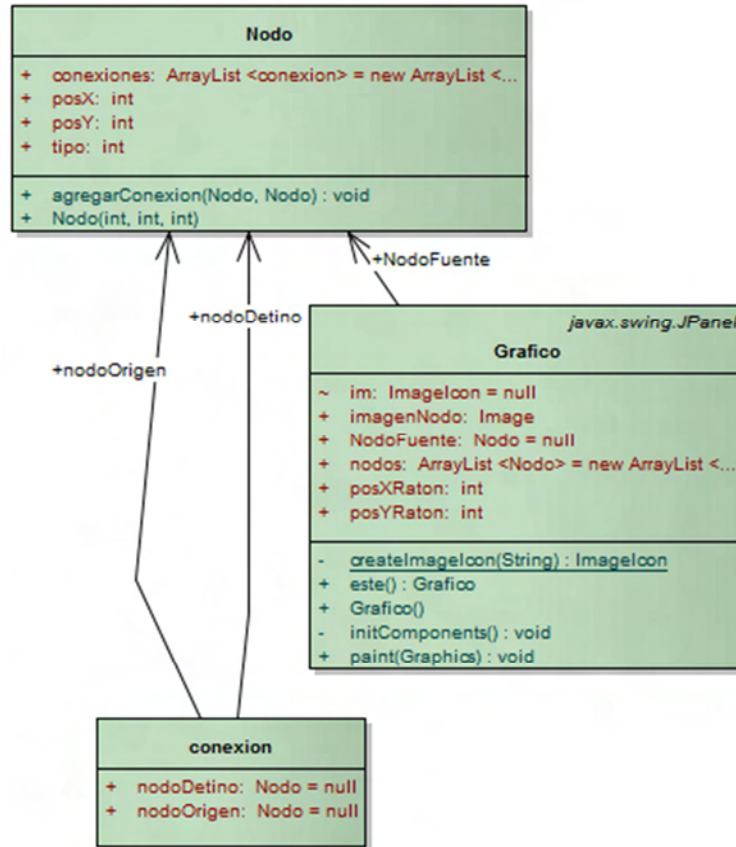
4.3.1.5 **Diagrama de Clases: Consultas Gráficas.** En la Figura 42, se muestra el diagrama de clases que permite administrar gráficamente el submódulo sentencias gráficas de Pg\_KDD.

Figura 42: Diagrama de Clases Consultas Gráficas



4.3.1.6 Diagrama de Clases: Administración Lógica KDD. En la Figura 43, se muestra el diagrama de clases que permite la administración lógica del submódulo minería de datos de Pg\_KDD.

Figura 43: Diagrama de Clases Adm. Logica KDD



#### 4.3.1.7 Diagrama de Clases: Administracion de servicios gráficos KDD.

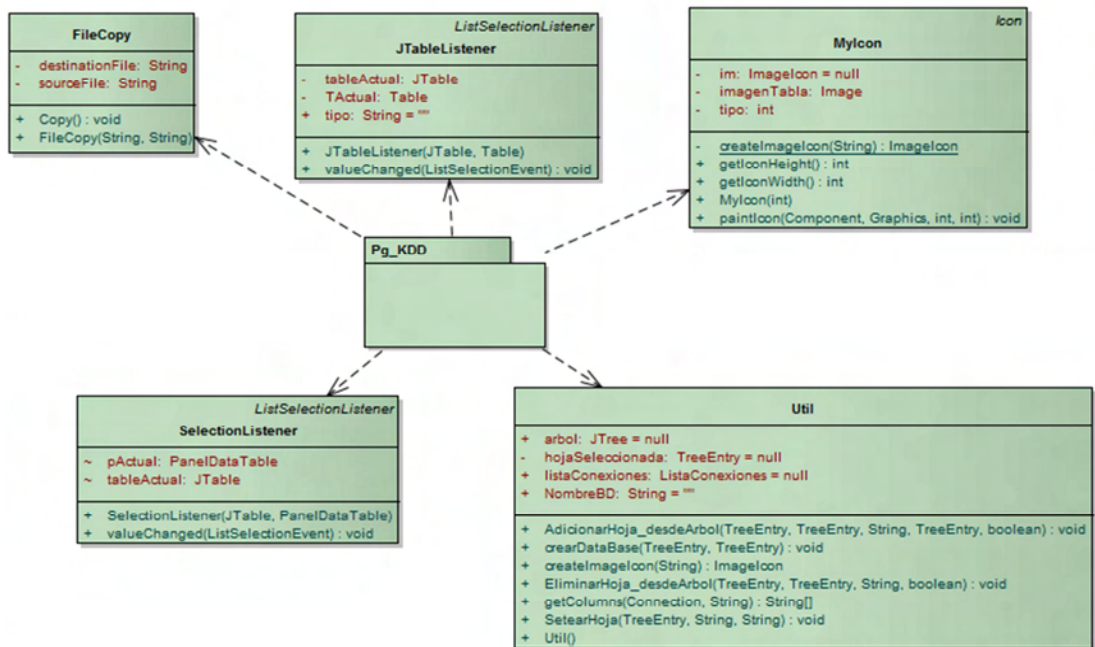
En la Figura 44, se muestra el diagrama de clases que permite la administración de los servicios gráficos del submódulo minería de datos de Pg\_KDD.

Figura 44: Diagrama de Clases Adm. de Servicios KDD



**4.3.1.8 Diagrama de Clases: Utilidades.** En la Figura 45, se muestra el diagrama de clases que son de gran utilidad dentro de la herramienta Pg\_KDD, para su uso general.

Figura 45: Diagrama de Clases Utilidades

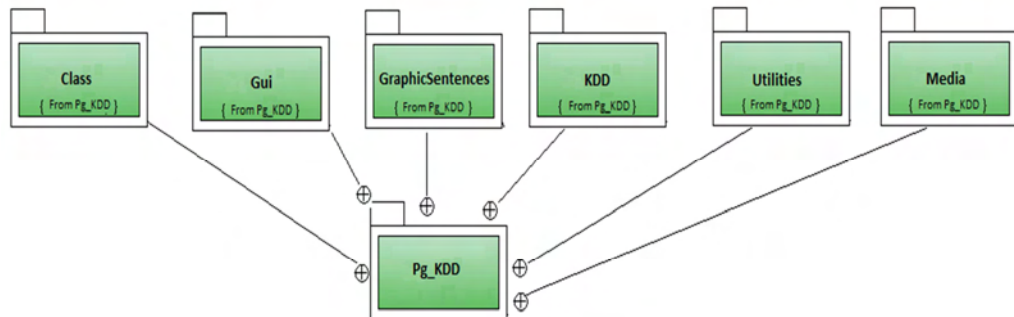


4.3.2 Diagramas de paquetes. A continuación se describen los diagramas de paquetes.

4.3.2.1 Diagrama: Paquete Principal. En el paquete principal de la herramienta Pg\_KDD, contiene los subpaquetes Class, Gui, GraphicSentences, KDD, Utilities y Media que serán descritos cada uno a continuación.

En la Figura 46, se muestra el diagrama paquetes general de la herramienta.

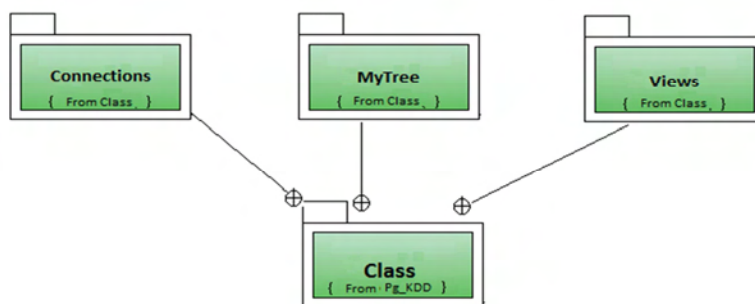
Figura 46: Diagrama de Paquetes Principal



**4.3.2.2 Diagrama: Paquete Class.** El subpaquete Class contiene los subpaquetes Connections, en el cual se encuentra las clases necesarias para el manejo de conexiones a la aplicación, MyTree, el cual contiene las clases necesarias para la administración del árbol de conexiones de la herramienta, Views, el cual contiene las clases necesarias para le manejo dinámico de los paneles de edición de archivos SQL.

En la Figura 47, se muestra el diagrama de paquetes que permite la administración lógica de Pg\_KDD.

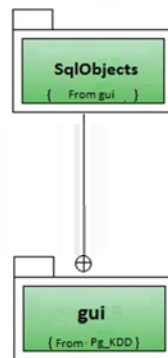
Figura 47: Diagrama de Paquetes Class



**4.3.2.3 Diagrama: Paquete Gui.** El subpaquete Gui, contiene las clases que permiten al administración gráfica de la herramienta como son ventanas emergentes, paneles de visualización de datos, marcos de visualización y también contiene el subpaquete Sql Objects, el cual contiene las clases necesarias para la administración gráfica de cada componente de una bases de datos, como son sus tablas, columnas, referencias, entre otras.

En la Figura 48, se muestra el diagrama de paquetes que permite la administración gráfica de Pg\_KDD.

Figura 48: Diagrama de Paquetes Gui



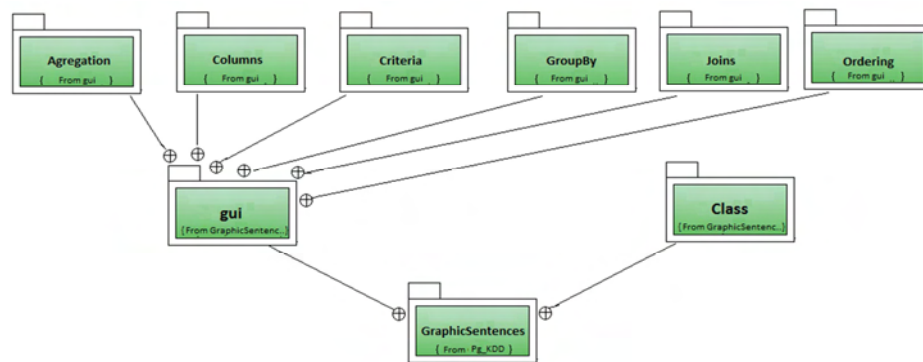
**4.3.2.4 Diagrama: Paquete GraphicSentences.** El subpaquete GraphicSentences contiene los subpaquetes, Gui, para la administración gráfica y Class, para la administración lógica.

El subpaquete Gui, contiene los subpaquetes Agregation, el cual contiene las clases necesarias para la visualización y aplicación de el concepto de funciones de agregación dentro de una consulta SQL, Columns, el cual contiene las clases necesarias para la visualización y selección de los atributos de una tabla de una base de datos, Criteria, el cual contiene las clases necesarias para la visualización y aplicación del concepto de restricción dentro de una consulta SQL, GroupBy , el cual contiene las clases necesarias para la visualización y aplicación del concepto de agrupación dentro de una consulta SQL, Joins, el cual contiene las clases necesarias para la visualización y aplicación del concepto de relación entre tablas de una base de datos dentro de una consulta SQLy Ordering, el cual contiene las

clases necesarias para la visualización y aplicación del concepto de ordenamiento de atributos dentro de una consulta SQL.

En la Figura 49, se muestra el diagrama de paquetes que permite la administración general del submódulo de sentencias gráficas de Pg\_KDD.

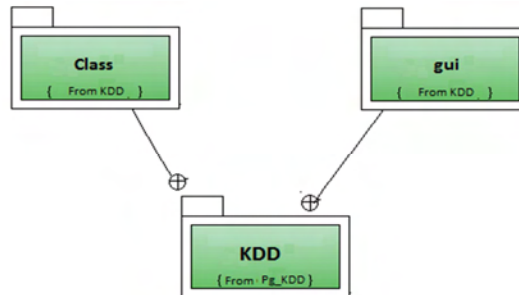
**Figura 49: Diagrama de Paquetes GraphicSentences**



**4.3.2.5 Diagrama: Paquete KDD.** El subpaquete KDD contiene los subpaquetes Class, el cual contiene las clases necesarias para la administración lógica del submódulo de minería de datos y Gui, el cual contiene las clases necesarias para la administración gráfica del módulo de minería de datos, estos dos subpaquetes permiten la aplicación de las tareas de minería de datos para el descubrimiento de conocimiento con bases de datos dentro de la herramienta.

En la Figura 50, se muestra el diagrama de paquetes que permite la administración general del submódulo de minería de datos de Pg\_KDD.

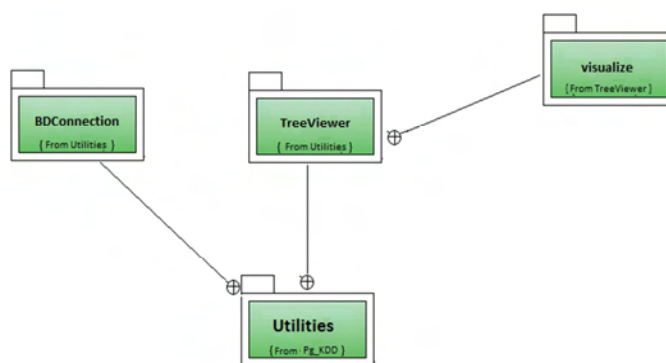
Figura 50: Diagrama de Paquetes KDD



**4.3.2.6 Diagrama: Paquete Utilities.** El subpaquete Utilities contiene los subpaquetes BDConnections, el cual contiene las clases necesarias para la conexión de la herramienta Pg\_KDD con el sistema de descubrimiento de conocimiento PostgresKDD, este contiene las clases necesarias para la administración lógica del árbol de visualización de reglas de la tarea de clasificación para el submódulo KDD, además contiene el subpaquete Visualize, el cual contiene las clases necesarias para la administración gráfica de dicho árbol.

En la Figura 51, se muestra el diagrama de paquetes que permite la administración general del submódulo utilidades de Pg\_KDD.

Figura 51: Diagrama de Paquetes Utilities

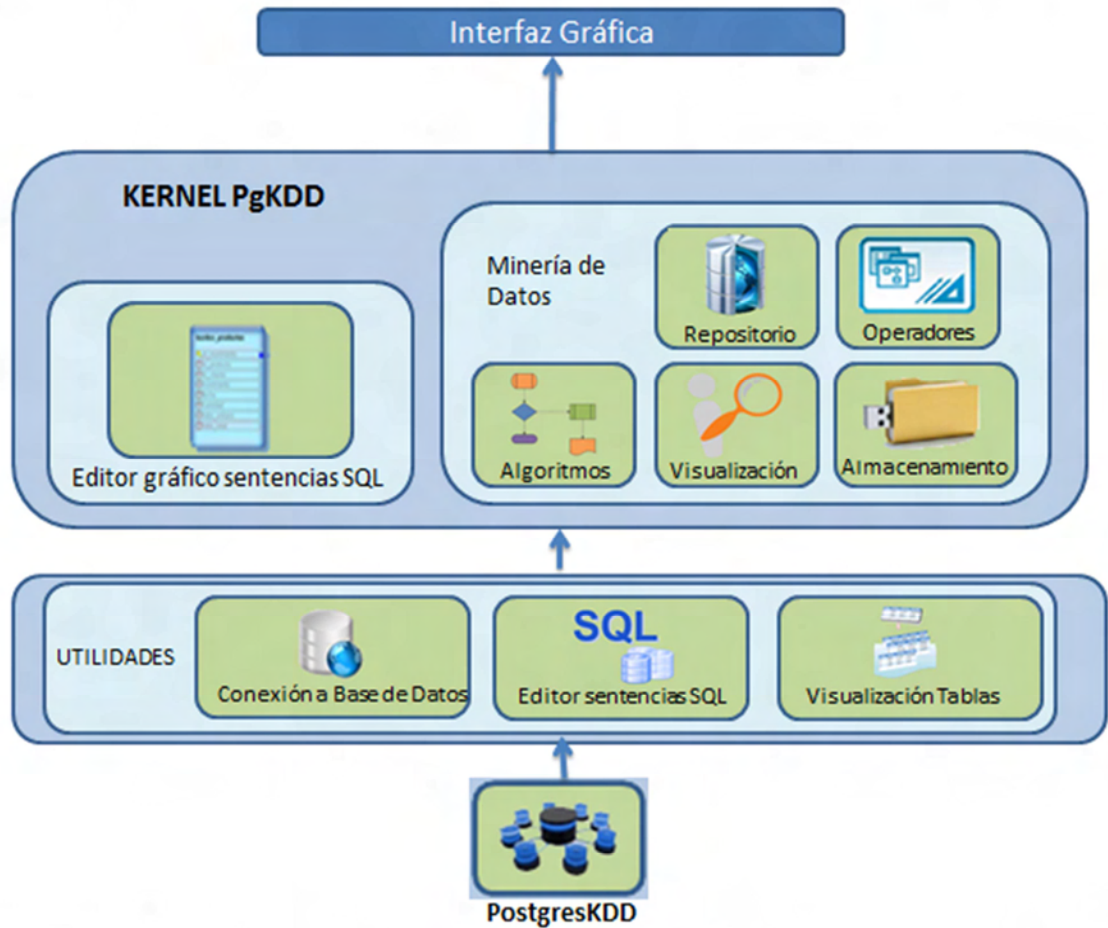


#### **4.4 ARQUITECTURA DE Pg\_KDD**

El sistema operativo sobre el cual se trabajó y se implementó Pg\_KDD es Slackware 12.0. El lenguaje de programación que se utilizó para el desarrollo de la herramienta es Java. Además se hizo uso de otras herramientas libres para el desarrollo de Pg\_KDD como NetBeans 7 como IDE de desarrollo, VirtualBox para la creación de máquinas virtuales, Gimp para edición de imágenes, entre otras.

Para el proceso de descubrir conocimiento en bases de datos, Pg\_KDD incluye las siguientes etapas: conexión a SGBD, tareas de manipulación de datos y de administración gráfica de bases de datos, proceso gráfico de minería de datos, manejo de repositorios, manejo de algoritmos y visualización de resultados. Esta estructura se indica en la Figura 52:

Figura 52: Arquitectura Pg\_KDD



**4.4.1 Módulo de Interfaz Grafica.** Este módulo da soporte visual a los módulos de utilidades, kernel Pg\_KDD, y se encarga de brindar al usuario una medio agradable y facil de usar para la aplicacion de las tareas de descubrimiento de conocimiento con bases de datos que involucren la interaccion entre los diferentes componentes de la herramienta.

**4.4.2 Módulo Gestión: KERNEL Pg\_KDD.** En este módulo reposan los paquetes primordiales para la ejecucion de las tareas de descubrimiento de

conocimiento. Este reúne los módulos de Utilidades, el módulo de Sentencias Gráficas y el módulo de Minería de Datos.

#### **4.4.2.1 Módulo Sentencias Gráficas (Editor Grafico de Sentencias SQL).**

Este módulo a través de una colección de clases y librerías permite la representación de resultados de operaciones de manipulación de datos y la administración de bases de datos de manera gráfica.

**4.4.2.2 Módulo Minería De Datos.** En este módulo se gestiona de manera gráfica los operadores, primitivas y algoritmos implementados en PostgresKDD para las tareas de minería de datos, facilitando la toma de decisiones e incrementando la eficiencia del proceso KDD.

Dentro de este módulo del sistema existen los siguientes submódulos:

- **Repositorio:** Este submódulo permite manipular los repositorios utilizados para la aplicación de las tareas de descubrimiento de conocimiento con bases de datos.
- **Operadores:** Este submódulo permite la utilización de los operadores las tareas de asociación y clasificación del proceso de descubrimiento de conocimiento en bases de datos.
- **Algoritmos:** Este submódulo permite la aplicación de los algoritmos de asociación y clasificación del proceso de descubrimiento de conocimiento en bases de datos.
- **Visualización:** Este submódulo contiene las clases necesarias para construir y desplegar las estructuras que nos permitirán visualizar de manera gráfica y dinámica los resultados de la aplicación de técnicas de minería.  
El objetivo de este módulo es permitir la interacción y comunicación entre los diferentes componentes de los módulos involucrados, así como de proveer soporte en el caso de no ser necesaria la utilización de alguno de ellos, por ejemplo, cuando una conexión del repositorio se hace directamente hacia uno de los algoritmos de minería sin pasar por algún operador para la tarea.
- **Almacenamiento:** Este submódulo permite el almacenamiento de las tareas de asociación o clasificación del proceso de descubrimiento de conocimiento con bases de datos.

**4.4.3 Módulo Utilidades.** En este módulo se mantiene una colección de clases principales y librerías que son utilizadas por otras clases a lo largo de la aplicación para el desarrollo de tareas comunes como, conexión con el sistema de descubrimiento de conocimiento postresKDD, manipulación de Scripts SQL y visualización, descritos a continuación:

- **Conexión a Bases de Datos:** Este submódulo es el encargado de la conexión a una fuente de datos. El objetivo de este módulo es proveer al usuario de herramientas amigables para la selección y construcción de una vista o conjunto de datos minables.
- **Editor Sentencias SQL:** Este submódulo es el encargado de la administración de los scripts SQL. El objetivo de este módulo es proveer al usuario de herramientas amigables para la selección, edición y ejecución de scripts SQL que estén dentro de su disco duro o simplemente la creación de uno nuevo.
- **Visualización de Tablas:** Este submódulo contiene las clases necesarias para construir y desplegar las estructuras que nos permitirán visualizar de manera gráfica y dinámica el contenido de las tablas existentes dentro de una base de datos.

## 4.5 ARQUITECTURA DE PAQUETES Y CLASES

A continuación se describen de manera general la estructura de paquetes de Pg\_KDD y las clases que pertenecen a cada paquete, dentro de ellas se describen los métodos implementados.

### 4.5.1 Utilidades

#### Pg\_KDD

Este es el paquete principal del proyecto, contiene la siguiente clase:

**Clase GUI:** Esta es la Clase principal del aplicativo que despliega la GUI principal, sus métodos son:

- **GUI ()** este es el constructor de la clase y se encarga de instanciar todo el fríame de la GUI principal del proyecto Pg\_KDD
- **iniciar\_controles ()** este método consiste en iniciar todos los controles de la interfaz principal.
- **crearArbol ()** este método instancia el árbol de conexión creando un evento para diferenciar los pop menús que se ejecutaran en cada tipo de hoja del árbol y así llamando a los pop menús de cada una de ellas.
- **AdicionarHoja\_ desdeArbol ()** este método es un método recursivo y adiciona una hoja creada desde los pop menús generados para cada tipo de hoja del árbol de conexión.
- **Abrir\_conexiones ()** este método se ejecuta al iniciar el aplicativo y carga todas las conexiones guardadas previamente en un archivo plano.
- **gráficar\_conexiones ()** este método gráfica en el árbol de conexiones, las conexiones guardadas en el archivo plano nombrado anterior mente llamando al método crear\_hojas de la clase IconTreeModel del Paquete Class.MyTree, si ocurre algún percance en el servicio de PostgreSQL instalado en el sistema y o ahí algún error en una conexión guardada este generara el TreeEntry pero con un icono de ERROR de carga.
- **createToolBar ()** este método crea la barra de herramientas, instanciando cada uno de sus botones asociados a ella.
- **createMenuBar ()** este método crea la Barra de Menús de la cabecera del aplicativo, instanciando cada uno de sus Menús y Menú Ítems asociados a ella.
- **eventosToolBar ()** este método trabaja todos los eventos relacionados con todos los botones asociados al ToolBar.
- **eventosMenuBar ()** este método trabaja todos los eventos relacionados con todos los menús asociados al Menú Bar, dentro de él se encuentra la creación de los “Views”, así llamados a las vistas que están contenidas en el TabView estos términos son generales del manejo de la librería Infonode que ayuda a mejorar la interfaz de usuario.
- **eventosFrame ()** este método trabaja todo lo relacionado con los eventos del GUI principal, cuando se cierra el FRAME: sobrescribe un archivo plano guardando todas la conexiones existentes en el aplicativo previamente creadas, al igual que con el mismo evento relacionado, guarda cada uno de los “Views”, de tipo “SQL Script” y “Archivo Script”, en un archivo \*.SQL respectivamente, que en un caso puede ya haber sido creado al instanciar un “View” y haber accionado el botón “guardar” del ToolBar o el Ítem “Guardar”

del Menú File del Menú Bar en donde cambia un estado de “Guardado”, del Panel contenido en el View, a true.

- **eventosArbol ()** este método trabaja todo lo relacionado con los eventos referentes a la selección de cada tipo de hoja del Árbol de conexión, en la ejecución de estos eventos se podrá visualizar las siguientes funciones:
  - Actualización el estado de habilitación o de estructura de algunos objetos tanto del Menú Bar como de la barra de herramientas (ToolBar).
  - Instanciar las interfaces de creación para cada tipo de hoja seleccionada, Tablas, Bases de Datos, Restricciones etc.
  - Visualizar el “View Datos” cuyo panel Contenido es de tipo “Panel Datos”, este será visualizado solo para las hojas seleccionadas de tipo “tabla” o “campo”, este “View Datos”, no se instanciara varias veces solo se actualiza su panel contenido dependiendo de la hoja tipo “tabla” seleccionada del Árbol de conexión.

En la parte final de la clase “GUI” se encuentran los métodos utilizados de la librería INFONODE para la manipulación y automatización de la interfaz Gráfica.

**Clase SplashScreen:** Esta clase es la que instancia la interfaz de bienvenida de la aplicación Pg\_KDD.

**Clase BarraProgreso:** Esta clase es la muestra una barra de progreso al recargar el árbol de conexiones.

## **Class**

Este paquete hace referencia a la parte lógica del aplicativo en cuanto a manejo del árbol de conexión y manejo de conexiones de la aplicación, contiene los siguientes paquetes:

### **MyTree**

Sus clases son:

**Clase TreeEntry:** Esta clase describe un nuevo tipo de dato, el cual va a ser objeto de relación para cada una de las hojas del árbol de conexión.

Sus métodos son acordes a Set () y Get () para cada uno de sus atributos, y también referentes a inserción, eliminación u Obtención de nodos hijo para un objeto de este tipo, entre otros.

**Clase IconTreeRenderer:** Esta clase tiene como función colocarle, a un TreeEntry u hoja del árbol de conexión, un color diferente para su título cuando esta es seleccionada, su metodo principal es.

- **getTreeCellRendererComponent ()** Este método tiene como función colocar el color de título diferente al TreeEntry cuando es seleccionada.

**Clase IconTreeModel:** Esta clase se encarga de crear todo el modelo a la instancia de un JTree de Java, en el caso del aplicativo se consume esta clase en las clases GUI y QueryGraph, sus métodos son:

- **IconTreeModel ()** es el constructor de la clase y es vital a la hora de instanciar el modelo del árbol ya que crea la primera hoja del árbol de conexión.
- **crear\_hojas ()** este método se encarga de crear todo el modelo del árbol de conexión de la clase "GUI" de acuerdo a la estructura de los objetos que contenga el usuario que crea la nueva conexión, en este caso serán las todas bases de datos relacionadas a él con sus respectivos objetos que se relacionan con cada una de ellas (Base de Datos).
- **crear\_hojas\_QueryGraph ()** este método crea el modelo del árbol instanciado para la clase "QueryGraph", en donde crea como hoja principal el nombre de la bases de datos seleccionada en un JComboBox y con base a esta selección genera hojas relacionadas a las tablas de esta base de datos.
- **getRoot ()** este método retorna el TreeEntry raíz del árbol.
- **getChild ()** este método retorna un TreeEntry hijo de un TreeEntry en específico el cual se lo manda por parámetro y un índice que se refiere a una posición en la lista de los TreeEntrys hijos.
- **getChildCount ()** retorna el número de hijos que contiene un TreeEntry.
- **isLeaf ()** este método retorna un booleano verificando como verdadero o falso si el TreeEntry contiene o no hijos.
- **getIndexOfChild ()** este método retorna la posición en que se encuentra un TreeEntry hijo con respecto a un TreeEntry padre.

- **addTreeModelListener ()** este método adiciona un Escuchador al modelo del de esta clase.
- **removeTreeModelListener ()** este método Elimina el Escuchador adicionado al modelo de esta clase.

## Connections

Sus clases son:

**Clase ConexionArbol** Esta clase describe el objeto conexión que será instanciado cada vez que se genere una nueva conexión al aplicativo.

sus métodos son acordes a Set () y Get () para cada uno de sus atributos, nombre, usuario, contraseña, host y puerto que representan generalmente a una conexión.

**Clase ListaConexiones** esta clase maneja una lista de objetos tipo ConexionArbol, en donde se gestiona las conexiones del aplicativo.

Sus métodos radican en la administración de las conexiones, Adición, Eliminación y Búsqueda.

## Views

**Clase DatosView:** Esta clase describe el objeto tipo View dinámico que se instanciará cada vez que se cree un “KDD Editor” o un Archivo de edición, la diferencia entre estos dos tipos de instancia es el atributo “tipo panel” en el panel contenido “Panel Script”.

**Clase ListaViews:** Esta clase administra una lista de objetos tipo DatosView, en la cual se gestiona las Views dinámicas instanciadas.

Sus métodos radican en la administración de los Views del área de gestión de pestañas o TabView del aplicativo con la funcionalidad de adición, eliminación y búsqueda.

## 4.5.2 Kernel Pg\_KDD: Sentencias gráficas

### GraphicSentences

Este paquete hace referencia a la administración de las sentencias gráficas que están directamente relacionadas con el GUI de la aplicación principal, contiene los siguientes paquetes:

#### Class

Este paquete administra la parte lógica o de razonamiento de las sentencias gráficas, contiene las siguientes clases:

**Clase Área de Trabajo:** Esta clase administra todos los elementos gráficos incluidos en ella como tablas, campos y conexiones, la lógica es sencilla de esta clase, contiene una lista de tablas inicializadas a *null* en su comienzo, y la tabla contiene una lista de campos, y estos últimos tienen una lista de conexiones, sus métodos son:

- **AreaTrabajo ():** Este método es el constructor de la clase y se encarga de inicializar el área de trabajo con parámetros de ancho y alto.
- **Paint ():** Este método se encarga de pintar todos elementos gráficos existentes en el área de trabajo, como tablas con sus respectivos campos y conexiones.
- **insertarTabla ():** Este método se encarga de agregar una nueva tabla al área de trabajo, la cual hace una sentencia SQL extrayendo el número de campos y su nombre en un ResultSet los cuales se le agregaran a la lista de la tabla creada, y a estos últimos les agrega tipo de llave si es foránea o primaria para el momento de su dibujo se aplique un icono correspondiente.
- **Buscar ():** Este método tiene como función la búsqueda de una tabla en las existentes dentro de las lista de tablas del área de trabajo.
- **Eliminar ():** Este método tiene como función eliminar una tabla del área de trabajo, internamente elimina también las conexiones que sean parte de esta tabla.

**GraficoTabla:** Esta clase describe el objeto tipo tabla del área de trabajo, sus métodos son los siguientes:

- **insertarCampo ()**: Este método tiene como función insertar un campo a la tabla.
- **setCoordenadas ()**: Este método se encarga de setear las coordenadas a la tabla dentro del área de trabajo, estas coordenadas son tomadas de la posición final del mouse.

**Clase CampoTabla:** Esta clase describe el objeto tipo campo de una Tabla del área de trabajo, sus métodos son los siguientes:

- **EliminarConexion ()**: Este método tiene como función eliminar una conexión del campo, recorriendo los campos que la contengan.
- **insertarConexion ()**: Este método tiene como función insertar una conexión al campo seleccionado.

**Clase ConexionCampo:** Esta clase describe el objeto tipo conexión que está relacionado con un campo en específico, sus métodos son los siguientes:

- **ecuacionRecta ()**: Este método se encarga de calcular la ecuación de la recta tomando los puntos, inicial y final de la conexión campo a campo de su respectiva tabla.
- **calculoDistancia ()**: Este método se encarga del cálculo de la distancia entre dos puntos.
- **estaEncima ()**: Este método se encarga de averiguar, si el mouse al accionar su evento de clic sobre el área de trabajo, se encuentra a una mínima distancia en donde se encuentra una conexión dibujada.

## GUI

Este paquete Administra la parte gráfica de las sentencias gráficas, contiene las siguientes clases:

**Clase QueryGraph:** Esta clase se encarga de administrar todo el fríame gráfico correspondiente a las sentencias gráficas, sus métodos son los siguientes:

- **Iniciar ()**: Este método se encarga de inicializar los controles utilizados para el control interno de las sentencias gráficas.
- **cargarBasesdeDatos ()**: Este método se encarga de cargar todas las bases de datos al JComboBox de la interfaz del usuario que instancio

QueryGraph, este usuario fue obtenido de la hoja seleccionada de del árbol de conexión del fríame principal.

- **eventosArbol ()**: Este método se encarga de los eventos realizados en el árbol que contiene las tablas de la base de datos seleccionada.
- **eventosAreaTrabajo ()**: Este método se encarga de los eventos realizados en el área de trabajo correspondiente a llamada de los pop menús de cada objeto tabla, campo y conexión.
- **realizarDibujo ()**: Este método se encarga de dibujar las conexiones gráficas provenientes de las conexiones manuales del **Panel de Administración manual de las sentencias**.
- **crearSql ()**: Este método se encarga de la creación del Código SQL que se visualiza en el "SQL Statement", este es llamado cada vez que se ejecuta un evento de realización de sentencia SQL.

Este paquete a su vez contiene los siguientes paquetes:

## Joins

Este paquete describe el manejo de uniones o relaciones entre las tablas de una base de datos, contiene las siguientes clases:

**Clase RowPanelJoins**: Esta clase contiene las características gráficas del objeto que representa una fila en una tabla del panel Joins, el metodo más representativo de la clase es:

- **programarEventos ()**: En este método se programa todos los eventos de los controles que contiene el panel de creación de JOINS manuales.

**Clase TablePanelJoins**: Esta clase administra las filas o los objetos tipo RowPanleJoins, contiene un metodo clave para la adición de nuevas filas a la tabla de uniones:

- **nuevaFila ()**: Este método se encarga de agregar Objetos tipo RowPanleJoins a la TablePanelJoins asignandole su posición individual en memoria.

## Columns

Este paquete describe el manejo las columnas incluidas dentro de la relación gráfica.

## **Criterio**

Este paquete describe el manejo de las restricciones que se realizan en la consulta SQL, trayendo consigo la utilización de los operadores OR o AND según sea el caso.

## **GroupBy**

Este paquete describe el concepto de agrupamiento dentro de una sentencia SQL, utilizando controles gráficos para su mejor entendimiento.

## **Ordering**

Este paquete describe el concepto de ordenamiento dentro de una sentencia SQL, utilizando controles gráficos para su mejor entendimiento.

## **Agregation**

Este paquete describe el concepto de agregación en donde se utiliza a las funciones agregadas y se las restringe, con el fin de ayudar en la mejora de la sentencia SQL que se esta realizando.

### **4.5.3 Kernel Pg\_KDD: Módulo Minería de Datos**

#### **KDD**

Este paquete hace referencia a la administración del módulo de minería de datos, en el cual se maneja tanto la parte gráfica como la parte lógica, contiene los siguientes paquetes:

#### **CLASS**

Este paquete describe el manejo de la parte lógica del módulo de minería de datos, contiene las siguientes clases:

**Clase Grafico:** Esta clase Administra todos los elementos gráficos incluidos en ella como son los nodos y las conexiones entre ellos, la lógica es sencilla de esta

clase, contiene una lista de nodos inicializadas a null en su comienzo, cada nodo contiene una lista de conexiones, sus métodos son:

- **Paint ()**: Este método se encarga de pintar todos elementos gráficos existentes en el área de trabajo, como como son los nodos con sus respectivas conexiones.
- **EliminarNodo ()**: Este método tiene como función eliminar un nodo del área de trabajo (gráfico), internamente elimina también las conexiones que sean parte de él.

**Clase Nodo**: Esta Clase describe el objeto tipo nodo del área de trabajo, sus métodos son los siguientes:

- **Nodo ()**: Este método es el constructor de la clase y se encarga de inicializar el nuevo nodo con parámetros de x,y como coordenadas de su posición en dentro de el área.
- **agregarConexion ()**: Este método se encarga de agregar una nueva conexión al nodo seleccionado, determinando el nodo destino con quien se va conectar.
- **EliminarConexion ()**: Este método tiene como función eliminar la conexión del nodo seleccionado.

**Clase conexionKDD**: Esta clase describe el objeto tipo conexión que está relacionado con un nodo en específico, sus métodos son los siguientes:

- **ecuacionRecta ()**: Este método se encarga de calcular la ecuación de la recta tomando los puntos, inicial y final de la conexión nodo a nodo de su respectiva tabla.
- **calculoDistancia ()**: Este método se encarga del cálculo de la distancia entre dos puntos.
- **estaEncima ()**: Este método se encarga de averiguar, si el mouse al accionar su evento de clic sobre el área de trabajo (gráfico), se encuentra a una mínima distancia en donde se encuentra una conexión dibujada.

**Clase TipoNodo**: Esta clase se encarga de declarar el tipo de nodo que va a ser agregado al área de trabajo (gráfico), ejemplo: REPOSITORIO, ASSOCIATOR, ASSOROW, EQUIPASO, etc, esta clase no contiene métodos solo un atributo de asignación.

**Clase EstadoNodo:** Esta clase se encarga de declarar el estado de nodo que va a ser agregado al área de trabajo, ejemplo (gráfico): CREADO, CONFIGURADO, CORRIENDO, etc, esta clase no contiene métodos solo un atributo de asignación.

## GUI

Este paquete describe el manejo de la parte gráfica del módulo de minería de datos, contiene las siguientes clases:

**Clase FrameKDD:** Esta clase se encarga de administrar todo el fríame gráfico correspondiente a la parte de minería de datos, sus métodos principales son los siguientes:

- **abrir():** Este método se encarga de abrir un archivo cuya extensión es \*.KDDg y cargar una previa configuración que el usuario haya guardado de los enlaces entre nodos.
- **guardar():** Este método se encarga de guardar un archivo cuya extensión es \*.KDDg en donde se encuentre la actual configuración de enlaces entre nodos en el área de trabajo.
- **crearPopmenu():** Este método se encarga de crear los popmenús para cada uno de los nodos.
- **verificarConexiones():** Este método se encarga de verificar que las conexiones que dese hacer el usuario sean las correctas, mostrando con mensajes de error emergentes las conexiones incorrectas.

**Clase ConfigurationRepository:** Esta clase se encarga de configurar el repositorio seleccionado por el usuario, el cual será utilizado a lo largo del proceso KDD.

- **Iniciar():** Este método se encarga de inicializar los controles utilizados para el control interno de las sentencias gráficas.
- **cargarDatos():** Este método se encarga de cargar los datos de la tabla repositorio a la tabla de visualización.
- **cargarCampos():** Este método se encarga de cargar los campos de la tabla repositorio a la tabla de visualización.

- **crear\_tabla\_repositorio():** Este método se encarga de cargar de crear la tabla `repositorio_nomtabla` tomando las columnas seleccionadas por el usuario, generando un script SQL y ejecutándolo para su creación.

**Clase ConfigurationAssociator:** Esta clase se encarga de configurar el operador `associator` el cual será utilizado a lo largo del algoritmo de asociación.

- **btnSaveActionPerformed():** Este es un método que ejecuta el evento de un botón el cual toma los valores de los controles gráficos y genera un script SQL relacionado con la sintaxis del operador `Associator`.

**Clase ConfigurationAssorow:** Esta clase se encarga de configurar el operador `assorow` el cual será utilizado a lo largo del algoritmo de asociación.

- **btnSaveActionPerformed():** Este es un método que ejecuta el evento de un botón el cual toma los valores de los controles gráficos y genera un script SQL relacionado con la sintaxis del operador `Assorow`.

**Clase ConfigurationEquipaso:** Esta clase se encarga de configurar el operador `equipaso` el cual será utilizado a lo largo del algoritmo de asociación.

- **btnSaveActionPerformed():** Este es un método que ejecuta el evento de un botón el cual toma los valores de los controles gráficos y genera un script SQL relacionado con la sintaxis del operador `Equipaso`.

**Clase ConfigurationAssociatorRules:** Esta clase se encarga de configurar los parámetros de la función `describe_associator_rules` la cual será utilizada para la obtención de las reglas de asociación.

- **btnSaveActionPerformed():** Este es un método que ejecuta el evento de un botón el cual toma los valores de los controles gráficos y genera un script SQL relacionado con la sintaxis de la función definida por el usuario `describe_associator_rules`.

**Clase ConfigurationMate:** Esta clase se encarga de configurar el operador `Mate` el cual será utilizado a lo largo del algoritmo de clasificación.

- **btnSaveActionPerformed():** Este es un metodo que ejecuta el evento de un botón el cual toma los valores de los controles gráficos y genera un script SQL relacionado con la sintaxis del operador Mate.

**Clase TableView:** Esta clase se encarga de visualizar el resultado que genera la utilización de los operadores o funciones definidas por usuario.

- **crear\_tablas():** Este método se encarga de seleccionar que sentencia sql debe tomar según sea el nodo conectado a él, ya sea operador o función definida por el usuario.
- **crear\_tabla\_view():** Este método se encarga de cargar los datos dependiendo del Script SQL que contenga el nodo conectado y crear un nuevo objeto JTable para su posterior visualización.
- **crearTabla():** Este método se encarga modificar el modelo del objeto JTable para una mejor visualización.

**Clase RulesView:** Esta clase se encarga de visualizar las reglas generadas por la utilización de las funciones definidas por usuario, dependiendo si son del proceso de asociación o de clasificación.

- **crear\_tablas():** Este método se encarga de seleccionar que sentencia sql debe tomar según sea el nodo conectado a él, y dependiendo de que función definida por el usuario (`describe_associator_rules` para asociación ó `arm_rules` para clasificación) se visualizara la relación de interpretación y/o las reglas obtenidas de ella.
- **crear\_tabla\_associator():** Este método se encarga de cargar los datos dependiendo del Script SQL que contenga el nodo conectado y crear un nuevo objeto JTable y/o un JTextArea describiendo las reglas.
- **crearTabla():** Este método se encarga de modificar el modelo de el objeto JTable para una mejor visualización.

#### 4.5.4 Módulo Interfaz Grafica

##### GUI

Este paquete hace referencia al manejo de gráfico de la aplicación aquí se encuentran todas las interfaces gráficas que hacen parte de Pg\_KDD, este tiene las siguientes Clases contenidas:

**Clase Connection\_settings:** Esta clase es una interfaz gráfica que tiene como función la creación de una nueva conexión que se graficará en el modelo del árbol de conexiones, sus métodos son:

- **Iniciar():** Este método tiene como función cargar todas las conexiones existentes a un JList de Java, nótese que la ejecución de este método se hace internamente.
- **CloseDialog():** Este evento tiene como función cerrar la Ventana y liberarla en memoria aplicando el método nativo de Java dispose ().
- **setVisible():** Aquí se aplica una sobrescritura al método básico setVisible de la clase JFrame de Java, la función de este método es refrescar el panel que contiene los diversos objetos funcionales de la ventana.
- **btn\_TestActionPerformed():** Este método es la ejecución del evento del botón Test de la ventana cuya función es testear la conexión con PostgreSQL con los elementos capturados de la interfaz correspondientes al nombre de usuario y contraseña.
- **btn\_GuardarActionPerformed():** Este método es la ejecución del evento del botón Guardar de la ventana cuya función es guardar la conexión digitada por el usuario en la lista de visualización, validando existencia del nombre de la conexión ingresada, del usuario.
- **btn\_LimpiarActionPerformed():** Este método es la ejecución del evento del botón Limpiar de la ventana cuya función es borrar todos los campos de ingreso de la interfaz.
- **btn\_CancelarActionPerformed():** Este método es la ejecución del evento del botón Cancelar de la ventana cuya función es la cancelación de la acción crear conexión.
- **btn\_AyudaActionPerformed():** Este método es la ejecución del evento del botón Ayuda de la ventana cuya función es llamar al blog donde está toda la documentación del manual de usuario del aplicativo o llamar a una interfaz donde le brinde explicación sobre la utilización de esta interfaz.

- **btn\_ConectarActionPerformed():** Este método es la ejecución del evento del botón Conectar de la ventana cuya función es realizar la nueva conexión del aplicativo he invocar a la clase IconTreeModel para gráficar el modelo de la nueva conexión cargando todos los elementos de Bases de datos existentes en el árbol de conexión.

**Clase PanelTexto:** Esta clase tiene como objetivo la aplicación de estilo a los Paneles cuyo contenido sea un texto con palabras reservadas SQL o de lenguaje procedural etc. Este panel tendrá varias instancias por ejemplo en el View dinámico “KDD Editor” el cual contiene un Panel de edición de sentencias SQL, así mismo con el “Lienzo SQL” del panel Conexiones, sus métodos son:

- **isGuardado():** Este método se encarga de averiguar el estado del atributo guardado del panel, este método es muy importante en el manejo de Views cuyo panel interno es de tipo “SQL Script” donde se controla este estado para poder guardar el contenido del texto en un archivo \*.SQL si así lo desea el usuario o sobrescribirlo si ya se lo ha hecho.
- **setGuardado():** Este método se encarga de setear el estado del atributo guardado descrito anteriormente.
- **IniciarVariables():** Este método se encarga de iniciar las variables de estilo para cada palabra reservada.
- **ProgramarEventos():** Este método se encarga de ejecutar el evento de presionar una tecla cualquiera y aplicarlo al panel que contiene el texto de edición, llamando así a la función “aplicarEstilo” la cual se ejecuta directamente con cada palabra del texto.
- **aplicarEstilo():** Siendo consecuente con la explicación del anterior método, el actual aplica el estilo a las palabras reservadas, en un comienzo llamando al método “Tokenizar”, para luego ir pintando cada palabra con su correspondiente color, así mismo se aplica estilo a los comentarios comunes en un editor de texto en este caso se accionar usando en su comienzo los caracteres “- -“ del mismo modo se aplica estilo color rojo a los caracteres que estén dentro de comillas, esto para diferenciar un nombre de un campo o de una celda en una tabla de PostgreSQL ejemplo: **“SELECT \* FROM Tabla1,Tabla2 WHERE Tabla1.campo1 LIKE ‘nombre’ – sentencia SQL”**.
- **Tokenizar():** este método es el encargado de separar el texto por palabras y retornar una matriz de Strings cuyo contenido es cada palabra con su posición inicial y final dentro del texto.

**Clase PanelDataTable:** En esta clase se instancia al momento en que se ha instanciado el View estático "ViewDatos" accionado cuando se selecciona una hoja del árbol de conexión cuyo tipo es "tabla", este panel contiene internamente un JTable en el cual se contendrán todos los datos de la tabla seleccionada y la descripción de la misma, también contiene unos botones de ejecución ya sea de inserción, eliminación o simplemente de reinicio que se aplican a la tabla.

Sus métodos son:

- **Iniciar ():** Este método instancia una conexión a PostgreSQL con la base de datos, el host y el puerto obtenidos de la hoja seleccionada de la árbol de conexión.
- **SetTipo\_panel():** Este método se encarga de setear el atributo "tipo panel" de la clase.
- **GetTipo\_panel():** Este método se encarga de obtener el valor del atributo "tipo panel" de la clase.
- **SetListaconexiones():** Este método se encarga de setear el atributo "ListaConexiones" de la clase.
- **GetListaconexiones():** Este método se encarga de obtener el valor del atributo "ListaConexiones" de la clase.
- **crear\_tablas():** Este método se encarga de cargar los datos de la tabla seleccionada del árbol de conexión al JTable, validando su edición con respecto a la existencia de llave primaria de la misma forma como lo hace internamente PostgreSQL.
- **eventosTabla():** Este método se encarga de adicionarle los eventos al JTable y crear el pop menú cuyos ítems de selección son la exportación de los datos a una página en HTML o un archivo de extensión csv, y también ejecutando el evento de tecla presionada para la post-modificación de la tabla llamando así al método "actualizarDB".
- **actualizarDB():** Este método se encarga de ejecutar las sentencias SQL correspondientes a la Inserción y modificación que se le realicen al Jtable.

**Clase PanelScript:** Esta clase se instancia cada vez que se instancia un View dinámico "KDD Editor", internamente contiene un objeto tipo PanelTexto descrito anteriormente y otro panel el cual contiene un Tabpanel que contiene dos pestañas, una de ellas visualiza el resultado textual de PostgreSQL con respecto a las sentencias SQL escritas en el PanelTexto y la otra pestaña describirá el

resultado de datos si es el caso de una sentencia que arroje como resultado un "ResultSet".

Sus métodos competen a Get () y Set () de sus atributos de clase.

El paquete GUI contiene el siguiente paquete:

## **SqlObjects**

**Clase DataBase:** Esta clase se instancia cada vez que se crea o modifica una base de datos, esta contiene un fríame en el cual se encuentra el nombre de la base de datos y una instancia de PanelTexto con el fin de visualizar el código sql generado para el objeto este varía dependiendo si es creación o modificación, sus métodos son los siguientes:

**btn\_OKActionPerformed():** Este es un metodo que ejecuta el evento de un botón el cual da como resultado la creación o modificación de la base de datos.

**Clase Table:** Esta clase instancia cada vez que se crea o modifica una tabla de base de datos, esta contiene un fríame en el cual se encuentra el nombre de la tabla y una instancia de PanelTexto con el fin de visualizar el código sql para el objeto este varia dependiendo si es creación o modificación, sus métodos son los siguientes:

**btn\_OKActionPerformed():** Este es un metodo que ejecuta el evento de un botón el cual da como resultado la creación o modificación de la tabla.

**Clase Column:** Esta clase instancia cada vez que se crea o modifica una columna de una tabla, esta contiene un fríame en el cual se encuentra el nombre de la columna y una instancia de PanelTexto con el fin de visualizar el código sql para el objeto este varia dependiendo si es creación o modificación,, sus métodos son los siguientes:

**btn\_OKActionPerformed():** Este es un metodo que ejecuta el evento de un botón el cual da como resultado la creación o modificación de la columna.

**Clase PrimaryKey:** Esta clase instancia cada vez que se crea o modifica una llave primaria, esta contiene un fríame que describe a la nueva o antigua llave

primaria con una instancia de PanelTexto con el fin de visualizar el código sql generado para el objeto, sus métodos son los siguientes:

**btn\_OKActionPerformed():** Este es un metodo que ejecuta el evento de un botón el cual da como resultado la creación o modificación de la llave primaria.

**Clase ForeignKey:** Esta clase instancia cada vez que se crea o modifica una llave foránea, esta contiene un fríame que describe a la nueva o antigua llave foránea con una instancia de PanelTexto con el fin de visualizar el código sql para el objeto, sus métodos son los siguientes:

**btn\_OKActionPerformed():** Este es un metodo que ejecuta el evento de un botón el cual da como resultado la creación o modificación de la llave foránea.

**Clase Function:** Esta clase instancia cada vez que se crea una función definida por el usuario, esta contiene un fríame que describe a la nueva funcion con una instancia de PanelTexto con el fin de visualizar el código sql generado para el objeto, sus métodos son los siguientes:

**btn\_OKActionPerformed():** Este es un metodo que ejecuta el evento de un botón el cual da como resultado la creación de la función.

**Clase Rule:** Esta clase instancia cada vez que se crea una regla, esta contiene un fríame que describe a la nueva regla con una instancia de PanelTexto con el fin de visualizar el código sql generado para el objeto, sus métodos son los siguientes:

**btn\_OKActionPerformed():** Este es un metodo que ejecuta el evento de un botón el cual da como resultado la creación de la regla.

**Clase Trigger:** Esta clase instancia cada ves que se crea un disparador, esta contiene un fríame que describe al nuevo disparador con una instancia de PanelTexto con el fin de visualizar el código sql generado para el objeto, sus métodos son los siguientes:

**btn\_OKActionPerformed():** Este es un metodo que ejecuta el evento de un botón el cual da como resultado la creación del disparador.

**Clase Vista:** Esta clase instancia cada vez que se crea una vista, esta contiene un fráme que describe la vista con una instancia de PanelTexto con el fin de visualizar el código sql generado para el objeto, sus métodos son los siguientes:

**btn\_OKActionPerformed():** Este es un metodo que ejecuta el evento de un botón el cual da como resultado la creación de la vista.

## 5. NUEVAS IMPLEMENTACIONES EN POSTGRESKDD

En el módulo medianamente acoplado de PostgresKDD en la ruta *.. contrib/KDD/clasificacionPg\_KDD/inicio\_clasificacion.sql*, tomando como referencia el trabajo realizado en [37] se realizaron los siguientes cambios, con el fin de que Pg\_KDD pueda ejecutarlos.

Para la tarea de clasificación el proyecto MATE\_KDD [37], propone la tarea de clasificación usando una arquitectura medianamente acoplada, usando para la generación de las reglas de decisión funciones definidas por el usuario. Estas funciones se describen brevemente a continuación.

### **Función matetree()**

La función *matetree()* recibe como parámetro un tipo de dato VARCHAR que contiene el nombre del atributo clase de la tabla. Esta función tiene como fin generar un modelo de clasificación. Esta es la función principal la cual utiliza las funciones *mateby()* y *entro()* y luego junto con las funciones *mate\_tree()* y *mate\_gain* construye el modelo de decisión [37].

### **Función mateby()**

La función *mateby()* recibe como parámetro un tipo de dato VARCHAR que contiene el nombre del atributo clase de la tabla. Esta función tiene como fin generar por cada una de las tuplas todas las posibles combinaciones formadas por los valores no nulos de los atributos decisión y el atributo clase de la tabla tratada y además los cuenta. Al aplicar la función *mateby(<atributo\_clase>)* sobre la tabla a minar se obtiene la tabla 'mate\_clase' [37].

### **Función mate\_gain()**

La función *mate\_gain()* trabaja con la tabla 'mate\_clase' creada por la función *mateby()* y calcula la ganancia de información de los atributos decisión que cumplan unas determinadas característica y devuelve este valor a la función

mate\_tree(). Esta es la función que se encarga de seguir los pasos lógicos para encontrar el nodo en turno [37].

### **Funciones entro() y gain()**

Finalmente se aplican las funciones entro() y gain() que calculan la entropía y la ganancia de los atributos condición respecto al atributo clase. El resultado de estas funciones alimentan las funciones translate\_rules() y arm\_rules() que se encargan respectivamente de convertir los valores discretizados de la tabla original a sus valores correspondientes y de generar la tabla mate\_rules con la estructura de las reglas generadas [37].

Para la implementación en Pg\_KDD de la tarea de minería se realizaron pequeñas modificaciones al proceso lógico que se utiliza en la función matetree(). Se determinó que era mucho más eficaz utilizar en los primeros pasos de la generación de las reglas la primitiva SQL *Mate by*, fuertemente acoplada dentro del motor de PostgreSQL. La información que este operador retorne será usado por las función entro() dentro del proceso lógico usado en esa implementación. Para lograr esta modificación se adicionó una función definida por el usuario llamada convertir\_mate() cuya definición esta en el archivo *inicio\_clasificacion.sql* en el directorio.../contrib/KDD/clasificaciónPg\_KDD y se muestra en el Anexo A.

Adicionalmente se modificó la función mateby() agregando una llamada a la función convertir\_mate() como se observa en el Anexo A. El propósito de esta función es tomar la tabla resultante de *Mate by* y con estos datos sobre escribir los resultados generados internamente por la función mateby(), además de discretizar los valores para el correcto funcionamiento de la función entro().

Igualmente por requerimientos de la aplicación se adicionó la inserción de los cálculos de los valores de la entropía para determinar el nodo raíz en una nueva tabla usada en la aplicación para despliegue de datos. Esta tabla, llamada *ganancias*, cuya declaración se realiza en el archivo *inicio\_clasificacion.sql* en el directorio .../contrib/KDD/clasificaciónPg\_KDD, como se muestra en el Anexo A, se visualiza en el proceso de clasificación conectando a un nodo *gain()* un nodo *table view*.

Para la tarea de asociación, también se modificó la manera en que son usados las funciones agregadas, operadores y primitivas de asociación y minera

implementados en PostgresKDD. En la implementación original es necesario configurar manualmente la tabla a la que se le aplicará la tarea de clasificación realizando borrado y creación de tipos y de lenguajes de programación. En Pg\_KDD esta configuración se realiza automáticamente al iniciar el módulo de Minería y realizar el cargado del repositorio inicial. De esta manera se resuelve una limitación importante de la implementación y se dota a la herramienta de una mejora importante.

## 6. PRUEBAS Y EVALUACION DE RESULTADOS

Estas pruebas se realizaron en un equipo cuyas especificaciones son las siguientes:

- Procesador Intel core Quad
- Memoria RAM de 8GB DDR3
- Disco duro de 1 TB.
- Sistema Operativo Linux Slackware 12.

Se utilizaron los repositorios de datos reales de un supermercado del municipio de Pasto y repositorios de datos obtenidos en el portal de repositorios de la universidad de California – Irvine (<http://archive.ics.uci.edu/ml/>). Las características de cada uno de ellos se muestran a continuación.

### 6.1 CARACTERISTICAS DE LOS REPOSITORIOS

A continuación se describe los repositorios usados para las pruebas de rendimiento.

**6.1.1 Base de datos Universitaria.** Esta base de datos fue tomado de [15] donde se relaciona información de promedios educativos.

- Número de registros: 25
- Número de atributos: 5
- Total de diferentes valores: 24

La Tabla 41, muestra los datos de los promedios educativos relacionando la especialización, el estatus, la edad, la nacionalidad y el GPA.

**Tabla 41: Repositorio Universitaria**

| Especialización    | Estatus | Edad    | Nacionalidad  | GPA       |
|--------------------|---------|---------|---------------|-----------|
| <b>French</b>      | M.A.    | Over_30 | Canada        | 2.8...3.2 |
| <b>Cs</b>          | Junior  | 16...20 | Europe        | 3.2...3.6 |
| <b>Physics</b>     | M.S.    | 26...30 | Latin_America | 3.2...3.6 |
| <b>Engineering</b> | Ph.D    | 26...30 | Asia          | 3.6...4.0 |

| Especialización    | Estatus | Edad    | Nacionalidad  | GPA       |
|--------------------|---------|---------|---------------|-----------|
| <b>Philosophy</b>  | Ph.D    | 26...30 | Europe        | 3.2...3.6 |
| <b>French</b>      | Senior  | 16...20 | Canada        | 3.2...3.6 |
| <b>Chemistry</b>   | Junior  | 21...25 | USA           | 3.6...4.0 |
| <b>Cs</b>          | Senior  | 16...20 | Canada        | 3.2...3.6 |
| <b>Philosophy</b>  | M.S.    | Over_30 | Canada        | 3.6...4.0 |
| <b>French</b>      | Junior  | 16...20 | USA           | 2.8...3.2 |
| <b>Philosophy</b>  | Junior  | 26...30 | Canada        | 2.8...3.2 |
| <b>Philosophy</b>  | M.S.    | 26...30 | Asia          | 3.2...3.6 |
| <b>French</b>      | Junior  | 16...20 | Canada        | 3.2...3.6 |
| <b>Math</b>        | Senior  | 16...20 | USA           | 3.6...4.0 |
| <b>Cs</b>          | Junior  | 16...20 | Canada        | 3.2...3.6 |
| <b>Philosophy</b>  | Ph.D    | 26...30 | Canada        | 3.6...4.0 |
| <b>Philosophy</b>  | senior  | 26...30 | Canada        | 2.8...3.2 |
| <b>French</b>      | Ph.D    | Over_30 | Canada        | 2.8...3.2 |
| <b>Engineering</b> | Junior  | 21...25 | Europe        | 3.2...3.6 |
| <b>Math</b>        | Ph.D    | 26...30 | Latin_America | 3.2...3.6 |
| <b>Chemistry</b>   | Junior  | 16...20 | USA           | 3.6...4.0 |
| <b>Engineering</b> | Junior  | 21...25 | Canada        | 3.2...3.6 |
| <b>French</b>      | M.S.    | Over_30 | Latin_America | 3.2...3.6 |
| <b>Philosophy</b>  | Junior  | 21...25 | USA           | 2.8...3.2 |
| <b>Math</b>        | Junior  | 16...20 | Canada        | 3.6...4.0 |

La base de datos Universitaria se transformó a ítems. En la Tabla 42 relaciona cada valor diferente de la Tabla 41 con un nombre de ítem.

**Tabla 42: Repositorio Universitaria en Ítems**

| ITEM       | DESCRIPCION | ITEM       | DESCRIPCION   |
|------------|-------------|------------|---------------|
| <b>I1</b>  | French      | <b>I13</b> | 16,,,,20      |
| <b>I2</b>  | Cs          | <b>I14</b> | 21,,,,25      |
| <b>I3</b>  | Physics     | <b>I15</b> | 26,,,,30      |
| <b>I4</b>  | engineering | <b>I16</b> | over_30       |
| <b>I5</b>  | philosophy  | <b>I17</b> | Canada        |
| <b>I6</b>  | chemistry   | <b>I18</b> | Europe        |
| <b>I7</b>  | Math        | <b>I19</b> | Latin_America |
| <b>I8</b>  | M.A.        | <b>I20</b> | Asia          |
| <b>I9</b>  | Junior      | <b>I21</b> | USA           |
| <b>I10</b> | M.S.        | <b>I22</b> | 2,8,,,,3,2    |
| <b>I11</b> | Ph.D        | <b>I23</b> | 3,2,,,,3,6    |
| <b>I12</b> | Senior      | <b>I24</b> | 3,6,,,,4,0    |

Finalmente, en la Tabla 43, se muestra la base de datos Universitaria transformada en una tabla tipo transacción-ítems, la cual será la usada en las diferentes pruebas. En la Tabla 43, se indica los valores no nulos para cada transacción.

- Número de transacciones: 25
- Número de atributos: 24 .

Tabla 43: Ítems

| TID | LISTA DE ITEMS     | TID | LISTA DE ITEMS     |
|-----|--------------------|-----|--------------------|
| T1  | I1,I8,I16,I17,I22  | T14 | I7,I12,I13,I21,I24 |
| T2  | I2,I9,I13,I18,I23  | T15 | I2,I9,I13,I17,I23  |
| T3  | I3,I10,I15,I19,I23 | T16 | I5,I11,I15,I17,I24 |
| T4  | I4,I11,I15,I20,I24 | T17 | I5,I12,I15,I17,I22 |
| T5  | I5,I11,I15,I18,I23 | T18 | I1,I11,I16,I17,I22 |
| T6  | I1,I12,I13,I17,I23 | T19 | I4,I9,I14,I18,I23  |
| T7  | I6,I9,I14,I21,I24  | T20 | I7,I11,I15,I19,I23 |
| T8  | I2,I12,I13,I17,I23 | T21 | I6,I9,I13,I21,I24  |
| T9  | I5,I10,I16,I17,I24 | T22 | I4,I9,I14,I17,I23  |
| T10 | I1,I9,I13,I21,I22  | T23 | I1,I10,I16,I19,I23 |
| T11 | I5,I9,I15,I17,I22  | T24 | I5,I9,I14,I21,I22  |
| T12 | I5,I10,I15,I20,I23 | T25 | I7,I9,I13,I17,I24  |
| T13 | I1,I9,I13,I17,I23  |     |                    |

**6.1.2 Base de Datos Mercado.** Esta base de datos contiene transacciones de uno de los supermercados más importantes del departamento de Nariño (Colombia) durante un periodo determinado.

- Número de transacciones: 27000
- Promedio Ítems por transacción: 20

La Tabla 44 muestra los valores discretizados de la base de datos Mercado

Tabla 44: Valores transformados de Mercado

| Item | Valor                  |
|------|------------------------|
| I1   | ANEJO VICTORIA         |
| I2   | ALPINETTE PASION VERDE |

| Item | Valor                               |
|------|-------------------------------------|
| I3   | AROMATICAS GOURMET YERBABUENA       |
| I4   | AROMATICAS HINDU TORONJIL           |
| I5   | ARROZ FLOR HUILA                    |
| I6   | CAFE PURO                           |
| I7   | CEREALES MAIZENA VAINILLA           |
| I8   | CHOCOLATE LUKER                     |
| I9   | CHOCOLYNE                           |
| I10  | COCA COLA BOT VIDRIO                |
| I11  | COLONIA SPLASH SPORT X200 -40%      |
| I12  | CREMA DENTAL CREST                  |
| I13  | CREMA LAVAPLATOS MENTA 500 GTS.250  |
| I14  | CUBIERTA DE CHOCOLATE               |
| I15  | DES. SERIES INV. W.R. 50% EXTRA CON |
| I16  | ENDULZANTE NATURAL BIODIET SOBRES   |
| I17  | ESCOBA AGUAMARINA MADERA            |
| I18  | ESPONJILLA BONBRIL                  |
| I19  | FELDENE GEL 0.5%                    |
| I20  | HUEVOS CODORNIZ                     |

**6.1.3 Base de Datos Guardería.** Esta base de datos relaciona información que fue usada para determinar el acceso o no a escuelas infantiles.

- Número de transacciones: 12960
- Promedio Ítems por transacción: 8

**Tabla 45: Repositorio Guardería**

| Nro | Nombre del Atributo |
|-----|---------------------|
| 1   | padres              |
| 2   | en_guarderia        |
| 3   | hijos               |
| 4   | hogar               |
| 5   | finanzas            |
| 6   | Social              |
| 7   | salud               |
| 8   | recomendacion       |

**6.1.4 Base de datos JugarTenis.** Esta base de datos contiene información de 14 casos de decisión, con 5 atributos (atributos nominales).

- Número de transacciones: 14
- Número de atributos: 5

**Tabla 46: Repositorio JugarTenis**

| Nro | Nombre del Atributo |
|-----|---------------------|
| 1   | Estado              |
| 2   | Temperatura         |
| 3   | Humedad             |
| 4   | Viento              |
| 5   | Jugar(Clase)        |

**6.1.5 Base de datos Productos.** Esta base de datos contiene las siguientes características:

- Número de transacciones: 9
- Número de atributos: 6

**Tabla 47: Repositorio Productos con transacciones**

| TID  | ITEM1 | ITEM2  | ITEM3 | ITEM4 | ITEM5 | ITEM6  |
|------|-------|--------|-------|-------|-------|--------|
| T100 | leche | huevos |       |       | café  |        |
| T101 |       | huevos |       | queso |       |        |
| T102 |       | huevos | pan   |       |       |        |
| T103 | leche | huevos |       | queso |       |        |
| T104 | leche |        | pan   |       |       | azucar |
| T105 |       | huevos | pan   |       |       |        |
| T106 | leche |        | pan   |       |       |        |
| T107 | leche | huevos | pan   |       | café  |        |
| T108 | leche | huevos | pan   |       |       |        |

Para efectos de las pruebas funcionales se tomara el repositorio productos de la Tabla 47 y se lo transformara siguiendo los valores representados en la Tabla 48.

**Tabla 48: Valores Transformados para Productos**

| Item | Valor  |
|------|--------|
| i1   | Leche  |
| i2   | Huevos |
| i3   | Pan    |
| i4   | Queso  |
| i5   | Café   |
| i6   | Azucar |

## 6.2 PRUEBA DE FUNCIONALIDAD DE Pg\_KDD

Estas pruebas tienen por objetivo probar el buen funcionamiento de la herramienta Pg\_KDD con las tareas de minería de datos Asociación y Clasificación. Para tal fin se realizan las pruebas manualmente y con Pg\_KDD y se evalúan los resultados obtenidos. Si estos coinciden se demuestra que la herramienta funciona correctamente.

**6.2.1 Asociación.** Para la tarea de Asociación se utilizó el repositorio transformado de la base de datos Productos denominado Ítems que se muestra en la Tabla 49. A este repositorio se le aplica el algoritmo EquipAsso para calcular el conjunto de ítems frecuentes y luego se generan las reglas:

**Tabla 49: Ítems**

| TID  | ITEM1 | ITEM2 | ITEM3 | ITEM4 | ITEM5 | ITEM6 |
|------|-------|-------|-------|-------|-------|-------|
| T100 | I1    | I2    |       |       | I5    |       |
| T101 |       | I2    |       | I4    |       |       |
| T102 |       | I2    | I3    |       |       |       |
| T103 | I1    | I2    |       | I4    |       |       |
| T104 | I1    |       | I3    |       |       | I6    |
| T105 |       | I2    | I3    |       |       |       |
| T106 | I1    |       | I3    |       |       |       |

| TID  | ITEM1 | ITEM2 | ITEM3 | ITEM4 | ITEM5 | ITEM6 |
|------|-------|-------|-------|-------|-------|-------|
| T107 | I1    | I2    | I3    |       | I5    |       |
| T108 | I1    | I2    | I3    |       |       |       |

### Aplicación Manual:

#### Paso 1:

Se calcula el soporte para los ítems frecuentes tamaño 1 hasta el grado de la tabla utilizando la primitiva SQL *Associator Range*. Esta consulta se realiza con la siguiente sentencia SQL, y se obtiene el resultado mostrado en la Tabla 50.

```
Select item,item2,item3,item4,item5,item6,count(*) as soporte
From D
Associator Range 1 until 7
Group by item1,item2,item3,item4,item5,item6;
```

#### Paso 2.

Se obtienen el conjunto de ítems frecuentes que cumplan con un soporte mínimo mayor o igual a 2 (*i.e. minsup=2/9=22%*). Para ello se ejecuta la siguiente sentencia SQL. El resultado se muestra en la Tabla 51.

```
Select item,item2,item3,item4,item5,item6,count(*) As soporte
INTO fre_directos
FROM items
ASSOCIATOR RANGE 1 Until 7
GROUP By item1,item2,item3,item4,item5,item6
HAVING count(*) >=2
```

Tabla 50: Associator Range

| item1 | item2 | item3 | item4 | item5 | item6 | soporte |
|-------|-------|-------|-------|-------|-------|---------|
| I1    | I2    | I3    |       | I5    |       | 1       |
| I1    | I2    | I3    |       |       |       | 2       |
| I1    | I2    |       | I4    |       |       | 1       |
| I1    | I2    |       |       | I5    |       | 2       |
| I1    | I2    |       |       |       |       | 4       |
| I1    |       | I3    |       | I5    |       | 1       |
| I1    |       | I3    |       |       | I6    | 1       |

| item1 | item2 | item3 | item4 | item5 | item6 | soporte |
|-------|-------|-------|-------|-------|-------|---------|
| I1    |       | I3    |       |       |       | 4       |
| I1    |       |       | I4    |       |       | 1       |
| I1    |       |       |       | I5    |       | 2       |
| I1    |       |       |       |       | I6    | 1       |
| I1    |       |       |       |       |       | 6       |
|       | I2    | I2    |       | I5    |       | 1       |
|       | I2    | I3    |       |       |       | 4       |
|       | I2    | I3    |       |       |       | 4       |
|       | I2    |       | I4    |       |       | 2       |
|       | I2    |       |       | I5    |       | 2       |
|       | I2    |       |       |       |       | 7       |
|       |       | I3    |       | I5    |       | 1       |
|       |       | I3    |       |       | I6    | 1       |
|       |       | I3    |       |       |       | 6       |
|       |       |       | I4    |       |       | 2       |
|       |       |       |       | I5    |       | 2       |
|       |       |       |       |       | I6    | 1       |

Tabla 51: *fre\_directos*

| Item1 | Item2 | Item3 | Item4 | Item5 | Item6 | soporte |
|-------|-------|-------|-------|-------|-------|---------|
| I1    | I1    | I3    |       |       |       | 2       |
| I1    | I2    |       |       | I5    |       | 2       |
| I1    | I2    |       |       |       |       | 4       |
| I1    |       | I3    |       |       |       | 4       |
| I1    |       |       |       | I5    |       | 2       |
| I1    |       |       |       |       |       | 6       |
|       | I2    | I3    |       |       |       | 4       |
|       | I2    |       | I5    |       |       | 2       |
|       | I2    |       |       |       |       | 7       |
|       |       | I3    |       |       |       | 6       |
|       |       |       | I4    |       |       | 2       |
|       |       |       |       | I5    |       | 2       |

### Paso 3

Se generan las reglas de tamaño 3 de la tabla *fre\_directos* con una confianza mínima del 20% se ejecuta la siguiente sentencia SQL. El resultado se muestra en la Tabla 52.

```
SELECT * FROM DESCRIBE_ASSOCIATOR_RULES('fre_directos,2,20).
```

Tabla 52: Resultado de Describe\_Associator\_Rules

| Item1 | Item2 | Item3 | Item4 | Item5 | n_regla | Implica | Soporte | confianza |
|-------|-------|-------|-------|-------|---------|---------|---------|-----------|
| I1    |       |       |       |       | 1       | A       | 6       | 33.33     |
|       | I2    | I3    |       |       | 1       | C       | 4       |           |
|       | I2    |       |       |       | 2       | A       | 7       | 28.57     |
| I1    |       | I3    |       |       | 2       | C       | 4       |           |
|       |       | I3    |       |       | 3       | A       | 6       | 33.33     |
| I1    | I2    |       |       |       | 3       | C       | 4       |           |
| I1    | I2    |       |       |       | 4       | A       | 4       | 50.00     |
|       |       | I3    |       |       | 4       | C       | 6       |           |
| I1    |       | I3    |       |       | 5       | A       | 4       | 50.00     |
|       | I2    |       |       |       | 5       | C       | 7       |           |
|       | I2    | I3    |       |       | 6       | A       | 4       | 50.00     |
| I1    |       |       |       |       | 6       | C       | 6       |           |
| I1    |       |       |       |       | 7       | A       | 6       | 33.33     |
|       | I2    |       |       | I5    | 7       | C       | 2       |           |
|       | I2    |       |       |       | 8       | A       | 7       | 28.57     |
| I1    |       |       |       | I5    | 8       | C       | 2       |           |
|       |       |       |       | I5    | 9       | A       | 2       | 100.00    |
| I1    | I2    |       |       |       | 9       | C       | 4       |           |
| I1    | I2    |       |       |       | 10      | A       | 4       | 50.00     |
|       |       |       |       | I5    | 10      | C       |         |           |
| I1    |       |       |       | I5    | 11      | A       | 2       | 100.00    |
|       | I2    |       |       |       | 11      | C       | 7       |           |
|       | I2    |       |       | I5    | 12      | A       | 2       | 100.00    |
| I1    |       |       |       |       | 12      | C       | 6       |           |

Con esta tabla se generan las reglas siguiendo el orden lógico que ella nos presenta. La columna *n\_regla* indica el número de la regla. Para cada regla en la columna *implica* se detalla cual es el antecedente y cual el consecuente. También se muestra el soporte de la regla.

Las reglas que se generan son las siguientes:

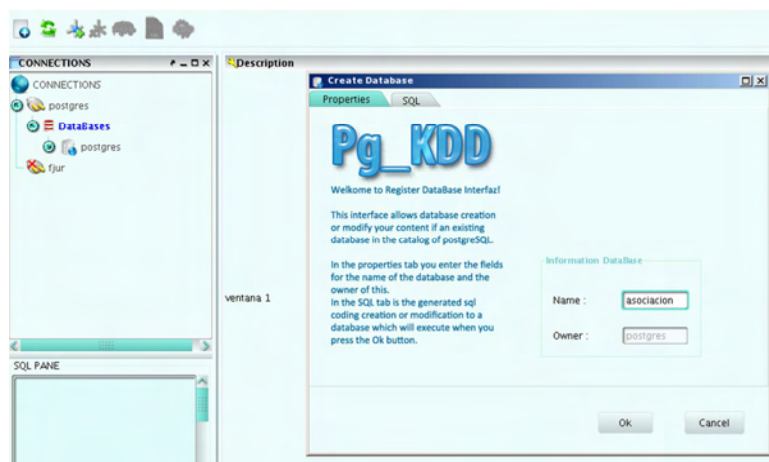
- $I1 \Rightarrow I2 \wedge I3$       confianza =  $2/6 = 33.33\%$
- $I2 \Rightarrow I1 \wedge I3$       confianza =  $2/7 = 28.57\%$
- $I3 \Rightarrow I1 \wedge I2$       confianza =  $2/2 = 33.33\%$
- $I1 \wedge I2 \Rightarrow I3$       confianza =  $2/4 = 50\%$
- $I1 \wedge I3 \Rightarrow I2$       confianza =  $2/2 = 50\%$
- $I2 \wedge I3 \Rightarrow I1$       confianza =  $2/2 = 50\%$

- $I1 \Rightarrow I2 \wedge I5$       confianza =  $2/6 = 33.33\%$
- $I2 \Rightarrow I1 \wedge I5$       confianza =  $2/7 = 28.57\%$
- $I5 \Rightarrow I1 \wedge I2$       confianza =  $2/2 = 100\%$
- $I1 \wedge I2 \Rightarrow I5$       confianza =  $2/4 = 50\%$
- $I1 \wedge I5 \Rightarrow I2$       confianza =  $2/2 = 100\%$
- $I2 \wedge I5 \Rightarrow I1$       confianza =  $2/2 = 100\%$

## Aplicación en Pg\_KDD

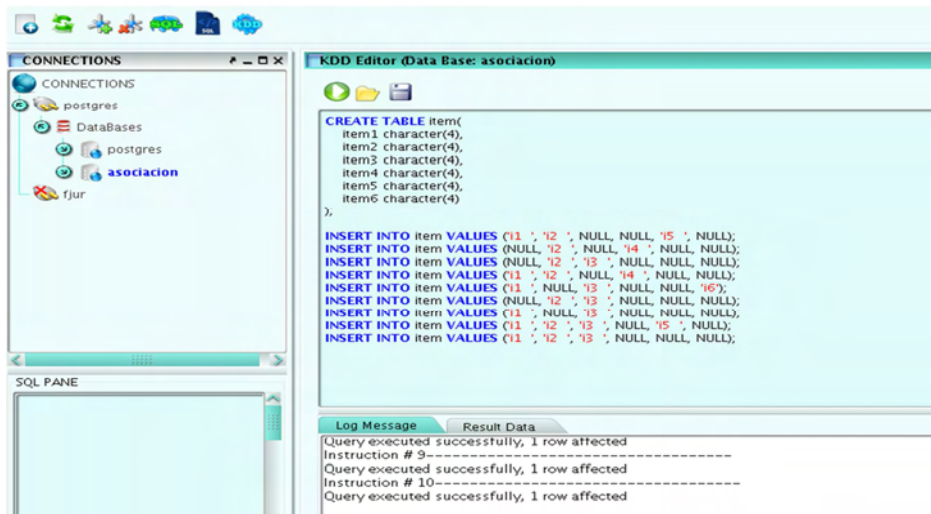
En la pantalla principal de la aplicación Pg\_KDD, se crea una nueva database llamada *asociación* como se lo observa en la Figura 53.

Figura 53: Creación de la nueva database asociación



En la Figura 54, se crea la tabla *ítem* dentro de la database *asociación*.

Figura 54: Script - Tabla item



### Paso 1

Se carga el módulo de data mining y agregamos hacia el work area el objeto KDD repository (Figura 55). Cargamos la tabla *ítem*, seleccionando todos sus atributos (Figura 56). Y con clic derecho se da clic en *Run* para que el nodo se ejecute.

Figura 55: Cargar repositorio para asociación

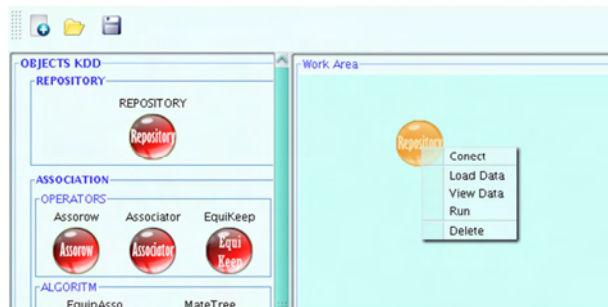
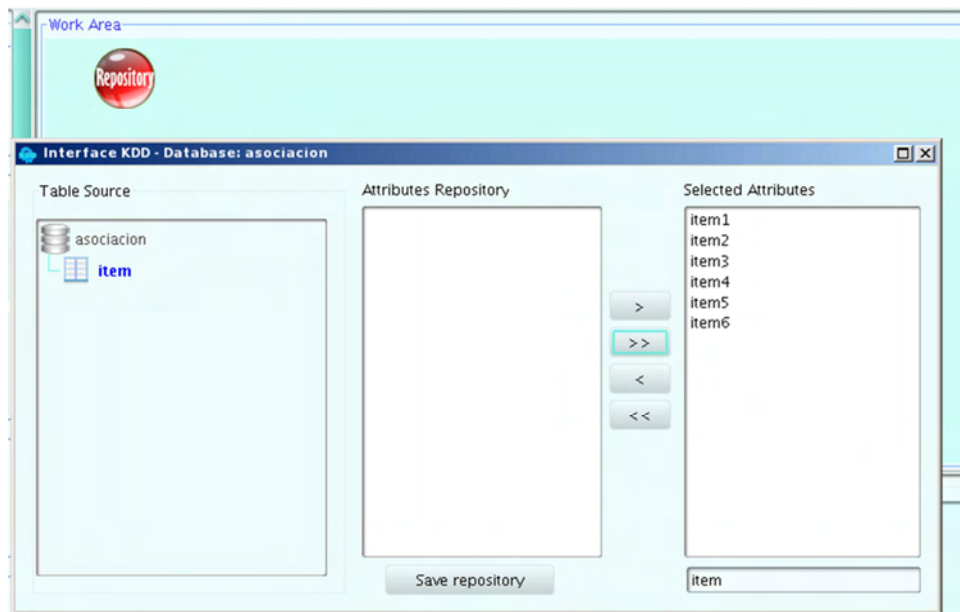


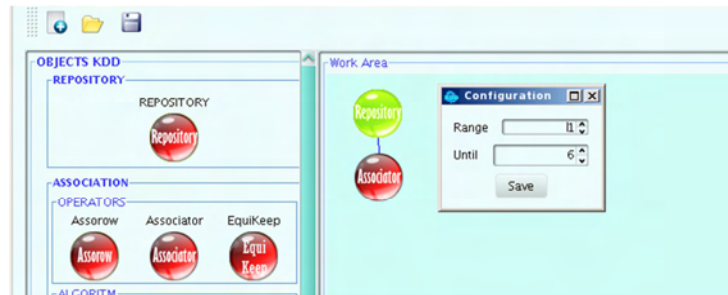
Figura 56: Selección de los atributos



## Paso 2

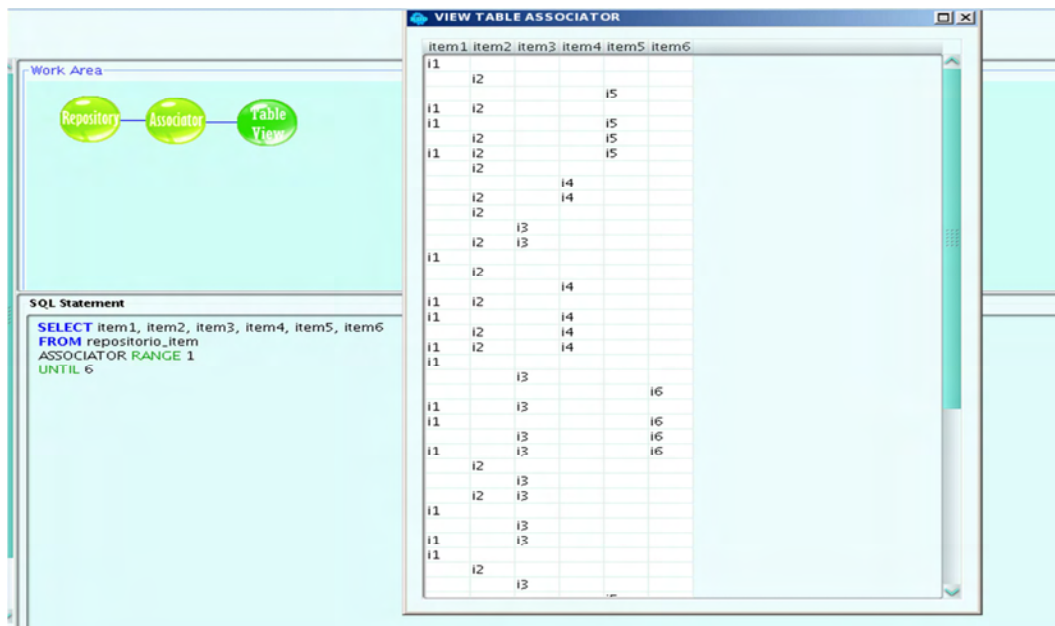
Se agrega un objeto tipo *Associator* al área de trabajo. Se conecta al nodo *repository* y se configura el rango de trabajo del operador *Associator* (Figura 57).

Figura 57: Configuración del operador Associator



Se agrega un nodo *Table View*, se lo conecta al nodo *Associator*. Con clic derecho se lo ejecuta, y se muestra la tabla resultado de la primitiva *Associator range* (Figura 58).

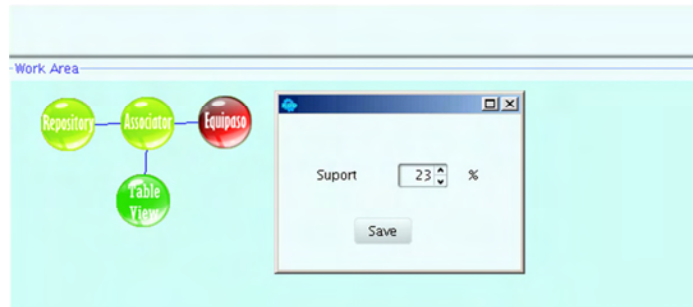
Figura 58: Resultado de Associator Range



### Paso 3

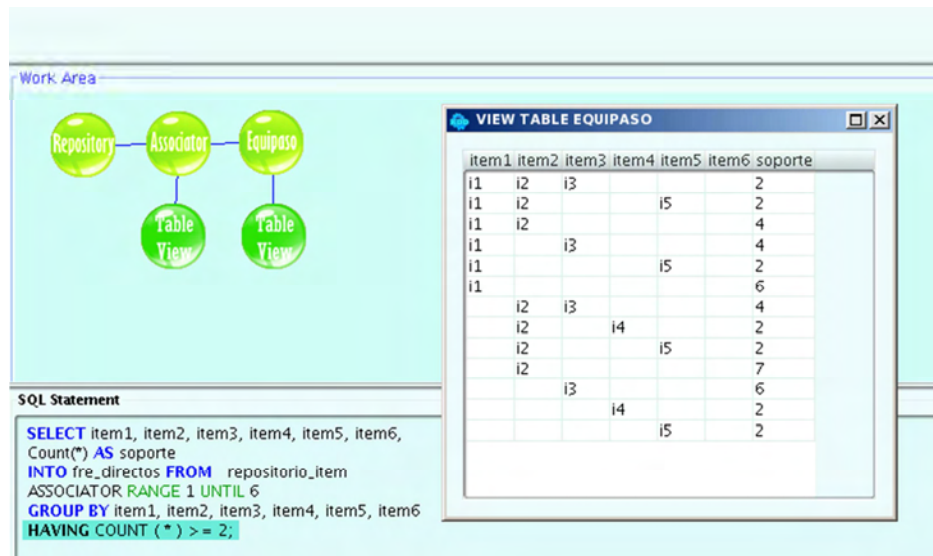
Se agrega un nodo *EquipAsso* al área de trabajo, se lo conecta al nodo *Associator*, se lo configura y se lo ejecuta (Figura 59).

Figura 59: Configuración del operador Equipaso



Se arrastra un nodo *Table view*, se lo conecta al nodo *EquipAsso*, se ejecuta y se muestra el resultado de la primitiva *Associator Range*, restringiendo el resultado a los ítemset con un soporte mayor igual a 2 (*i.e. minsup=2/9=22%*) (Figura 60), en este caso el soporte es mayor al que nos da como resultado al calcularlo debido a requerimientos de la aplicación.

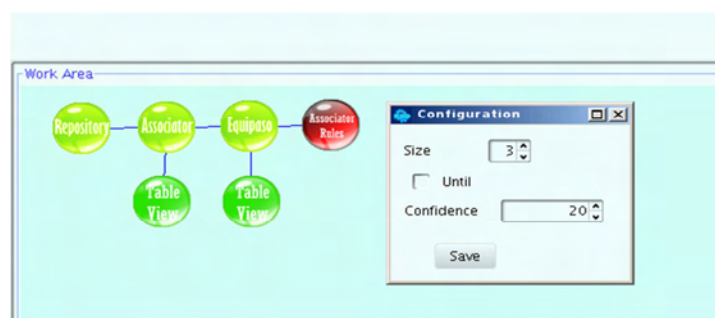
Figura 60: Resultado de Equipaso



#### Paso 4

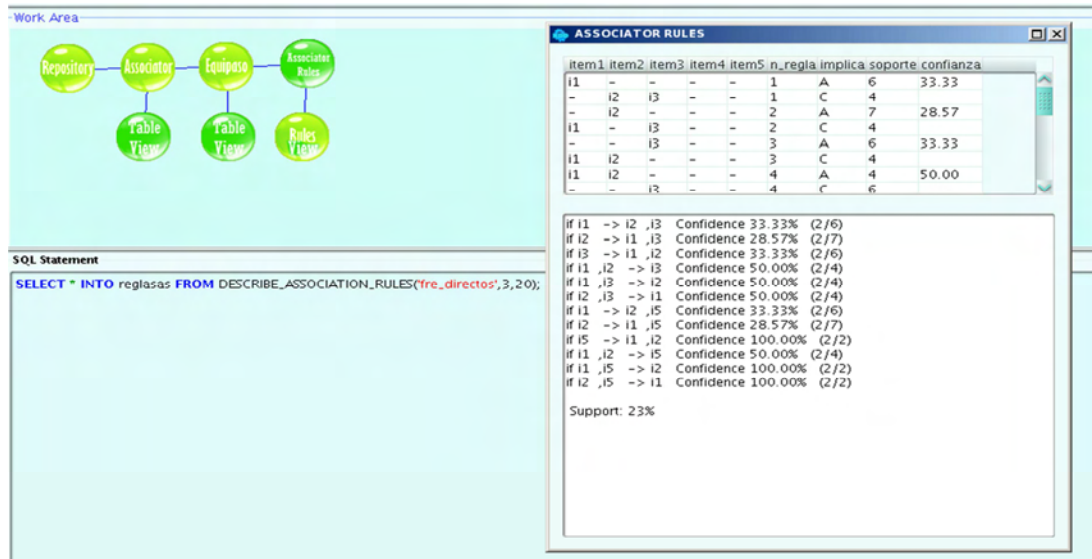
Finalmente se agrega un nodo *Associator Rules*, conectado al nodo *EquipAsso*.. Se configura el tamaño de las reglas, si se desea generar todas las reglas hasta ese tamaño, y la confianza mínima (Figura 61).

Figura 61: Configuración del nodo *Associator Rules*



Se agrega un nodo *Rules View*, se lo ejecuta y se muestran las reglas generadas por la función agregada *describe\_associator\_rules* (Figura 62),

Figura 62: Reglas generadas por la función *describe\_associator\_rules*



Del proceso anterior se verifica que las reglas que se generan corresponden a las encontradas anteriormente, como lo indica la Figura 63.

Figura 63: Comparación de resultados

|                               | Proceso Manual              | Pg_KDD                                   |
|-------------------------------|-----------------------------|--|
| $I1 \Rightarrow I2 \wedge I3$ | confianza = $2/6 = 33.33\%$ | if i1 -> i2 ,i3 Confidence 33.33% (2/6)  |
| $I2 \Rightarrow I1 \wedge I3$ | confianza = $2/7 = 28.57\%$ | if i2 -> i1 ,i3 Confidence 28.57% (2/7)  |
| $I3 \Rightarrow I1 \wedge I2$ | confianza = $2/2 = 33.33\%$ | if i3 -> i1 ,i2 Confidence 33.33% (2/6)  |
| $I1 \wedge I2 \Rightarrow I3$ | confianza = $2/4 = 50\%$    | if i1 ,i2 -> i3 Confidence 50.00% (2/4)  |
| $I1 \wedge I3 \Rightarrow I2$ | confianza = $2/2 = 50\%$    | if i1 ,i3 -> i2 Confidence 50.00% (2/4)  |
| $I2 \wedge I3 \Rightarrow I1$ | confianza = $2/2 = 50\%$    | if i2 ,i3 -> i1 Confidence 50.00% (2/4)  |
| $I1 \Rightarrow I2 \wedge I5$ | confianza = $2/6 = 33.33\%$ | if i1 -> i2 ,i5 Confidence 33.33% (2/6)  |
| $I2 \Rightarrow I1 \wedge I5$ | confianza = $2/7 = 28.57\%$ | if i2 -> i1 ,i5 Confidence 28.57% (2/7)  |
| $I5 \Rightarrow I1 \wedge I2$ | confianza = $2/2 = 100\%$   | if i5 -> i1 ,i2 Confidence 100.00% (2/2) |
| $I1 \wedge I2 \Rightarrow I5$ | confianza = $2/4 = 50\%$    | if i1 ,i2 -> i5 Confidence 50.00% (2/4)  |
| $I1 \wedge I5 \Rightarrow I2$ | confianza = $2/2 = 100\%$   | if i1 ,i5 -> i2 Confidence 100.00% (2/2) |
| $I2 \wedge I5 \Rightarrow I1$ | confianza = $2/2 = 100\%$   | if i2 ,i5 -> i1 Confidence 100.00% (2/2) |

Analizando los resultados obtenidos manualmente en las Tabla 50, Tabla 51 y Tabla 52 y los obtenidos en las Figura 58, Figura 60 y Figura 62 se puede observar que estos coinciden en su totalidad, por lo tanto la herramienta Pg\_KDD funciona correctamente para la tarea de Asociación.

**6.2.2 Clasificación.** Para la tarea de Clasificación se utilizó el repositorio denominado *Jugartenis* que se muestra en la Figura 64. A este repositorio se le aplica el algoritmo Mate-Tree para calcular construir el árbol de decisión y luego se generar las reglas de clasificación.

Figura 64: jugartenis

| DIA | ESTADO   | TEMPER.  | HUMEDAD | VIENTO | JUGAR TENIS |
|-----|----------|----------|---------|--------|-------------|
| D1  | Soleado  | Caliente | Alta    | Debil  | No          |
| D2  | Soleado  | Caliente | Alta    | Fuerte | No          |
| D3  | Nublado  | Caliente | Alta    | Debil  | SI          |
| D4  | Lluvioso | Templado | Alta    | Debil  | Si          |
| D5  | Lluvioso | Fresco   | Normal  | Debil  | Si          |
| D6  | Lluvioso | Fresco   | Normal  | Fuerte | No          |
| D7  | Nublado  | Fresco   | Normal  | Fuerte | Si          |
| D8  | Soleado  | Templado | Alta    | Debil  | No          |
| D9  | Soleado  | Fresco   | Normal  | Debil  | Si          |
| D10 | Lluvioso | Templado | Normal  | Debil  | Si          |
| D11 | Soleado  | Templado | Normal  | Fuerte | Si          |
| D12 | Nublado  | Templado | Alta    | Fuerte | Si          |
| D13 | Nublado  | Caliente | Normal  | Debil  | Si          |
| D14 | Lluvioso | Templado | Alta    | Fuerte | No          |

Fuente. Timarán,R. Clasificación: una tarea de descubrimiento de conocimiento en bases de datos, Pasto, 2009 [49].

### Aplicación manual [49]:

#### Paso 1

Se define como atributo clase S, al atributo *jugarTennis*, se calcula la Entropía de S, el conteo para Si=9 y el conteo para No=5. Aplicando la fórmula para la entropía, tenemos:

$Entropía(S)=E(S)= -\sum pi \log_2 pi$ , donde pi es la probabilidad que un ejemplo arbitrario pertenezca a la clase Ci.

$$E([9+,5-])=-(9/14)\log_2(9/14)-(5/14)\log_2(5/14)$$

$$E([9+,5-])=0.940$$

#### Calculo de ganancias

Se calcula la ganancia de cada atributo condición respecto al atributo clase S, usando la siguiente formula:

$$Gain(S,A) = Entropía(S) - \sum v \in Valores(A) * |S_v| / |S| * Entropía(S_v)$$

Donde:

$Valores(A)$  es el conjunto de todos los valores del atributo A

$S_v$  es el subconjunto de S para el atributo A que toma valores v.

- Para el Atributo viento.

valores(viento)=debil,fuerte.

Sdebil=[6+,2-]

Sfuerte=[3+,3-]

$$Gain(S, Viento) = Entropía(S) - (Sdebil * Entropía(Sdebil) + Sfuerte * Entropía(Sfuerte))$$

$$= 0.940 - 8/14 Entropía(Sdebil) - 6/14 Entropía(Sfuerte)$$

$$\text{Calculamos } E(Sdebil) = E([6+, 2-]) = -(6/8) \log_2(6/8) - (2/8) \log_2(2/8) = 0.811$$

$$E(Sfuerte) = E([3+, 3-]) = -(3/6) \log_2(3/6) - (3/6) \log_2(3/6) = 1$$

Reemplazando:

$$Gain(S, viento) = 0.940 - 0.463 - 0.428 = \mathbf{0.048}$$

- Para el atributo Estado:

$$Gain(S, Estado) = 0.940 - (5/14)0.97 - (4/14)0 - (5/14)0.97 = \mathbf{0.246}$$

- Para el atributo Humedad:  $Gain(S, Humedad) = \mathbf{0.151}$

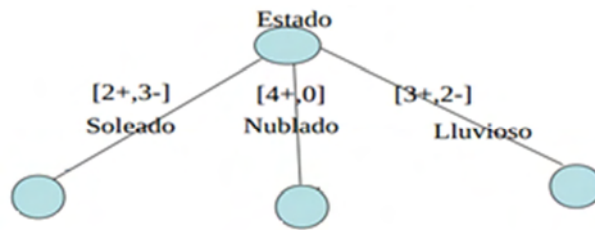
- Para el atributo Temperatura:  $Gain(S, Temperatura) = \mathbf{0.029}$

## Paso 2

### Construcción del árbol

De acuerdo a la ganancia de información, el atributo que mayor ganancia tiene es estado, por lo tanto este atributo se selecciona como el nodo raíz. Los nodos hijos son los diferentes valores que puede tomar estado ( Figura 65).

Figura 65: Árbol de decisión 1



Fuente. Timarán,R. Clasificación: una tarea de descubrimiento de conocimiento en bases de datos, Pasto, 2009 [49].

De la

Figura 66, siguiente podemos determinar que para el estado *nublado* para cada valor se presenta una clasificación positiva sobre la clase decisión *JugarTenis*, por lo tanto el nodo esta formado por un nodo hoja en la cual la clasificación *JugarTenis=si* y el valor de la entropía = 0.

Figura 66: Valores para el estado nublado

| DIA | ESTADO  | TEMPERATURA | HUMEDAD | VIENTO | JUGARTENIS |
|-----|---------|-------------|---------|--------|------------|
|     |         |             |         |        |            |
| D3  | Nublado | Caliente    | Alta    | Débil  | Si         |
|     |         |             |         |        |            |
|     |         |             |         |        |            |
| D7  | Nublado | Fresco      | Normal  | Fuerte | Si         |
|     |         |             |         |        |            |
|     |         |             |         |        |            |
| D12 | Nublado | Templado    | Alta    | Fuerte | Si         |
| D13 | Nublado | Caliente    | Normal  | Débil  | Si         |

Fuente. Timarán,R. Clasificación: una tarea de descubrimiento de conocimiento en bases de datos, Pasto, 2009 [49].

### Paso 3

La Figura 67, nos muestra que para el estado soleado, no hay uniformidad en los valores sobre la clase decisión, por lo que es necesario calcular la ganancia de *soleado* respecto los atributos condición y su valor en el atributo clase.

Figura 67: Valores para el estado soleado

| DIA | ESTADO  | TEMPERATURA | HUMEDAD | VIENTO | JUGARTENIS |
|-----|---------|-------------|---------|--------|------------|
| D1  | Soleado | Caliente    | Alta    | Débil  | No         |
| D2  | Soleado | Caliente    | Alta    | Fuerte | No         |
|     |         |             |         |        |            |
|     |         |             |         |        |            |
|     |         |             |         |        |            |
|     |         |             |         |        |            |
| D8  | Soleado | Templado    | Alta    | Débil  | No         |
| D9  | Soleado | Fresco      | Normal  | Débil  | Si         |
|     |         |             |         |        |            |
| D11 | Soleado | Templado    | Normal  | Fuerte | Si         |
|     |         |             |         |        |            |
|     |         |             |         |        |            |

Fuente. Timarán,R. Clasificación: una tarea de descubrimiento de conocimiento en bases de datos, Pasto, 2009 [49].

- Ganancia en Soleado

Humedad (Alta[0+,3-])  
 Humedad(Normal[2+,0-])

$$Gain(Soleado, Humedad) = 0.970 - (3/5) * 0 - (2/5) * 0 = 0.970$$

- Ganancia en Temperatura

Temperatura(Caliente[0+,0-])  
 Temperatura(Templado[1+,1-])  
 Temperatura(fresco[1+,0-])

$$Gain(Soleado, Temperatura) = 0.970 - (2/5) * 0 - (2/5) * 1 - (1/5) * 0 = 0.570$$

- Ganancia en Viento

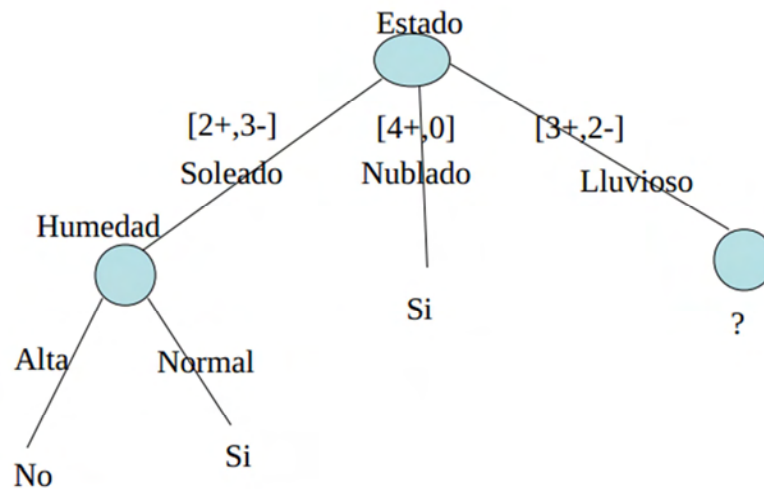
Viento(Débil[1+,2-])  
 Viento(fuerte[1+,1-])

$$Gain(Soleado, Viento) = 0.970 - (3/5) * 0.918 - (2/5) * 1 = 0.019$$

## Construcción del árbol

De los cálculos anteriores determinamos que el atributo con mayor ganancia de información es humedad, por lo tanto lo ubicamos como nodo hijo de *estado=soleado*.

Figura 68: Árbol de Decisión 2



Fuente. Timarán,R. Clasificación: una tarea de descubrimiento de conocimiento en bases de datos, Pasto, 2009 [49].

De la Figura 68, se observa que para *humedad=alta*, para cada valor se presenta una clasificación negativa sobre la clase decisión *JugarTenis*, por lo tanto el nodo está formado por un nodo hoja en la cual la clasificación *JugarTenis=no* y el valor de la entropía =0. Igual ocurre con *humedad=normal*, con un valor para *JugarTenis=si*.

Ahora para *estado=lluvioso*, la Figura 69, nos muestra que no hay uniformidad en los valores sobre la clase decisión, por lo que es necesario calcular la ganancia de *lluvioso* respecto los atributos condición y su valor en el atributo clase.

Figura 69: Valores para estado lluvioso

| DIA | ESTADO   | TEMPERATURA | HUMEDAD | VIENTO | JUGARTENIS |
|-----|----------|-------------|---------|--------|------------|
|     |          |             |         |        |            |
| D4  | Lluvioso | Templado    | Alta    | Débil  | Si         |
| D5  | Lluvioso | Fresco      | Normal  | Débil  | Si         |
| D6  | Lluvioso | Fresco      | Normal  | Fuerte | No         |
|     |          |             |         |        |            |
|     |          |             |         |        |            |
| D10 | Lluvioso | Templado    | Normal  | Débil  | Si         |
|     |          |             |         |        |            |
|     |          |             |         |        |            |

Fuente. Timarán,R. Clasificación: una tarea de descubrimiento de conocimiento en bases de datos, Pasto, 2009 [49].

- Ganancia en Lluvioso

Humedad(Alta[1+,1-])

Humedad(Normal[2+,1-])

$$Gain(Lluvioso, Humedad) = 0.97 - (2/5) * 1 - (3/5) * 0.917 = 0.0198$$

- Ganancia en Temperatura

Temperatura(Caliente[0+,0-])

Temperatura(Templado[2+,1-])

Temperatura(fresco[1+,1-])

$$Gain(Lluvioso, Temperatura) = 0.97 - 0 - (3/5) * 0.917 - (2/5) * 1 = 0.0198$$

- Ganancia en Viento

Viento(Debil[3+,0-])

Viento(Fuerte[0+,2-])

$$Gain(Lluvioso, Viento) = 0.970 - (3/5) * 0 - (2/5) * 0 = 0.970$$

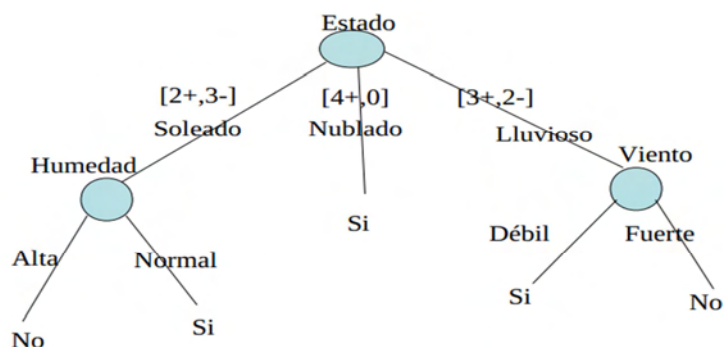
## Construcción del árbol

De los cálculos anteriores determinamos que el atributo con mayor ganancia de información es viento, por lo tanto lo ubicamos que el nodo hijo de *estado=lluvioso*.

De la Figura 69, se observa que para *viento=debil*, para cada valor se presenta una clasificación positiva sobre la clase decisión *JugarTennis*, por lo tanto el nodo esta formado por un nodo hoja en la cual la clasificación *JugarTennis=si* y el valor de la entropía =0. Igual ocurre con *viento=fuerte*, con un valor para *JugarTennis=no*.

Aquí termina la construcción del árbol pues todos los nodos finales son nodos hoja. El árbol generado se muestra en la Figura 70. Ahora es posible generar las reglas siguiendo el orden establecido en el árbol. Las reglas se muestran en la Figura 71.

Figura 70: Árbol de decisión final



Fuente. Timarán,R. Clasificación: una tarea de descubrimiento de conocimiento en bases de datos, Pasto, 2009 [49].

Figura 71: Reglas Generadas

|  |
|--|
| 1. Estado=Soleado ^ Humedad=alta -->JugarTennis = No   |
| 2. Estado=Soleado ^ Humedad=Normal -->JugarTennis = Si |
| 3. Estado=Nublado -->JugarTennis = Si                  |
| 4. Estado=Lluvioso ^ Viento=Debil -->JugarTennis = Si  |
| 5. Estado=Lluvioso ^ Viento=Fuerte -->JugarTennis = No |

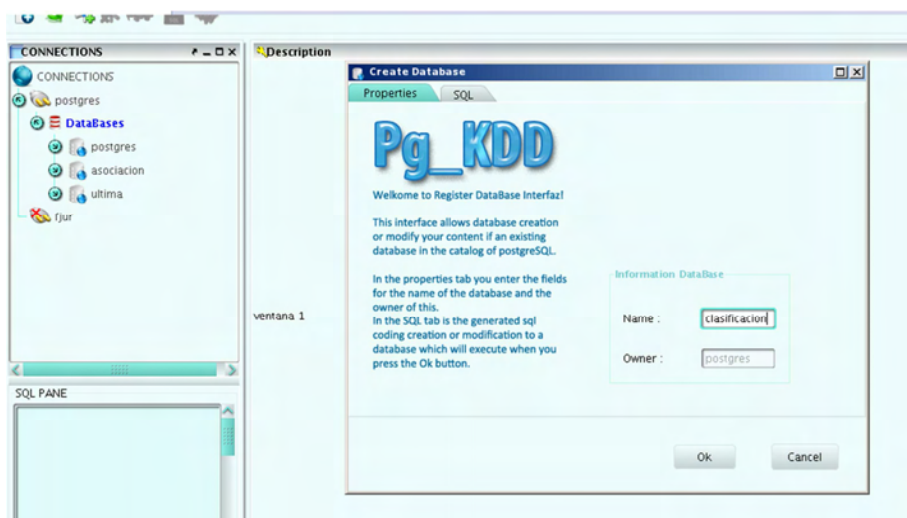
Fuente. Timarán,R. Clasificación: una tarea de descubrimiento de conocimiento en bases de datos, Pasto, 2009 [49].

## Aplicación con Pg\_KDD:

Para la tarea de clasificación, Pg\_KDD hace uso de los operadores Mate by, y de las funciones agregadas matetree(), test\_rules(), translate\_rules() y arm\_rules() para la creación del árbol de decisión.

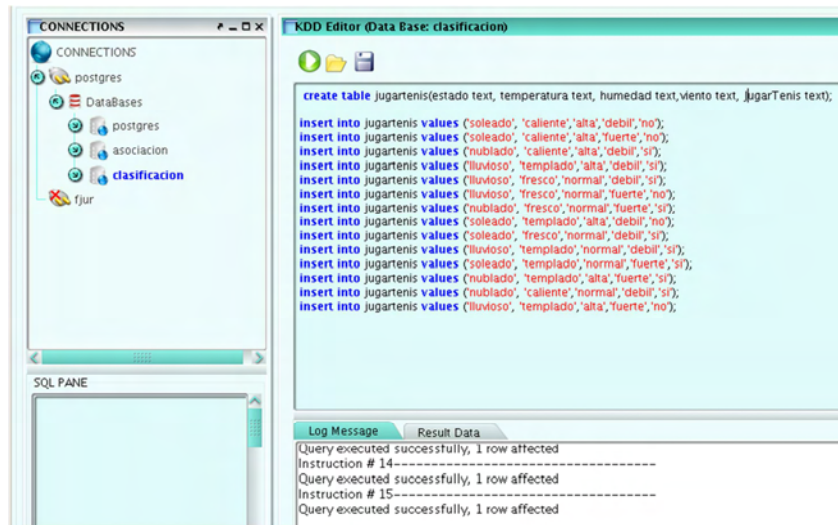
En la pantalla principal de la aplicación Pg\_KDD, se crea una nueva database llamada *clasificación* ( Figura 72).

Figura 72: Creación de la nueva database clasificacion



Se crea la tabla *JugarTenis* dentro de la database *clasificación* Figura 73.

Figura 73: Creación de la table jugartenis



Se carga el módulo de data mining y agregamos en el work area el objeto KDD repository (Figura 74). Cargamos la tabla *jugartenis*, seleccionando todos sus atributos (Figura 75). Y con clic derecho se da clic en *Run* para que el nodo se ejecute.

Figura 74: Carar repositorio para clasificación

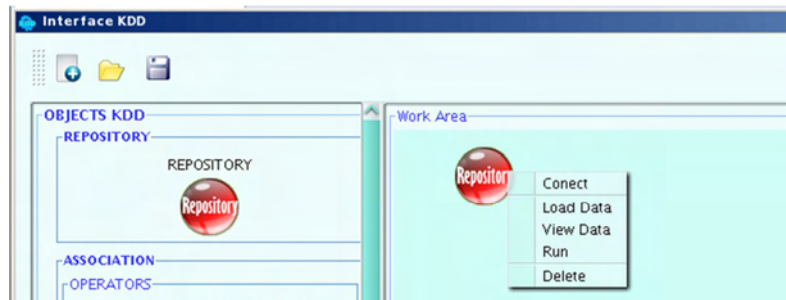
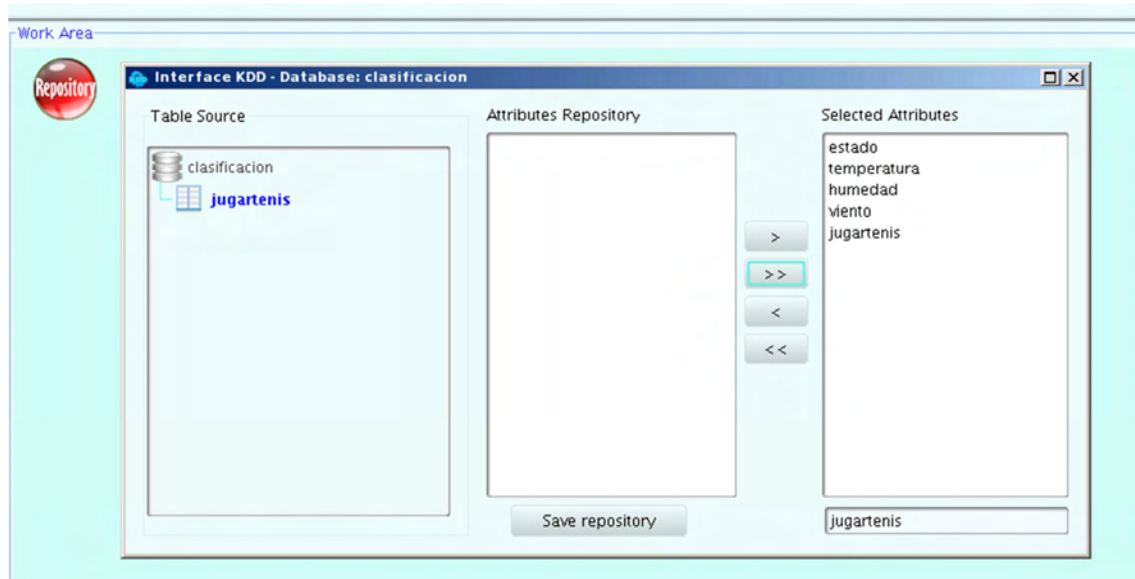


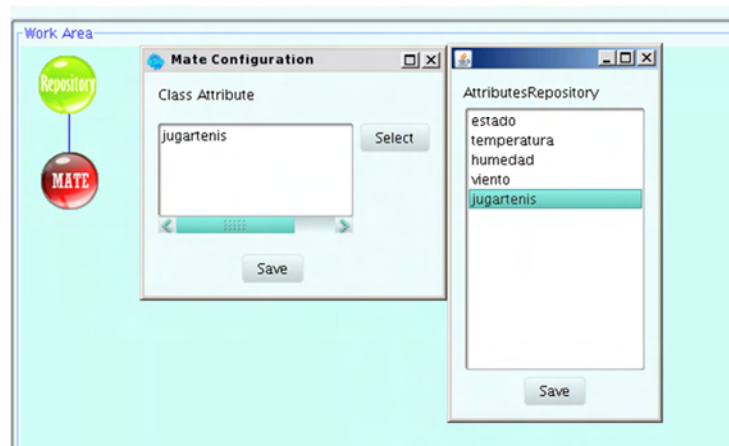
Figura 75: Selección de los atributos



### Paso 1.

Se agrega un objeto tipo *Mate* al área de trabajo. Se conecta al nodo *repository* y se configura el atributo clase (Figura 76).

Figura 76: Configuración del operador Mate



Se agrega un nodo *Table view*, se lo ejecuta y visualizamos la tabla generada por el operador *Mate By* (Figura 77).

Figura 77: Resultado del operador Mate By

The screenshot shows a database management tool interface. On the left, the 'Work Area' contains a diagram with three nodes: 'Repository', 'MATE', and 'Table View'. Below this, the 'SQL Statement' window contains the following query:

```

SELECT estado, temperatura, humedad, viento, jugartenis
_count(*) INTO tablemate
FROM repositorio_jugartenis
MATE BY estado, temperatura, humedad, viento
WITH jugartenis
GROUP BY estado, temperatura, humedad, viento
    
```

On the right, the 'VIEW TABLE MATE BY' window displays the following data:

| estado   | temperatura | humedad | viento | jugartenis | count |
|----------|-------------|---------|--------|------------|-------|
| lluvioso | fresco      | normal  | debil  | si         | 1     |
| lluvioso | fresco      | normal  | fuerte | no         | 1     |
| lluvioso | fresco      | normal  |        | no         | 1     |
| lluvioso | fresco      | normal  |        | si         | 1     |
| lluvioso | fresco      |         | debil  | si         | 1     |
| lluvioso | fresco      |         | fuerte | no         | 1     |
| lluvioso | fresco      |         |        | no         | 1     |
| lluvioso | fresco      |         |        | si         | 1     |
| lluvioso | templado    | alta    | debil  | si         | 1     |
| lluvioso | templado    | alta    | fuerte | no         | 1     |
| lluvioso | templado    | alta    |        | no         | 1     |
| lluvioso | templado    | alta    |        | si         | 1     |
| lluvioso | templado    | normal  | debil  | si         | 1     |
| lluvioso | templado    | normal  |        | si         | 1     |
| lluvioso | templado    |         | debil  | si         | 2     |
| lluvioso | templado    |         | fuerte | no         | 1     |
| lluvioso | templado    |         |        | no         | 1     |
| lluvioso | templado    |         |        | si         | 2     |
| lluvioso |             | alta    | debil  | si         | 1     |
| lluvioso |             | alta    | fuerte | no         | 1     |
| lluvioso |             | alta    |        | no         | 1     |
| lluvioso |             | alta    |        | si         | 1     |
| lluvioso |             | normal  | debil  | si         | 2     |
| lluvioso |             | normal  | fuerte | no         | 1     |
| lluvioso |             | normal  |        | no         | 1     |
| lluvioso |             | normal  |        | si         | 2     |
| lluvioso |             |         | debil  | si         | 3     |
| lluvioso |             |         | fuerte | no         | 2     |
| lluvioso |             |         |        | no         | 2     |
| lluvioso |             |         |        | si         | 3     |
| nublado  | caliente    | alta    | debil  | si         | 1     |
| nublado  | caliente    | alta    |        | si         | 1     |

## Paso 2.

Ahora, se agrega un nodo *entro* conectado al nodo *mate*, igualmente se agrega un nodo *table view* conectado al nodo *entro*, se lo configura y se visualiza el resultado de la función `matetree()` (Figura 78).

Figura 78: Cálculo de la entropía

Work Area

```

    graph LR
      Repository((Repository)) --> MATE((MATE))
      MATE --> ENTRO((ENTRO))
      MATE --> TV1((Table View))
      ENTRO --> TV2((Table View))
  
```

SQL Statement

```
select * from matetree("jugartenis");
```

VIEW TABLE ENTRO RESULT

| estado   | temperatura | humedad | viento | jugartenis | count | entro  |
|----------|-------------|---------|--------|------------|-------|--------|
|          |             |         | debil  | no         | 2     | 0.5000 |
|          |             |         | debil  | si         | 6     | 0.3113 |
|          |             |         | fuerte | no         | 3     | 0.5000 |
|          |             |         | fuerte | si         | 3     | 0.5000 |
|          | alta        |         |        | no         | 4     | 0.4613 |
|          | alta        |         |        | si         | 3     | 0.5239 |
|          | alta        | debil   |        | no         | 2     | 0.5000 |
|          | alta        | debil   |        | si         | 2     | 0.5000 |
|          | alta        | fuerte  |        | no         | 2     | 0.3900 |
|          | alta        | fuerte  |        | si         | 1     | 0.5283 |
|          | normal      |         |        | no         | 1     | 0.4011 |
|          | normal      |         |        | si         | 6     | 0.1906 |
|          | normal      | debil   |        | si         | 4     | 0.0000 |
|          | normal      | fuerte  |        | no         | 1     | 0.5283 |
|          | normal      | fuerte  |        | si         | 2     | 0.3900 |
| caliente |             |         |        | no         | 2     | 0.5000 |
| caliente |             |         |        | si         | 2     | 0.5000 |
| caliente |             |         | debil  | no         | 1     | 0.5283 |
| caliente |             |         | debil  | si         | 2     | 0.3900 |
| caliente |             |         | fuerte | no         | 1     | 0.0000 |
| caliente | alta        |         |        | no         | 2     | 0.3900 |
| caliente | alta        |         |        | si         | 1     | 0.5283 |
| caliente | alta        | debil   |        | no         | 1     | 0.5000 |
| caliente | alta        | debil   |        | si         | 1     | 0.5000 |
| caliente | alta        | fuerte  |        | no         | 1     | 0.0000 |
| caliente | normal      |         |        | si         | 1     | 0.0000 |
| caliente | normal      | debil   |        | si         | 1     | 0.0000 |
| fresco   |             |         |        | no         | 1     | 0.5000 |
| fresco   |             |         |        | si         | 3     | 0.3113 |
| fresco   |             |         | debil  | si         | 2     | 0.0000 |
| fresco   |             |         | fuerte | no         | 1     | 0.5000 |
| fresco   |             |         | fuerte | si         | 1     | 0.5000 |
| fresco   | normal      |         |        | no         | 1     | 0.5000 |
| fresco   | normal      |         |        | si         | 2     | 0.3113 |

### Paso 3.

Ahora para determinar el nodo raíz y crear el árbol de decisión, se agrega un nodo *gain()* conectado al nodo *entro()*, también se agrega un *Table view*, se ejecuta estos nodos y se visualiza el resultado de la operación de este nodo, el cual genera dos tablas, una indica el calculo de las entropías para determinar el nodo raíz, y otra con las reglas de decisión finales aunque con valores discretizados para los nodos (Figura 79). En esta última tabla se puede observar la lógica de la estructura del árbol.

Figura 79: Ganancias y Tabla con el árbol de decisión.

The screenshot displays a software interface with three main sections:

- Work Area:** A diagram showing a decision tree structure. At the top is a 'Repository' node, which connects to three nodes: 'MATE', 'ENTRO', and 'GAIN'. Each of these nodes is connected to a corresponding 'Table View' node below it.
- VIEW TABLE GAIN <2>:** A window showing a table of node gains. The table has columns: 'nodo padre', 'atributo', 'valor hijo', 'herm', and 'clase'.
 

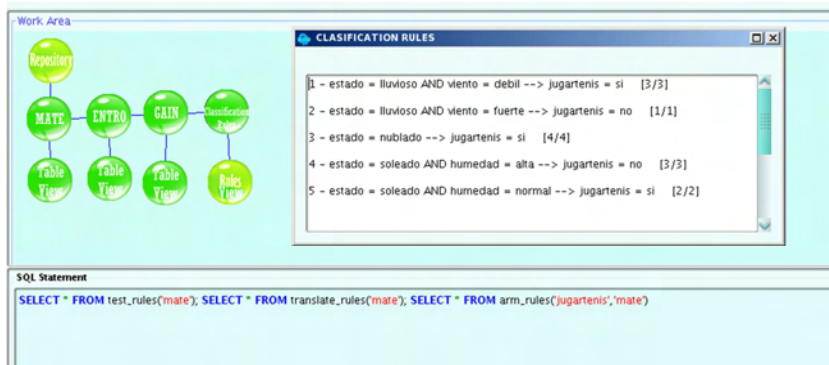
| nodo padre | atributo | valor hijo | herm | clase |
|------------|----------|------------|------|-------|
| 0          | estado   |            |      |       |
| 1          | estado   | 0          | 4    | 2     |
| 4          | viento   | 0          |      | 5     |
| 5          | viento   | 1          |      | 0     |
| 2          | estado   | 1          |      | 3     |
| 3          | estado   | 2          | 6    |       |
| 6          | humedad  | 0          |      | 7     |
| 7          | humedad  | 1          |      | 1     |
- SQL Statement:** A window showing the SQL query: `SELECT * FROM ganancias`.
- VIEW TABLE GAIN:** A window showing a table of attribute gains. The table has columns: 'atributo', 'clase', and 'ganancia'.
 

| atributo    | clase      | ganancia                |
|-------------|------------|-------------------------|
| estado      | jugartenis | 0.693571428571428571428 |
| temperatura | jugartenis | 0.911071428571428571429 |
| humedad     | jugartenis | 0.788450000000000000000 |
| viento      | jugartenis | 0.892171428571428571429 |

#### Paso 4.

Finalmente se agrega un nodo *classification rules* conectado al nodo *gain*, y un nodo *rules view*. Se ejecutan estos nodos y se visualiza las reglas ya generadas (Figura 80). En este paso se utilizan las funciones agregadas `test_rules()`, `translate_rules()` y `arm_rules()` que verifican las reglas generadas, transforman a su estado inicial los valores de las tablas y arman cada una de las reglas y devuelve reglas del tipo “*condicion1 AND condicion2.. --> Decisión*” respectivamente.

Figura 80: Reglas generadas



Del proceso anterior se verifica que las reglas que se generan corresponden a las encontradas anteriormente, como lo indica la

.Figura 81: Comparación de resultados

| Proceso Manual  | Pg_KDD  |
|---|---|
| 1. Estado=Soleado ^ Humedad=alta -->JugarTenis = No   | 1 - estado = soleado AND humedad = alta --> jugartenis = no [3/3]   |
| 2. Estado=Soleado ^ Humedad=Normal -->JugarTenis = Si | 2 - estado = soleado AND humedad = normal --> jugartenis = si [2/2] |
| 3. Estado=Nublado -->JugarTenis = Si                  | 3 - estado = nublado --> jugartenis = si [4/4]                      |
| 4. Estado=Lluvioso ^ Viento=Debil -->JugarTenis = Si  | 4 - estado = lluvioso AND viento = debil --> jugartenis = si [3/3]  |
| 5. Estado=Lluvioso ^ Viento=Fuerte -->JugarTenis = No | 5 - estado = lluvioso AND viento = fuerte --> jugartenis = no [1/1] |

Observando los resultados obtenidos del proceso manual y del proceso con Pg\_KDD, para la tarea de clasificacion el árbol construido y las reglas coinciden en su totalidad, por lo que se puede inferir que la herramienta funciona para esta tarea.

## 6.3 PRUEBAS DE RENDIMIENTO

Para verificar el rendimiento de la aplicación en escenarios reales se plantearon diferentes pruebas con diversos repositorios de datos para verificar su comportamiento. A continuación se detallan los procedimientos y los resultados.

**6.3.1 Asociación.** Para realizar las pruebas de rendimiento se seleccionaron los conjuntos de datos Universitaria y mercado. El objetivo principal es comparar el tiempo de procesamiento de los algoritmos con respecto al número de atributos y de instancias, así como medir la capacidad de la herramienta con BD grandes.

### **Prueba de rendimiento variando el numero de atributos para Associator con Ítems**

En esta prueba se medirá el tiempo de procesamiento para el objeto KDD o nodo *Associator*, realizando variaciones en el número de atributos, sin variar el número de transacciones.

En esta prueba se utilizó la base de datos Universitaria transformada (Items. Ver Tabla 43), la cual tiene 24 atributos y 25 transacciones. Se modificó este repositorio hasta alcanzar un número de transacciones igual a los 25000. Todos los registros que se agregaron con el fin de aumentar la cardinalidad de la tabla, pertenecen al mismo repositorio, así, el número de ítems clase obtenidos con el primer conjunto de transacciones es el mismo que se obtiene al procesar el último conjunto, es decir las reglas que se obtienen son similares, solamente se quiso observar el comportamiento de la herramienta con la cantidad de datos existentes en el repositorio.

Al realizar la prueba se configuró un soporte mínimo de 10% y un repositorio con 25000 transacciones para cada variación en el número de atributos. El promedio de transacciones por ítems es de 5. El nodo *Associator*, que utiliza la primitiva SQL *Associator Range*, se configuró de la siguiente manera:

- Tamaño Inicial: 1.
- Tamaño Final: 5.

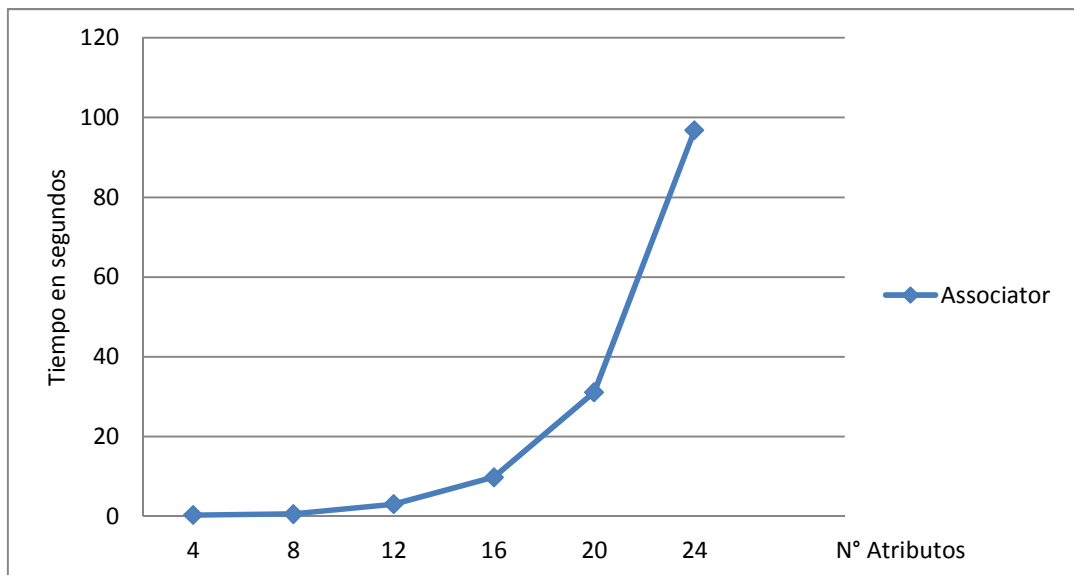
En la Tabla 53, se presentan los resultados.

Tabla 53: Prueba de rendimiento de Associator con Ítems variando N° Atributos

| Base de Datos Ítems |                                       |
|---------------------|---------------------------------------|
| N° Atributos        | Tiempo (seg) - N° transacciones 25000 |
|                     | <b>Associator</b>                     |
| 4                   | 0,3                                   |
| 8                   | 0,6                                   |
| 12                  | 3,06                                  |
| 16                  | 9,8                                   |
| 20                  | 31,14                                 |
| 24                  | 96,83                                 |

En la Figura 82, se muestra el rendimiento en segundos del operador *Associator* con diferente número de atributos.

Figura 82: Prueba de rendimiento de Associator con Ítems variando N° Atributos



De los tiempos de respuesta del nodo *Associator*, se observa que el comportamiento de la primitiva *Associator Range* tiende a ser exponencial cuando el número de atributos aumenta.

### Prueba de rendimiento variando el número de transacciones para *Associator* con Ítems

En esta prueba se utilizó la base de datos Universitaria transformada (Tabla 43), la cual tiene 24 atributos y 25 transacciones. De igual manera como se hizo en la prueba anterior se modificó este repositorio hasta alcanzar un número de transacciones igual a los 25000 con variaciones de 5000 transacciones.

Se configuró un soporte mínimo de 10% para todos los conjuntos de registros, y para cada iteración se varió el número de registros del repositorio aumentando 5000 transacciones al repositorio. Todos los registros que se agregaron con el fin de aumentar la cardinalidad de la tabla, pertenecen a las mismas divisiones de clases del repositorio. El número de atributos se mantuvo fijo en 24 para todas las iteraciones. El promedio de ítems por transacción es de 5. El nodo *Associator*, que utiliza la primitiva SQL *Associator Range*, se configuró de la siguiente manera:

- Tamaño Inicial: 1.
- Tamaño Final: 5 (Promedio de ítems por transacción).

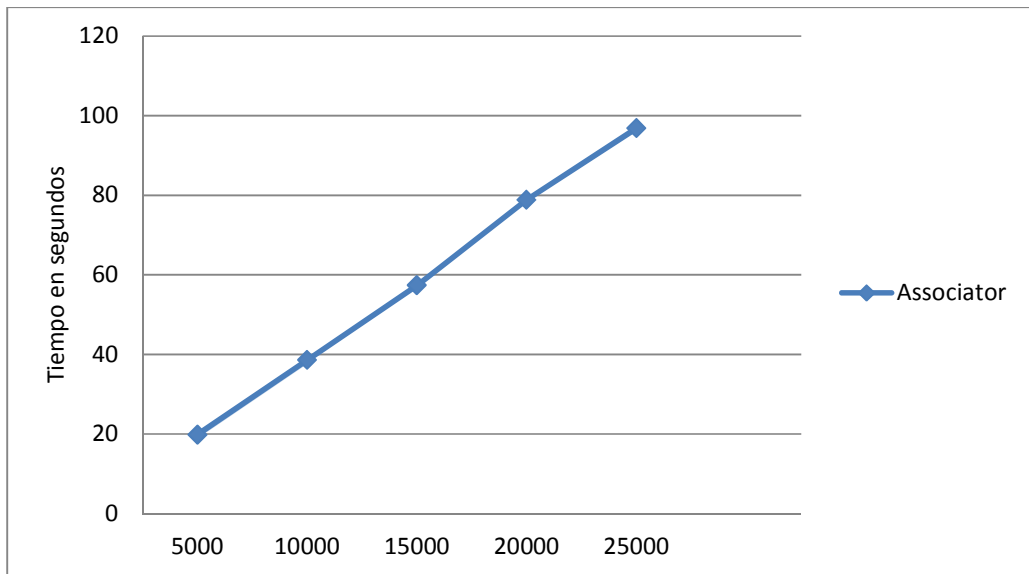
En la Tabla 54, se muestran los resultados obtenidos.

**Tabla 54: Prueba de rendimiento de *Associator* con Ítems variando N° Transacciones**

| N° Transacciones | Base de Datos Ítems            |
|------------------|--------------------------------|
|                  | Tiempo (seg) - N° Atributos 24 |
|                  | <b>Associator</b>              |
| <b>5000</b>      | 19,84                          |
| <b>10000</b>     | 38,6                           |
| <b>15000</b>     | 57,36                          |
| <b>20000</b>     | 78,85                          |
| <b>25000</b>     | 96,83                          |

En la Figura 83, se muestra el rendimiento en segundos del operador *Associator* con diferente número de transacciones.

**Figura 83: Prueba de rendimiento de Associator con Ítems variando N°Transacciones**



La gráfica anterior nos muestra que el operador *Associator* mantiene una tendencia lineal cuando se incrementa el número de registros, lo cual indica proporcionalidad entre tiempo de respuesta y número de transacciones.

### **Prueba de rendimiento variando el número de transacciones y el número de atributos para Associator con Ítems**

En esta prueba se utilizó la base de datos Universitaria transformada (Tabla 43), la cual tiene 24 atributos y 25 transacciones. De igual manera como se hizo en las dos pruebas anteriores se varía el número de transacciones (5000 – 25000 con intervalos de 5000) y además también se varía el número de atributos (4-24 con intervalos de 4).

Se configuró un soporte mínimo de 10% para todos los conjuntos de registros, y para cada iteración se varió el número de registros del repositorio aumentando 5000 transacciones al repositorio. Todos los registros que se agregaron con el fin de aumentar la cardinalidad de la tabla, pertenecen a las mismas divisiones de clases del repositorio. En cada iteración el número de atributos se aumentó en 4 hasta llegar a los 24 atributos. El nodo *Associator*, que utiliza la primitiva SQL *Associator Range*, se configuró de la siguiente manera:

- Tamaño Inicial: 1.
- Tamaño Final: 5 (Promedio de ítems por transacción)..

En la Tabla 55, se muestran los resultados obtenidos.

**Tabla 55: Prueba de rendimiento de Associator con ítems variando N° Transacciones y N° Atributos**

|                  |  | Base de Datos Ítems       |      |      |      |       |       |
|------------------|--|---------------------------|------|------|------|-------|-------|
|                  |  | Associator - Tiempo (seg) |      |      |      |       |       |
|                  |  | N° Atributos              |      |      |      |       |       |
| N° Transacciones |  | 4                         | 8    | 12   | 16   | 20    | 24    |
| <b>5000</b>      |  | 0,1                       | 0,2  | 0,5  | 1,9  | 6,93  | 19,84 |
| <b>10000</b>     |  | 0,13                      | 0,35 | 1,32 | 4,45 | 13,8  | 38,6  |
| <b>15000</b>     |  | 0,17                      | 0,46 | 1,8  | 6,09 | 18,25 | 57,36 |
| <b>20000</b>     |  | 0,25                      | 0,55 | 2,49 | 8,74 | 27,83 | 78,85 |
| <b>25000</b>     |  | 0,3                       | 0,64 | 3,06 | 9,8  | 31,14 | 96,83 |

La Figura 84 y Figura 85, nos da una perspectiva del comportamiento del operador *Associator* al variar tanto el número de atributos como el número de transacciones del repositorio sobre el cual se lo aplica.

En las dos figuras se observa el mismo comportamiento que se detalla en las 2 pruebas anteriores, con una tendencia exponencial del operador *Associator* cuando varía el número de atributos con número de transacciones constantes, y una tendencia lineal cuando la variación es del número de transacciones.

Figura 84: Prueba de rendimiento de Associator con Ítems variando N° Transacciones y N° Atributos- Tiempo vs N° de atributos

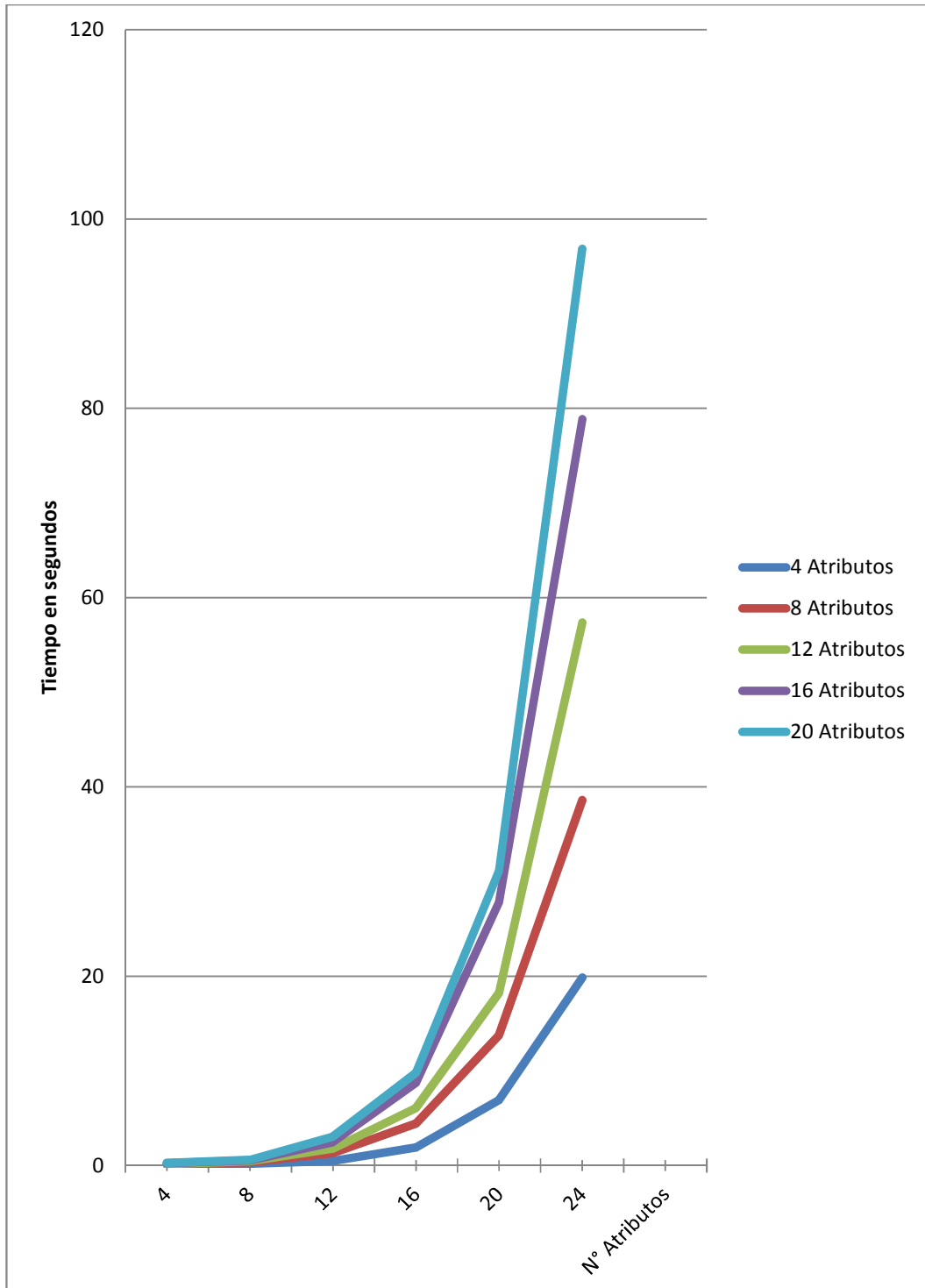
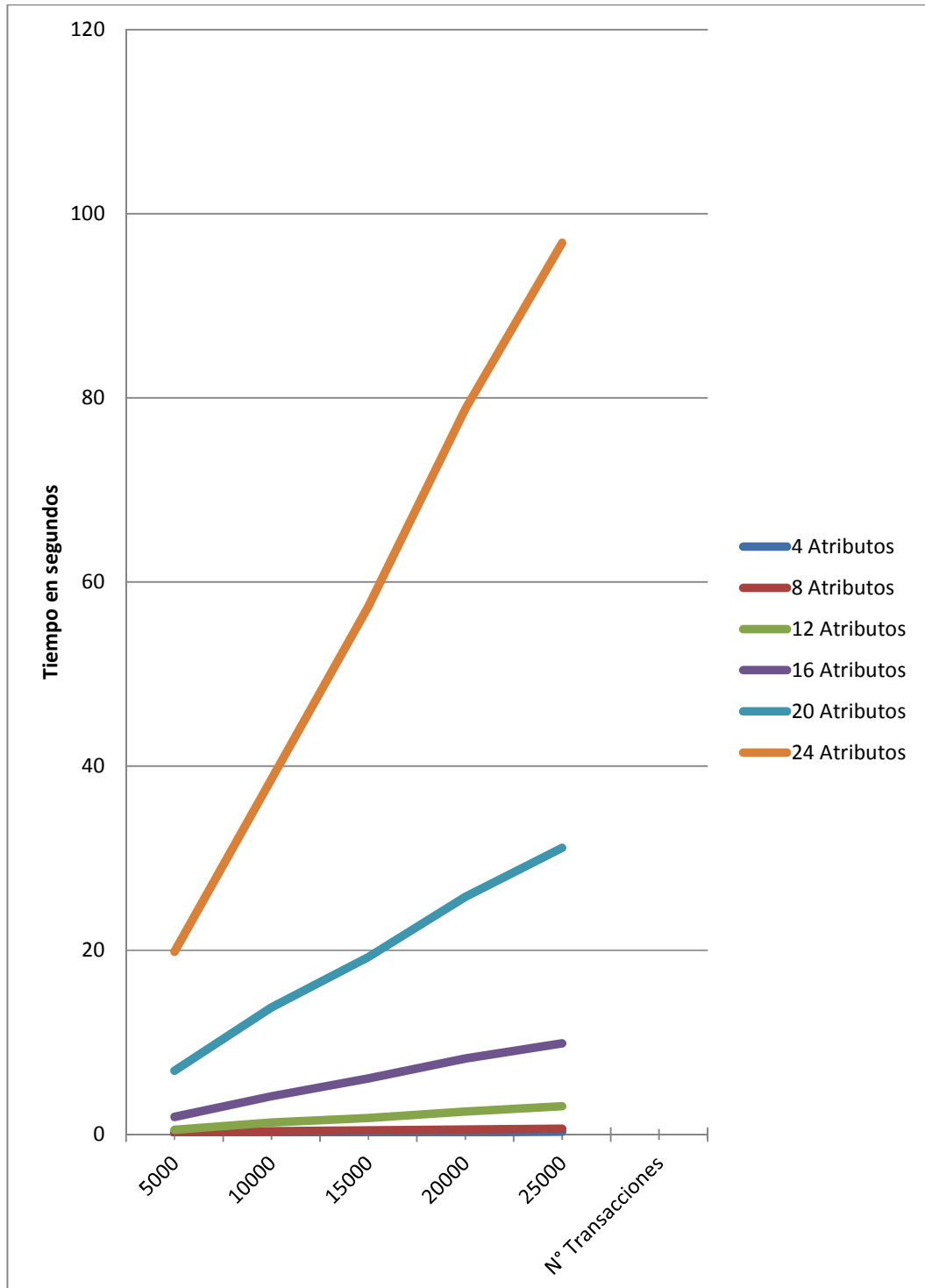


Figura 85: Prueba de rendimiento de Associator con ítems variando N° Transacciones y N° Atributos. Tiempo vs N° de transacciones.



## Prueba de rendimiento variando el número de transacciones y el número de atributos en EquipAsso con Ítems

En esta prueba se utilizó la base de datos Universitaria transformada (Tabla 43) y se varía tanto el número de transacciones (5000 – 25000 con intervalos de 5000 ) y el número de atributos (4-24 con intervalos de 4). Se tomará el tiempo de respuesta del algoritmo EquipAsso. El nodo se lo configura con un soporte del 10%. La primitiva SQL *Associator Range*, se configuró de la siguiente manera:

- Tamaño Inicial: 1.
- Tamaño Final: 5 (Promedio de ítems por transacción)..

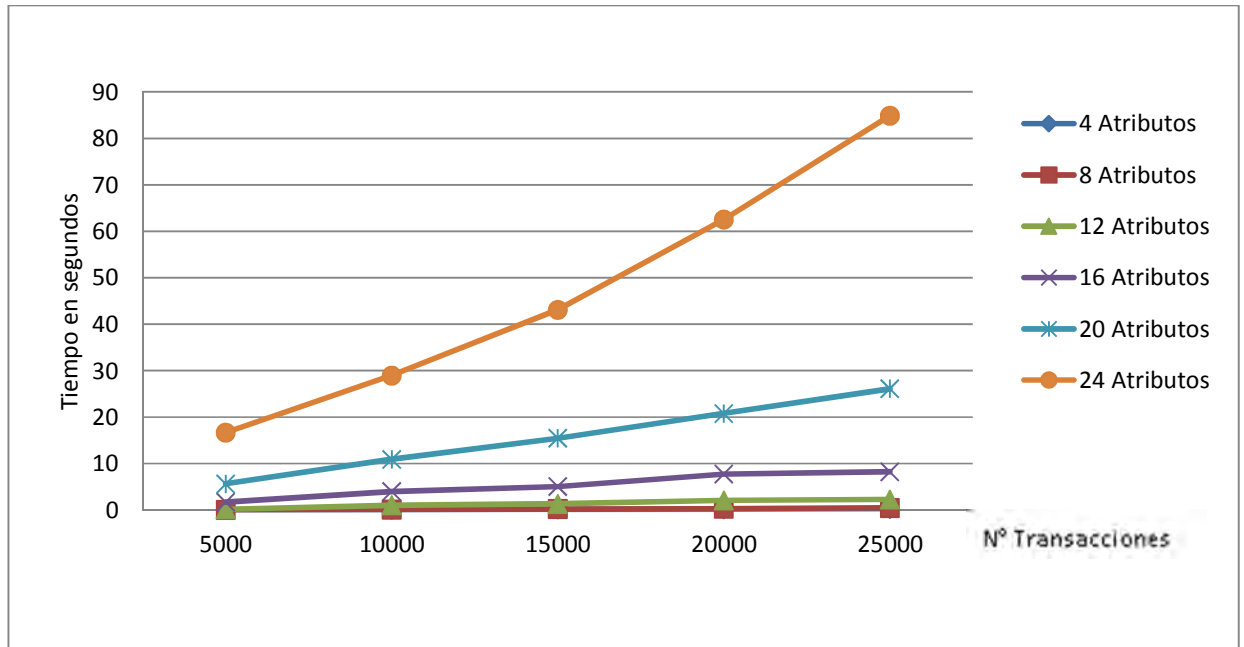
La Tabla 56, muestra los tiempos de respuesta.

**Tabla 56: Rendimiento EquipAsso con Ítems variando N° Atributos y N° Transacciones**

| N° Transacciones | Base de Datos Ítems      |      |      |      |       |       |
|------------------|--------------------------|------|------|------|-------|-------|
|                  | EquipAsso - Tiempo (seg) |      |      |      |       |       |
|                  | N° Atributos             |      |      |      |       |       |
|                  | 4                        | 8    | 12   | 16   | 20    | 24    |
| 5000             | 0,1                      | 0,1  | 0,2  | 1,74 | 5,65  | 16,7  |
| 10000            | 0,13                     | 0,17 | 1,08 | 4,02 | 10,95 | 28,98 |
| 15000            | 0,18                     | 0,25 | 1,37 | 5,1  | 15,49 | 43,13 |
| 20000            | 0,24                     | 0,33 | 2,12 | 7,76 | 20,8  | 62,6  |
| 25000            | 0,3                      | 0,5  | 2,32 | 8,27 | 26,13 | 84,93 |

La Figura 86, indica, como en la prueba de rendimiento con *Associator*, una tendencia a un comportamiento lineal para el nodo *EquipAsso*.

Figura 86: Rendimiento de EquipAsso con Ítems variando N° de Atributos y N° de Transacciones.  
Tiempo vs N° de transacciones



### Prueba de rendimiento variando el número de atributos para Associator con mercado

En esta prueba se medirá el tiempo de procesamiento para el objeto KDD o nodo *Associator*, realizando variaciones en el número de atributos, sin variar el número de transacciones.

En esta prueba se utilizó la base de datos mercado, la cual tiene 20 atributos y 27000 transacciones.

Al realizar la prueba se configuró un soporte mínimo de 10% y un repositorio con 27000 transacciones para cada incremento en el número de atributos. El promedio de transacciones por ítems es de 4. El nodo *Associator*, que utiliza la primitiva SQL *Associator Range*, se configuró de la siguiente manera:

- Tamaño Inicial: 1.
- Tamaño Final: 4 (Promedio de ítems por transacción).

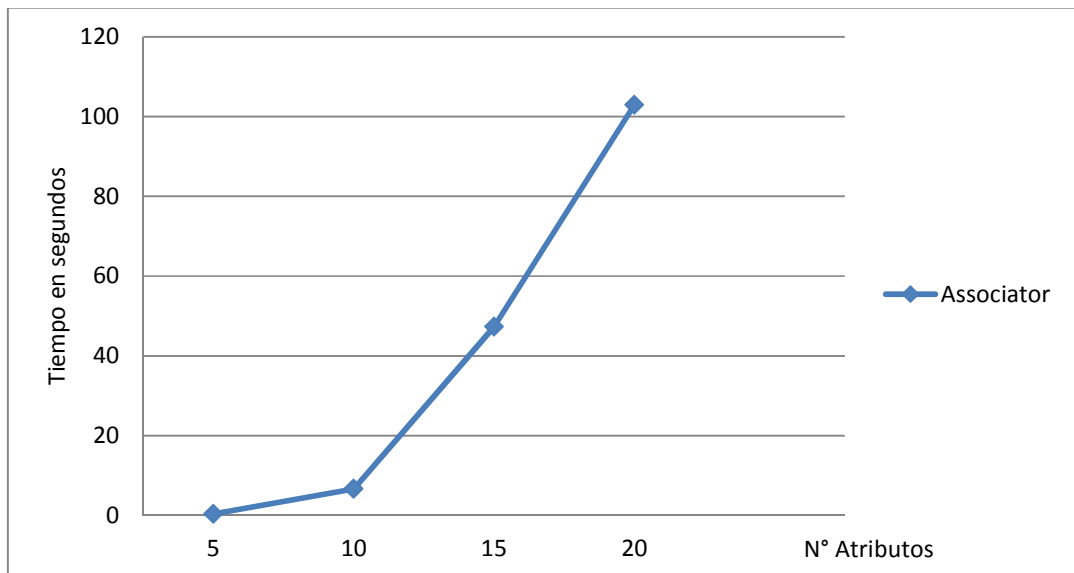
En la Tabla 57, se presentan los resultados.

**Tabla 57: Prueba de rendimiento de Associator con mercado variando N° Atributos**

| Base de Datos mercado |                                       |
|-----------------------|---------------------------------------|
| N° Atributos          | Tiempo (seg) - N°transacciones 2 7000 |
|                       | <b>Associator</b>                     |
| 5                     | 0,35                                  |
| 10                    | 6,62                                  |
| 15                    | 47,27                                 |
| 20                    | 102,9                                 |

En la Figura 87, se muestra el rendimiento en segundos del operador *Associator* con diferente número de atributos. El comportamiento de los datos tiene una tendencia a ser exponencial como se ha descrito en la prueba anterior.

**Figura 87: Prueba de rendimiento de Associator con mercado variando N° Atributos**



## Prueba de rendimiento variando el número de transacciones para Associator con mercado

En esta prueba se utilizó la base de datos mercado, la cual tiene 20 atributos y 27000 transacciones.

Al realizar la prueba se configuró un soporte mínimo de 10% para todos los conjuntos de registros y para cada iteración se varió el número de registros del repositorio aumentando 9000 transacciones al repositorio. Los registros que se adicionaron al repositorio prueba para cada iteración pertenecen al repositorio mercado, el cual se subdividió en 3 grupos de registros cada uno de 9000 transacciones.

El número de atributos se mantuvo fijo en 20 para todas las iteraciones. El promedio de transacciones por ítems es de 4. El nodo *Associator*, que utiliza la primitiva SQL *Associator Range*, se configuró de la siguiente manera:

- Tamaño Inicial: 1.
- Tamaño Final: 4 (Promedio de ítems por transacción).

En la

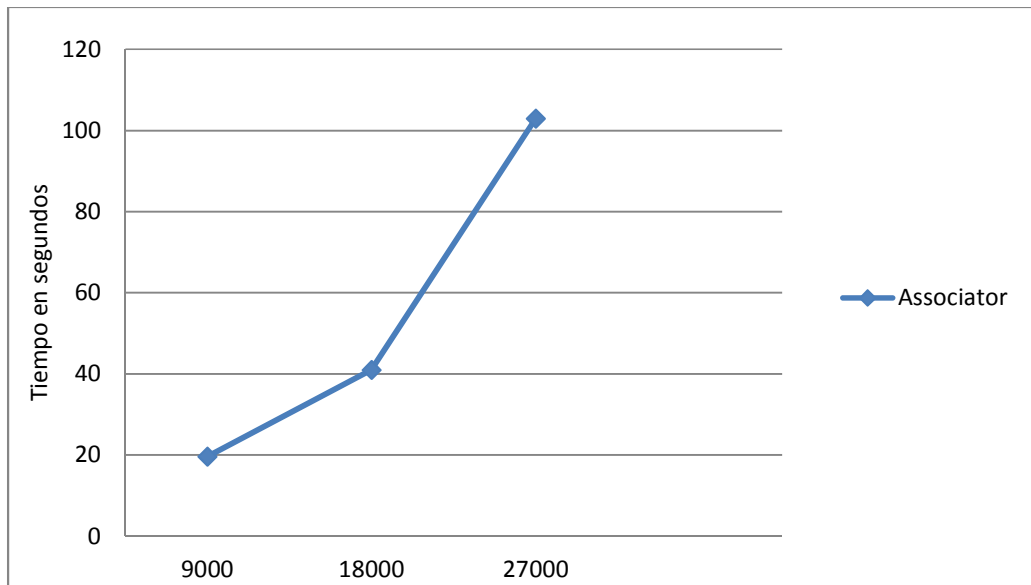
Tabla 58, se muestran los resultados obtenidos.

**Tabla 58: Prueba de rendimiento de Associator con mercado variando N° Transacciones**

| N° Transacciones | Base de Datos mercado          |
|------------------|--------------------------------|
|                  | Tiempo (seg) - N° Atributos 20 |
|                  | Associator                     |
| <b>9000</b>      | 19,6                           |
| <b>18000</b>     | 40,98                          |
| <b>27000</b>     | 102,9                          |

En la Figura 88, se muestra el rendimiento en segundos del operador *Associator* con diferente número de transacciones.

**Figura 88: Prueba de rendimiento de Associator con mercado variando N° Transacciones**



### **Prueba de rendimiento variando el número de transacciones y el número de atributos para Associator con mercado**

En esta prueba se utilizó la base de datos mercado, la cual tiene 20 atributos y 27000 transacciones.

De igual manera como se hizo en las dos pruebas anteriores se varía el número de transacciones (9000 – 27000 con intervalos de 9000 ) y además también se varía el número de atributos (5-20 con intervalos de 5).

Se configuró un soporte mínimo de 10% para todos los conjuntos de registros y para cada iteración se varió el número de registros del repositorio aumentando 9000 transacciones al repositorio. Los registros que se adicionaron al repositorio prueba para cada iteración pertenecen al repositorio mercado, el cual se subdividió en 3 grupos de registros cada uno de 9000 transacciones.

El número de atributos se mantuvo fijo en 20 para todas las iteraciones. El promedio de transacciones por ítems es de 4. El nodo *Associator*, que utiliza la primitiva SQL *Associator Range*, se configuró de la siguiente manera:

- Tamaño Inicial: 1.
- Tamaño Final: 4 (Promedio de ítems por transacción).

En la Tabla 59 , se muestran los resultados obtenidos.

**Tabla 59: Prueba de rendimiento de Associator con mercado variando N° Transacciones y N° Atributos**

| N° Transacciones | Base de Datos mercado     |      |       |       |
|------------------|---------------------------|------|-------|-------|
|                  | Associator - Tiempo (seg) |      |       |       |
|                  | N° Atributos              |      |       |       |
|                  | 5                         | 10   | 15    | 20    |
| <b>9000</b>      | 0,23                      | 1,92 | 14,24 | 32,9  |
| <b>18000</b>     | 0,38                      | 4,34 | 28,89 | 69,92 |
| <b>27000</b>     | 0,56                      | 6,62 | 45,27 | 102,9 |

La Figura 89 y la Figura 90, nos da una perspectiva del comportamiento del operador *Associator* al variar tanto el número de atributos como el número de transacciones del repositorio mercado.

Figura 89: Prueba de rendimiento de Associator con mercado variando N° Transacciones y N° Atributos. Tiempo vs N° Atributos

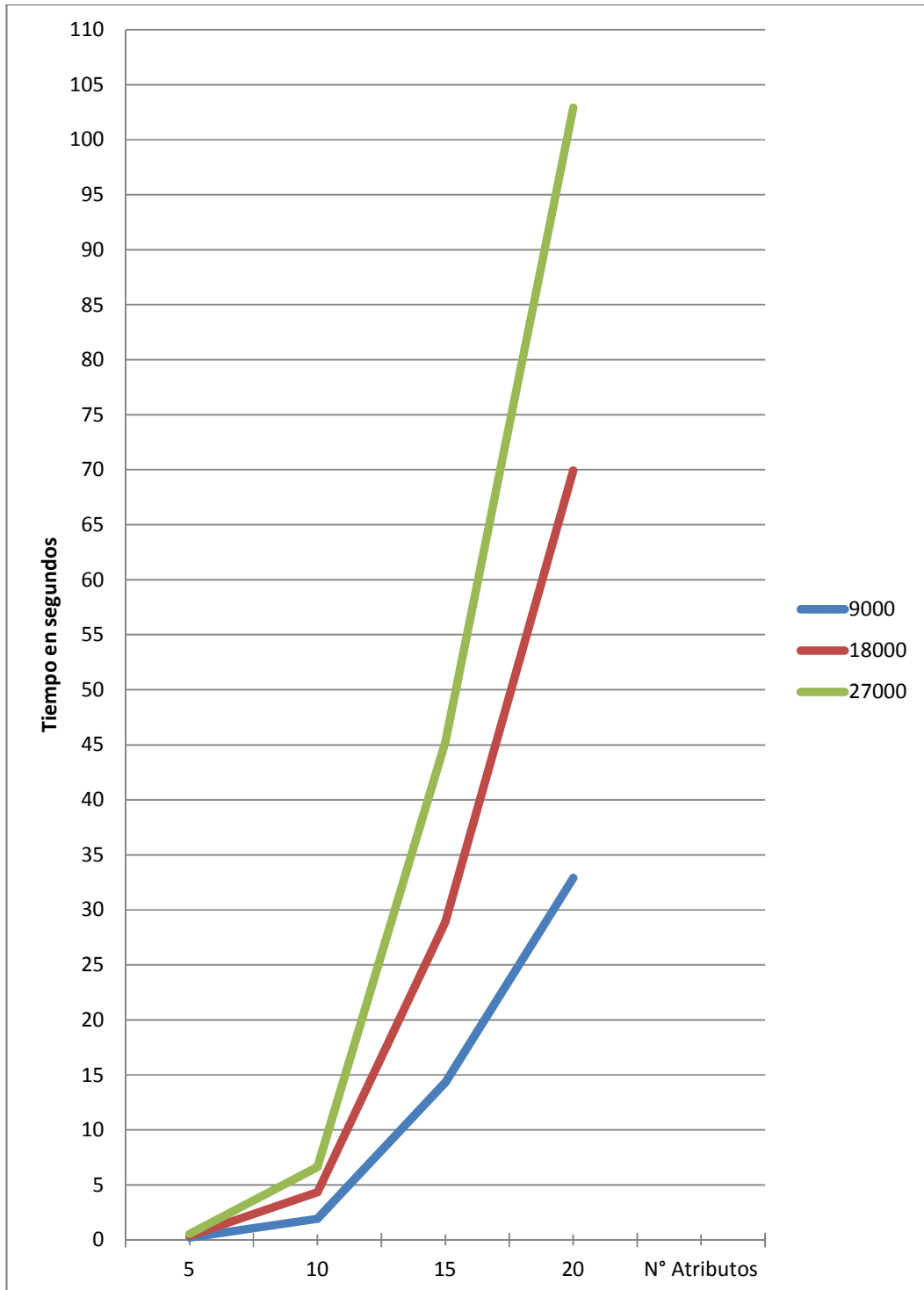
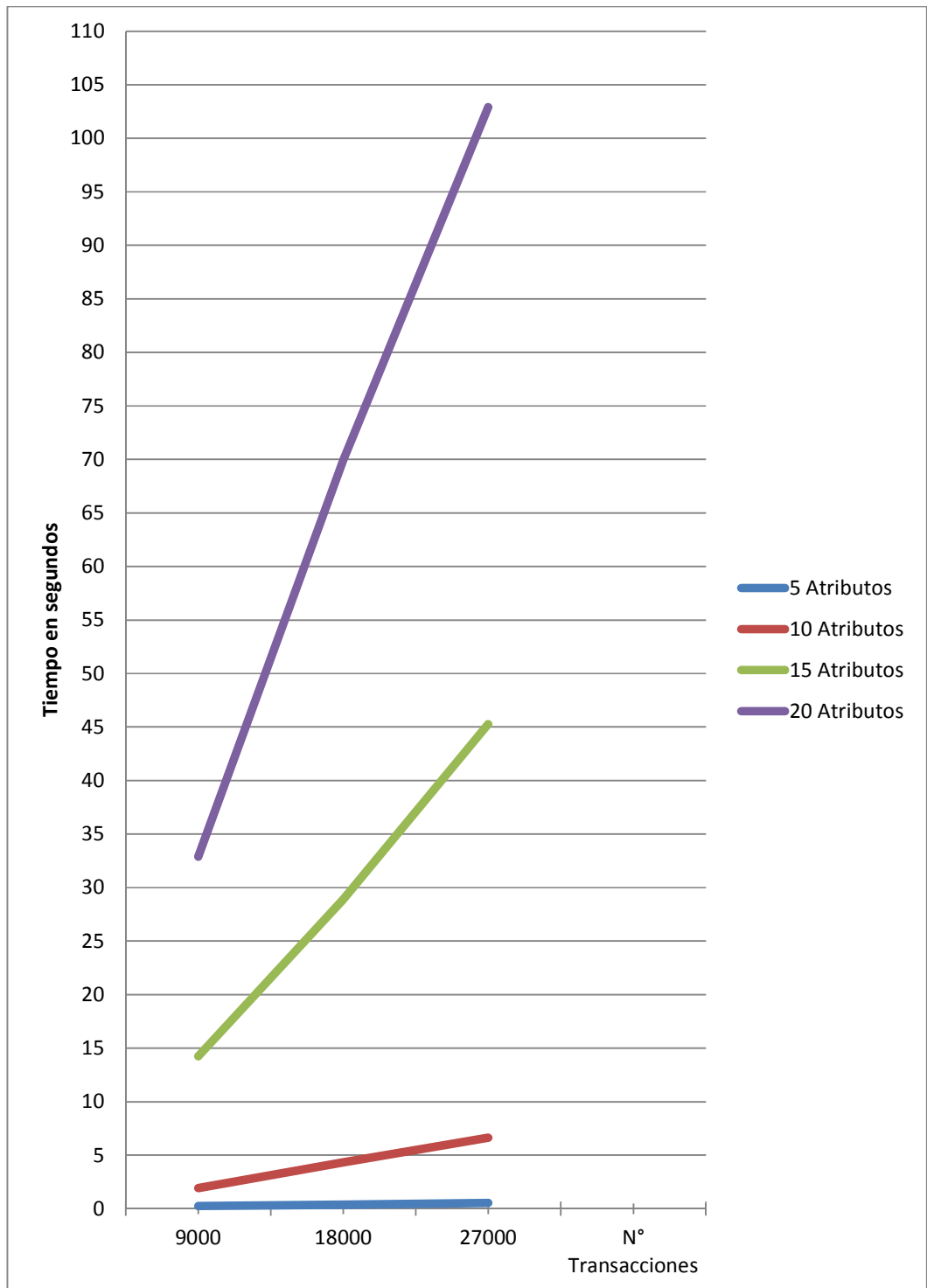


Figura 90: Prueba de rendimiento de Associator con mercado variando N° Transacciones y N° Atributos. Tiempo vs N° Transacciones



## Prueba de rendimiento variando el número de transacciones y el número de atributos en EquipAsso con mercado

En esta prueba se utilizó la base de datos mercado y se varía tanto el número de transacciones (9000 – 27000 con intervalos de 9000 ) y el número de atributos 5-20 con intervalos de 5). Se tomará el tiempo de respuesta del algoritmo EquipAsso. El nodo se lo configura con un soporte del 10%. La primitiva SQL *Associator Range*, se configuró de la siguiente manera:

- Tamaño Inicial: 1.
- Tamaño Final: 4 (Promedio de ítems por transacción)..

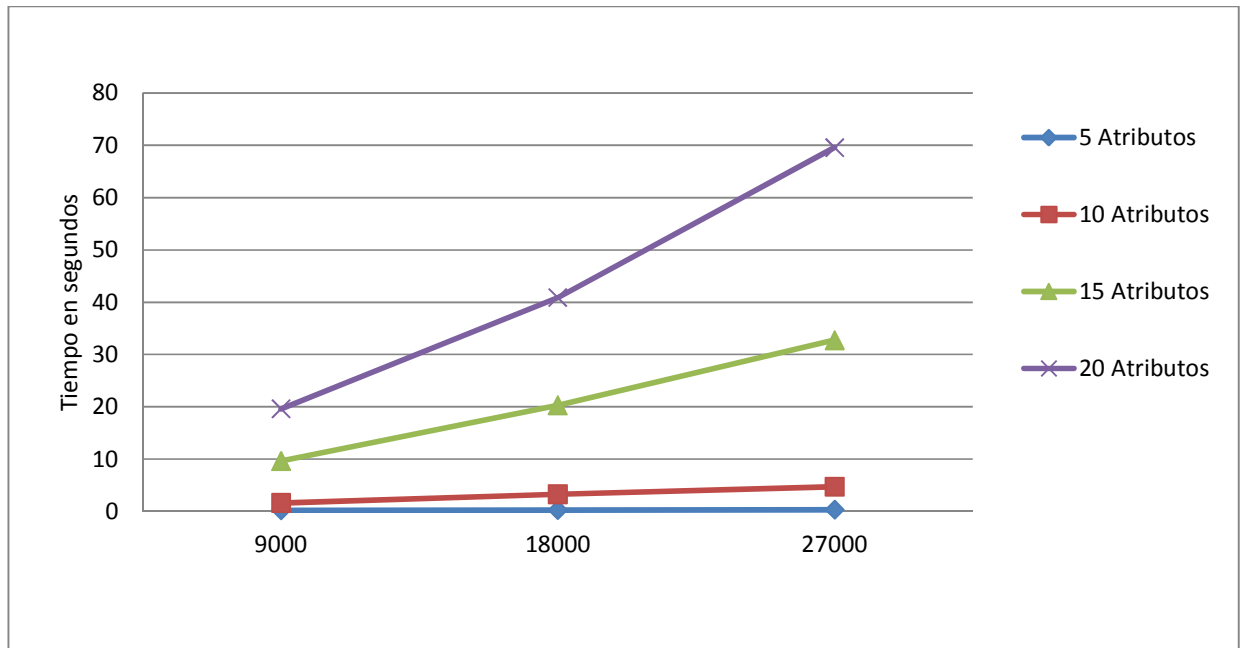
La Tabla 60, muestra los tiempos de respuesta.

**Tabla 60: Rendimiento EquipAsso con mercado variando N° Atributos y N° Transacciones**

| N° Transacciones | Base de Datos mercado    |      |       |       |
|------------------|--------------------------|------|-------|-------|
|                  | EquipAsso - Tiempo (seg) |      |       |       |
|                  | N° Atributos             |      |       |       |
|                  | 5                        | 10   | 15    | 20    |
| <b>9000</b>      | 0,18                     | 1,61 | 9,61  | 19,6  |
| <b>18000</b>     | 0,24                     | 3,26 | 20,28 | 40,89 |
| <b>27000</b>     | 0,28                     | 4,7  | 32,74 | 69,57 |

En la Figura 91Figura 83 se muestra el rendimiento en segundos del operador *Associator* con diferente número de transacciones y atributos.

Figura 91: Rendimiento de EquipAsso con Ítems variando N° de Atributos y N° de Transacciones



Para la anterior prueba se realizó una comparación de resultados con la herramienta Weka [52]. Primero se indica los itemset obtenidos con Pg\_KDD para los valores máximos de las pruebas para el repositorio mercado.

En la Figura 92 y en la Figura 93, se presentan los resultados obtenidos con Pg\_KDD para el repositorio mercado y las siguientes configuraciones para la primitiva Associator Range y para el algoritmo EquipAsso:

Configuración Associator range:

Range: 1

Until: 4 (Tamaño de la transacción)

Para EquipAsso se configuró un soporte de 10%.



Figura 93: Resultados EquipAsso con mercado

| item1 | item2 | item3 | item4 | item5 | item6 | item7 | item8 | item9 | item10 | item11 | item12 | item13 | item14 | item15 | item16 | item17 | item18 | item19 | item20 | soporte |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|---------|
| i3    |       |       |       |       |       | i8    |       |       | i11    |        |        |        |        |        |        |        |        |        |        | 2683    |
| i3    |       |       |       |       |       | i8    |       |       |        |        |        | i14    |        |        |        |        |        |        |        | 3483    |
| i3    |       |       |       |       |       | i8    |       |       |        |        |        |        |        |        |        |        |        |        |        | 3860    |
| i3    |       |       |       |       |       |       |       |       | i11    |        |        |        | i14    |        |        |        |        |        |        | 4503    |
| i3    |       |       |       |       |       |       |       |       | i11    |        |        |        |        | i15    |        |        |        |        |        | 2952    |
| i3    |       |       |       |       |       |       |       |       | i11    |        |        |        |        |        |        |        |        |        |        | 6166    |
| i3    |       |       |       |       |       |       |       |       |        |        | i12    |        |        |        |        |        |        |        |        | 2835    |
| i3    |       |       |       |       |       |       |       |       |        |        |        |        | i14    | i15    |        |        |        |        |        | 3165    |
| i3    |       |       |       |       |       |       |       |       |        |        |        |        | i14    |        |        |        |        |        |        | 7179    |
| i3    |       |       |       |       |       |       |       |       |        |        |        |        |        | i15    |        |        |        |        |        | 3861    |
| i3    |       |       |       |       |       |       |       |       |        |        |        |        |        |        |        |        |        |        |        | 10343   |
|       |       |       | i5    |       |       |       | i8    |       |        | i11    |        |        | i14    |        |        |        |        |        |        | 2684    |
|       |       |       | i5    |       |       |       | i8    |       |        | i11    |        |        |        |        |        |        |        |        |        | 3007    |
|       |       |       | i5    |       |       |       | i8    |       |        |        |        |        | i14    |        |        |        |        |        |        | 3378    |
|       |       |       | i5    |       |       |       | i8    |       |        |        |        |        |        |        |        |        |        |        |        | 3861    |
|       |       |       | i5    |       |       |       |       |       |        | i11    |        |        | i14    |        |        |        |        |        |        | 4186    |
|       |       |       | i5    |       |       |       |       |       |        | i11    |        |        |        |        |        |        |        |        |        | 5580    |
|       |       |       | i5    |       |       |       |       |       |        |        |        |        | i14    |        |        |        |        |        |        | 5792    |
|       |       |       | i5    |       |       |       |       |       |        |        |        |        |        | i15    |        |        |        |        |        | 2843    |
|       |       |       | i5    |       |       |       |       |       |        |        |        |        |        |        |        |        |        |        |        | 8037    |
|       |       |       |       | i6    |       |       |       |       |        |        |        |        |        |        |        |        |        |        |        | 3058    |
|       |       |       |       |       |       | i7    |       |       |        | i11    |        |        |        |        |        |        |        |        |        | 2947    |
|       |       |       |       |       |       | i7    |       |       |        |        |        |        |        | i14    |        |        |        |        |        | 2943    |
|       |       |       |       |       |       | i7    |       |       |        |        |        |        |        |        |        |        |        |        |        | 3796    |
|       |       |       |       |       |       |       | i8    |       |        | i11    |        |        |        | i14    |        |        |        |        |        | 3861    |
|       |       |       |       |       |       |       | i8    |       |        | i11    |        |        |        |        |        |        |        |        |        | 4558    |
|       |       |       |       |       |       |       | i8    |       |        |        |        |        |        | i14    |        |        |        |        |        | 5410    |
|       |       |       |       |       |       |       | i8    |       |        |        |        |        |        |        |        |        |        |        |        | 7551    |
|       |       |       |       |       |       |       |       | i9    |        |        |        |        |        |        |        |        |        |        |        | 2996    |
|       |       |       |       |       |       |       |       |       | i10    |        |        |        |        |        |        |        |        |        |        | 3217    |
|       |       |       |       |       |       |       |       |       |        | i11    | i12    |        |        |        |        |        |        |        |        | 3266    |
|       |       |       |       |       |       |       |       |       |        | i11    |        |        |        | i14    | i15    |        |        |        |        | 3381    |
|       |       |       |       |       |       |       |       |       |        | i11    |        |        |        | i14    |        |        |        |        |        | 8258    |
|       |       |       |       |       |       |       |       |       |        | i11    |        |        |        |        | i15    |        |        |        |        | 3969    |
|       |       |       |       |       |       |       |       |       |        | i11    |        |        |        |        |        |        |        |        |        | 13237   |
|       |       |       |       |       |       |       |       |       |        |        | i12    |        |        | i14    |        |        |        |        |        | 3959    |
|       |       |       |       |       |       |       |       |       |        |        | i12    |        |        |        |        |        |        |        |        | 5938    |
|       |       |       |       |       |       |       |       |       |        |        |        |        |        | i14    | i15    |        |        |        |        | 4399    |
|       |       |       |       |       |       |       |       |       |        |        |        |        |        | i14    |        |        |        |        |        | 15373   |
|       |       |       |       |       |       |       |       |       |        |        |        |        |        |        | i15    |        |        |        |        | 5685    |
|       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |        |        |        |        | i20    | 3159    |

En la Figura 94, se presentan las reglas obtenidas para una confianza del 50% y tamaño de regla 2, 3 y 4:

Figura 94: Reglas de asociación generadas con mercado

| ASSOCIATOR RULES |       |       |       |       |       |       |       |       |        |        |        |        |        |        |        |        |        |        |         |         |         |          |
|------------------|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|---------|---------|---------|----------|
| item1            | item2 | item3 | item4 | item5 | item6 | item7 | item8 | item9 | item10 | item11 | item12 | item13 | item14 | item15 | item16 | item17 | item18 | item19 | n_regla | implica | soporte | confianz |
| i1               | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 1       | A       | 5676    | 53.82    |
| -                | -     | i3    | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 1       | C       | 10343   |          |
| i1               | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 2       | A       | 5676    | 62.33    |
| -                | -     | -     | -     | i5    | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 2       | C       | 8037    |          |
| i1               | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 3       | A       | 5676    | 78.37    |
| -                | -     | -     | -     | -     | -     | -     | -     | -     | -      | i11    | -      | -      | -      | -      | -      | -      | -      | -      | 3       | C       | 13237   |          |
| i1               | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 4       | A       | 5676    | 74.63    |

|   |
|---|
| if i8,i11 -> i5 Confidence 65.97% (3007/4558)     |
| if i5,i8 -> i14 Confidence 87.49% (3378/3861)     |
| if i5,i14 -> i8 Confidence 58.32% (3378/5792)     |
| if i8,i14 -> i5 Confidence 62.44% (3378/5410)     |
| if i5 -> i11,i14 Confidence 52.08% (4186/8037)    |
| if i5,i11 -> i14 Confidence 75.02% (4186/5580)    |
| if i5,i14 -> i11 Confidence 72.27% (4186/5792)    |
| if i11,i14 -> i5 Confidence 50.69% (4186/8258)    |
| if i8 -> i11,i14 Confidence 51.13% (3861/7551)    |
| if i8,i11 -> i14 Confidence 84.71% (3861/4558)    |
| if i8,i14 -> i11 Confidence 71.37% (3861/5410)    |
| if i15 -> i11,i14 Confidence 59.47% (3381/5685)   |
| if i11,i15 -> i14 Confidence 85.19% (3381/3969)   |
| if i14,i15 -> i11 Confidence 76.86% (3381/4399)   |
| if i1,i5 -> i11,i14 Confidence 75.89% (2685/3538) |
| if i1,i11 -> i5,i14 Confidence 60.36% (2685/4448) |
| if i1,i14 -> i5,i11 Confidence 63.39% (2685/4236) |
| if i1,i5,i11 -> i14 Confidence 89.32% (2685/3006) |
| if i1,i5,i14 -> i11 Confidence 87.77% (2685/3059) |
| if i1,i11,i14 -> i5 Confidence 74.69% (2685/3595) |
| if i5,i11,i14 -> i1 Confidence 64.14% (2685/4186) |
| if i3,i5 -> i11,i14 Confidence 62.55% (2682/4288) |
| if i3,i5,i11 -> i14 Confidence 81.94% (2682/3273) |
| if i3,i5,i14 -> i11 Confidence 76.96% (2682/3485) |
| if i3,i11,i14 -> i5 Confidence 59.56% (2682/4503) |
| if i5,i11,i14 -> i3 Confidence 64.07% (2682/4186) |
| if i5,i8 -> i11,i14 Confidence 69.52% (2684/3861) |
| if i8,i11 -> i5,i14 Confidence 58.89% (2684/4558) |
| if i5,i8,i11 -> i14 Confidence 89.26% (2684/2007) |
| if i5,i8,i14 -> i11 Confidence 79.46% (2684/3378) |
| if i5,i11,i14 -> i8 Confidence 64.12% (2684/4186) |
| if i8,i11,i14 -> i5 Confidence 69.52% (2684/3861) |

Support: 10%

En la Figura 95 se detallan las reglas tamaño 2:

Figura 95: Reglas de asociación tamaño 2 generadas con mercado

| item1 | item2 | item3 | item4 | item5 | item6 | item7 | item8 | item9 | item10 | item11 | item12 | item13 | item14 | item15 | item16 | item17 | item18 | item19 | n_regla | implica | soporte | confianz |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|---------|---------|---------|----------|
| i1    | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 1       | A       | 5676    | 53.82    |
| -     | -     | i3    | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 1       | C       | 10343   |          |
| i1    | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 3       | A       | 5676    | 62.33    |
| -     | -     | -     | i5    | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 3       | C       | 8037    |          |
| i1    | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 7       | A       | 5676    | 78.37    |
| -     | -     | -     | -     | -     | -     | -     | -     | -     | -      | i11    | -      | -      | -      | -      | -      | -      | -      | -      | 7       | C       | 13237   |          |
| i1    | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 9       | A       | 5676    | 74.63    |

```

if i1 -> i3 Confidence 53.82% (3055/5676)
if i1 -> i5 Confidence 62.33% (3538/5676)
if i1 -> i11 Confidence 78.37% (4448/5676)
if i1 -> i14 Confidence 74.63% (4236/5676)
if i2 -> i11 Confidence 63.42% (4448/7014)
if i2 -> i14 Confidence 60.29% (4229/7014)
if i5 -> i3 Confidence 53.35% (4288/8037)
if i8 -> i3 Confidence 51.12% (3860/7551)
if i3 -> i11 Confidence 59.62% (6166/10343)
if i3 -> i14 Confidence 69.41% (7179/10343)
if i15 -> i3 Confidence 67.92% (3861/5685)
if i8 -> i5 Confidence 51.13% (3861/7551)
if i5 -> i11 Confidence 69.43% (5580/8037)
if i5 -> i14 Confidence 72.07% (5792/8037)
if i15 -> i5 Confidence 50.01% (2843/5685)
if i7 -> i11 Confidence 77.63% (2947/3796)
if i7 -> i14 Confidence 77.53% (2943/3796)
if i8 -> i11 Confidence 60.36% (4558/7551)
if i8 -> i14 Confidence 71.65% (5410/7551)
if i12 -> i11 Confidence 55.00% (3266/5938)
if i11 -> i14 Confidence 62.39% (8258/13237)
if i14 -> i11 Confidence 53.72% (8258/15373)
if i15 -> i11 Confidence 69.82% (3969/5685)
if i12 -> i14 Confidence 66.67% (3959/5938)
if i15 -> i14 Confidence 77.38% (4399/5685)
    
```

Support: 10%

En la Figura 96 se detallan las reglas tamaño 3:

Figura 96: Reglas de asociación tamaño 3 generadas con mercado

| ASSOCIATOR RULES |       |       |       |       |       |       |       |       |        |        |        |        |        |        |        |        |        |        |         |         |         |          |
|------------------|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|---------|---------|---------|----------|
| item1            | item2 | item3 | item4 | item5 | item6 | item7 | item8 | item9 | item10 | item11 | item12 | item13 | item14 | item15 | item16 | item17 | item18 | item19 | n_regla | implica | soporte | confianz |
| i1               | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 1       | A       | 5676    | 52.96    |
| -                | -     | -     | -     | i5    | -     | -     | -     | -     | -      | i11    | -      | -      | -      | -      | -      | -      | -      | -      | 1       | C       | 5580    |          |
| i1               | -     | -     | -     | i5    | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 4       | A       | 3538    | 84.96    |
| -                | -     | -     | -     | -     | -     | -     | -     | -     | -      | i11    | -      | -      | -      | -      | -      | -      | -      | -      | 4       | C       | 13237   |          |
| i1               | -     | -     | -     | -     | -     | -     | -     | -     | -      | i11    | -      | -      | -      | -      | -      | -      | -      | -      | 5       | A       | 4448    | 67.58    |
| -                | -     | -     | -     | i5    | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 5       | C       | 8037    |          |
| -                | -     | -     | -     | i5    | -     | -     | -     | -     | -      | i11    | -      | -      | -      | -      | -      | -      | -      | -      | 6       | A       | 5580    | 53.87    |

```

if i5,i11 -> i3 Confidence 58.66% (3273/5580)
if i3,i5 -> i14 Confidence 81.27% (3485/4288)
if i5,i14 -> i3 Confidence 60.17% (3485/5792)
if i3,i8 -> i11 Confidence 69.51% (2683/3860)
if i8,i11 -> i3 Confidence 58.86% (2683/4558)
if i3,i8 -> i14 Confidence 90.23% (3483/3860)
if i8,i14 -> i3 Confidence 64.38% (3483/5410)
if i3,i11 -> i14 Confidence 73.03% (4503/6166)
if i3,i14 -> i11 Confidence 62.72% (4503/7179)
if i11,i14 -> i3 Confidence 54.53% (4503/8258)
if i15 -> i3,i11 Confidence 51.93% (2952/5685)
if i3,i15 -> i11 Confidence 76.46% (2952/3861)
if i11,i15 -> i3 Confidence 74.38% (2952/3969)
if i15 -> i3,i14 Confidence 55.67% (3165/5685)
if i3,i15 -> i14 Confidence 81.97% (3165/3861)
if i14,i15 -> i3 Confidence 71.95% (3165/4399)
if i5,i8 -> i11 Confidence 77.88% (3007/3861)
if i5,i11 -> i8 Confidence 53.89% (3007/5580)
if i8,i11 -> i5 Confidence 65.97% (3007/4558)
if i5,i8 -> i14 Confidence 87.49% (3378/3861)
if i5,i14 -> i8 Confidence 58.32% (3378/5792)
if i8,i14 -> i5 Confidence 62.44% (3378/5410)
if i5 -> i11,i14 Confidence 52.08% (4186/8037)
if i5,i11 -> i14 Confidence 75.02% (4186/5580)
if i5,i14 -> i11 Confidence 72.27% (4186/5792)
if i11,i14 -> i5 Confidence 50.69% (4186/8258)
if i8 -> i11,i14 Confidence 51.13% (3861/7551)
if i8,i11 -> i14 Confidence 84.71% (3861/4558)
if i8,i14 -> i11 Confidence 71.37% (3861/5410)
if i15 -> i11,i14 Confidence 59.47% (3381/5685)
if i11,i15 -> i14 Confidence 85.19% (3381/3969)
if i14,i15 -> i11 Confidence 76.86% (3381/4399)

```

Support: 10%

En la Figura 97 se detallan las reglas tamaño 4:

Figura 97: Reglas de asociación tamaño 4 generadas con mercado

| item1 | item2 | item3 | item4 | item5 | item6 | item7 | item8 | item9 | item10 | item11 | item12 | item13 | item14 | item15 | item16 | item17 | item18 | item19 | n_regla | implica | soporte | confianz |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|---------|---------|---------|----------|
| i1    | -     | -     | -     | i5    | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -      | -      | -      | -      | 5       | A       | 3538    | 75.89    |
| -     | -     | -     | -     | -     | -     | -     | -     | -     | -      | i11    | -      | -      | i14    | -      | -      | -      | -      | -      | 5       | C       | 8258    |          |
| i1    | -     | -     | -     | -     | -     | -     | -     | -     | -      | i11    | -      | -      | -      | -      | -      | -      | -      | -      | 6       | A       | 4448    | 60.36    |
| -     | -     | -     | -     | i5    | -     | -     | -     | -     | -      | -      | -      | -      | i14    | -      | -      | -      | -      | -      | 6       | C       | 5792    |          |
| i1    | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | i14    | -      | -      | -      | -      | -      | 7       | A       | 4236    | 63.39    |
| -     | -     | -     | -     | i5    | -     | -     | -     | -     | -      | i11    | -      | -      | -      | -      | -      | -      | -      | -      | 7       | C       | 5580    |          |
| i1    | -     | -     | -     | i5    | -     | -     | -     | -     | -      | i11    | -      | -      | -      | -      | -      | -      | -      | -      | 11      | A       | 3006    | 89.32    |

```

if i1,i5 -> i11,i14 Confidence 75.89% (2685/3538)
if i1,i11 -> i5,i14 Confidence 60.36% (2685/4448)
if i1,i14 -> i5,i11 Confidence 63.39% (2685/4236)
if i1,i5,i11 -> i14 Confidence 89.32% (2685/3006)
if i1,i5,i14 -> i11 Confidence 87.77% (2685/3059)
if i1,i11,i14 -> i5 Confidence 74.69% (2685/3595)
if i5,i11,i14 -> i1 Confidence 64.14% (2685/4186)
if i3,i5 -> i11,i14 Confidence 62.55% (2682/4288)
if i3,i5,i11 -> i14 Confidence 81.94% (2682/3273)
if i3,i5,i14 -> i11 Confidence 76.96% (2682/3485)
if i3,i11,i14 -> i5 Confidence 59.56% (2682/4503)
if i5,i11,i14 -> i3 Confidence 64.07% (2682/4186)
if i5,i8 -> i11,i14 Confidence 69.52% (2684/3861)
if i8,i11 -> i5,i14 Confidence 58.89% (2684/4558)
if i5,i8,i11 -> i14 Confidence 89.26% (2684/3007)
if i5,i8,i14 -> i11 Confidence 79.46% (2684/3378)
if i5,i11,i14 -> i8 Confidence 64.12% (2684/4186)
if i8,i11,i14 -> i5 Confidence 69.52% (2684/3861)
    
```

Support: 10%

A continuación en la Tabla 61 se muestra más detalladamente las reglas generadas para mercado con Pg\_KDD.

Tabla 61: Reglas de asociación tamaño 2, 3 y 4 generadas con mercado

| Reglas generadas: tamaño 2, 3 y 4           |
|---|
| if i1 -> i3 Confidence 53.82% (3055/5676)   |
| if i1 -> i5 Confidence 62.33% (3538/5676)   |
| if i1 -> i11 Confidence 78.37% (4448/5676)  |
| if i1 -> i14 Confidence 74.63% (4236/5676)  |
| if i2 -> i11 Confidence 63.42% (4448/7014)  |
| if i2 -> i14 Confidence 60.29% (4229/7014)  |
| if i5 -> i3 Confidence 53.35% (4288/8037)   |
| if i8 -> i3 Confidence 51.12% (3860/7551)   |
| if i3 -> i11 Confidence 59.62% (6166/10343) |
| if i3 -> i14 Confidence 69.41% (7179/10343) |
| if i15 -> i3 Confidence 67.92% (3861/5685)  |
| if i8 -> i5 Confidence 51.13% (3861/7551)   |
| if i5 -> i11 Confidence 69.43% (5580/8037)  |
| if i5 -> i14 Confidence 72.07% (5792/8037)  |
| if i15 -> i5 Confidence 50.01% (2843/5685)  |
| if i7 -> i11 Confidence 77.63% (2947/3796)  |

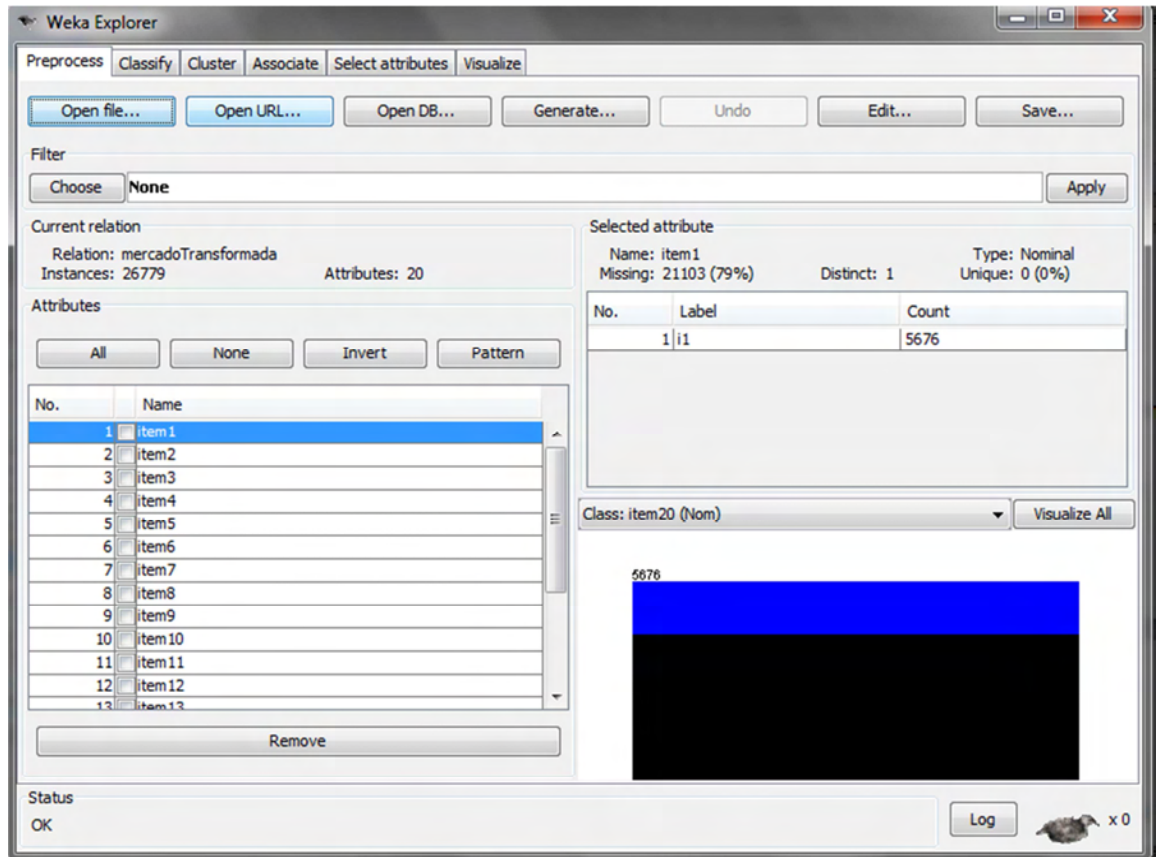
| Reglas generadas: tamaño 2, 3 y 4              |  |
|--|--|
| if i7 -> i14 Confidence 77.53% (2943/3796)     |  |
| if i8 -> i11 Confidence 60.36% (4558/7551)     |  |
| if i8 -> i14 Confidence 71.65% (5410/7551)     |  |
| if i12 -> i11 Confidence 55.00% (3266/5938)    |  |
| if i11 -> i14 Confidence 62.39% (8258/13237)   |  |
| if i14 -> i11 Confidence 53.72% (8258/15373)   |  |
| if i15 -> i11 Confidence 69.82% (3969/5685)    |  |
| if i12 -> i14 Confidence 66.67% (3959/5938)    |  |
| if i15 -> i14 Confidence 77.38% (4399/5685)    |  |
| if i1 -> i5,i11 Confidence 52.96% (3006/5676)  |  |
| if i1,i5 -> i11 Confidence 84.96% (3006/3538)  |  |
| if i1,i11 -> i5 Confidence 67.58% (3006/4448)  |  |
| if i5,i11 -> i1 Confidence 53.87% (3006/5580)  |  |
| if i1 -> i5,i14 Confidence 53.89% (3059/5676)  |  |
| if i1,i5 -> i14 Confidence 86.46% (3059/3538)  |  |
| if i1,i14 -> i5 Confidence 72.21% (3059/4236)  |  |
| if i5,i14 -> i1 Confidence 52.81% (3059/5792)  |  |
| if i1 -> i11,i14 Confidence 63.34% (3595/5676) |  |
| if i1,i11 -> i14 Confidence 80.82% (3595/4448) |  |
| if i1,i14 -> i11 Confidence 84.87% (3595/4236) |  |
| if i2,i11 -> i14 Confidence 69.81% (3105/4448) |  |
| if i2,i14 -> i11 Confidence 73.42% (3105/4229) |  |
| if i3,i5 -> i11 Confidence 76.33% (3273/4288)  |  |
| if i3,i11 -> i5 Confidence 53.08% (3273/6166)  |  |
| if i5,i11 -> i3 Confidence 58.66% (3273/5580)  |  |
| if i3,i5 -> i14 Confidence 81.27% (3485/4288)  |  |
| if i5,i14 -> i3 Confidence 60.17% (3485/5792)  |  |
| if i3,i8 -> i11 Confidence 69.51% (2683/3860)  |  |
| if i8,i11 -> i3 Confidence 58.86% (2683/4558)  |  |
| if i3,i8 -> i14 Confidence 90.23% (3483/3860)  |  |
| if i8,i14 -> i3 Confidence 64.38% (3483/5410)  |  |
| if i3,i11 -> i14 Confidence 73.03% (4503/6166) |  |
| if i3,i14 -> i11 Confidence 62.72% (4503/7179) |  |
| if i11,i14 -> i3 Confidence 54.53% (4503/8258) |  |
| if i15 -> i3,i11 Confidence 51.93% (2952/5685) |  |
| if i3,i15 -> i11 Confidence 76.46% (2952/3861) |  |
| if i11,i15 -> i3 Confidence 74.38% (2952/3969) |  |
| if i15 -> i3,i14 Confidence 55.67% (3165/5685) |  |
| if i3,i15 -> i14 Confidence 81.97% (3165/3861) |  |
| if i14,i15 -> i3 Confidence 71.95% (3165/4399) |  |
| if i5,i8 -> i11 Confidence 77.88% (3007/3861)  |  |
| if i5,i11 -> i8 Confidence 53.89% (3007/5580)  |  |
| if i8,i11 -> i5 Confidence 65.97% (3007/4558)  |  |
| if i5,i8 -> i14 Confidence 87.49% (3378/3861)  |  |
| if i5,i14 -> i8 Confidence 58.32% (3378/5792)  |  |
| if i8,i14 -> i5 Confidence 62.44% (3378/5410)  |  |
| if i5 -> i11,i14 Confidence 52.08% (4186/8037) |  |

| Reglas generadas: tamaño 2, 3 y 4                 |  |
|---|--|
| if i5,i11 -> i14 Confidence 75.02% (4186/5580)    |  |
| if i5,i14 -> i11 Confidence 72.27% (4186/5792)    |  |
| if i11,i14 -> i5 Confidence 50.69% (4186/8258)    |  |
| if i8 -> i11,i14 Confidence 51.13% (3861/7551)    |  |
| if i8,i11 -> i14 Confidence 84.71% (3861/4558)    |  |
| if i8,i14 -> i11 Confidence 71.37% (3861/5410)    |  |
| if i15 -> i11,i14 Confidence 59.47% (3381/5685)   |  |
| if i11,i15 -> i14 Confidence 85.19% (3381/3969)   |  |
| if i14,i15 -> i11 Confidence 76.86% (3381/4399)   |  |
| if i1,i5 -> i11,i14 Confidence 75.89% (2685/3538) |  |
| if i1,i11 -> i5,i14 Confidence 60.36% (2685/4448) |  |
| if i1,i14 -> i5,i11 Confidence 63.39% (2685/4236) |  |
| if i1,i5,i11 -> i14 Confidence 89.32% (2685/3006) |  |
| if i1,i5,i14 -> i11 Confidence 87.77% (2685/3059) |  |
| if i1,i11,i14 -> i5 Confidence 74.69% (2685/3595) |  |
| if i5,i11,i14 -> i1 Confidence 64.14% (2685/4186) |  |
| if i3,i5 -> i11,i14 Confidence 62.55% (2682/4288) |  |
| if i3,i5,i11 -> i14 Confidence 81.94% (2682/3273) |  |
| if i3,i5,i14 -> i11 Confidence 76.96% (2682/3485) |  |
| if i3,i11,i14 -> i5 Confidence 59.56% (2682/4503) |  |
| if i5,i11,i14 -> i3 Confidence 64.07% (2682/4186) |  |
| if i5,i8 -> i11,i14 Confidence 69.52% (2684/3861) |  |
| if i8,i11 -> i5,i14 Confidence 58.89% (2684/4558) |  |
| if i5,i8,i11 -> i14 Confidence 89.26% (2684/3007) |  |
| if i5,i8,i14 -> i11 Confidence 79.46% (2684/3378) |  |
| if i5,i11,i14 -> i8 Confidence 64.12% (2684/4186) |  |
| if i8,i11,i14 -> i5 Confidence 69.52% (2684/3861) |  |
|   |  |
| Support: 10%                                      |  |

Para generar las reglas de asociación para mercado con Weka se siguieron los siguientes pasos:

- Selección del repositorio Items (Figura 98)

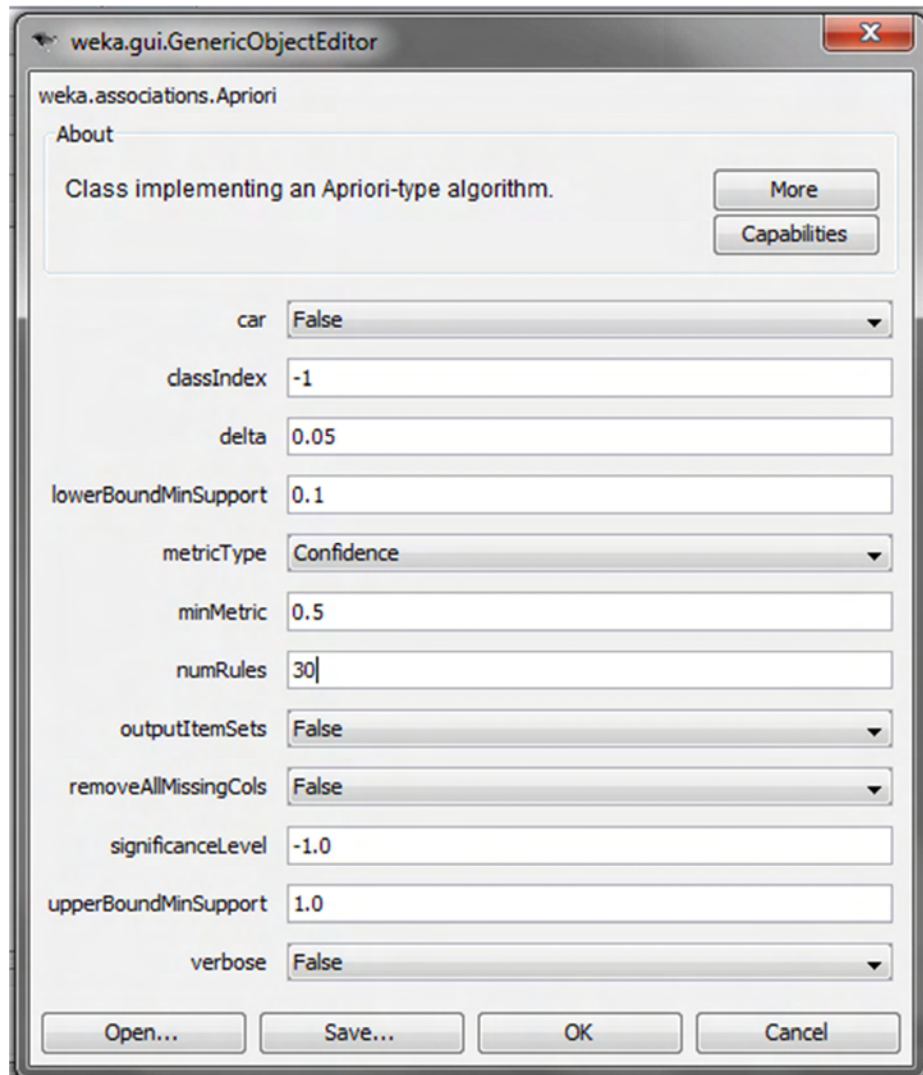
Figura 98: Selección de Repositorio Ítems en Weka



Fuente. Esta investigación

- Configuración del algoritmo Apriori para descubrimiento de reglas de asociación (Figura 99).

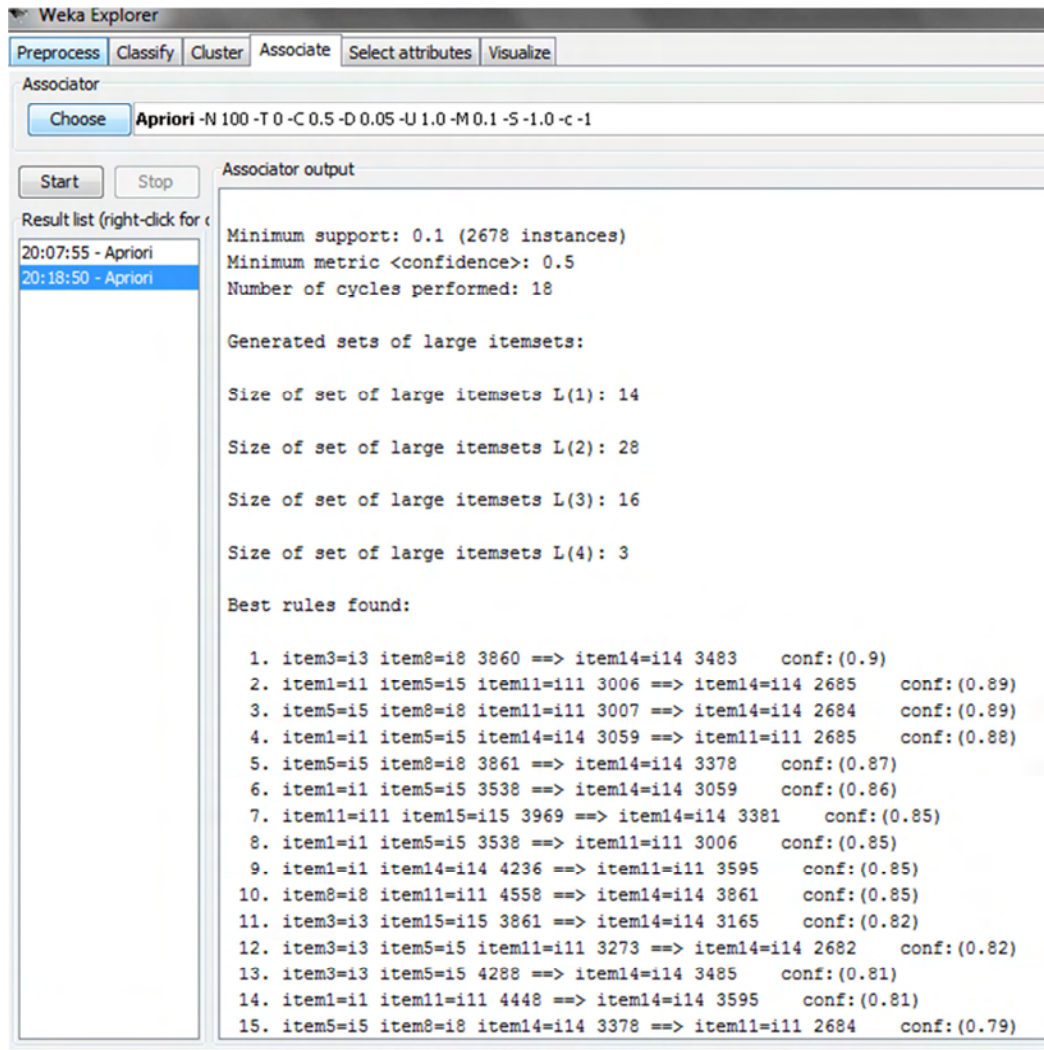
Figura 99: Configuración algoritmo Apriori en Weka



Fuente. Esta investigación

- Generación de reglas en Weka (ver Figura 100).

Figura 100: Generación de reglas con Weka



Fuente. Esta investigación

La Tabla 62 muestra las reglas generadas con Weka para el repositorio mercado.

Tabla 62: Reglas generadas con Weka para mercado

| Reglas generadas con Weka para mercado                   |             |
|--|-------------|
| 1. item3=i3 item8=i8 3860 ==> item14=i14 3483            | conf:(0.9)  |
| 2. item1=i1 item5=i5 item11=i11 3006 ==> item14=i14 2685 | conf:(0.89) |
| 3. item5=i5 item8=i8 item11=i11 3007 ==> item14=i14 2684 | conf:(0.89) |

| Reglas generadas con Weka para mercado |   |
|--|---|
| 4.                                     | item1=i1 item5=i5 item14=i14 3059 ==> item11=i11 2685 conf:(0.88) |
| 5.                                     | item5=i5 item8=i8 3861 ==> item14=i14 3378 conf:(0.87)            |
| 6.                                     | item1=i1 item5=i5 3538 ==> item14=i14 3059 conf:(0.86)            |
| 7.                                     | item11=i11 item15=i15 3969 ==> item14=i14 3381 conf:(0.85)        |
| 8.                                     | item1=i1 item5=i5 3538 ==> item11=i11 3006 conf:(0.85)            |
| 9.                                     | item1=i1 item14=i14 4236 ==> item11=i11 3595 conf:(0.85)          |
| 10.                                    | item8=i8 item11=i11 4558 ==> item14=i14 3861 conf:(0.85)          |
| 11.                                    | item3=i3 item15=i15 3861 ==> item14=i14 3165 conf:(0.82)          |
| 12.                                    | item3=i3 item5=i5 item11=i11 3273 ==> item14=i14 2682 conf:(0.82) |
| 13.                                    | item3=i3 item5=i5 4288 ==> item14=i14 3485 conf:(0.81)            |
| 14.                                    | item1=i1 item11=i11 4448 ==> item14=i14 3595 conf:(0.81)          |
| 15.                                    | item5=i5 item8=i8 item14=i14 3378 ==> item11=i11 2684 conf:(0.79) |
| 16.                                    | item1=i1 5676 ==> item11=i11 4448 conf:(0.78)                     |
| 17.                                    | item5=i5 item8=i8 3861 ==> item11=i11 3007 conf:(0.78)            |
| 18.                                    | item7=i7 3796 ==> item11=i11 2947 conf:(0.78)                     |
| 19.                                    | item7=i7 3796 ==> item14=i14 2943 conf:(0.78)                     |
| 20.                                    | item15=i15 5685 ==> item14=i14 4399 conf:(0.77)                   |
| 21.                                    | item3=i3 item5=i5 item14=i14 3485 ==> item11=i11 2682 conf:(0.77) |
| 22.                                    | item14=i14 item15=i15 4399 ==> item11=i11 3381 conf:(0.77)        |
| 23.                                    | item3=i3 item15=i15 3861 ==> item11=i11 2952 conf:(0.76)          |
| 24.                                    | item3=i3 item5=i5 4288 ==> item11=i11 3273 conf:(0.76)            |
| 25.                                    | item1=i1 item5=i5 3538 ==> item11=i11 item14=i14 2685 conf:(0.76) |
| 26.                                    | item5=i5 item11=i11 5580 ==> item14=i14 4186 conf:(0.75)          |
| 27.                                    | item1=i1 item11=i11 item14=i14 3595 ==> item5=i5 2685 conf:(0.75) |
| 28.                                    | item1=i1 5676 ==> item14=i14 4236 conf:(0.75)                     |
| 29.                                    | item11=i11 item15=i15 3969 ==> item3=i3 2952 conf:(0.74)          |
| 30.                                    | item2=i2 item14=i14 4229 ==> item11=i11 3105 conf:(0.73)          |
| 31.                                    | item3=i3 item11=i11 6166 ==> item14=i14 4503 conf:(0.73)          |
| 32.                                    | item5=i5 item14=i14 5792 ==> item11=i11 4186 conf:(0.72)          |
| 33.                                    | item1=i1 item14=i14 4236 ==> item5=i5 3059 conf:(0.72)            |
| 34.                                    | item5=i5 8037 ==> item14=i14 5792 conf:(0.72)                     |
| 35.                                    | item14=i14 item15=i15 4399 ==> item3=i3 3165 conf:(0.72)          |
| 36.                                    | item8=i8 7551 ==> item14=i14 5410 conf:(0.72)                     |
| 37.                                    | item8=i8 item14=i14 5410 ==> item11=i11 3861 conf:(0.71)          |
| 38.                                    | item15=i15 5685 ==> item11=i11 3969 conf:(0.7)                    |
| 39.                                    | item2=i2 item11=i11 4448 ==> item14=i14 3105 conf:(0.7)           |
| 40.                                    | item8=i8 item11=i11 item14=i14 3861 ==> item5=i5 2684 conf:(0.7)  |
| 41.                                    | item5=i5 item8=i8 3861 ==> item11=i11 item14=i14 2684 conf:(0.7)  |
| 42.                                    | item3=i3 item8=i8 3860 ==> item11=i11 2683 conf:(0.7)             |
| 43.                                    | item5=i5 8037 ==> item11=i11 5580 conf:(0.69)                     |
| 44.                                    | item3=i3 10343 ==> item14=i14 7179 conf:(0.69)                    |
| 45.                                    | item15=i15 5685 ==> item3=i3 3861 conf:(0.68)                     |
| 46.                                    | item1=i1 item11=i11 4448 ==> item5=i5 3006 conf:(0.68)            |
| 47.                                    | item12=i12 5938 ==> item14=i14 3959 conf:(0.67)                   |
| 48.                                    | item8=i8 item11=i11 4558 ==> item5=i5 3007 conf:(0.66)            |
| 49.                                    | item8=i8 item14=i14 5410 ==> item3=i3 3483 conf:(0.64)            |
| 50.                                    | item5=i5 item11=i11 item14=i14 4186 ==> item1=i1 2685 conf:(0.64) |

| Reglas generadas con Weka para mercado |   |  |
|--|---|--|
| 51.                                    | item5=i5 item11=i11 item14=i14 4186 ==> | item8=i8 2684 conf:(0.64)              |
| 52.                                    | item5=i5 item11=i11 item14=i14 4186 ==> | item3=i3 2682 conf:(0.64)              |
| 53.                                    | item2=i2 7014 ==>                       | item11=i11 4448 conf:(0.63)            |
| 54.                                    | item1=i1 item14=i14 4236 ==>            | item5=i5 item11=i11 2685 conf:(0.63)   |
| 55.                                    | item1=i1 5676 ==>                       | item11=i11 item14=i14 3595 conf:(0.63) |
| 56.                                    | item3=i3 item14=i14 7179 ==>            | item11=i11 4503 conf:(0.63)            |
| 57.                                    | item3=i3 item5=i5 4288 ==>              | item11=i11 item14=i14 2682 conf:(0.63) |
| 58.                                    | item8=i8 item14=i14 5410 ==>            | item5=i5 3378 conf:(0.62)              |
| 59.                                    | item11=i11 13237 ==>                    | item14=i14 8258 conf:(0.62)            |
| 60.                                    | item1=i1 5676 ==>                       | item5=i5 3538 conf:(0.62)              |
| 61.                                    | item1=i1 item11=i11 4448 ==>            | item5=i5 item14=i14 2685 conf:(0.6)    |
| 62.                                    | item8=i8 7551 ==>                       | item11=i11 4558 conf:(0.6)             |
| 63.                                    | item2=i2 7014 ==>                       | item14=i14 4229 conf:(0.6)             |
| 64.                                    | item5=i5 item14=i14 5792 ==>            | item3=i3 3485 conf:(0.6)               |
| 65.                                    | item3=i3 10343 ==>                      | item11=i11 6166 conf:(0.6)             |
| 66.                                    | item3=i3 item11=i11 item14=i14 4503 ==> | item5=i5 2682 conf:(0.6)               |
| 67.                                    | item15=i15 5685 ==>                     | item11=i11 item14=i14 3381 conf:(0.59) |
| 68.                                    | item8=i8 item11=i11 4558 ==>            | item5=i5 item14=i14 2684 conf:(0.59)   |
| 69.                                    | item8=i8 item11=i11 4558 ==>            | item3=i3 2683 conf:(0.59)              |
| 70.                                    | item5=i5 item11=i11 5580 ==>            | item3=i3 3273 conf:(0.59)              |
| 71.                                    | item5=i5 item14=i14 5792 ==>            | item8=i8 3378 conf:(0.58)              |
| 72.                                    | item15=i15 5685 ==>                     | item3=i3 item14=i14 3165 conf:(0.56)   |
| 73.                                    | item12=i12 5938 ==>                     | item11=i11 3266 conf:(0.55)            |
| 74.                                    | item11=i11 item14=i14 8258 ==>          | item3=i3 4503 conf:(0.55)              |
| 75.                                    | item1=i1 5676 ==>                       | item5=i5 item14=i14 3059 conf:(0.54)   |
| 76.                                    | item5=i5 item11=i11 5580 ==>            | item8=i8 3007 conf:(0.54)              |
| 77.                                    | item5=i5 item11=i11 5580 ==>            | item1=i1 3006 conf:(0.54)              |
| 78.                                    | item1=i1 5676 ==>                       | item3=i3 3055 conf:(0.54)              |
| 79.                                    | item14=i14 15373 ==>                    | item11=i11 8258 conf:(0.54)            |
| 80.                                    | item5=i5 8037 ==>                       | item3=i3 4288 conf:(0.53)              |
| 81.                                    | item3=i3 item11=i11 6166 ==>            | item5=i5 3273 conf:(0.53)              |
| 82.                                    | item1=i1 5676 ==>                       | item5=i5 item11=i11 3006 conf:(0.53)   |
| 83.                                    | item5=i5 item14=i14 5792 ==>            | item1=i1 3059 conf:(0.53)              |
| 84.                                    | item5=i5 8037 ==>                       | item11=i11 item14=i14 4186 conf:(0.52) |
| 85.                                    | item15=i15 5685 ==>                     | item3=i3 item11=i11 2952 conf:(0.52)   |
| 86.                                    | item8=i8 7551 ==>                       | item5=i5 3861 conf:(0.51)              |
| 87.                                    | item8=i8 7551 ==>                       | item11=i11 item14=i14 3861 conf:(0.51) |
| 88.                                    | item8=i8 7551 ==>                       | item3=i3 3860 conf:(0.51)              |
| 89.                                    | item11=i11 item14=i14 8258 ==>          | item5=i5 4186 conf:(0.51)              |
| 90.                                    | item15=i15 5685 ==>                     | item5=i5 2843 conf:(0.5)               |

Fuente. Esta investigación

La Tabla 63 muestra el comparativo entre las reglas que coinciden entre las dos herramientas. De acuerdo a estos resultados se puede inferir que Pg\_KDD es una herramienta viable para el descubrimiento de reglas de asociación.

**Tabla 63: Comparación reglas generadas por Pg\_KDD y Weka para mercado**

| Reglas generadas en Pg_KDD                        | Reglas generadas en Weka  |
|---|---|
| if i3,i8 -> i14 Confidence 90.23% (3483/3860)     | 1. item3=i3 item8=i8 3860 ==> item14=i14 3483 conf:(0.9)              |
| if i1,i5,i11 -> i14 Confidence 89.32% (2685/3006) | 2. item1=i1 item5=i5 item11=i11 3006 ==> item14=i14 2685 conf:(0.89)  |
| if i5,i8,i11 -> i14 Confidence 89.26% (2684/3007) | 3. item5=i5 item8=i8 item11=i11 3007 ==> item14=i14 2684 conf:(0.89)  |
| if i1,i5,i14 -> i11 Confidence 87.77% (2685/3059) | 4. item1=i1 item5=i5 item14=i14 3059 ==> item11=i11 2685 conf:(0.88)  |
| if i5,i8 -> i14 Confidence 87.49% (3378/3861)     | 5. item5=i5 item8=i8 3861 ==> item14=i14 3378 conf:(0.87)             |
| if i1,i5 -> i14 Confidence 86.46% (3059/3538)     | 6. item1=i1 item5=i5 3538 ==> item14=i14 3059 conf:(0.86)             |
| if i11,i15 -> i14 Confidence 85.19% (3381/3969)   | 7. item11=i11 item15=i15 3969 ==> item14=i14 3381 conf:(0.85)         |
| if i1,i5 -> i11 Confidence 84.96% (3006/3538)     | 8. item1=i1 item5=i5 3538 ==> item11=i11 3006 conf:(0.85)             |
| if i1,i14 -> i11 Confidence 84.87% (3595/4236)    | 9. item1=i1 item14=i14 4236 ==> item11=i11 3595 conf:(0.85)           |
| if i8,i11 -> i14 Confidence 84.71% (3861/4558)    | 10. item8=i8 item11=i11 4558 ==> item14=i14 3861 conf:(0.85)          |
| if i3,i5 -> i14 Confidence 81.27% (3485/4288)     | 13. item3=i3 item5=i5 4288 ==> item14=i14 3485 conf:(0.81)            |
| if i3,i15 -> i14 Confidence 81.97% (3165/3861)    | 11. item3=i3 item15=i15 3861 ==> item14=i14 3165 conf:(0.82)          |
| if i1,i11 -> i14 Confidence 80.82% (3595/4448)    | 14. item1=i1 item11=i11 4448 ==> item14=i14 3595 conf:(0.81)          |
| if i3,i5,i11 -> i14 Confidence 81.94% (2682/3273) | 12. item3=i3 item5=i5 item11=i11 3273 ==> item14=i14 2682 conf:(0.82) |
| if i1,i11,i14 -> i5 Confidence 74.69% (2685/3595) | 27. item1=i1 item11=i11 item14=i14 3595 ==> item5=i5 2685 conf:(0.75) |
| if i5,i11,i14 -> i3 Confidence 64.07% (2682/4186) | 52. item5=i5 item11=i11 item14=i14 4186 ==> item3=i3 2682 conf:(0.64) |
| if i5,i11,i14 -> i8 Confidence 64.12% (2684/4186) | 51. item5=i5 item11=i11 item14=i14 4186 ==> item8=i8 2684 conf:(0.64) |
| if i5,i11,i14 -> i1 Confidence 64.14% (2685/4186) | 50. item5=i5 item11=i11 item14=i14 4186 ==> item1=i1 2685 conf:(0.64) |
| if i3,i5,i14 -> i11 Confidence 76.96% (2682/3485) | 21. item3=i3 item5=i5 item14=i14 3485 ==> item11=i11 2682 conf:(0.77) |
| if i3,i11,i14 -> i5 Confidence 59.56% (2682/4503) | 66. item3=i3 item11=i11 item14=i14 4503 ==> item5=i5 2682 conf:(0.6)  |
| if i1,i11 -> i5,i14 Confidence 60.36% (2685/4448) | 61. item1=i1 item11=i11 4448 ==> item5=i5 item14=i14 2685 conf:(0.6)  |

| Reglas generadas en Pg_KDD                        | Reglas generadas en Weka   |
|---|--|
| if i3,i5 -> i11,i14 Confidence 62.55% (2682/4288) | 57. item3=i3 item5=i5 4288 ==> item11=i11<br>item14=i14 2682 conf:(0.63) |
| if i5,i8 -> i11,i14 Confidence 69.52% (2684/3861) | 41. item5=i5 item8=i8 3861 ==> item11=i11<br>item14=i14 2684 conf:(0.7)  |
| if i5,i8,i14 -> i11 Confidence 79.46% (2684/3378) | 15. item5=i5 item8=i8 item14=i14 3378 ==><br>item11=i11 2684 conf:(0.79) |
| if i8,i11,i14 -> i5 Confidence 69.52% (2684/3861) | 40. item8=i8 item11=i11 item14=i14 3861 ==><br>item5=i5 2684 conf:(0.7)  |
| if i1,i14 -> i5,i11 Confidence 63.39% (2685/4236) | 54. item1=i1 item14=i14 4236 ==> item5=i5<br>item11=i11 2685 conf:(0.63) |
| if i8,i11 -> i5,i14 Confidence 58.89% (2684/4558) | 68. item8=i8 item11=i11 4558 ==> item5=i5<br>item14=i14 2684 conf:(0.59) |
| if i1,i5 -> i11,i14 Confidence 75.89% (2685/3538) | 25. item1=i1 item5=i5 3538 ==> item11=i11<br>item14=i14 2685 conf:(0.76) |

Fuente. Esta investigación

De los resultados anteriores se toman la regla generada con mayor confianza, cuyo resultado con valores sin discretizar se observa en la Tabla 64 .

Tabla 64: Regla transformada a valores reales

| Regla Generada por Pg_KDD                        | Regla con valores sin discretizar  |
|--|--|
| if i3,i8 -> i14 Confidence 90.23%<br>(3483/3860) | <b>Si compra AROMATICAS GOURMET YERBABUENA y CHOCOLATE LUKER, es posible que compre CUBIERTA DE CHOCOLATE.</b> |

Fuente. Esta investigación

**6.3.2 Clasificación.** Para realizar las pruebas de rendimiento de los algoritmos, se seleccionó el conjunto de datos *guardería* (Ver Tabla 45). El objetivo principal es comparar el tiempo de procesamiento de los algoritmos con respecto al número de atributos y de instancias, así como medir la capacidad de la herramienta con BD grandes.

Las pruebas de rendimiento se ejecutan teniendo en cuenta dos aspectos, primero rendimiento mirando el comportamiento de los algoritmos en la medida en que se incrementa el número de instancias, y el segundo aspecto a evaluar es el comportamiento en la medida en que la Base de Datos crece en número de atributos.

## Prueba de rendimiento variando el número de transacciones para Mate

En esta prueba se medirá el tiempo de procesamiento para el nodo *Mate*, realizando variaciones en el número de transacciones, sin variar el número de atributos.

En la prueba se usará la Base de Datos *Guarderia* (Tabla 45). Se modificó el repositorio de tal manera que se aumentó la cardinalidad de la tabla con datos pertenecientes al mismo conjunto de clases hasta obtener las necesarias para la realización de las pruebas.

Al final se obtuvieron conjuntos de datos con 5000, 10000, 15000, 20000 y 25000 transacciones.

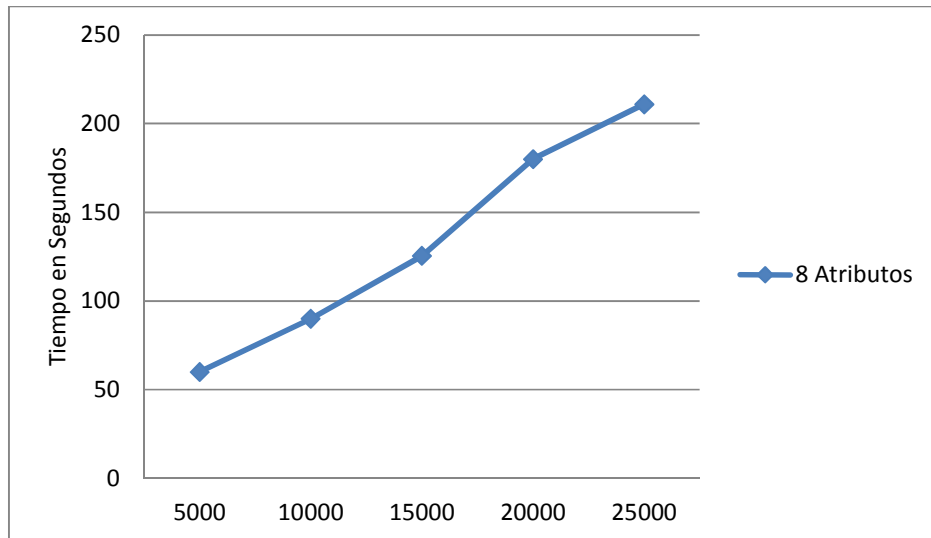
Para cada conjunto de registros se midió el tiempo de respuesta del nodo *Mate*. La Tabla 65, muestra los resultados.

**Tabla 65: Rendimiento con Mate variando N° de trans acciones**

| N° Transacciones | Base de Datos Guarderia       |
|------------------|-------------------------------|
|                  | Tiempo (seg) - N° Atributos 8 |
|                  | <b>Mate</b>                   |
| <b>5000</b>      | 60,09                         |
| <b>10000</b>     | 90,01                         |
| <b>15000</b>     | 125,5                         |
| <b>20000</b>     | 180                           |
| <b>25000</b>     | 210,9                         |

En la Figura 101, se muestra el rendimiento en segundos del nodo *Mate* con diferente número de transacciones. La gráfica nos muestra que el tiempo de respuesta para el nodo *Mate* se incrementa a medida que el número de instancias crece, con una leve tendencia a aumentar exponencialmente con un mayor número de transacciones. El tiempo de respuesta se podrían considerar eficiente y razonable aunque la tendencia exponencial de la gráfica indica que para valores mas grandes estos tiempos crecerán mucho mas y el rendimiento no será el deseado.

**Figura 101: Prueba de rendimiento con Guarderia variando N°transacciones**



### Prueba de rendimiento variando el número de atributos para Mate

También es necesario probar los algoritmos con una base de datos flexible en cuanto al manejo de número de atributos y evaluar el rendimiento de cada algoritmo y aprovechar para comparar los resultados en cuanto a número de reglas y porcentaje de clasificación. Se elige para esta tarea la base *guarderia* ya que por su estructura permite evaluar la escalabilidad de acuerdo al número de atributos. Para ello se comienza trabajando con un repositorio cuya cardinalidad se incrementó hasta obtener 25000 transacciones, se tomaran en principio los 2 primeros atributos incluido el atributo clase y se aumentará en cada iteración el número atributo en 2. El tiempo expresado en las tablas está dado en segundos. Para cada conjunto de registros se midió el tiempo de respuesta para el nodo *Mate*.

En la Tabla 66, se muestran los tiempos obtenidos en la prueba:

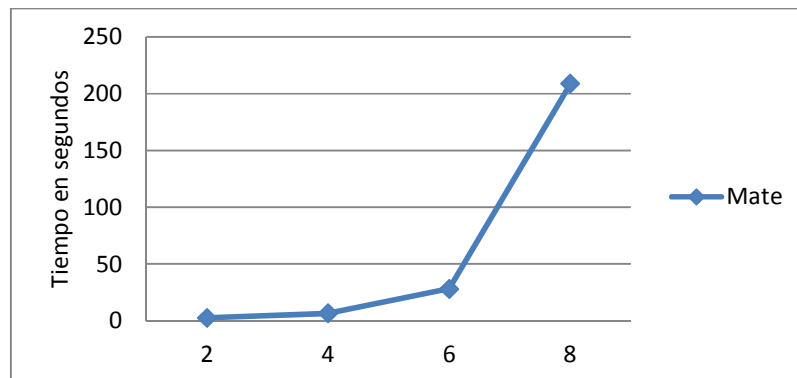
**Tabla 66: Rendimiento con Mate variando N° de Atrib utos**

| N° Atributos | Base de Datos Guarderia                  |  |
|--------------|--|--|
|              | Tiempo (seg) - N° de Transacciones 25000 |  |
|              | <b>Mate</b>                              |  |
| <b>2</b>     | 2,61                                     |  |

| Base de Datos Guardería |       |
|-------------------------|-------|
| 4                       | 6,58  |
| 6                       | 28,12 |
| 8                       | 208,8 |

En la Figura 102, se muestra el rendimiento en segundos del nodo *Mate* con diferente número de atributos. La prueba muestran un comportamiento evidentemente exponencial del nodo *Mate* cuando los atributos se incrementan, es decir, se muestra que los tiempos de ejecución están directamente relacionados con el número de atributos y las distintas categorías del atributo clase. El tiempo de ejecución del algoritmo crece significativamente de acuerdo al grado y cardinalidad de la tabla.

**Figura 102: Rendimiento con guardería variando N°a tributos**



### **Prueba de rendimiento variando el número de atributos y transacciones para *Mate***

Esta prueba se realizó con la misma base de datos transformada *Guardería* De igual manera como se hizo en las dos pruebas anteriores se varía el número de transacciones (5000 – 25000 con intervalos de 5000) y además también se varia el número de atributos (2-8 con intervalos de 2).

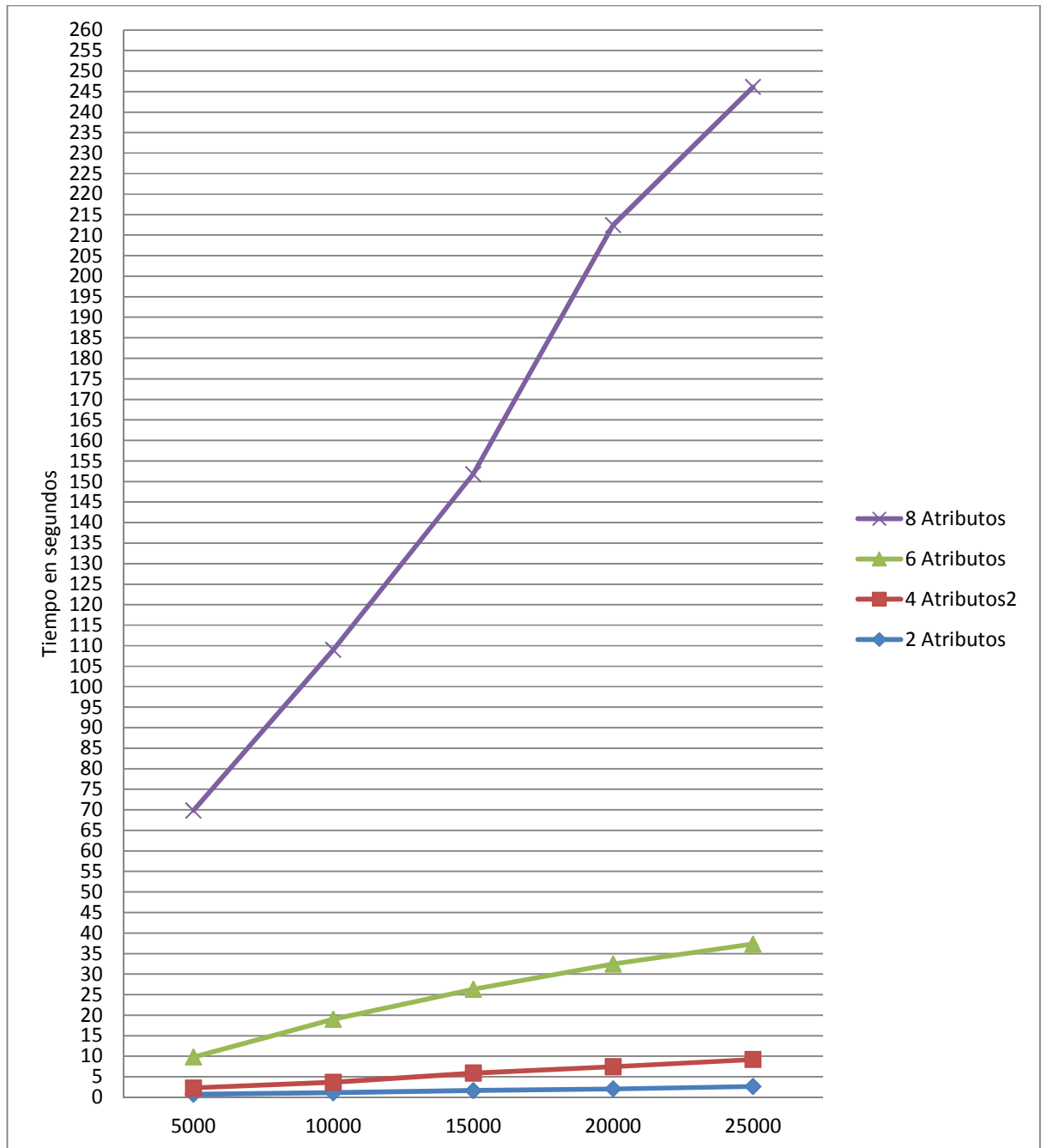
En la Tabla 67, se muestran los tiempos obtenidos en la prueba:

**Tabla 67: Rendimiento con Mate variando N° de Atributos y N° Transacciones**

| N° Transacciones | Base de Datos Guardería |      |       |       |
|------------------|-------------------------|------|-------|-------|
|                  | Mate - Tiempo (seg)     |      |       |       |
|                  | N° Atributos            |      |       |       |
|                  | 2                       | 4    | 6     | 8     |
| <b>5000</b>      | 0,74                    | 1,51 | 7,53  | 60,09 |
| <b>10000</b>     | 1,09                    | 2,57 | 15,33 | 90,01 |
| <b>15000</b>     | 1,61                    | 4,28 | 20,39 | 125,5 |
| <b>20000</b>     | 2,04                    | 5,39 | 25,03 | 180   |
| <b>25000</b>     | 2,61                    | 6,58 | 28,12 | 208,8 |

En la Figura 103, se muestra el rendimiento en segundos del nodo *Mate* con diferente número de atributos y transacciones, y claramente indica la pérdida de eficiencia y la baja de rendimiento que ocurre cuando la cantidad de transacciones crece demasiado y el número de atributos es igualmente mayor con respecto a la totalidad de campos del repositorio.

Figura 103: Rendimiento con Mate variando N° de Atributos y N° Transacciones



## Prueba de rendimiento variando el número de transacciones y el número de atributos en Mate-Tree

Esta prueba se realizó con la misma base de datos transformada *Guardería*. De igual manera como se hizo en las dos pruebas anteriores se varía el número de transacciones (5000 – 25000 con intervalos de 5000) y además también se varía el número de atributos (2-8 con intervalos de 2). Se tomará el tiempo de respuesta del algoritmo Mate-Tree.

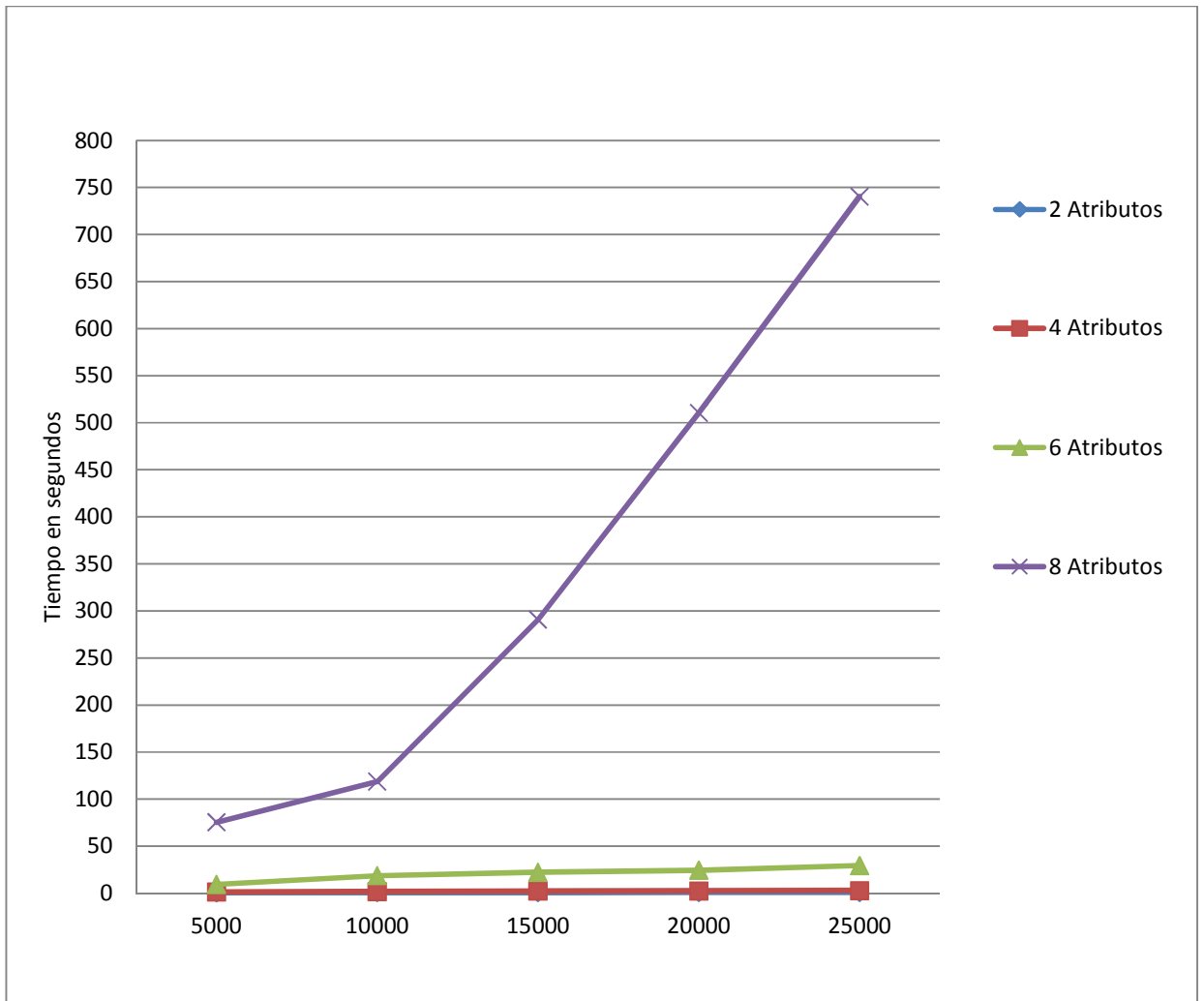
La Tabla 68, se muestra los tiempos de respuesta de la prueba:

**Tabla 68: Rendimiento Mate-Tree variando N° Atributos y N° Transacciones**

| N° Transacciones | Base de Datos Guardería  |      |       |       |
|------------------|--------------------------|------|-------|-------|
|                  | Mate-Tree - Tiempo (seg) |      |       |       |
|                  | N° Atributos             |      |       |       |
|                  | 2                        | 4    | 6     | 8     |
| <b>5000</b>      | 0,41                     | 1,19 | 9,22  | 75,3  |
| <b>10000</b>     | 0,5                      | 1,67 | 18,46 | 118,6 |
| <b>15000</b>     | 0,6                      | 2,12 | 22,21 | 290,7 |
| <b>20000</b>     | 0,7                      | 2,42 | 24,35 | 510,5 |
| <b>25000</b>     | 0,75                     | 2,62 | 29,22 | 740,5 |

La Figura 104, indica un comportamiento exponencial para el nodo *Mate-Tree*. Igualmente puede observarse la pérdida de eficiencia al incrementar el número de atributos más allá de los 6. Con lo anterior se puede inferir que el tiempo de ejecución del algoritmo sobre conjuntos de datos pequeños es relativamente bajo. La diferencia en tiempo de ejecución se hace más notoria en la medida en que el conjunto de datos es mayor, es decir, el tiempo de ejecución crece proporcionalmente con respecto al tamaño del conjunto de datos.

Figura 104: Rendimiento con Mate variando N° de atributos y N° Transacciones



Para la anterior prueba se realizó una comparación de resultados con la herramienta Weka[52]. Primero se indican los resultados obtenidos con Pg\_KDD.

En la Figura 105, Figura 106 y Figura 107 se presentan los resultados obtenidos para el operador Mate con atributo clase “recomendación”, para la función agregada Entro y Gain.

Figura 105: Resultado Mate con Guarderia

| VIEW TABLE MATE BY |          |          |            |            |               |               |       |
|--------------------|----------|----------|------------|------------|---------------|---------------|-------|
| parents            | has_nurs | children | housing    | finance    | social        | recomendacion | count |
| great_pret         | critical | more     | convenient | convenient | nonprob       | not_recom     | 8     |
| great_pret         | critical | more     | convenient | convenient | nonprob       | priority      | 10    |
| great_pret         | critical | more     | convenient | convenient | nonprob       | spec_prior    | 6     |
| great_pret         | critical | more     | convenient | convenient | problematic   | not_recom     | 8     |
| great_pret         | critical | more     | convenient | convenient | problematic   | priority      | 4     |
| great_pret         | critical | more     | convenient | convenient | problematic   | spec_prior    | 12    |
| great_pret         | critical | more     | convenient | convenient | slightly_prob | not_recom     | 8     |
| great_pret         | critical | more     | convenient | convenient | slightly_prob | priority      | 10    |
| great_pret         | critical | more     | convenient | convenient | slightly_prob | spec_prior    | 6     |
| great_pret         | critical | more     | convenient | convenient |               | not_recom     | 24    |
| great_pret         | critical | more     | convenient | convenient |               | priority      | 24    |
| great_pret         | critical | more     | convenient | convenient |               | spec_prior    | 24    |
| great_pret         | critical | more     | convenient | incony     | nonprob       | not_recom     | 8     |
| great_pret         | critical | more     | convenient | incony     | nonprob       | priority      | 6     |
| great_pret         | critical | more     | convenient | incony     | nonprob       | spec_prior    | 10    |
| great_pret         | critical | more     | convenient | incony     | problematic   | not_recom     | 8     |
| great_pret         | critical | more     | convenient | incony     | problematic   | spec_prior    | 16    |
| great_pret         | critical | more     | convenient | incony     | slightly_prob | not_recom     | 8     |
| great_pret         | critical | more     | convenient | incony     | slightly_prob | priority      | 6     |
| great_pret         | critical | more     | convenient | incony     | slightly_prob | spec_prior    | 10    |
| great_pret         | critical | more     | convenient | incony     |               | not_recom     | 24    |
| great_pret         | critical | more     | convenient | incony     |               | priority      | 12    |
| great_pret         | critical | more     | convenient | incony     |               | spec_prior    | 36    |
| great_pret         | critical | more     | convenient |            | nonprob       | not_recom     | 16    |
| great_pret         | critical | more     | convenient |            | nonprob       | priority      | 16    |
| great_pret         | critical | more     | convenient |            | nonprob       | spec_prior    | 16    |
| great_pret         | critical | more     | convenient |            | problematic   | not_recom     | 16    |
| great_pret         | critical | more     | convenient |            | problematic   | priority      | 4     |
| great_pret         | critical | more     | convenient |            | problematic   | spec_prior    | 28    |
| great_pret         | critical | more     | convenient |            | slightly_prob | not_recom     | 16    |
| great_pret         | critical | more     | convenient |            | slightly_prob | priority      | 16    |
| great_pret         | critical | more     | convenient |            | slightly_prob | spec_prior    | 16    |
| great_pret         | critical | more     | convenient |            |               | not_recom     | 48    |
| great_pret         | critical | more     | convenient |            |               | priority      | 36    |
| great_pret         | critical | more     | convenient |            |               | spec_prior    | 60    |
| great_pret         | critical | more     | critical   | convenient | nonprob       | not_recom     | 8     |
| great_pret         | critical | more     | critical   | convenient | nonprob       | priority      | 2     |
| great_pret         | critical | more     | critical   | convenient | nonprob       | spec_prior    | 14    |
| great_pret         | critical | more     | critical   | convenient | problematic   | not_recom     | 8     |
| great_pret         | critical | more     | critical   | convenient | problematic   | spec_prior    | 16    |

Figura 106: Resultado Entro con Guarderia

| VIEW TABLE ENTRO RESULT |          |          |            |            |               |               |             |
|-------------------------|----------|----------|------------|------------|---------------|---------------|-------------|
| parents                 | has_nurs | children | housing    | finance    | social        | recomendacion | count entro |
|                         |          |          |            |            | nonprob       | not_recom     | 539 0.4165  |
|                         |          |          |            |            | nonprob       | priority      | 1616 0.5141 |
|                         |          |          |            |            | nonprob       | spec_prior    | 1329 0.5304 |
|                         |          |          |            |            | problematic   | not_recom     | 60 0.0864   |
|                         |          |          |            |            | problematic   | priority      | 1690 0.5293 |
|                         |          |          |            |            | problematic   | spec_prior    | 2522 0.4489 |
|                         |          |          |            |            | slightly_prob | not_recom     | 549 0.4200  |
|                         |          |          |            |            | slightly_prob | priority      | 1607 0.5150 |
|                         |          |          |            |            | slightly_prob | spec_prior    | 1329 0.5304 |
|                         |          |          |            | convenient |               | not_recom     | 728 0.3849  |
|                         |          |          |            | convenient |               | priority      | 2538 0.5158 |
|                         |          |          |            | convenient |               | spec_prior    | 2270 0.5274 |
|                         |          |          |            | convenient | nonprob       | not_recom     | 349 0.4697  |
|                         |          |          |            | convenient | nonprob       | priority      | 806 0.5098  |
|                         |          |          |            | convenient | nonprob       | spec_prior    | 538 0.5256  |
|                         |          |          |            | convenient | problematic   | not_recom     | 20 0.0628   |
|                         |          |          |            | convenient | problematic   | priority      | 934 0.5224  |
|                         |          |          |            | convenient | problematic   | spec_prior    | 1194 0.4709 |
|                         |          |          |            | convenient | slightly_prob | not_recom     | 359 0.4743  |
|                         |          |          |            | convenient | slightly_prob | priority      | 798 0.5117  |
|                         |          |          |            | convenient | slightly_prob | spec_prior    | 538 0.5255  |
|                         |          |          |            | inconv     |               | not_recom     | 420 0.2771  |
|                         |          |          |            | inconv     |               | priority      | 2375 0.5263 |
|                         |          |          |            | inconv     |               | spec_prior    | 2910 0.4954 |
|                         |          |          |            | inconv     | nonprob       | not_recom     | 190 0.3434  |
|                         |          |          |            | inconv     | nonprob       | priority      | 810 0.5177  |
|                         |          |          |            | inconv     | nonprob       | spec_prior    | 791 0.5207  |
|                         |          |          |            | inconv     | problematic   | not_recom     | 40 0.1079   |
|                         |          |          |            | inconv     | problematic   | priority      | 756 0.5305  |
|                         |          |          |            | inconv     | problematic   | spec_prior    | 1328 0.4236 |
|                         |          |          |            | inconv     | slightly_prob | not_recom     | 190 0.3435  |
|                         |          |          |            | inconv     | slightly_prob | priority      | 809 0.5178  |
|                         |          |          |            | inconv     | slightly_prob | spec_prior    | 791 0.5207  |
|                         |          |          | convenient |            |               | not_recom     | 566 0.4208  |
|                         |          |          | convenient |            |               | priority      | 1813 0.4970 |
|                         |          |          | convenient |            |               | spec_prior    | 1200 0.5286 |
|                         |          |          | convenient |            | nonprob       | not_recom     | 268 0.4993  |
|                         |          |          | convenient |            | nonprob       | priority      | 556 0.4924  |
|                         |          |          | convenient |            | nonprob       | spec_prior    | 253 0.4909  |
|                         |          |          | convenient |            | problematic   | not_recom     | 20 0.0864   |
|                         |          |          | convenient |            | problematic   | priority      | 710 0.5006  |

Figura 107: Resultado Gain con Guarderia

| node | padre | atributo | valor | hijo | herm | clase |
|------|-------|----------|-------|------|------|-------|
| 0    |       | has_nurs |       |      |      |       |
| 1    | 0     | has_nurs | 0     | 6    | 2    |       |
| 6    | 1     | parents  | 0     | 9    | 7    |       |
| 9    | 6     | housing  | 0     | 12   | 10   |       |
| 12   | 9     | finance  | 0     | 14   | 13   |       |
| 14   | 12    | social   | 0     | 17   | 15   |       |
| 17   | 14    | children | 0     | 18   | 1    |       |
| 18   | 14    | children | 1     | 19   | 0    |       |
| 19   | 14    | children | 2     | 20   | 0    |       |
| 20   | 14    | children | 3     |      | 0    |       |
| 15   | 12    | social   | 1     | 16   | 3    |       |
| 16   | 12    | social   | 2     | 21   |      |       |
| 21   | 16    | children | 0     | 22   | 1    |       |
| 22   | 16    | children | 1     | 23   | 0    |       |
| 23   | 16    | children | 2     | 24   | 0    |       |
| 24   | 16    | children | 3     |      | 0    |       |
| 13   | 9     | finance  | 1     |      | 3    |       |
| 10   | 6     | housing  | 1     | 11   | 3    |       |
| 11   | 6     | housing  | 2     |      | 3    |       |
| 7    | 1     | parents  | 1     | 25   | 8    |       |
| 25   | 7     | housing  | 0     | 28   | 26   |       |
| 28   | 25    | finance  | 0     | 30   | 29   |       |
| 30   | 28    | social   | 0     | 33   | 31   |       |
| 33   | 30    | children | 0     | 34   | 1    |       |
| 34   | 30    | children | 1     | 35   | 0    |       |
| 35   | 30    | children | 2     | 36   | 0    |       |
| 36   | 30    | children | 3     |      | 0    |       |
| 31   | 28    | social   | 1     | 32   | 3    |       |
| 32   | 28    | social   | 2     | 37   |      |       |
| 37   | 32    | children | 0     | 38   | 1    |       |
| 38   | 32    | children | 1     | 39   | 0    |       |
| 39   | 32    | children | 2     | 40   | 0    |       |
| 40   | 32    | children | 3     |      | 0    |       |
| 29   | 25    | finance  | 1     |      | 3    |       |
| 26   | 7     | housing  | 1     | 27   | 3    |       |
| 27   | 7     | housing  | 2     |      | 3    |       |
| 8    | 1     | parents  | 2     | 41   |      |       |
| 41   | 8     | social   | 0     | 44   | 42   |       |
| 44   | 41    | children | 0     | 45   | 1    |       |
| 45   | 41    | children | 1     | 48   | 46   |       |

| atributo | clase         | ganancia                 |
|----------|---------------|--------------------------|
| parents  | recomendacion | 1 271083800373632239325  |
| has_nurs | recomendacion | 0 9494604127746641757854 |
| children | recomendacion | 1 337347495774397295614  |
| housing  | recomendacion | 1 345107303620674317232  |
| finance  | recomendacion | 1 362478035761942887643  |
| social   | recomendacion | 1 311717302731073747887  |

En la Figura 108, se observan las reglas generadas:

Figura 108: Resultado Reglas para Guardería

| CLASIFICATION RULES   |           |
|---|-----------|
| 1 - parents = great_pret AND has_nurs = critical AND children = more AND housing = convenient AND finance = convenient AND social = nonprob --> recomendacion = priority        | [10/24]   |
| 2 - parents = great_pret AND has_nurs = critical AND housing = convenient AND finance = inconv --> recomendacion = spec_prior   | [8/24]    |
| 3 - parents = pretentious AND has_nurs = improper AND children = more AND housing = less_conv AND social = slightly_prob --> recomendacion = not_recom                          | [8/24]    |
| 4 - parents = pretentious AND has_nurs = improper AND children = more AND housing = convenient AND finance = convenient AND social = slightly_prob --> recomendacion = priority | [8/24]    |
| 5 - parents = pretentious AND has_nurs = improper AND children = more AND housing = convenient AND finance = inconv AND social = slightly_prob --> recomendacion = not_recom    | [60/96]   |
| 6 - parents = pretentious AND has_nurs = improper AND children = more AND housing = critical AND social = slightly_prob --> recomendacion = not_recom                           | [10/24]   |
| 7 - parents = pretentious AND has_nurs = improper AND children = more AND housing = less_conv AND social = slightly_prob --> recomendacion = not_recom                          | [8/24]    |
| 8 - parents = usual AND has_nurs = improper AND housing = convenient AND finance = convenient AND social = nonprob --> recomendacion = not_recom                                | [8/24]    |
| 9 - parents = usual AND has_nurs = improper AND housing = convenient AND finance = convenient AND social = problematic --> recomendacion = priority                             | [8/24]    |
| 10 - parents = usual AND has_nurs = improper AND housing = convenient AND finance = convenient AND social = slightly_prob --> recomendacion = not_recom                         | [172/288] |
| 11 - parents = usual AND has_nurs = improper AND housing = convenient AND finance = inconv --> recomendacion = priority   | [376/576] |
| 12 - parents = usual AND has_nurs = improper AND housing = critical --> recomendacion = priority  | [344/576] |
| 13 - parents = great_pret AND has_nurs = critical AND housing = critical --> recomendacion = spec_prior   | [7/15]    |
| 14 - parents = usual AND has_nurs = improper AND housing = less_conv --> recomendacion = priority   | [5/15]    |
| 15 - parents = great_pret AND has_nurs = less_proper AND children = more AND social = nonprob --> recomendacion = priority  | [4/12]    |
| 16 - parents = great_pret AND has_nurs = less_proper AND children = more AND housing = convenient AND social = nonprob --> recomendacion = priority                             | [4/12]    |
| 17 - parents = great_pret AND has_nurs = less_proper AND children = more AND housing = critical AND social = nonprob --> recomendacion = not_recom                              | [32/54]   |
| 18 - parents = great_pret AND has_nurs = less_proper AND children = more AND housing = less_conv AND social = nonprob --> recomendacion = priority                              | [7/15]    |
| 19 - parents = great_pret AND has_nurs = less_proper AND children = more AND housing = convenient AND finance = convenient AND social = nonprob --> recomendacion = priority    | [5/15]    |
| 20 - parents = great_pret AND has_nurs = less_proper AND children = more AND housing = convenient AND finance = inconv AND social = nonprob --> recomendacion = not_recom       | [4/12]    |
| 21 - parents = great_pret AND has_nurs = less_proper AND children = more AND housing = critical AND social = nonprob --> recomendacion = not_recom                              | [4/12]    |
| 22 - parents = great_pret AND has_nurs = less_proper AND children = more AND housing = less_conv AND social = nonprob --> recomendacion = not_recom                             | [94/162]  |

En la Tabla 69Tabla 62 se detallan las reglas generadas con Pg\_KDD para Guardería.

Tabla 69: Reglas generadas por Pg\_KDD para Guardería

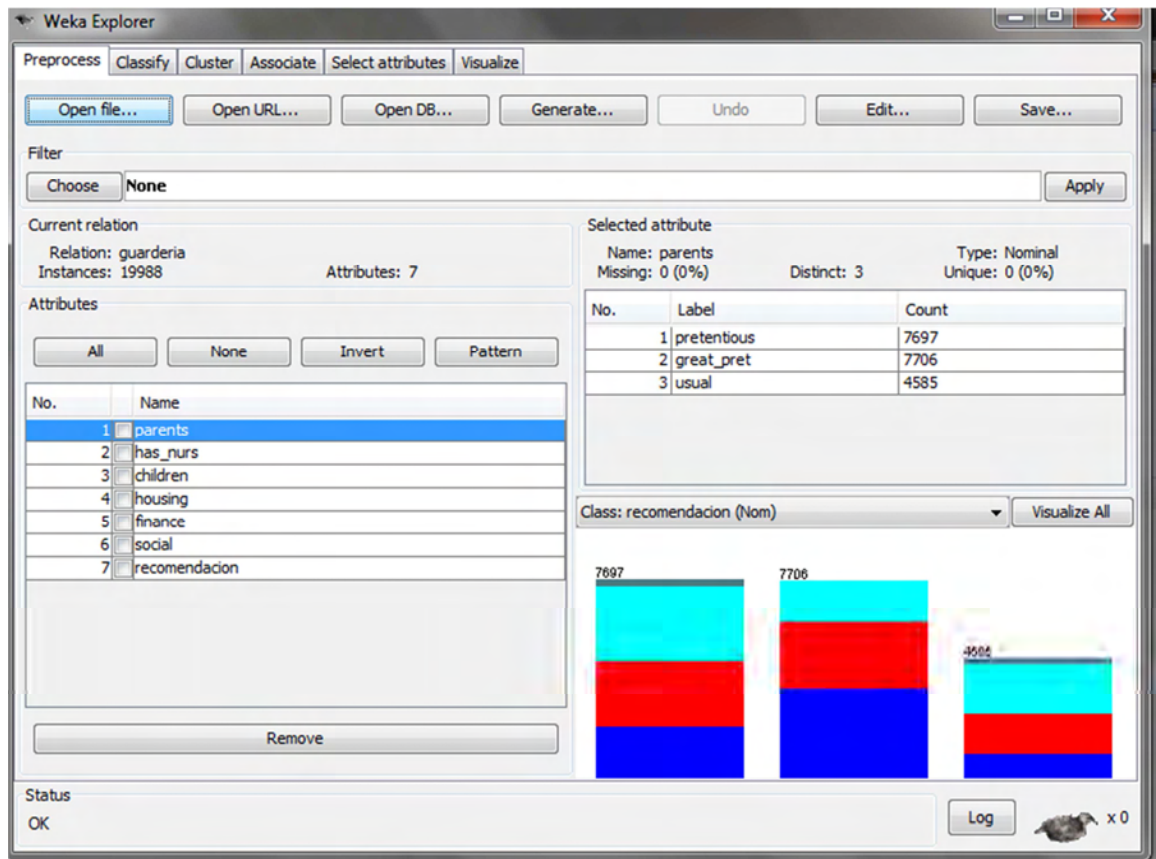
| Reglas generadas para Guardería  |
|--|
| 1 - parents = great_pret AND has_nurs = critical AND children = more AND housing = convenient AND finance = convenient AND social = nonprob --> recomendacion = priority [10/24]       |
| 2 - parents = great_pret AND has_nurs = critical AND housing = convenient AND finance = inconv --> recomendacion = spec_prior [8/24]   |
| 3 - parents = pretentious AND has_nurs = improper AND children = more AND housing = less_conv AND social = slightly_prob --> recomendacion = not_recom [8/24]                          |
| 4 - parents = pretentious AND has_nurs = improper AND children = more AND housing = convenient AND finance = convenient AND social = slightly_prob --> recomendacion = priority [8/24] |
| 5 - parents = pretentious AND has_nurs = improper AND children = more AND housing = convenient AND finance = inconv AND social = slightly_prob --> recomendacion = not_recom [60/96]   |
| 6 - parents = pretentious AND has_nurs = improper AND children = more AND housing = critical AND social = slightly_prob --> recomendacion = not_recom [10/24]                          |
| 7 - parents = pretentious AND has_nurs = improper AND children = more AND housing = less_conv AND social = slightly_prob --> recomendacion = not_recom [8/24]                          |
| 8 - parents = usual AND has_nurs = improper AND housing = convenient AND finance = convenient AND social = nonprob --> recomendacion = not_recom [8/24]                                |

| Reglas generadas para Guardería   |
|---|
| 9 - parents = usual AND has_nurs = improper AND housing = convenient AND finance = convenient AND social = problematic --> recomendacion = priority [8/24]                          |
| 10 - parents = usual AND has_nurs = improper AND housing = convenient AND finance = convenient AND social = slightly_prob --> recomendacion = not_recom [172/288]                   |
| 11 - parents = usual AND has_nurs = improper AND housing = convenient AND finance = inconv --> recomendacion = priority [376/576]   |
| 12 - parents = usual AND has_nurs = improper AND housing = critical --> recomendacion = priority [344/576]  |
| 13 - parents = great_pret AND has_nurs = critical AND housing = critical --> recomendacion = spec_prior [7/15]  |
| 14 - parents = usual AND has_nurs = improper AND housing = less_conv --> recomendacion = priority [5/15]  |
| 15 - parents = great_pret AND has_nurs = less_proper AND children = more AND social = nonprob --> recomendacion = priority [4/12]   |
| 16 - parents = great_pret AND has_nurs = less_proper AND children = more AND housing = convenient AND social = nonprob --> recomendacion = priority [4/12]                          |
| 17 - parents = great_pret AND has_nurs = less_proper AND children = more AND housing = critical AND social = nonprob --> recomendacion = not_recom [32/54]                          |
| 18 - parents = great_pret AND has_nurs = less_proper AND children = more AND housing = less_conv AND social = nonprob --> recomendacion = priority [7/15]                           |
| 19 - parents = great_pret AND has_nurs = less_proper AND children = more AND housing = convenient AND finance = convenient AND social = nonprob --> recomendacion = priority [5/15] |
| 20 - parents = great_pret AND has_nurs = less_proper AND children = more AND housing = convenient AND finance = inconv AND social = nonprob --> recomendacion = not_recom [4/12]    |
| 21 - parents = great_pret AND has_nurs = less_proper AND children = more AND housing = critical AND social = nonprob --> recomendacion = not_recom [4/12]                           |
| 22 - parents = great_pret AND has_nurs = less_proper AND children = more AND housing = less_conv AND social = nonprob --> recomendacion = not_recom [94/162]                        |
| 19 - parents = usual AND has_nurs = critical AND children = more AND housing = convenient AND finance = inconv --> recomendacion = spec_prior [32/72]                               |
| 20 - parents = usual AND has_nurs = critical AND children = more AND housing = critical --> recomendacion = spec_prior [32/72]  |
| 21 - parents = usual AND has_nurs = critical AND children = more AND housing = less_conv --> recomendacion = spec_prior [24/36]   |
| 22 - parents = great_pret AND has_nurs = improper AND children = more AND housing = convenient AND finance = convenient --> recomendacion = not_recom [16/36]                       |

Para generar las reglas de asociación para mercado con Weka se siguieron los siguientes pasos:

- Selección del repositorio Guardería (ver Figura 109)

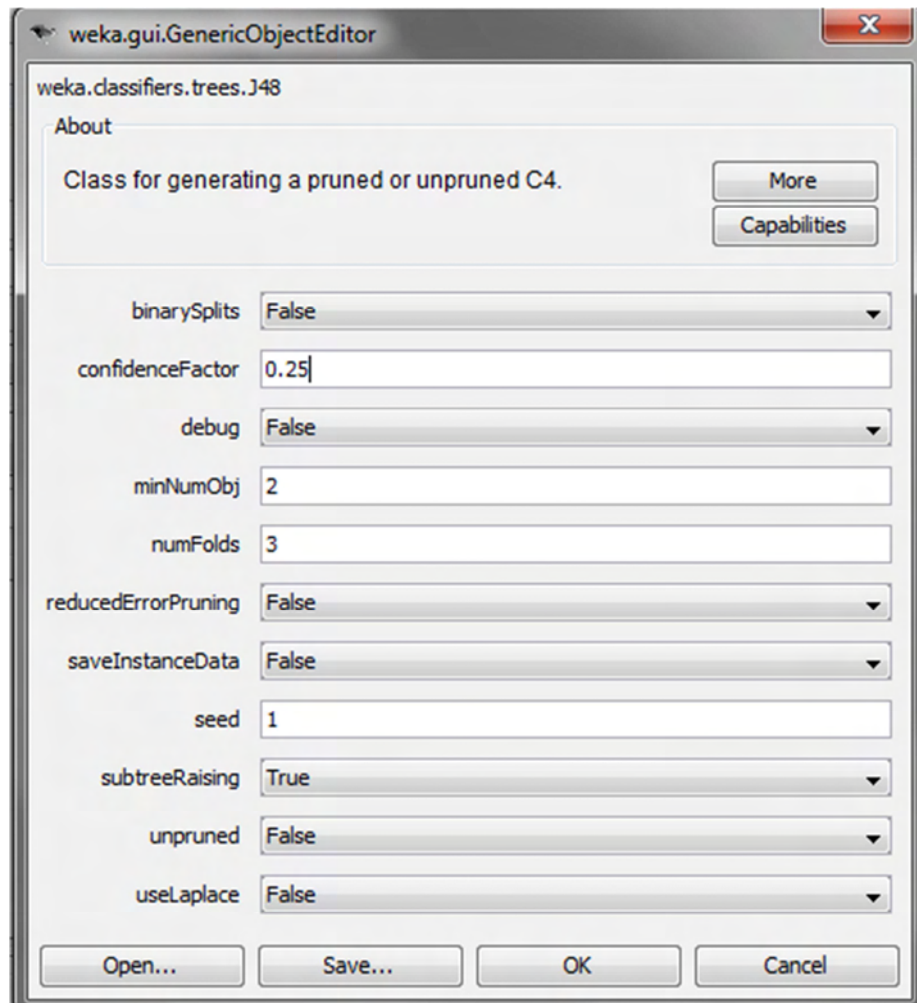
Figura 109: Selección de Repositorio Guardería en Weka



Fuente. Esta investigación

- Configuración del algoritmo J48 para descubrimiento de reglas de asociación (ver).

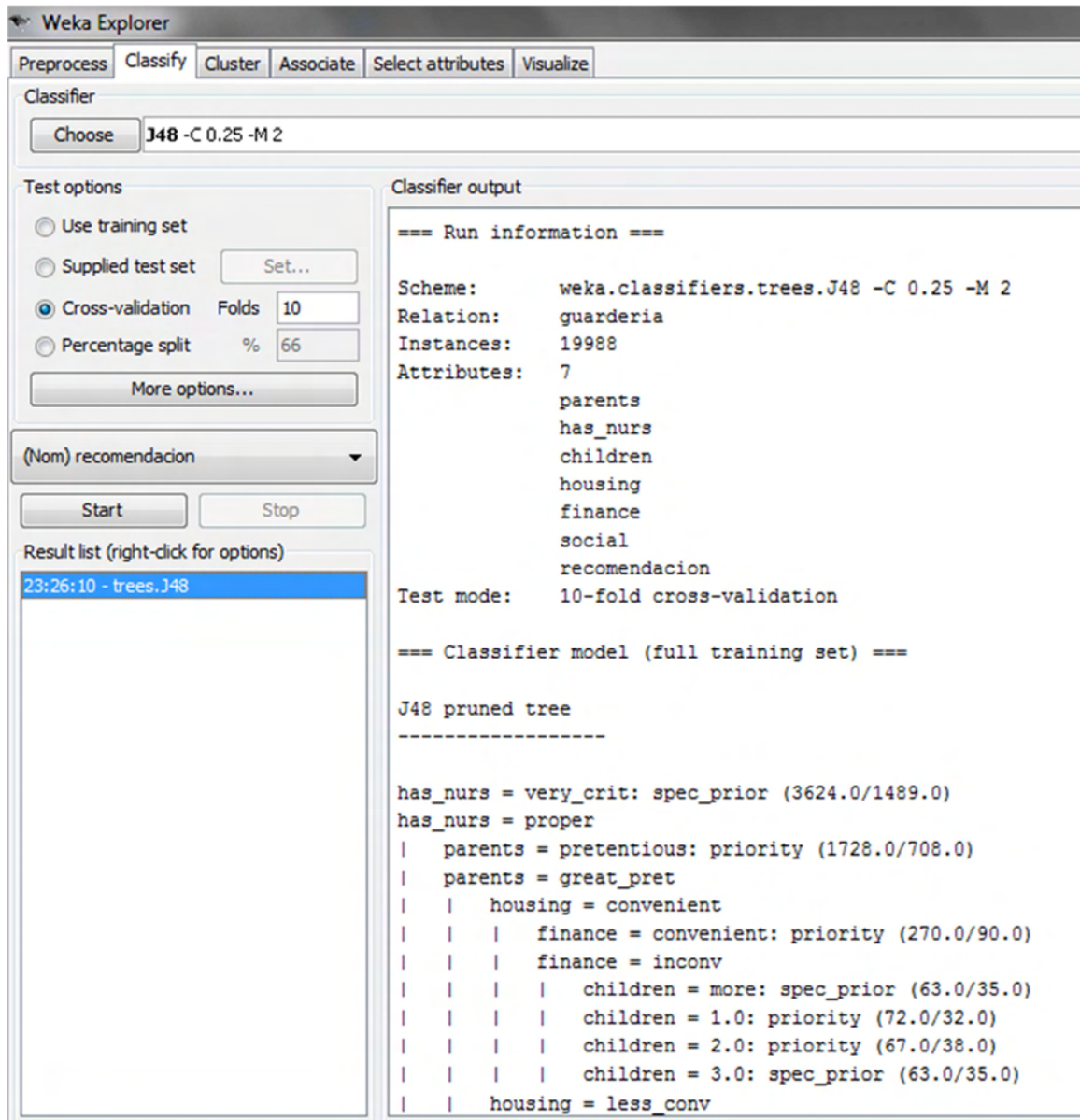
Figura 110: Configuración algoritmo J48 en Weka



Fuente. Esta investigación

- Generación de reglas de clasificación en Weka (ver Figura 111Figura 100).

Figura 111: Generación de reglas de clasificación con Weka



Fuente. Esta investigación

La Tabla 70 muestra el modelo del árbol de clasificación generado con Weka para el repositorio Guardería, donde se marca aquellos caminos que coinciden con las reglas generadas por Pg\_KDD.

Tabla 70: Reglas generadas con Weka para Guardería

| Reglas generadas con Weka para Guardería         |
|--|
| J48 pruned tree                                  |
| has_nurs = very_crit: spec_prior (3624.0/1489.0) |
| has_nurs = proper                                |
| parents = pretentious: priority (1728.0/708.0)   |
| parents = great_pret                             |
| housing = convenient                             |
| finance = convenient: priority (270.0/90.0)      |
| finance = inconv                                 |
| children = more: spec_prior (63.0/35.0)          |
| children = 1.0: priority (72.0/32.0)             |
| children = 2.0: priority (67.0/38.0)             |
| children = 3.0: spec_prior (63.0/35.0)           |
| housing = less_conv                              |
| children = more: spec_prior (126.0/70.0)         |
| children = 1.0: priority (144.0/64.0)            |
| children = 2.0                                   |
| social = slightly_prob: priority (42.0/22.0)     |
| social = problematic: spec_prior (42.0/26.0)     |
| social = nonprob: priority (42.0/22.0)           |
| children = 3.0: spec_prior (126.0/70.0)          |
| housing = critical                               |
| social = slightly_prob: priority (177.0/114.0)   |
| social = problematic: spec_prior (178.0/68.0)    |
| social = nonprob: priority (177.0/114.0)         |
| parents = usual: priority (864.0/354.0)          |
| has_nurs = less_proper                           |
| parents = pretentious: priority (1728.0/708.0)   |
| parents = great_pret                             |
| housing = convenient                             |
| finance = convenient: priority (288.0/96.0)      |
| finance = inconv                                 |
| children = more: spec_prior (72.0/40.0)          |
| children = 1.0: priority (72.0/32.0)             |
| children = 2.0: priority (72.0/40.0)             |
| children = 3.0: spec_prior (72.0/40.0)           |
| housing = less_conv                              |
| children = more: spec_prior (144.0/80.0)         |
| children = 1.0: priority (144.0/64.0)            |
| children = 2.0: priority (144.0/80.0)            |
| children = 3.0: spec_prior (144.0/80.0)          |
| housing = critical                               |
| social = slightly_prob: priority (192.0/124.0)   |
| social = problematic: spec_prior (192.0/72.0)    |
| social = nonprob: priority (192.0/124.0)         |
| parents = usual: priority (864.0/354.0)          |

### Reglas generadas con Weka para Guardería

```
has_nurs = improper
| parents = pretentious
| | housing = convenient
| | | finance = convenient: priority (288.0/96.0)
| | | finance = inconv
| | | | children = more: spec_prior (72.0/40.0)
| | | | children = 1.0: priority (72.0/32.0)
| | | | children = 2.0: priority (72.0/40.0)
| | | | children = 3.0: spec_prior (72.0/40.0)
| | housing = less_conv
| | | children = more: spec_prior (144.0/80.0)
| | | children = 1.0: priority (144.0/64.0)
| | | children = 2.0: priority (144.0/80.0)
| | | children = 3.0: spec_prior (144.0/80.0)
| | housing = critical
| | | social = slightly_prob: priority (192.0/124.0)
| | | social = problematic: spec_prior (192.0/72.0)
| | | social = nonprob: priority (192.0/124.0)
| parents = great_pret: spec_prior (1728.0/716.0)
| parents = usual: priority (864.0/354.0)
has_nurs = critical
| parents = pretentious: spec_prior (951.0/411.0)
| parents = great_pret: spec_prior (1728.0/716.0)
| parents = usual
| | housing = convenient
| | | finance = convenient: priority (144.0/48.0)
| | | finance = inconv
| | | | children = more: spec_prior (36.0/20.0) Regla 1
| | | | children = 1.0: priority (36.0/16.0)
| | | | children = 2.0: priority (36.0/20.0)
| | | | children = 3.0: spec_prior (36.0/20.0)
| | housing = less_conv
| | | children = more: spec_prior (72.0/40.0) (Regla 2)
| | | children = 1.0: priority (72.0/32.0)
| | | children = 2.0: priority (72.0/40.0)
| | | children = 3.0: spec_prior (72.0/40.0)
| | housing = critical
| | | social = slightly_prob: priority (96.0/62.0)
| | | social = problematic: spec_prior (96.0/36.0)
| | | social = nonprob: priority (96.0/62.0)
```

Number of Leaves : 59

Size of the tree : 85

Fuente. Esta investigación

La Tabla 71 muestra el comparativo entre las reglas que coinciden entre las dos herramientas. De acuerdo a estos resultados se puede inferir que Pg\_KDD es una herramienta viable para el descubrimiento de reglas de clasificación.

**Tabla 71: Comparación reglas generadas por Pg\_KDD y Weka para mercado**

| Reglas generadas en Pg_KDD  | Reglas generadas en Weka   |
|---|--|
| 19 - parents = usual AND has_nurs = critical AND children = more AND housing = convenient AND finance = inconv --> recomendacion=spec_prior [32/72] | Regla 1: has_nurs = critical AND parents = usual AND housing = convenient AND finance = inconv AND children = more: spec_prior (36.0/20.0) |
| 21 - parents = usual AND has_nurs = critical AND children = more AND housing = less_conv --> recomendacion=spec_prior [24/36]                       | Regla 2: has_nurs = critical AND parents = usual AND housing = less_conv AND children = more: spec_prior (72.0/40.0)                       |

**Fuente.** Esta investigación

## 7. CONCLUSIONES

Partiendo de los resultados obtenidos en las pruebas de rendimiento se puede concluir que:

- ✓ Para las tareas de Asociación, la tendencia de los tiempos de ejecución del operador Associator Range y del algoritmo EquipAsso es exponencial, cuando el número de combinaciones de los ítemsets frecuentes aumenta y permanece constante el número de transacciones, y es lineal cuando se aumenta tanto el número de transacciones como el número de combinaciones de los itemsets frecuentes, por lo cual se recomienda hacer énfasis en la etapa de pre-procesamiento del repositorio a utilizar, donde se limpien los datos en blanco, datos inconsistentes o fuera de rango y así poder determinar cual será el tamaño máximo de combinaciones que deberá realizar el operador, con el cual se puede realizar una configuración adecuada del operador Associator Range.
- ✓ Según los resultados obtenidos en la generación de reglas de asociación con Pg\_KDD, en comparación con los obtenidos con la herramienta Weka aplicando el algoritmo Apriori, se presentan resultados similares, por consiguiente se demuestra la confiabilidad de la herramienta en la tarea de asociación.
- ✓ Teniendo en cuenta los resultados obtenidos con la aplicación del operador Mate y el algoritmo MateTree donde se obtuvieron reglas para la tarea de clasificación, se concluye que el incremento de atributos condición aumenta los tiempos de respuesta, no obstante, el incremento en el número de transacciones no afecta los tiempos de respuesta.
- ✓ A partir de la similitud en los resultados obtenidos con Pg\_KDD y Weka, para la generación de reglas de asociación y clasificación, donde se confirma la confiabilidad de la herramienta, se recomienda el uso de esta, conjuntamente con PostgresKDD en las electivas de descubrimiento de conocimiento en la Universidad de Nariño.

## 8. RECOMENDACIONES

Una vez descrito los resultados más relevantes que se han obtenido durante el desarrollo de este trabajo, se sugiere una serie de puntos de partida para futuros trabajos:

- ✓ Implementar los operadores que Timarán [31] ha propuesto para Asociación y Clasificación como son Enhance, Assocol. De igual manera, implementarlas dentro del módulo de minería de datos de Pg\_KDD.
- ✓ Implementar de manera fuertemente acoplada las funciones definidas por el usuario para Asociación y Clasificación, que se encuentran medianamente acopladas dentro de PostgresKDD, y realizar los cambios necesarios en el módulo de minería de Pg\_KDD.

Finalmente, este trabajo permitió aplicar los conocimientos adquiridos en el programa de Ingeniería de Sistemas, específicamente en la electiva de Minería de Datos y las diferentes cátedras de desarrollo de software.

## REFERENCIAS

1. Agrawal R., Mannila H., Srikant R., Toivonen H., Verkamo A.I., Fast Discovery of Association Rules, in Advances in Knowledge Discovery and Data Mining, AAAI Press / The MIT Press, 1996.
2. Chaudhuri S., Data Mining and Database Systems: Where is the Intersection?, Bulletin of the Technical Committee on Data Engineering, Vol.21 No. 1, Marzo, 1998.
3. Chen M., Han J., Yu P., Data Mining: An Overview from Database Perspective, IEEE Transactions on Knowledge and Data Engineering, 1996.
4. Clear, J., Dunn, D., Harvey, B., Heytens, M., Lohman, P., Mehta, A., Melton, M., Rohrberg, L., Savasere, A., Wehrmeister, R., Xu, M., NonStop SQL/MX Primitives for Knowledge Discovery, KDD-99, San Diego, USA, 1999.
5. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., From Data Mining to Knowledge Discovery: An Overview, in Advances in Knowledge Discovery and Data Mining, AAAI Pres/ The MIT Press, 1996.
6. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., The KDD Process for Extracting Useful Knowledge from Volumes of Data, Communications of the ACM, Vol 39, No. 11, November, 1996.
7. Fayyad U., Piatetsky-Shapiro G., Smyth P., Knowledge Discovery and Data Mining: Towards a Unifying Framework, The Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, 1996.
8. Fayyad U., Mining Databases: Towards Algorithms for Knowledge Discovery, Bulletin of the IEEE Computer Society, Vol.21, No. 1, March, 1998.
9. Freitas, A.A., Generic, Set-oriented Primitives to Support Data-parallel Knowledge Discovery in Relational Database Systems, Ph.D. diss.,

<http://cswww.essex.ac.uk/SystemsArchitecture/DataMining/alex/thesis.html>, 1997.

10. Freitas, A.A., Lavington, S.H., Using SQL Primitives and Parallel DBServers to Speed Up Knowledge Discovery in large relational databases, University of Essex, UK, <http://citeseer.nj.nec.com/cs>, 1997.
11. Han, J., Fu, Y., Wang, W., Chiang, J., Zaiane, O., Koperski, K., DBMiner: Interactive Mining of Multiple-Level Knowledge in Relational Databases, ACM SIGMOD, Montreal, Canada, 1996.
12. Han, J., Fu Y., Wang, W., Chiang, J., Koperski, K., Li, D., Lu Y., Rajan, A., Stefanovic, N., Xia, B., Zaiane, O., DBMiner: A System for Mining Knowledge in Large Relational Databases, The second International Conference on Knowledge Discovery & Data Mining, Portland, Oregon, 1996.
13. Han, J., Fu Y., Wang, W., Koperski, K., Zaiane, O., DMQL: A Data Mining Query Language for Relational Databases, SIGMOD 96 Workshop, On research issues on Data Mining and Knowledge Discovery DMKD 96, Montreal, Canada, 1996.
14. Han, J., Chiang, J., Chee, S., Chen, J., Chen, Q., Cheng, S., Gong, W., Kamber, M., Koperski, K., Liu, G., Lu, Y., Stefanovic, N., Winstone, L., Xia, B., Zaiane, O., Zhang, S., Zhu, H., DBMiner: A System for Data Mining in Relational Databases and Data Warehouses, CASCON: Meeting of Minds, Toronto, Canada, 1997.
15. Han, J., Kamber, M., Data Mining Concepts and Techniques, Morgan Kaufmann Publishers, San Francisco, 2001.
16. Han J., Cai Y., Cercone N., Knowledge Discovery in Databases: An Attribute-Oriented Approach, VLDB Conference, Vancouver, Canada, 1992.
17. Han J., Data mining, in J. Urban and P. Dasgupta (eds) Encyclopedia of Distributed Computing, Kluwer Academic Publishers, 1999.

18. Han J., Pei J., Yin Y., Mining Frequent Patterns without Candidate Generation, ACM SIGMOD, Dallas, Texas, USA, 2000.
19. Imielinski, T., Mannila, H., A Database Perspective on Knowledge Discovery, Communications of the ACM, Vol. 39, No 11, November 1996.
20. Imielinski, T., Virmani, A., Abdulghani, A., Data Mine: Application Programming Interface and Query Language for database Mining, 2<sup>o</sup> Conference KDD y Data Mining, Portland, Oregon, 1996.
21. Imielinski, T., Virmani, A., MSQL: A Query Language for Database Mining, Data Mining and Knowledge Discovery, Kluwer Academic Publishers, Vol 3, Number 4, pp .373-408, 1999.
22. Klosgen W., Some implementation aspects of a Discovery System, Knowledge Discovery in Databases Workshop, 1993.
23. Meo R., Psaila G., Ceri S., A New SQL-like Operator for Mining Association Rules, VLDB Conference, Bombay, India, 1996.
24. Meo R., Psaila G., Ceri S., An Extension to SQL for Mining Association Rules, Data Mining and Knowledge Discovery, Kluwer Academic Publishers, Vol 2, pp . 195-224, Boston, 1998.
25. Meo R., Psaila G., Ceri S., A Tightly-Coupled Architecture for Data Mining, 14th. International Conference on Data Engineering ICDE98, 1998.
26. Rajamani, K., Cox, A., Iyer, B., Chadha, A., Efficient Mining for Association Rules with Relational Database Systems, International Database Engineering and Application Symposium, p. 148-155, 1999.
27. Sarawagi S., Thomas S., Agrawal R., Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications, ACM SIGMOD, 1998.
28. Sarawagi S., Thomas S., Agrawal R., Integrating Association Rule Mining with Relational Database Systems: Alternatives and

Implications, Data Mining and Knowledge Discovery, Kluwer Academic Publishers, Vol 4, 2000.

29. Sarawagi S., Thomas S., Agrawal R., Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications, ACM SIGMOD, 1998.
30. Timarán, R., Arquitecturas de Integración del Proceso de Descubrimiento de Conocimiento con Sistemas de Gestión de bases de datos: UN ESTADO DEL ARTE, Ingeniería y Competitividad, Universidad del Valle, Volumen 3, No. 2, Cali, diciembre de 2001.
31. Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, INFORME AVANCE TESIS DOCTORAL, Universidad del Valle, diciembre 2002.
32. Timarán, R., Millán, M., Machuca, F., New Algebraic Operators and SQL Primitives for Mining Association Rules, in proceedings of the IASTED International Conference on Neural Networks and Computational Intelligence (NCI 2003), International Association of Science and Technology for Development, Cancun, Mexico, mayo 2003.
33. Schmuller, Joseph. Aprendiendo UML en 24 horas. Prentice-Hall, 2001
34. Internet - Wikipedia, La Enciclopedia libre- Paradigma Orientado a Objetos, [http://es.wikipedia.org/wiki/Java\\_\(lenguaje\\_de\\_programaci%C3%B3n\)](http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)). Visitado agosto de 2011.
35. Internet - Look & Feel - Nilo J. González, Madrid, <http://personales.ya.com/nimrod/index.html>. Visitado agosto de 2011.
36. Internet – InfoNode – Doking Windows – Tabbed Panel, NNL Technology AB. 2009, <http://www.infonode.net/>. Visitado agosto de 2011.
37. Castro C, Cabrera M., MATE-KDD: Una Herramienta Genérica Para El Descubrimiento De Reglas De Clasificación Medianamente Acoplada Al SGBD PostgreSQL, SAN JUAN DE PASTO, 2007. Tesis de Pregrado,

Universidad de Nariño.  
<http://biblioteca.udenar.edu.co:8085/bibliotecavirtual/viewer.aspx?&var=70974>. Visitado Abril 2012.

38. Booch, Grady; Rumbaugh, James; Jacobson, Ivar. El Lenguaje Unificado de Modelado. Madrid, 1999, Addison Wesley.
39. Booch, Grady; Rumbaugh, James; Jacobson, Ivar. El Proceso Unificado de Desarrollo. Madrid, 2000, Addison Wesley.
40. Quinlan, J. Ross. Induction of Decision Trees, Machine Learning, Kluwer Academic Publishers Hingham, MA, USA 1986.
41. Quinlan, J. Ross: C4-5: Programs for Machine Learning, Kluwer Academic Publishers Hingham, MA, USA, 1992.
42. Mehta, M., Agrawal, R., Rissanen. SLIQ: A Fast Scalable Classifier for Data Mining, 1996.
43. Cerquera C., Armero S., Diaz M., Guerrero M.: Implantación De Primitivas Sql Para El Descubrimiento De Reglas De Asociación Y Clasificación Al Interior Del Motor Del Sistema Gestor De Base De Datos PostgreSQL, 2005, Tesis de Pregrado, Universidad de Nariño.  
<http://biblioteca.udenar.edu.co:8085/bibliotecavirtual/viewer.aspx?&var=67436>
44. Internet – Wikipedia, La Enciclopedia libre, NetBeans - <http://es.wikipedia.org/wiki/NetBeans>. Visitado Agosto 2011
45. Internet – Wikipedia, La Enciclopedia libre, JDBC - <http://es.wikipedia.org/wiki/JDBC>. Visitado Agosto 2011
46. Internet – scribd, El ABC de Jdbc, Otero. Abraham, 2003 . <http://es.scribd.com/doc/84530902/EI-ABC-Jdbc2>. Visitado Agosto 2012
47. Internet - Wikipedia, La Enciclopedia libre, Swing (biblioteca gráfica) [http://es.wikipedia.org/wiki/Swing\\_%28biblioteca\\_gr%C3%A1fica%29](http://es.wikipedia.org/wiki/Swing_%28biblioteca_gr%C3%A1fica%29). Visitado agosto 2011.

48. Internet - Look and Feel  
[http://chuwiki.chuidiang.org/index.php?title=Look\\_and\\_Feel](http://chuwiki.chuidiang.org/index.php?title=Look_and_Feel). Visitado agosto 2011.
49. Timarán,R. Clasificación: una tarea de descubrimiento de conocimiento en bases de datos, San Juan de Pasto, Nariño, Colombia, 2009.
50. Timarán,R. Asociación: una método de descubrimiento de conocimiento en bases de datos, San Juan de Pasto, Nariño, Colombia, 2009.
51. Calderón A, Ramirez I, Alvarado J., Guevara A., TariyKDD: Una herramienta genérica De Descubrimiento De Conocimiento En Bases de Datos débilmente acoplada con El SGBD PostgreSQL”, 2007, Tesis de Pregrado, Universidad de Nariño.
52. Internet - Weka 3 - Data Mining with Open Source Machine Learning Software in Java, <http://www.cs.waikato.ac.nz/ml/weka/>. Visitado agosto 2012

## ANEXOS

### ANEXO A. Nuevas implementaciones en postgresKDD

#### Implementación Funcion definida por el usuario *convertir\_mate()*.

Figura A. 1 Implementación Funcion definida por el usuario *convertir\_mate()*.

```
CREATE OR REPLACE FUNCTION convertir_mate() RETURNS SETOF varchar AS '  
DECLARE  
.   sql      text DEFAULT ''''';  
.   sql1     text DEFAULT ''''';  
.   cad      text DEFAULT ''''';  
.   cada     text DEFAULT ''''';  
.  
.   rec1     record;  
.   rec2     record;  
.   rec3     record;  
BEGIN  
.   sql := 'SELECT attname,relnatts FROM pg_class, pg_attribute';  
.   sql := sql || ' WHERE pg_class.relname = ''kdd ent''';  
.   sql := sql || ' AND pg_attribute.attrelid = pg_class.oid';  
.   sql := sql || ' AND attnum > 0 ORDER BY attnum';  
.  
.   sql := 'SELECT atributo, valor, discret FROM kdd_values';  
.  
.   FOR rec1 IN EXECUTE 'SELECT relname FROM pg_class WHERE relname = ''tablemate_aux'' LOOP  
.     EXECUTE 'DROP TABLE tablemate_aux';  
.   END LOOP;  
.  
.   EXECUTE 'CREATE TABLE tablemate_aux AS SELECT * FROM tablemate';  
.  
.   FOR rec1 IN EXECUTE sql1 LOOP  
.     .   FOR rec2 IN EXECUTE sql LOOP  
.       .   .   IF cast(rec2.atributo as varchar) = cast(rec1.attname as varchar) THEN  
.           .   .   .   cad := 'UPDATE tablemate_aux set ||cast(rec1.attname as varchar)|| = ||cast(rec2.discret as varchar)||';  
.           .   .   .   cad := cad || ' WHERE ||rec1.attname|| = ||rec2.valor||';  
.           .   .   .   EXECUTE cad;  
.           .   .   .   END IF;  
.       .   .   END LOOP;  
.     .   END LOOP;  
.  
.   EXECUTE 'DELETE FROM kdd_clase';  
.   EXECUTE 'INSERT INTO kdd_clase SELECT * FROM tablemate_aux';  
.   EXECUTE 'DROP TABLE tablemate_aux';  
.  
.   RETURN NEXT ''OK'';  
.   RETURN;  
END;'  
LANGUAGE 'plpgsql' WITH (isstrict);
```

Fuente. Estas investigación.

## Modificación a la función definida por el usuario mate\_by()

Figura A. 2: Modificación a la Función definida por el usuario mate\_by().

```
CREATE OR REPLACE FUNCTION mateby(varchar) RETURNS SETOF varchar AS '  
DECLARE  
. sql . text DEFAULT ''';  
. sql1 . text DEFAULT ''';  
. cad . text DEFAULT NULL;.  
. cada . text DEFAULT ''';  
.  
. rec1 . record;.  
. rec2 . record;.  
. rec3 . record;.  
.  
. flag_val . integer;.  
. localset . integer;.  
. attr . integer;.  
. active . integer;.  
. control1 . integer;.  
. control2 . integer;.  
. control3 . integer;.  
. flag . integer;.  
. i . integer;.  
. num . integer;.  
. machete . integer;.  
.  
BEGIN  
. FOR rec1 IN EXECUTE ''SELECT relname AS flag_val FROM pg_class WHERE relname =''''pg_columna'''' LOOP  
END LOOP;  
. IF rec1.flag_val IS NOT NULL THEN  
. EXECUTE ''DROP TABLE pg_columna'';.  
END IF;  
. EXECUTE ''CREATE TABLE pg_columna(num int2,att bpchar)''; .  
.  
. ...  
.  
. ...  
.  
. EXECUTE ''DROP TABLE pg_columna'';  
. EXECUTE ''DROP TABLE pg_pos'';.  
. EXECUTE ''DROP TABLE pg_att'';.  
. EXECUTE ''DROP TABLE pg_mate'';.  
.  
. --Llamado a la función convertir_mate(), que convierte la información de la tabla Tablemate.  
. --generada por el operador Mate en la fuente de datos de la función entro().  
EXECUTE ''select * from convertir_mate()'';.  
.  
. RETURN NEXT ''CREADA kdd_clase'';  
RETURN;  
END;  
LANGUAGE 'plpgsql' WITH (isstrict);
```

Fuente. Estas investigación.

## Creación tabla ganancias en función definida por el usuario matetree().

Figura A. 3: Creación tabla ganancias en función definida por el usuario matetree().

```
CREATE OR REPLACE FUNCTION matetree(varchar) RETURNS SETOF varchar AS',
DECLARE
.   i,           integer;,
.   numatt, integer;,
.   nodo,       integer;,
.   ban,        integer;,
.   cp,         integer;,
.
.   Ent,        numeric;,
.   Ent_clase,  numeric;,
.   Gan,        numeric;,
.   G_min,     numeric :=999;
.
.   sql,        text DEFAULT '';
.   sql_g,     text DEFAULT '';
.   txt,        text DEFAULT '';
.   rta,        text DEFAULT NULL;
.
.   rec1,      record;
.   rec2,      record;
.
BEGIN
EXECUTE 'SELECT * FROM mateby(''*****||$1||*****)'';
FOR rec1 IN EXECUTE 'SELECT * FROM entro(''*****||$1||*****)' LOOP
.   Ent_clase := CAST(rec1.entro as numeric);
.   END LOOP;
.
.   ...
.   ...
.   ...
.
.   sql := 'SELECT atname,atnum FROM pg_class,pg_attribute WHERE ''
.   .   ||''pg_class.relname='''kdd ent''*****
.   .   || '' AND pg_attribute.attrelid = pg_class.oid AND atname !=*****
.   .   || $1 || ***** AND atnum >0 ORDER BY atnum'';
.
.   --Creación tabla Ganancias
.   FOR rec1 IN EXECUTE 'SELECT relname FROM pg_class WHERE relname = *****ganancias***** LOOP
.   .   EXECUTE 'DROP TABLE ganancias';
.   .   END LOOP;
.   .   EXECUTE 'CREATE TABLE ganancias(atributo varchar, clase varchar, ganancia numeric)';
.
.   FOR rec1 IN EXECUTE sql LOOP
.   .   .
.   .   EXECUTE 'INSERT INTO pg_pila VALUES(1,0,***** || rec1.atname || *****,-1)'';
.   .   .
.   .   sql_g := 'SELECT * FROM mate_gain(0,*****||$1||*****)'';
.   .   .
.   .   FOR rec2 IN EXECUTE sql_g LOOP
.   .   .   Gan := rec2.mate_gain;
.   .   .   --insert a la tabla ganancias
.   .   .   EXECUTE 'INSERT INTO ganancias VALUES(***** || rec1.atname || ***** ,*****|| $1 || ***** ,*****|| Gan ||*****)'';
.   .   .   END LOOP;
.   .   .
.   .   IF Gan < G_min THEN
.   .   .   G_min := Gan;
.   .   .   txt := rec1.atname;
.   .   .   numatt := rec1.atnum;
.   .   .   END IF;
.   .   .
.   .   EXECUTE 'DELETE FROM pg_pila WHERE num=1'';
.   .   END LOOP;
.
.   ...
.   ...
.   ...
END;'
LANGUAGE 'plpgsql' WITH (isstrict);
```

Fuente. Estas investigación.

## ANEXO B. Manual de instalación

Este manual explica la instalación para usar las diferentes funcionalidades de Pg\_KDD, un entorno gráfico para el sistema de descubrimiento de conocimiento en bases de datos PostgresKDD.

### 1. Instalación de PostgresKDD

- Se ingresa como usuario *root* y se desplaza al directorio `</usr/local/src/>` donde se ubicarán las fuentes de PostgresKDD.

```
$ su  
# cd /usr/local/src/  
# cp /cdrom/PostgresKDD.tar.gz.
```

- Se desempaqueta el archivo y se ingresa en el directorio generado.

```
# tar xvfz PostgresKDD.tar.gz  
# cd PostgreSQL-7.3.4
```

- Una vez se ha ubicado en el directorio es necesario, antes de la instalación, configurar el código fuente de la siguiente manera:

```
# ./configure
```

- Para compilar e instalar, se digitan las siguientes secuencias de comandos:

```
# gmake  
# gmake install
```

- Para el correcto funcionamiento de la herramienta es necesario la creación de un grupo para el súper-usuario *postgres*.

```
# groupadd postgres
```

- Se crea el súper-usuario *postgres*. Es indispensable la creación de este usuario para el buen funcionamiento de Pg\_KDD.

```
# adduser postgres
```

Dentro del proceso de creación del usuario es necesario realizar los ajustes que se indican a continuación.

- Se establece el “*Initial group*” o grupo inicial como *postgres*.

```
Initial group [users]: postgres
```

- Se establece el “*Home directory*” o el directorio home o de inicio a */usr/local/pgsql*.

```
Home directory [ /home/postgres ] : /usr/local/pgsql
```

```
Warning '/usr/local/pgsql/' already exists'
```

```
Do you to change the home directory path? (Y/n) n
```

```
Do you want to chown postgres.postgres /usr/local/pgsql/ ? (Y/n) y
```

- Se establece la contraseña del usuario *postgres*, preferiblemente como *postgres*.
- Se crea el directorio de datos y de generación de archivos de seguimiento y se le asigna permisos de propietario al usuario *postgres*.

```
# mkdir /usr/local/pgsql/data
```

```
# chown -R postgres:postgres /usr/local/pgsql/
```

Para los siguientes pasos es necesaria la identificación como usuario *postgres*.

- Identificación como usuario *postgres*.

```
# login postgres
```

- Se crea el archivo de configuración para las variables de ambiente necesarias para el funcionamiento de PostgresKDD.

```
$ pico .bash_profile
```

- En el archivo que se crea se ingresa lo siguiente:

```
PGUSER=postgres
PGDATA=/usr/local/pgsql/data
PGLIB=/usr/local/pgsql/lib
PATH=$PATH:/usr/local/pgsql/bin
```

```
Export PGUSER PGDATA PGLIB PATH
```

- Una vez se ha guardado el archivo, se realiza un *logout* y un *login* sobre el usuario *postgres*, así las variables de ambiente creadas se integrarán al sistema. Para verificar la correcta especificación de las variables ejecutar como usuario *postgres* el comando *env* y verificar la existencia de las variables creadas en el paso anterior.

```
$ env
```

- Iniciar las bases de datos básicas para el correcto funcionamiento de PostgresKDD.

```
$ /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
```

- Se inicia el demonio de PostgresKDD

```
$ /usr/local/pgsql/bin/postmaster -i -D /usr/local/pgsql/data&
```

- Se crea una base de datos llamada “*postgres*” y se inicia la terminal interactiva de PostgreSQL. La creación de esta base de datos es necesaria para el funcionamiento adecuado de la aplicación Pg\_KDD.

```
# /usr/local/pgsql/bin/createdb postgres
# psql postgres
```

Antes de utilizar las funciones de asociación es necesario configurarlas adecuadamente.

- Se ingresa como usuario *root* y se desplaza al directorio de asociación.

```
$ su
```

```
# cd /usr/local/src/PostgreSQL-7.3.4/contrib/KDD/asociacion
```

- Para compilar e instalar, se digitan las siguientes secuencias de comandos:

```
# gmake
# gmake install
```

Las FDO (Funciones definidas por el usuario) creadas para asociación y para clasificación se crearan en momento de ejecución del módulo del minería de datos, en un proceso transparente al usuario.

## 2 Instalación de Pg\_KDD

Una vez tenemos instalado y configurado PostgresKDD, se procede a instalar la herramienta Pg\_KDD en el directorio `</opt/>`

- Se ingresa como usuario *root*, se desplaza al directorio de asociación y se copia la herramienta desde el cd o medio de instalación.

```
$ su
# cd /opt/
# cp /cdrom/Pg_KDD.tar.gz.
```

- Se desempaqueta el archivo y se ingresa en el directorio generado.

```
# tar -xvfz Pg_KDD.tar.gz.
```

Para ejecutar la herramienta se ingresa al directorio `<.../dist>` y se ejecuta el comando `Java -jar`.

```
# cd /opt/Pg_KDD/dist
# Java -jar Pg_KDD.jar
```

## ANEXO C. Manual de usuario

Este manual explica cada funcionalidad que brinda Pg\_KDD, un entorno gráfico para el sistema de descubrimiento de conocimiento en bases de datos PostgresKDD

### 1 Inicio de la aplicación

El proceso de inicio de la aplicación carga un archivo de configuración donde se almacenan las conexiones existentes creadas en ejecuciones previas de la aplicación, si es la primera vez que se ejecuta la aplicación, este procedimiento se obviara.

Figura C. 1 Inicio de la aplicación

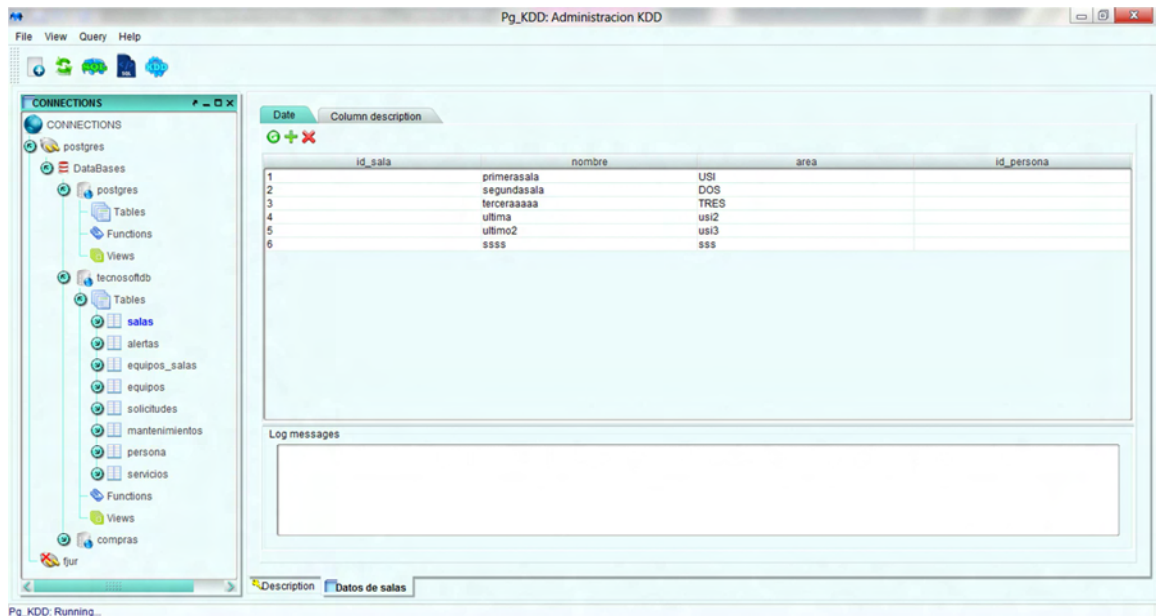


### 2 Marco principal

El marco principal de la aplicación muestra todos los controles y paneles de visualización para la gestión de las diferentes funcionalidades que Pg\_KDD provee,

A continuación se explicará el funcionamiento de cada uno de los controles y paneles del marco principal.

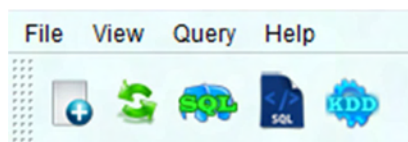
Figura C. 2 Marco principal



### 3 Barra de herramientas

En la barra de herramientas se ubica la barra de menú con sus diferentes opciones: File, View, Query y Help, al igual que los botones de acción New, Refresh, New Object, Delete Object, Graphics Statements, SQL WorkSheet y Data Mining. La funcionalidad de cada uno de ellos será descrita a continuación:

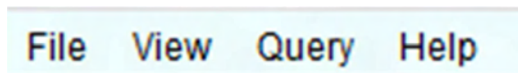
Figura C. 3 Barra de herramientas



#### Barra De Menú

Para acceder a cada opción de la barra de menú, simplemente ubique el cursor en la opción que desee y de clic con el botón izquierdo del mouse.

Figura C. 4 Barra de menú



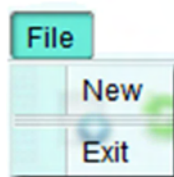
### Opción *File*.

Dentro de esta opción se tienen los siguientes ítems:

**New.** Permite la creación de un nuevo archivo en el sistema, abriendo una nueva ventana tipo “SQL WorkSheet” en donde se podrá editar código SQL y guardarlo en el disco de la computadora.

**Exit.** Permite salir de la aplicación.

Figura C. 5 Opción File



### Opción *View*.

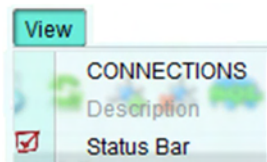
Dentro de esta opción se tienen los siguientes ítems:

**CONNECTIONS:** Permite ocultar o visualizar el panel “Tree Connections” o árbol de conexión.

**Description:** Permite ocultar o visualizar el panel “Description”.

**Status Bar:** Permite ocultar o visualizar la barra de estado de la aplicación.

Figura C. 6 Opción View



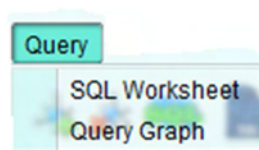
### Opción *Query*.

Dentro de esta opción se tienen los siguientes ítems:

**SQL WorkSheet:** Permite visualizar una nueva hoja de trabajo SQL o “SQL WorkSheet”, descrita en este manual en la sección 5 PANELES DE VISUALIZACION.

**Query Graph:** Permite visualizar el módulo “Query Graph” para la creación de sentencias gráficas para que será descrito detalladamente en la...sección SENTENCIAS GRAFICAS.

Figura C. 7 Opción Query



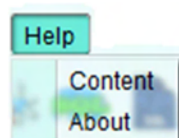
#### Opción Help.

Dentro de este menú se tienen las siguientes opciones:

**Content:** Permite visualizar el actual manual de usuario de Pg\_KDD.

**About:** Permite visualizar la versión y cierta información técnica de la aplicación, como también una breve descripción de los desarrolladores de Pg\_KDD.

Figura C. 8 Opción Help



#### 4 Botones de acción

Para acceder a cada opción de la barra de botones de acción, simplemente ubique el cursor en el botón que desee y de clic con el botón izquierdo del mouse.

Figura C. 9 Botones de Acción



#### Botón New

Crea una nueva conexión en el sistema desplegando una ventana “*Crear Conexión*” que será descrita detalladamente en la...sección “VENTANAS EMERGENTES”....

Figura C. 10 Botón New



### **Botón Refresh**

Actualiza o refresca la aplicación, cargando nuevamente el árbol de conexiones o “*Tree Conections*” y dejándolo en su estado inicial (cerradas todas sus hojas).

Figura C. 4 Botón Refresh



### **Botón New Object**

Dependiendo de la hoja u objeto seleccionado en el árbol de conexiones o “*Tree Conections*”, se mostrará una ventana emergente para realizar la acción de crear el objeto correspondiente.

### **Botón Graphics Statements**

Permite visualizar el módulo “*Query Graph*” para la creación de sentencias gráficas.

Figura C. 12 Botón Graphics Statements



### **Botón SQL WorkSheet**

Permite visualizar una nueva hoja de trabajo SQL o “SQL WorkSheet”, que esta descrita en este manual en la sección de “PANELES DE VISUALIZACION”.

Figura C. 13 Botón Sql WorkSheet



### **Boton Data Mining**

La acción de este botón permite visualizar el módulo “Data Mining” para la gestión de minería de datos que será descrito detalladamente mas adelante.

Figura C. 14 Botón Data Mining



## **5 Paneles de visualizacion**

### **Panel Connections**

Este panel contiene el objeto principal de la aplicación que es el “TREE CONNECTIONS”..

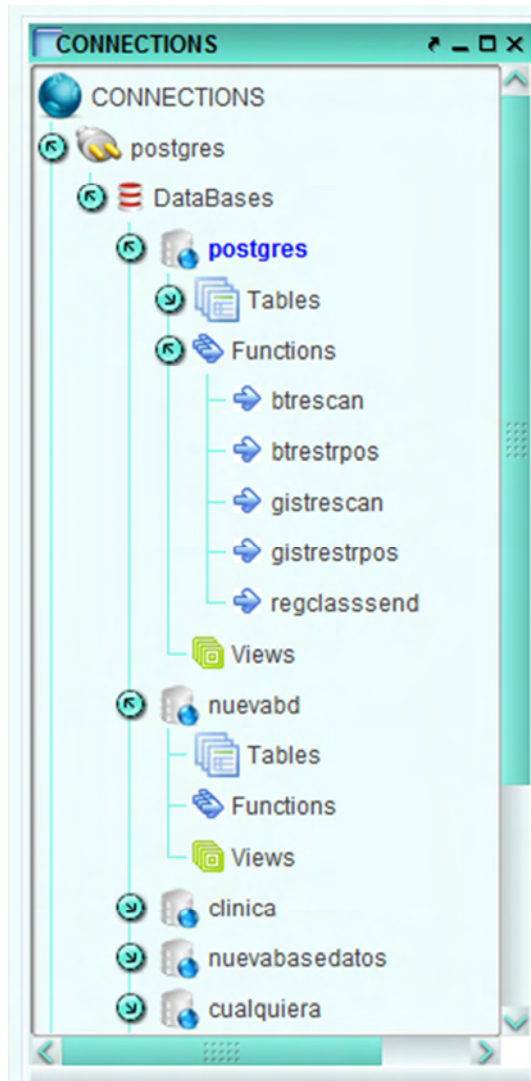
### **Tree Connections**

En el árbol de conexión se encuentran todos los objetos de la aplicación relacionados con bases de datos del gestor PostgresKDD, a diferencia de el objeto conexión que sirve para crear conexiones dentro de la aplicación, esta última despliega todos los objetos relacionados con un usuario como son bases de datos, tablas, funciones, reglas, disparadores, entre otros.

Cada uno de estos objetos tienen eventos de acción, al ubicarse en alguno de ellos y pulsar clic derecho se desplegará un pop menú con ítems específicos para cada objeto del árbol, como son generalmente, *new* (nuevo), *delete* (borrar), *properties* (propiedades), etc.

Si el objeto seleccionado es una tabla de una base de datos, este desplegará inmediatamente un panel “*Data Table*” donde se mostrarán los datos pertenecientes a esa tabla al igual que la descripción de los campos de la misma. Este panel esta descrito más detalladamente en la sección PANELES DE VISUALIZACION.

Figura C. 15 Tree Connection



### Panel Description

En este panel se muestra un mensaje la bienvenida a la aplicación y un breve resumen de las funcionalidades de Pg\_KDD.

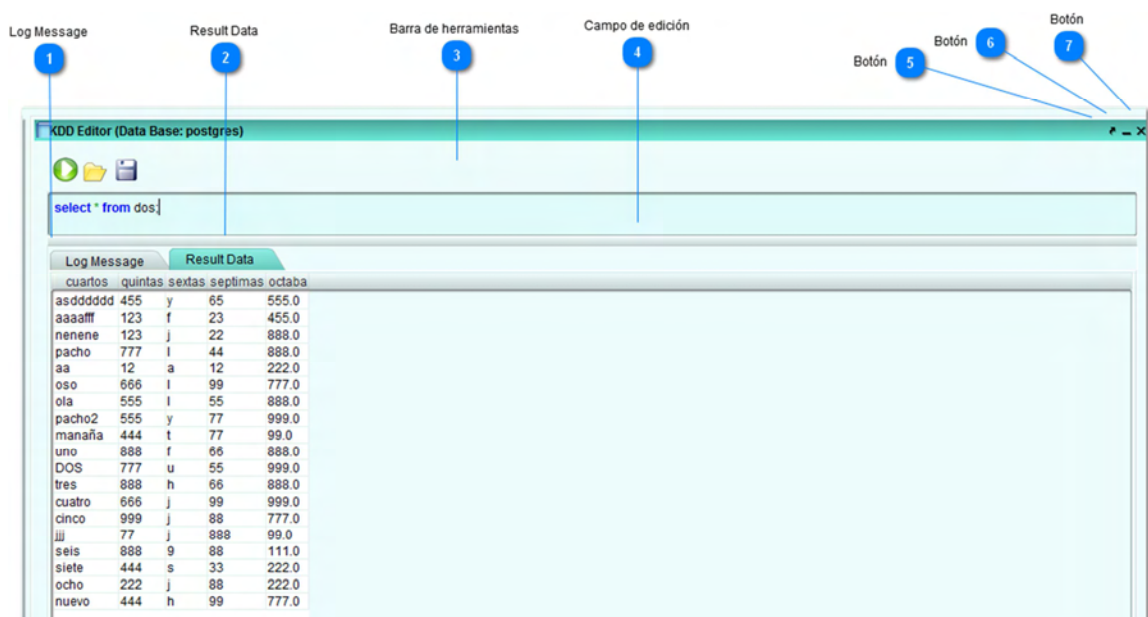
### Panel Sql WorkSheet

Este panel se ejecuta por dos motivos, primero cuando se crea un nuevo archivo desde la opción del menú "File", ítem "new" y segundo cuando el usuario crea una

nueva hoja de trabajo dirigiéndose al botón “SQL WorkSheet” de la barra de botones de acción.

En el primer caso el panel tomara el nombre del archivo que se esta creando, y en el segundo caso tomara el nombre de la base de datos que se esté manejando en el árbol de conexiones, no importando que objeto o nodo derivado a uno de tipo “Data Base” del árbol de conexiones este seleccionado.

Figura C. 16 Panel Sql WorkSheet



A continuación se describe los diferentes controles y elementos de este panel.

1

### Log Message:

Este es un panel donde se muestran los mensajes que resultan de las sentencias SQL escritas en el campo de edición, si una de las sentencias SQL escritas en el campo de texto se refiere a mas de una selección de datos de una base de datos, desplegará una tabla describiendo cada columna con su nombre y fila con su dato correspondiente.

2

### Result Data:

Al igual que el panel “*Log Message*” este panel muestra los resultados de las sentencias SQL escritas en el campo de edición solo si esta es una única sentencia de selección de datos, de lo contrario se mostrará en el panel *Log Message*.



### **Barra de herramientas:**

Esta barra de herramientas contiene los siguientes botones de acción:

- **Run:**

Este botón permite la ejecución de las sentencias escritas en el campo de edición, y visualizar sus resultados ya sea en “*Log Message*” como en “*Result Data*” según sea el caso.

Si las sentencias escritas producen cambios de inserción, actualización o eliminación en la base de datos actual, la ejecución de las mismas generara una barra de progreso dependiendo del número de sentencias escritas, y actualizará el “*Tree Connections*” o árbol de conexiones.

Figura C. 17 Botón Run



- **Open:**

Este botón permite abrir un archivo previamente guardado en disco y cargar su contenido en el campo de edición.

Figura C. 18 Botón Open



- **Save:**

Este botón permite guardar el contenido del campo de edición en un archivo.

Figura C. 19 Botón Save



4

**Campo de edición:**

Es un panel que contiene un área de texto en donde el usuario puede escribir sentencias SQL.

5

**Botón (Acoplamiento):**

Este botón permite el acoplamiento y desacoplamiento del panel "SQL WorkSheet" del marco principal, con el fin de poder trabajar con ella por separado.

6

**Botón (Minimizar):**

Este botón permite minimizar el panel dentro de la aplicación, si el panel esta minimizado se convertirá en botón (restaurar).

7

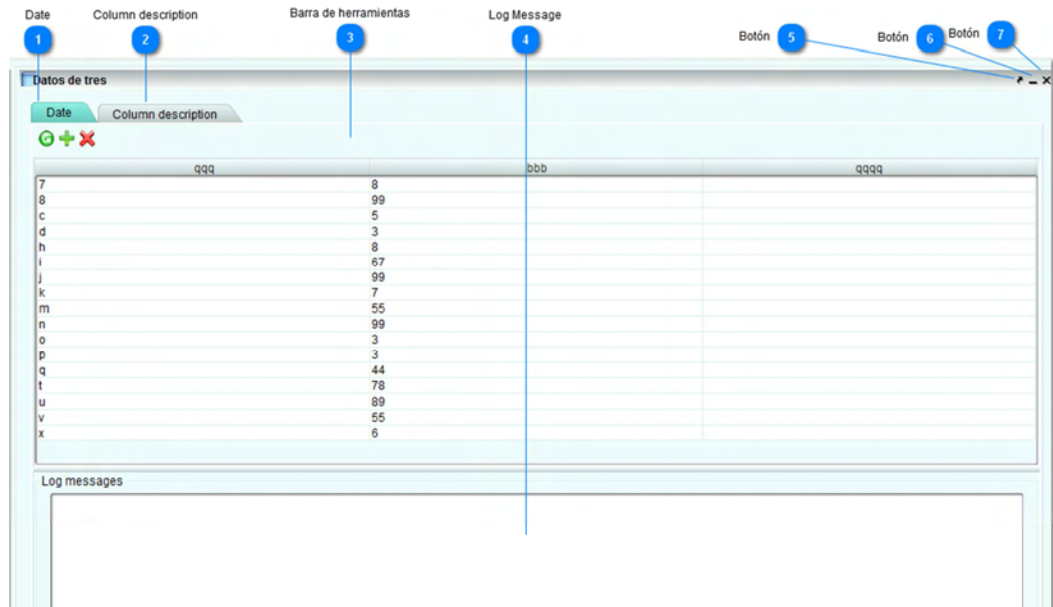
**Botón (Cerrar):**

Este botón permite cerrar el panel y si su contenido no ha sido guardado aun, se solicitará al usuario que lo haga si lo desea.

**Panel Data Table**

Este panel se mostrará cuando se seleccione del *Tree Connectios* un objeto tipo Tabla. En el panel se indicarán los registros de la tabla e información de las columnas de la misma.

Figura C. 20 Panel Data Table



1

**Data:**

Este es un panel que contiene una tabla describiendo los datos de la tabla seleccionada del árbol de conexiones.

Esta tabla permite la edición de las celdas contenidas, además si se sitúa en cualquier celda de la tabla y se presiona clic derecho, se generará un pop menú con la opción de exportar los datos a una archivo \*.HTML. El resultado de esto se muestra a continuación:

Figura C. 21 Reporte en Html

PG\_KDD  
KDD REPORTS

TABLE DESCRIPT:  
dos

| cuartos  | quintas | sextas | septimas | octaba |
|----------|---------|--------|----------|--------|
| asdddddd | 455     | y      | 65       | 555    |
| aaaafff  | 123     | f      | 23       | 455    |
| nenene   | 123     | j      | 22       | 888    |
| pacho    | 777     | l      | 44       | 888    |
| aa       | 12      | a      | 12       | 222    |
| oso      | 666     | l      | 99       | 777    |
| ola      | 555     | l      | 55       | 888    |
| pacho2   | 555     | y      | 77       | 999    |
| mana a   | 444     | t      | 77       | 99     |
| uno      | 888     | f      | 66       | 888    |
| DOS      | 777     | u      | 55       | 999    |
| tres     | 888     | h      | 66       | 888    |
| cuatro   | 666     | j      | 99       | 999    |
| cinco    | 999     | j      | 88       | 777    |
| jjj      | 77      | j      | 888      | 99     |

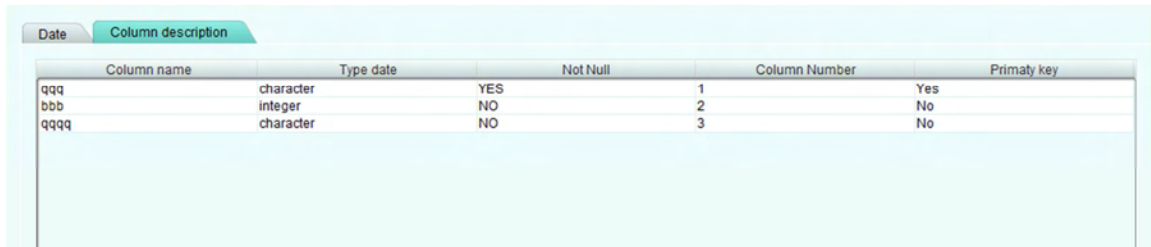
Este reporte permite realizar búsquedas para cada columna de la tabla, solo se debe hacer clic en el icono correspondiente e ingresar el valor deseado.

2

**Column Description:**

Este panel contiene una tabla que describe cada campo de la tabla. La información mostrada incluye nombre del campo, tipo de dato, si la columna acepta nulos o no, la posición del campo en la tabla, y si es llave primaria o no.

Figura C. 22 Column Description



| Column name | Type date | Not Null | Column Number | Primaty key |
|-------------|-----------|----------|---------------|-------------|
| qqq         | character | YES      | 1             | Yes         |
| bbb         | integer   | NO       | 2             | No          |
| qqqq        | character | NO       | 3             | No          |

3

### Barra de Herramientas:

Esta barra de herramientas del panel susodicho tiene los siguientes botones de acción:

- **Reload (Recargar):**

Ejecuta los cambios realizados en la tabla.

Figura C. 23 Botón Reload



- **Add Row (Adicionar Fila):**

Permite la adición de una nueva fila a la tabla.

Figura C. 24 Botón Add Row



- **Delete Row (Eliminar Fila):**

Elimina la fila seleccionada de la tabla.

Figura C. 25 Boton Delete Row



4

### Log Message:

Este es un panel donde se muestran los mensajes que resultan de las modificaciones que se le hacen a la tabla.

5

**Botón (Acoplamiento):**

Este botón permite el acoplamiento y desacoplamiento del panel “*Data Table*” del marco principal, con el fin de poder trabajar con ella por separado.

6

**Botón (Minimizar):**

Este botón permite minimizar el panel dentro de la aplicación, si el panel esta minimizado se convertirá en botón (restaurar).

7

**Botón (Cerrar):**

Este botón permite cerrar el panel eliminándola de las el panel pestañas.

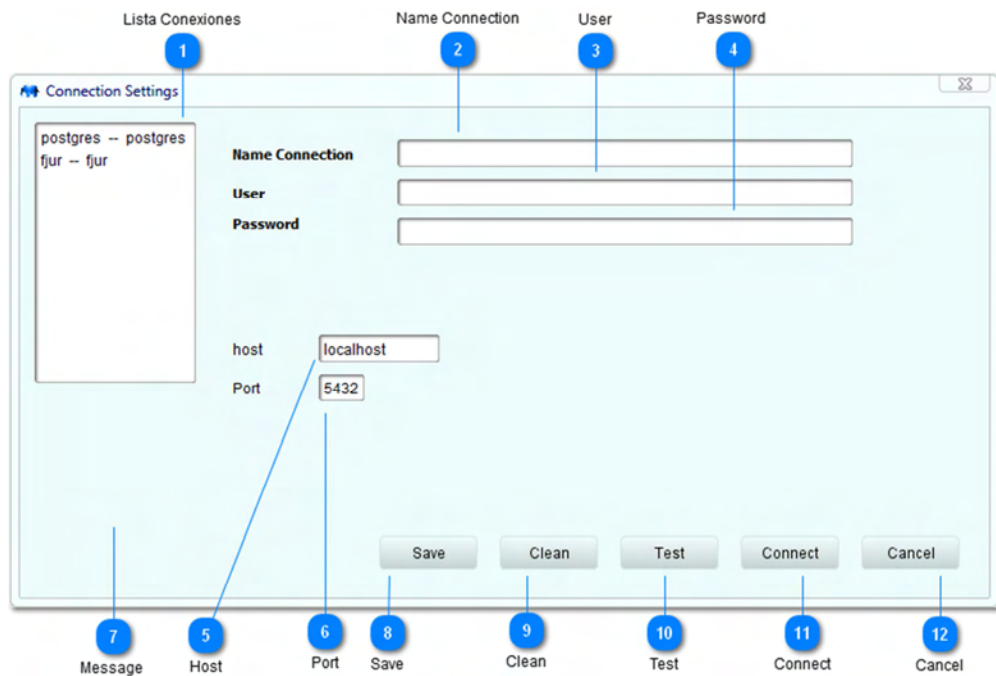
## 6 Ventanas emergentes

Esta sección del manual, se refiere a todas aquellas ventanas de visualización que resultan de la utilización de cada uno de los elementos de la aplicación Pg\_KDD. Cabe resaltar que cada una de las ventanas emergentes siguientes tiene doble funcionalidad: de creación o la edición para cada objeto del árbol de conexiones o “*Tree Connections*”. A continuación se describirán cada una de ellas.

### Crear/Modificar Conexión

Esta ventana emergente se mostrará al hacer clic derecho sobre un elemento *Connection* del *Tree Connection*, y seleccionar del pop menú la opción Nueva Conexión o Properties si es que se desea modificar una conexión existente.

Figura C. 26 Ventana Crear/Modificar conexión



1

**Lista de conexiones:**

En esta lista se visualizan todas las conexiones que se encuentran activas en el sistema.

2

**Name conexión:**

En este campo de texto se ingresa el nombre de la conexión.

3

**User:**

En este campo de texto se ingresa el nombre del usuario, con el que se iniciara la nueva conexión en el sistema.

4

**Password:**

En este campo de texto se ingresa la clave del usuario para poderse conectar con el gestor de base de datos PostgresKDD.

5

**Host:**

En este campo de texto se ingresa la ubicación del equipo donde se encuentra alojado el sistema gestor de base de datos.

6

**Port:**

En este campo de texto se ingresa el puerto por donde el servidor (Servicio de PostgresKDD) escucha las peticiones que se le hacen referidas a consultas al gestor. Este puerto deberá ser cambiado dependiendo del servicio creado previamente por el usuario.

7

Aquí reside un campo oculto, el cual tiene como funcionalidad visualizar los mensajes de testeado de la conexión, error de conexión y testeado de conexión exitosa.

8

**Save:**

Este botón tiene como funcionalidad grabar la conexión si así lo desea el usuario, mostrándola automáticamente en la lista de conexiones.

9

**Clean:**

Este botón tiene como funcionalidad limpiar los campos de la ventana emergente.

10

**Test:**

Este botón tiene como funcionalidad testear la conexión con los datos ingresados por el usuario, activará el campo oculto mostrando los mensajes pertinentes.

11

**Connect:**

Este botón tiene como funcionalidad crear la nueva conexión, recargando el árbol de conexiones y aumentándole la nueva conexión.

Pg\_KDD no permite la creación de conexiones con el mismo nombre ni múltiples conexiones con un mismo usuario.



**Cancel:**

Este botón tiene como funcionalidad cancelar la acción de creación de una conexión o edición de la conexión, cerrando la ventana emergente.

**Crear/Modificar Base De Datos**

Esta ventana emergente se mostrará al hacer clic derecho sobre un elemento *Data Base* del *Tree Connection* y seleccionar del pop menú la opción New DataBase o Properties si se desea modificar una ya existente.

Figura C. 27 Ventana New DataBase



1

**Properties:**

Este es un panel que contiene dos campos de texto uno para el nombre de la base de datos y otro para el propietario de la misma, este último es cargado automáticamente del árbol de conexión.

2

### SQL:

Este panel contendrá el código SQL generado de la creación o modificación que se realice a un objeto tipo base de datos del árbol de conexión o “*TREE CONNECTIONS*”.

Figura C. 28 Pestaña Sql en Ventana New DataBase



3

### OK:

Este botón tiene como funcionalidad la de ejecutar la sentencia SQL de creación o de modificación de la base de datos.

4

### Cancel:

Este botón tiene como funcionalidad cancelar la operación sobre el objeto base de datos del árbol de conexiones, cerrando la ventana emergente.

**Crear/Modificar Tablas**

Esta ventana emergente se mostrará al hacer clic derecho sobre un elemento *Tabla* del *Tree Connection*, y seleccionar del pop menú la opción *New Table* o *Properties* si es que se desea modificar una tabla existente.

Figura C. 29 Ventana Crear/Modificar Tablas



1

**Properties:**

Este es un panel que contiene dos campos de texto uno para el nombre de la tabla y otro para el propietario de misma, este ultimo es cargado automáticamente del árbol de conexión.

2

**SQL:**

Este panel contendrá el código SQL generado de la creación o modificación que se realice a un objeto tipo base de datos del árbol de conexión o "TREE CONNECTIONS".

Figura C. 30 Pestaña Sql en ventana crear/modificar tabla



3

**OK:**

Este botón tiene como funcionalidad la ejecución del SQL de creación o de modificación de una tabla.

4

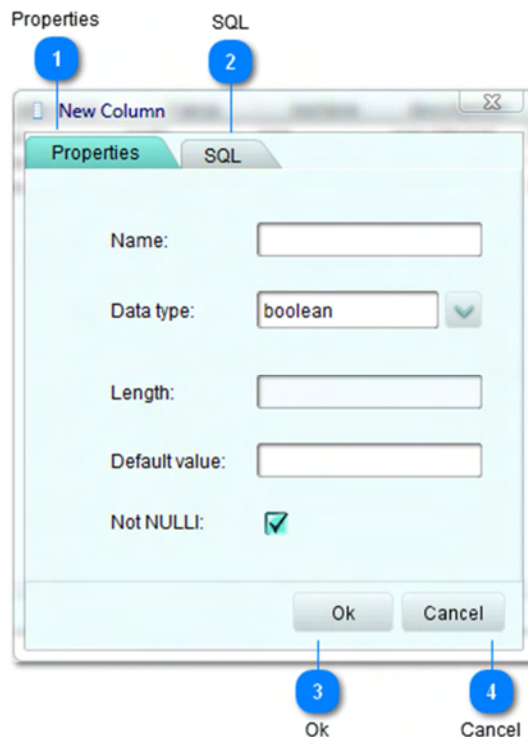
**Cancel:**

Este botón tiene como funcionalidad cancelar la operación sobre el objeto tabla del árbol de conexiones, cerrando la ventana emergente.

### Crear/Modificar Columnas

Esta ventana emergente se mostrará al hacer clic derecho sobre un elemento *Field* del *Tree Connection*, y seleccionar del pop menú la opción *New Column* o *Properties* si es que se desea modificar un columna existente.

Figura C. 31 Ventana Crear/Modificar columnas



1

#### Properties:

Este panel contiene los siguientes controles:

**Name:** En este campo se digitara o cargará el nombre de la columna.

**Data Type:** En esta lista desplegable se encuentran los tipos de datos mas representativos que se utiliza en PostgresKDD.

**Length:** Según sea el tipo de dato, este contendrá un tamaño que será ingresado por el usuario en este campo. Ejemplo: char (2).

**Default Value:** En este campo se ingresa el valor defecto de la columna.

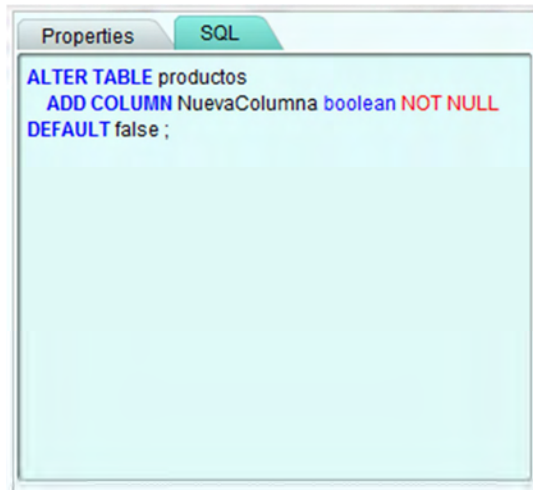
**Not NULL:** Indica si la columna acepte o no valores nulos.

2

#### SQL:

Este panel contendrá el código SQL generado de la creación o modificación que se realice a un objeto tipo columna del árbol de conexión o "Tree Connections".

Figura C. 32 Pestaña SQL en ventana Crear/Modificar Columna



3

**OK:**

Este botón tiene como funcionalidad la ejecución del SQL de creación o de modificación de una columna.

4

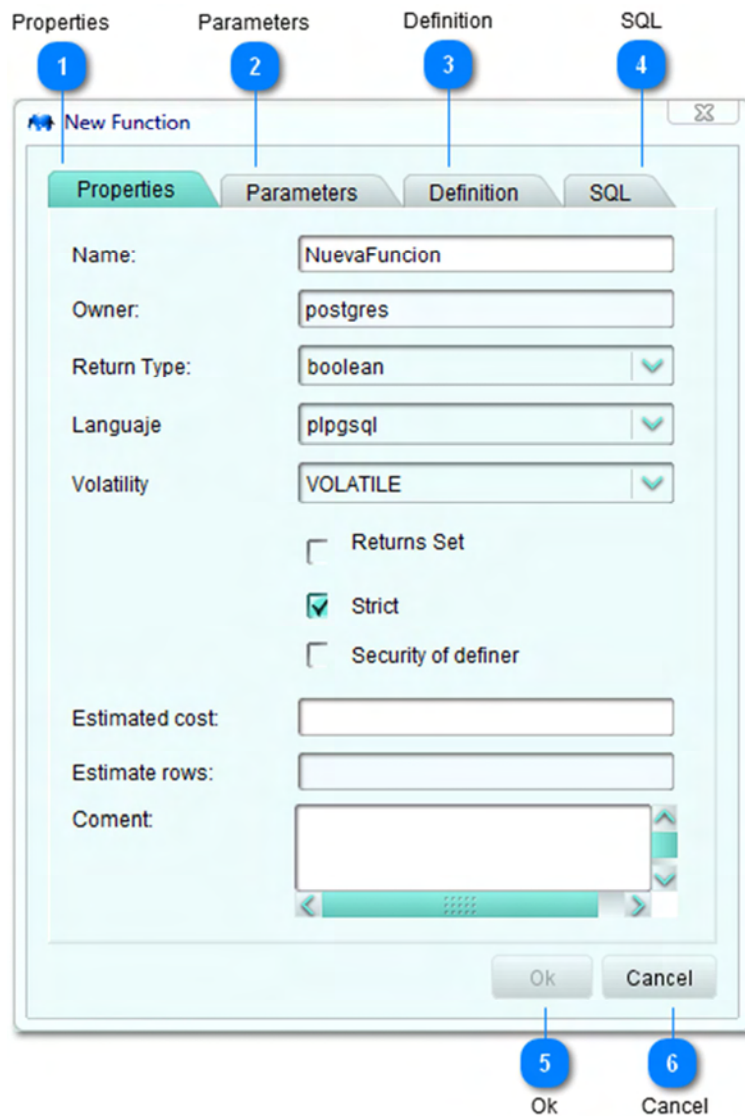
**Cancel:**

Este botón tiene como funcionalidad cancelar la operación sobre el objeto columna del árbol de conexiones, cerrando la ventana emergente.

### Crear/Modificar Funciones

Esta ventana emergente se mostrará al hacer clic derecho sobre un elemento *Función* del *Tree Connection*, y seleccionar del pop menú la opción *New Function* o *Properties* si es que se desea modificar una función existente.

Figura C. 5 Ventana Crear/Modificar Función



**1 Properties:**

Este panel contiene los siguientes controles:

**Name:** En este campo se ingresara o cargara el nombre de la función.

**Owner:** En este campo se cargará automáticamente el propietario de la función.

**Return Type:** En esta lista desplegable se encuentran los tipos de datos que retornara la función.

**Lenguaje:** En esta lista desplegable se encuentran los diferentes tipos de lenguajes que puede tener la definición de la función.

**Volatility:** Esta lista desplegable se encuentra las opciones de volatilidad de la función.

**Return Set:** Esta caja de chequeo se utiliza cuando se desea retornar un grupo de datos.

**Strict:** Se utiliza para especificar que la volatilidad de la función es estricta.

**Security of definer:** Se utiliza para especificar que la definición de la funciones es totalmente segura.

**Estimated cost:** En este campo se especifica el costo estimado de la función.

**Estimate rows:** En este campo se especifica el las filas estimadas para la función.

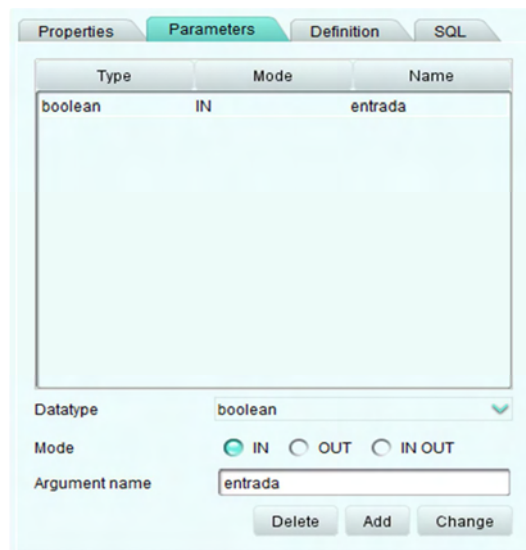
**Comment:** En este campo se especifica el comentario que se le puede dar a la función.

2

### Parameters:

Panel para ingreso de parámetros a la función.

Figura C. 6 Pestaña Parameters en Ventana de Función



En este panel se tienen los siguientes controles:

**Datatype:** Tipo de dato del parámetro.

**Mode:** Indica si el parámetro es de entrada o de salida o tipo InOut

**Argument Name:** Nombre del parámetro.

3

### Definition

Ingrese aquí la definición de la función.

Figura C. 7 Pestaña Definition en Ventana Función

```
BEGIN
RAISE NOTICE "Cantidad contiene aqui %",cantidad;

cantidad := 50;

DECLARE
cantidad INTEGER := 80;
BEGIN
RAISE NOTICE "Cantidad contiene aqui %",cantidad;

END;
RAISE NOTICE "Cantidad contiene aqui %",cantidad;

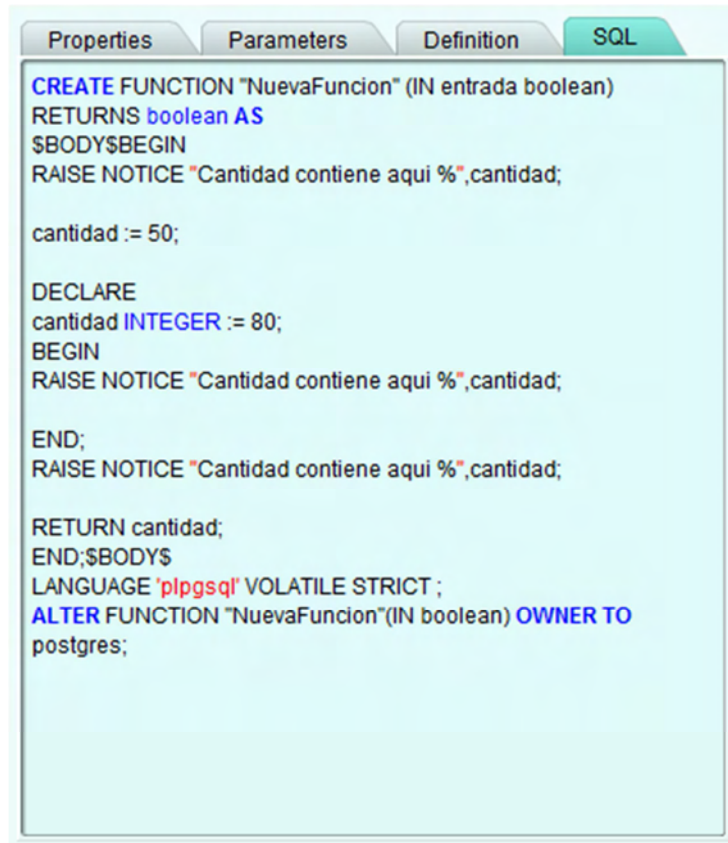
RETURN cantidad;
END;
```



#### SQL

Código SQL generado para la función que se desea crear.

Figura C. 8 Pestaña SQL en Ventana Función



```
CREATE FUNCTION "NuevaFuncion" (IN entrada boolean)
RETURNS boolean AS
$BODY$BEGIN
RAISE NOTICE "Cantidad contiene aqui %",cantidad;

cantidad := 50;

DECLARE
cantidad INTEGER := 80;
BEGIN
RAISE NOTICE "Cantidad contiene aqui %",cantidad;

END;
RAISE NOTICE "Cantidad contiene aqui %",cantidad;

RETURN cantidad;
END;$BODY$
LANGUAGE 'plpgsql' VOLATILE STRICT ;
ALTER FUNCTION "NuevaFuncion"(IN boolean) OWNER TO
postgres;
```

5

**OK:**

Una vez haya definido las propiedades, parámetros y definición de la función haga clic en el botón OK para guardar o modificar la función.

6

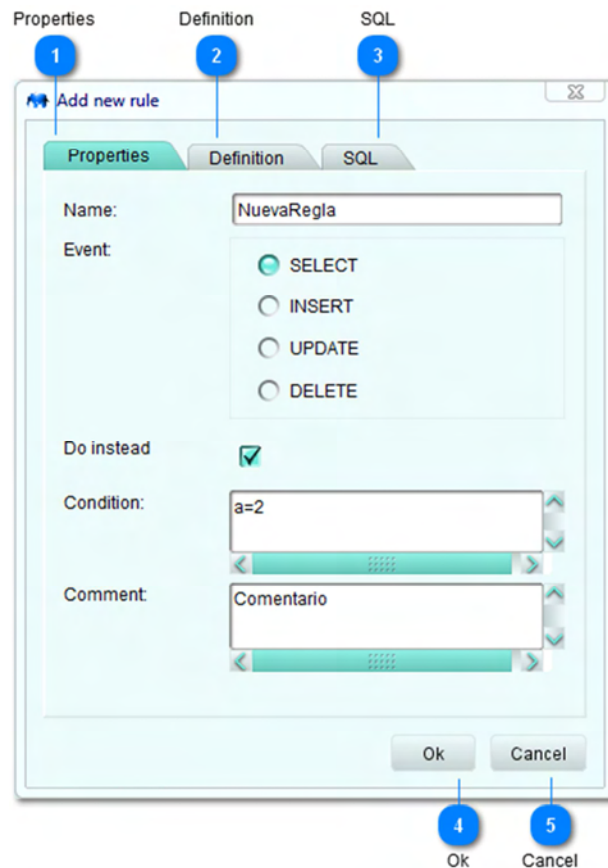
**CANCEL:**

Este botón tiene como funcionalidad cancelar la operación sobre el objeto columna del árbol de conexiones, cerrando la ventana emergente.

## Crear/Modificar Reglas

Esta ventana emergente se mostrará al hacer clic derecho sobre un elemento *Rules* del *Tree Connection*, y seleccionar del pop menú la opción *New Rule* o *Properties* si es que se desea modificar una regla existente.

Figura C. 37 Ventana Crear/Modificar Reglas



### 1 Properties

Este panel contiene los siguientes controles

**Name:** En este campo se cargará o ingresará el nombre de la regla.

**Event:** Indica cuando debe ejecutarse la regla.

**Do instead:** Si este control está seleccionado indica que los comandos deben ser ejecutados en lugar del comando original.

**Condition:** Indica la condición que debe cumplir la regla para su ejecución.

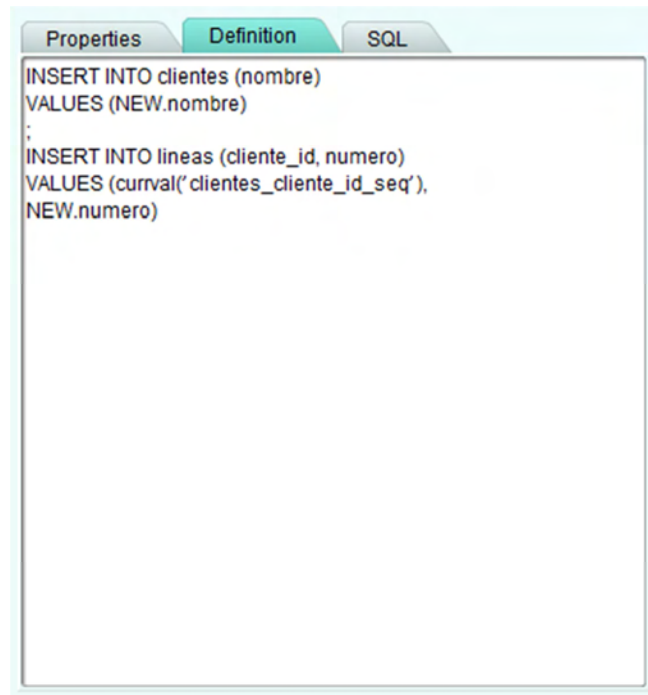
**Comment:** Comentario para la regla.

2

### Definition

Ingrese aquí la definición para la regla que desea ingresar o que desea modificar.

Figura C. 38 Pestaña Definition en Ventana Reglas

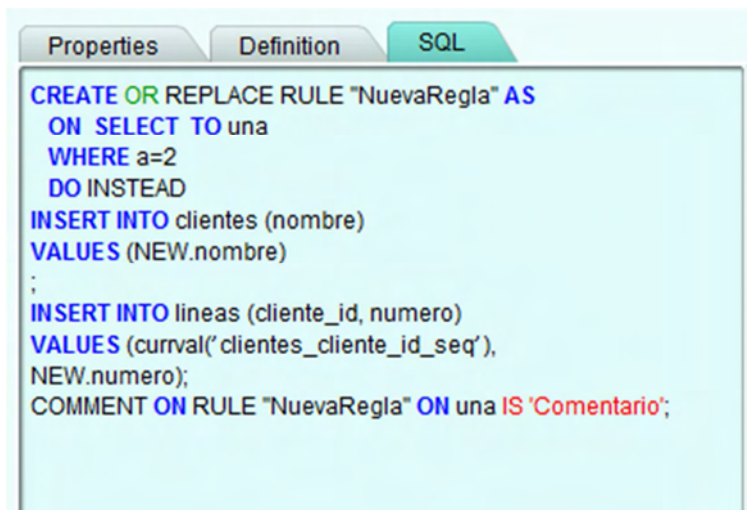


3

### SQL

Este panel muestra el código SQL generado para la regla que se esta ingresando o modificando

Figura C. 39 Pestaña SQL en Ventana Reglas



```
CREATE OR REPLACE RULE "NuevaRegla" AS
ON SELECT TO una
WHERE a=2
DO INSTEAD
INSERT INTO clientes (nombre)
VALUES (NEW.nombre)
;
INSERT INTO lineas (cliente_id, numero)
VALUES (curval('clientes_cliente_id_seq'),
NEW.numero);
COMMENT ON RULE "NuevaRegla" ON una IS 'Comentario';
```

4

**OK:**

Este botón tiene como funcionalidad la ejecución del SQL de creación o de modificación de la regla.

5

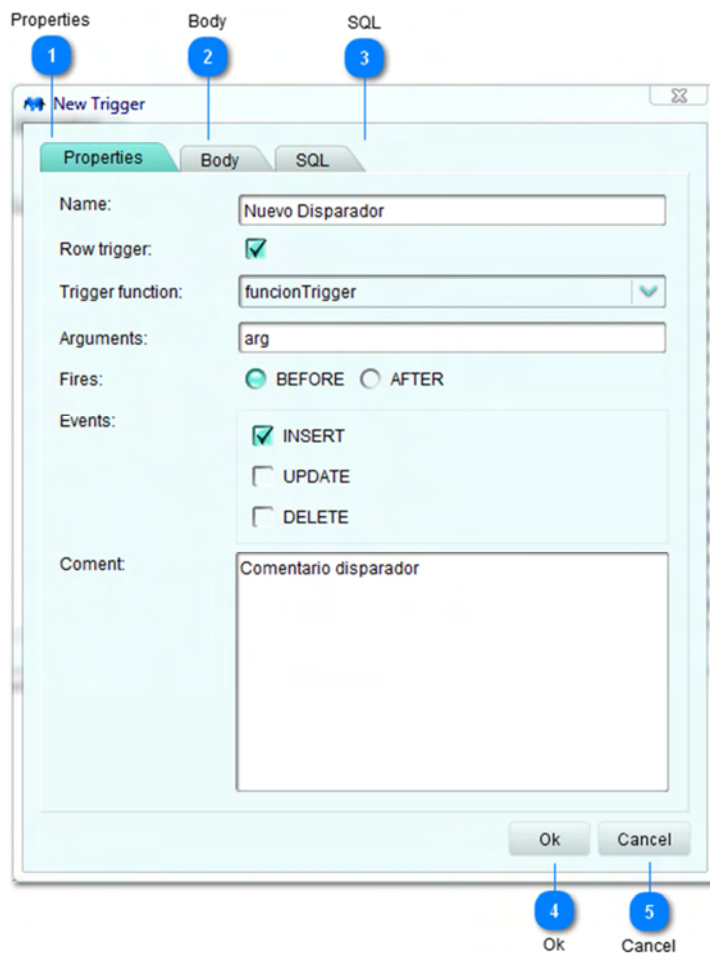
**CANCEL:**

Este botón tiene como funcionalidad cancelar la operación sobre el objeto columna del árbol de conexiones, cerrando la ventana emergente.

### **Crear/Modificar Disparadores**

Esta ventana emergente se mostrará al hacer clic derecho sobre un elemento *Trigger* del *Tree Connection*, y seleccionar del pop menú la opción New Trigger o Properties si es que se desea modificar una disparador existente.

Figura C. 9 Ventana Crear/Modificar Disparadores



### 1 Properties

Este panel contiene los siguientes controles:

**Name:** Nombre del disparador.

**Row trigger:** Si esta seleccionado indica que el procedimiento del disparador debe ser ejecutado por cada fila afectada por el evento especificado para el disparador. Si no se especifica la ejecución se hará por cada sentencia SQL.

**Trigger Function:** Muestra el nombre de la función que ejecutará el disparador.

**Arguments:** Una lista de argumentos que se proporcionan a la función cuando se ejecute el disparador.

**Fires:** Seleccione para determinar si la función es llamada antes o después del evento.

**Events:** Especifica el evento que lanza el disparador.

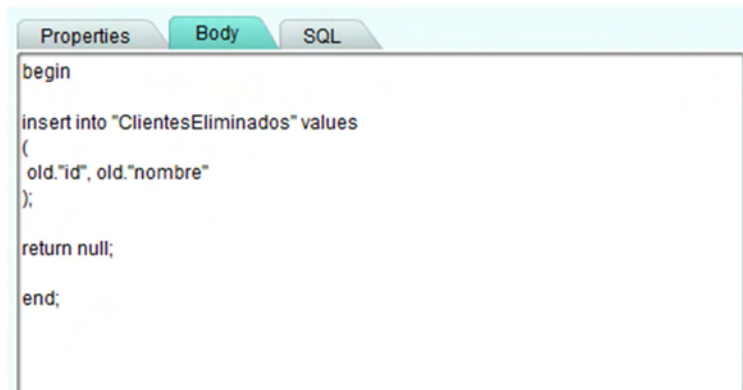
**Comments:** Comentarios al trigger.

2

## Body

Ingrese en este panel la definición para el disparador.

Figura C. 10 Pestaña Body en Ventana Disparadores



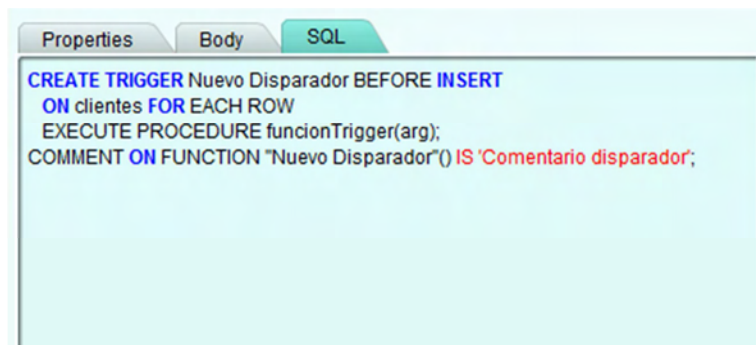
```
begin
insert into "ClientesEliminados" values
(
old."id", old."nombre"
);
return null;
end;
```

3

## SQL

Este panel muestra el código SQL generado para el disparador que se esta ingresando o modificando.

Figura C. 11 Pestaña SQL en Ventana Disparadores



```
CREATE TRIGGER Nuevo Disparador BEFORE INSERT
ON clientes FOR EACH ROW
EXECUTE PROCEDURE funcionTrigger(arg);
COMMENT ON FUNCTION "Nuevo Disparador"() IS 'Comentario disparador';
```

4

## OK

Este botón tiene como funcionalidad la ejecución del SQL de creación o de modificación del disparador según sea el caso.



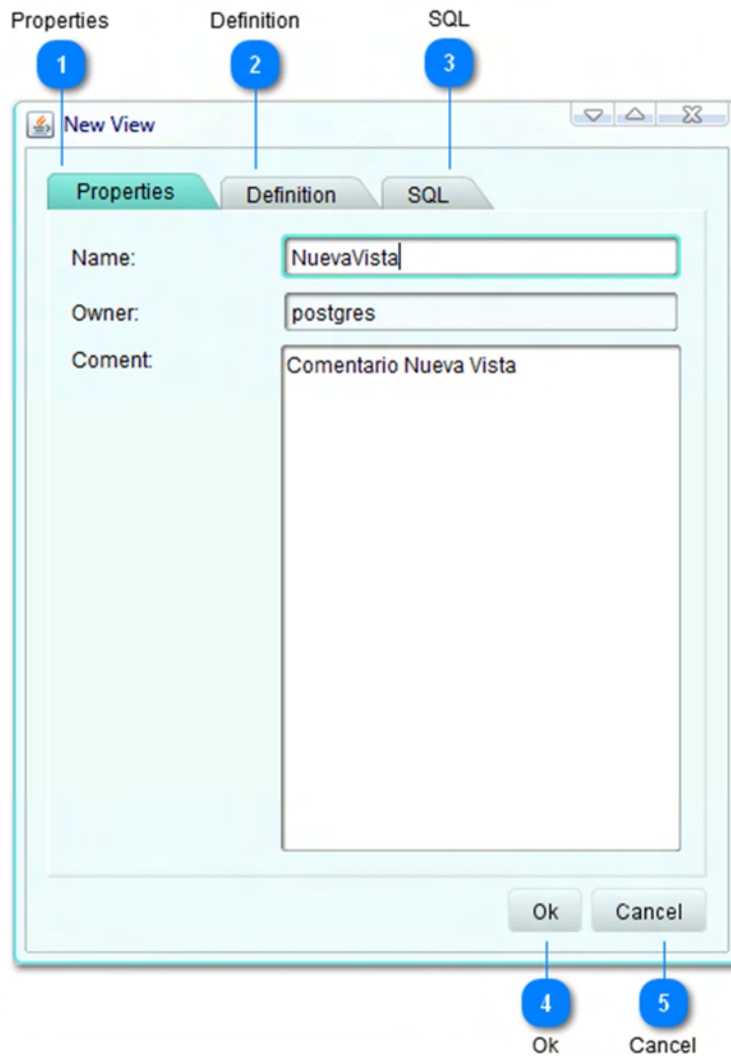
## **CANCEL**

Este botón tiene como funcionalidad cancelar la operación sobre el objeto columna del árbol de conexiones, cerrando la ventana emergente.

## **Crear/Modificar Vistas**

Esta ventana emergente se mostrará al hacer clic derecho sobre un elemento *View* del *Tree Connection*, y seleccionar del pop menú la opción *New View* o *Properties* si es que se desea modificar una vista existente.

Figura C. 12 Ventana Vistas



### 1 Properties

Este panel contiene los siguientes controles:

**Name:** Nombre de la vista que se esta creando o modificando.

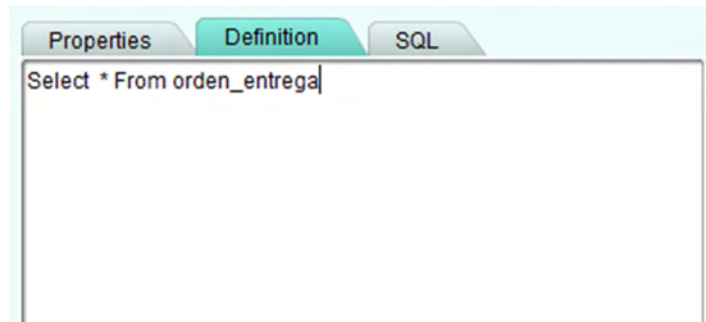
**Owner:** En este campo se cargara automáticamente el propietario de la función obtenida a la hora de la creación e la conexión.

**Comment:** Comentarios para la vista.

### 2 Definition

Especifique en este panel la definición de la vista.

Figura C. 13 Pestaña Definition en Ventana Vista



### 3 SQL

Este panel muestra el código SQL generado para la vista.

Figura C. 14 Pestaña SQL en Ventana Vista



### 4 OK

Este botón tiene como funcionalidad la ejecución del SQL de creación o de modificación del disparador.

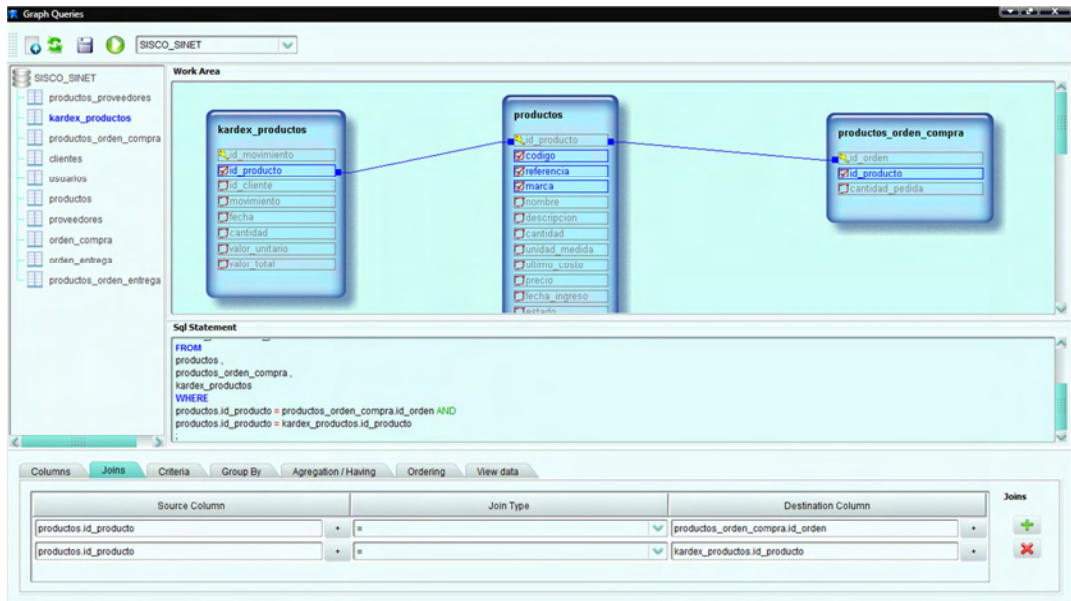
### 5 CANCEL

Este botón tiene como funcionalidad cancelar la operación sobre el objeto columna del árbol de conexiones, cerrando la ventana emergente.

## 7 SENTENCIAS GRAFICAS

Esta interfaz permite de manera gráfica realizar operaciones de consulta sobre las tablas en el repositorio.

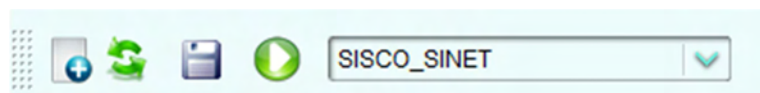
Figura C. 15 Módulo Query Graph



### 7.1 Barra de herramientas

En la barra de herramientas encontramos visibles los botones de acción New, Refresh, New Object, Save, Play y una lista desplegable denominada Data Base, cada uno de ellos nos brindan funcionalidades que serán descritas a continuación:

Figura C. 47 Barra de herranmentas Query Graph



#### Botón New

Crema una nueva ventana para la realización de nuevas consultas.

Figura C. 48 Botón New



### **Botón Refresh**

Actualiza la ventana de trabajo actual.

Figura C. 49 Botón Refresh



### **Botón Save**

Despliega un cuadro de búsqueda que muestra la estructura de archivos del sistema permitiéndole guardar su trabajo.

Figura C. 50 Botón Save



### **Botón Play**

Ejecuta la sentencia que se haya especificado.

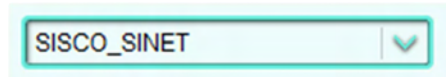
Figura C. 16 Botón Play



### **Combo Data Base**

Esta lista muestra las diferentes bases de datos que están creadas en el sistema. Al seleccionar una se cargarán las tablas correspondientes a esta base de datos.

Figura C. 17 Combo Data Base

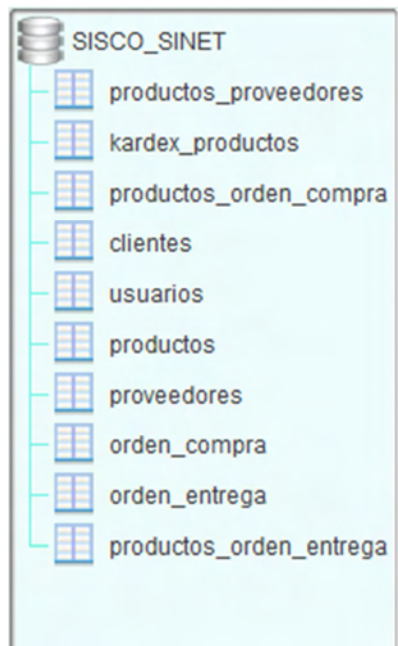


## 7.2 Paneles de visualización

### Tree Tables

Este panel muestra las tablas que pertenecen a la base de datos seleccionada en el combo *Data Base*.

Figura C. 18 Panel Tree Tables



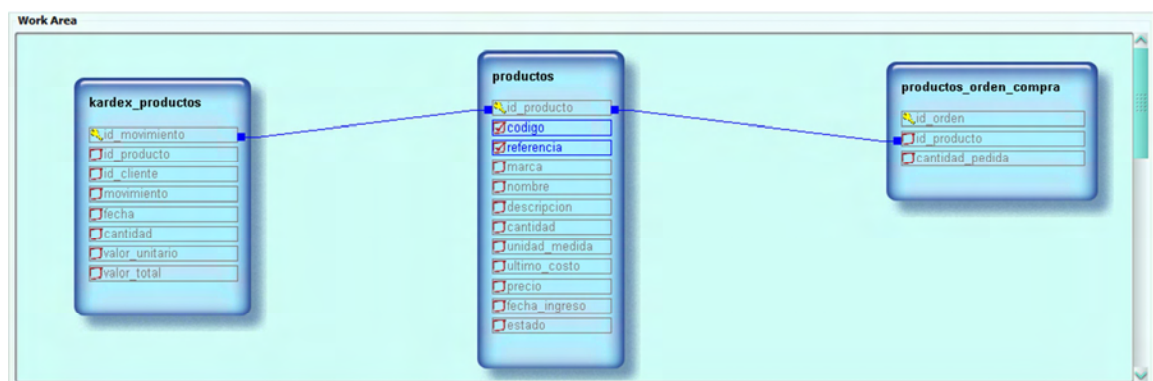
### Work Area

Es el área donde se muestran las tablas que ha seleccionado para realizar la consulta y las diferentes acciones que se ejecutaran sobre ellas. Para agregar una tabla al área de trabajo, selecciónela del *tree tables* y con clic sostenido arrástrela hasta el *work área*. Una representación de la tabla y sus campos se mostrará. Junto a cada nombre de columna de cada tabla hay una caja de selección.

Márquela si se desea que el campo sea incluido en la consulta. Si no se marca ninguna la consulta mostrará todos los campos.

Si desea hacer consulta sobre más de una tabla, seleccione la nueva tabla y arrástrela hasta el área de trabajo. Para especificar relación entre ellas defina un *join* en la pestaña del panel inferior o con clic sostenido marque el campo deseado en la tabla uno y arrastre hasta el campo en la tabla dos. Esto creará una relación que se mostrará como una línea que une los dos campos.

Figura C. 19 Panel Work Area



## Sql Statement

Este panel muestra en tiempo real la consulta que se ejecutará para la o las tablas en el *work área*.

Figura C. 20 Panel Sql Statement

```
Sql Statement
SELECT
productos.codigo ,
productos.referencia
FROM
kardex_productos ,
productos ,
productos_orden_compra
WHERE
productos.id_producto = kardex_productos.id_movimiento AND
productos_orden_compra.id_producto = productos.id_producto
;
```

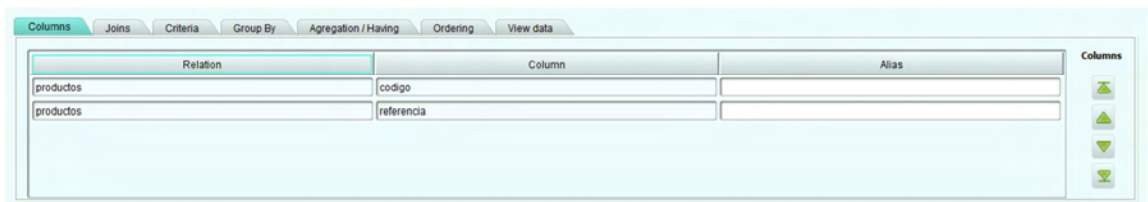
## Panel Pestañas

Este panel permite realizar ajustes a la consulta que se ejecutará para las tablas del work area. Incluye las siguientes pestañas.

## Columns

Si en las tablas del *work area* se ha marcado la casilla de chequeo de algún campo, este se mostrará en esta pestaña. Se indica el nombre de la relación, el nombre de la columna y si se desea puede especificar un alias para el campo. Los controles a la derecha permiten ajustar el orden en que se mostrarán los campos seleccionados.

Figura C. 21 Pestaña Columns



## Joins

Permite establecer joins entre las tablas del *work area*.

Para agregar un nuevo *join* haga clic en el control de la derecha con figura de un mas (+). Seleccione el campo en la *Source Column* de la lista desplegable, este es el campo en la tabla uno que se relacionará con otro campo en la tabla dos.

Seleccione el tipo de *join* de la lista desplegable, y al final selección de la *Destination Column* el campo de la tabla destino.

En el *work area* se dibujará una línea entre los campos relacionados.

Para eliminar el *join*, selecciónelo y haga clic en el control de la derecha con figura de una equis (X)

Figura C. 57 Pestpaña Joins



## Criterio

Permite especificar criterios de restricción para los campos de las tablas seleccionadas

Para agregar un nuevo criterio haga clic en el botón de la derecha con una figura de mas (+).

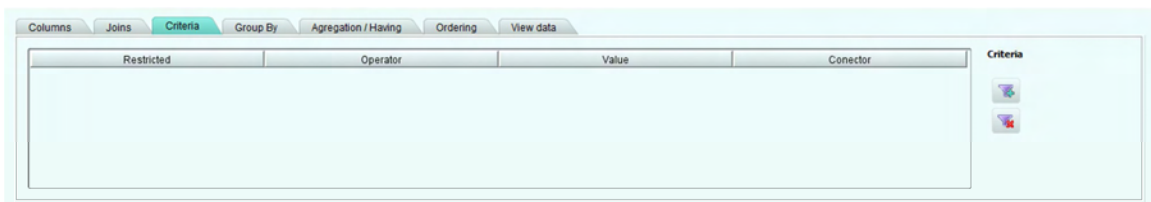
Seleccione el campo que se verá afectado por la restricción.

Seleccione el tipo de operador para la restricción de la lista desplegable.

Seleccione el campo con el cual deberá analizarse la restricción para el campo primero.

Si se desea agregar más restricciones seleccione el tipo de conector que los relacionará.

Figura C. 58 Pestaña Criterio



## Group By

Permite establecer los parámetros de agrupamiento para la consulta actual.

En la columna de la derecha se mostrarán los campos de las tablas en el *work area*. Seleccione el campo por el que desea agrupar los resultados de la consulta y con los controles en la columna central agréguelo a la columna de la derecha. Si desea eliminar un criterio de agrupamiento, selecciónelo y con el control de la columna central quítelo de la columna de la derecha.

Figura C. 59 Pestaña Group by



## Agregation/Having

Permite establecer funciones de agregado y si se desea restringir los resultados de estas funciones.

En el panel se mostrarán los campos seleccionados de las tablas en el *work area*. Especifique, si es necesario, el tipo de función de agregado que se aplicará sobre el campo. Ingrese un alias y restrinja los resultados de la función de agregado seleccionada seleccionando un valor de la lista desplegable en la columna "*having*", indique el tipo de operador y el valor con el que se comparará el campo seleccionado.

Figura C. 22 Pestaña Agregacion/Having

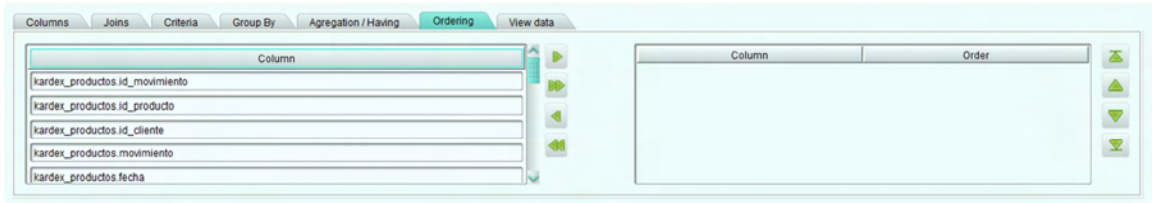


## Ordering

Permite especificar los campos por los que los resultados se ordenaran.

Seleccione de la columna de la izquierda los campos para la clausula de ordenamiento. En la columna de la derecha seleccione el criterio de ordenamiento (*ASC*, *DESC*).

Figura C. 23 Pestaña Ordering



## View Data

Este panel muestra los resultados de la consulta generada para las tablas del *work area*.

Figura C. 24 Pestaña View Data



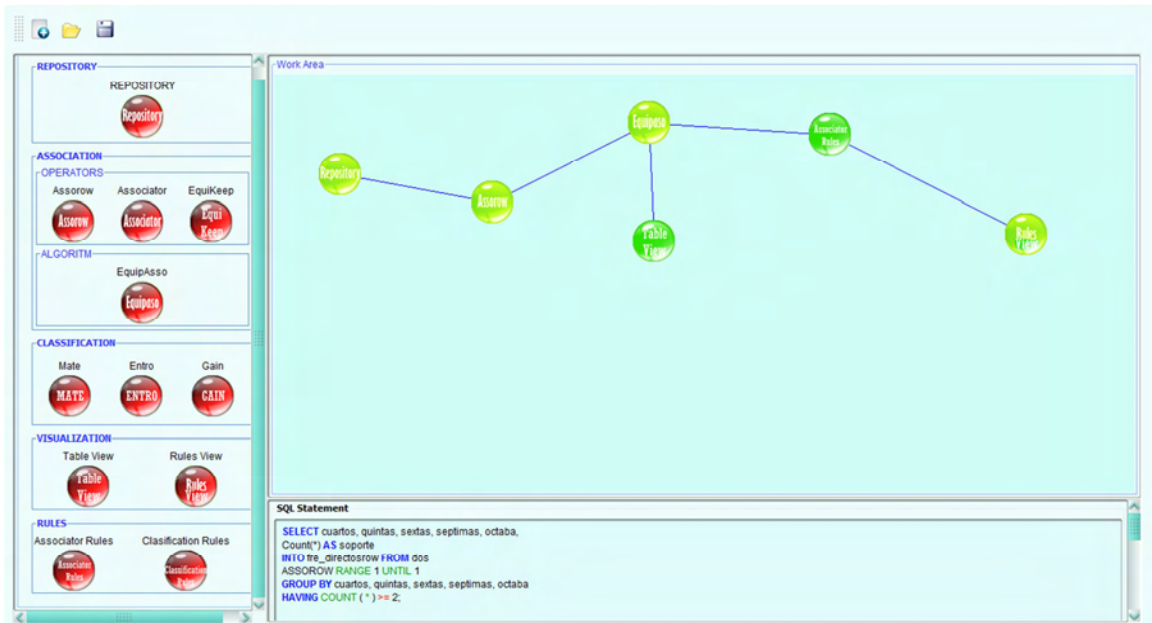
| productos.id_producto | productos.codigo | productos.referencia | productos.marca | productos.nombre | productos.descripcion | productos.cantidad | productos.unidad_medida | productos.ultimo_costo | productos.precio | productos.fecha_ingreso | productos.proveedor |
|-----------------------|------------------|----------------------|-----------------|------------------|-----------------------|--------------------|-------------------------|------------------------|------------------|-------------------------|---------------------|
| 2                     | una              | mia                  | arroz           | arroz carolina   | 1                     | bullo              | 60000.00                | 60000.00               | 12-12-2011       | a                       |                     |
| 3                     | tres             | mia                  | zapato          | zapato fino      | 3                     | Ud                 | 30000.00                | 30000.00               | 12-12-2011       | a                       |                     |
| 1                     | dsa              | asda                 | pan             | pan integral     | 25                    | Ud                 | 100.00                  | 100.00                 | 12-12-2011       | a                       |                     |

## 8 MINERIA DE DATOS

### 8.1 Frame principal

Este módulo permite realizar tarea de minería de datos sobre nuestra base de datos.

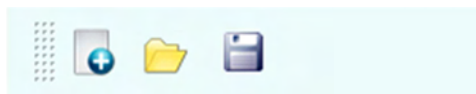
Figura C. 25 Módulo Minería de Datos



## 8.2 Barra de herramientas

En la barra de herramientas encontramos visibles los botones de acción New, Open y Save. Cada uno de ellos nos brindan funcionalidades que serán descritas a continuación:

Figura C. 26 Barra de Herramientas



### Botón New

Crea una nueva ventana de trabajo.

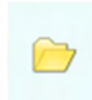
Figura C. 27 Botón New



### **Botón Abrir**

Abre una ventana de trabajo anteriormente guardada.

Figura C. 28 Botón Abrir



### **Botón Guardar**

Despliega un cuadro de búsqueda que muestra la estructura de archivos del sistema permitiéndole guardar su trabajo.

Figura C. 67 Botón Guardar

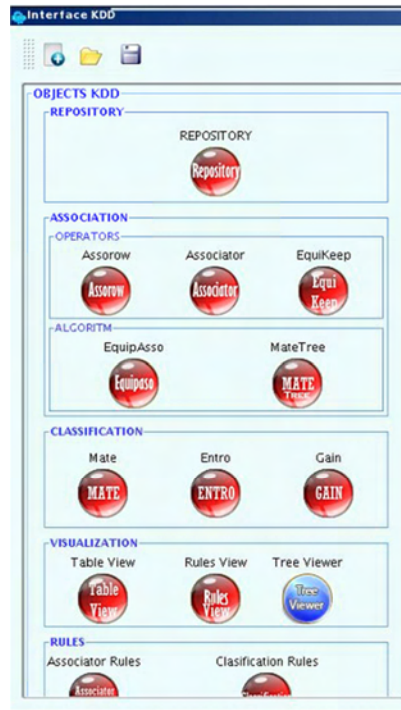


## **8.3 Paneles de visualización**

### **Objects KDD**

Muestra los diferentes objetos que se pueden utilizar para el proceso de minería de datos.

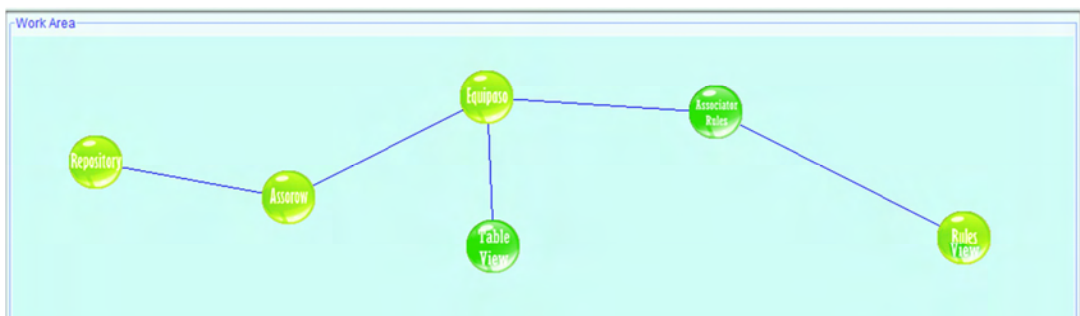
Figura C. 68 Panel Objects KDD



### Work Area

Permite especificar relaciones y acciones entre los objetos KDD en el. Seleccione del panel de objetos KDD, el elemento que necesite para realizar una tarea de minería, y con clic sostenido arrástrela hasta esta área de trabajo.

Figura C. 69 Panel Work Area



## Sql Statement

Este panel muestra en tiempo real la sentencia SQL que se ejecutará para los objetos en el área de trabajo

Figura C. 29 Panel Sql Statement



```
SQL Statement
SELECT cuartos, quintas, sextas, septimas, octaba,
Count(*) AS soporte
INTO fre_directosrow FROM dos
ASSOROW RANGE 1 UNTIL 1
GROUP BY cuartos, quintas, sextas, septimas, octaba
HAVING COUNT (*) >= 2;
```

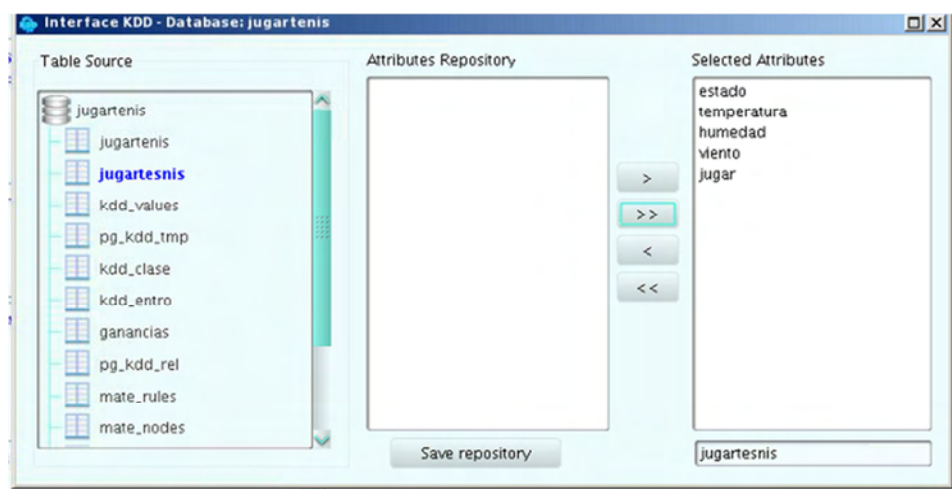
## 8.4 Ventanas Emergentes

Esta sección del manual, se refiere a todas aquellas ventanas de visualización que resultan de la utilización de cada uno de los objetos. A continuación se describirán cada una de ellas.

### Configuration Repository

Se muestra cuando se arrastra un objeto *Repository* al *Work Area*, y se hace clic derecho sobre él y se selecciona del pop menú la opción *Load Data*.

Figura C. 30 Load Data Objeto Repository



En la ventana emergente se muestran tres paneles:

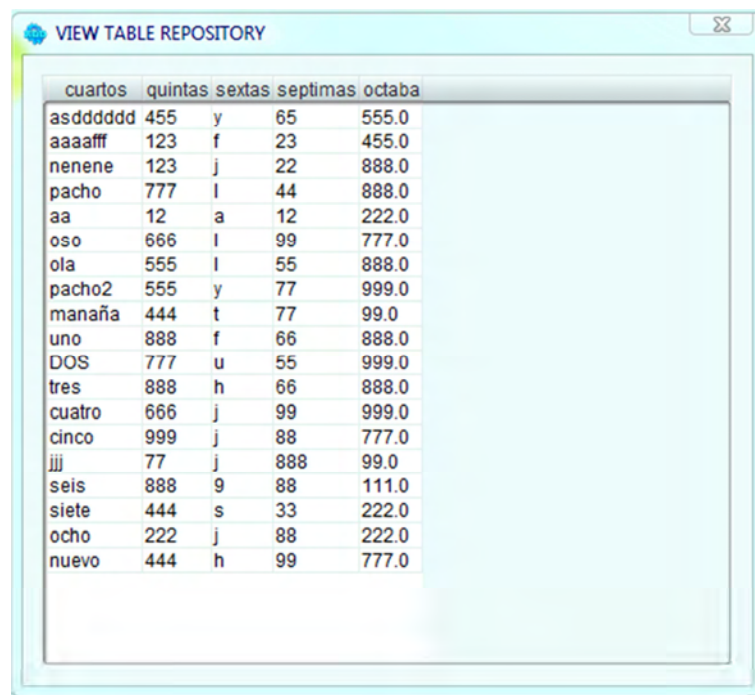
**Table Source:** Muestra las diferentes conexiones y las tablas pertenecientes a estas. Para seleccionar una tabla selecciónela con clic. Los campos se mostrarán en el panel *Attributes Repository*.

**Attributes Repository:** Muestra los campos o columnas de la tabla seleccionada.

**Selected Attributes:** Muestra los atributos seleccionados de la tabla. Para seleccionar un atributo haga clic sobre él y luego sobre las figuras entre los paneles *Attributes Repository* y *Selected Attributes*.

El botón *Save Repository* guarda la selección actual y cierra la ventana emergente. Para observar los datos de la tabla, haga clic derecho sobre el objeto *Repository* y luego sobre *View Data*, se mostrará una ventana con los datos de la selección.

Figura C. 31 View Data objeto Repository

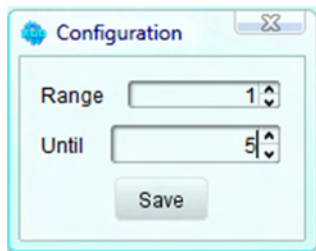


| cuartos  | quintas | sextas | septimas | octaba |
|----------|---------|--------|----------|--------|
| asdddddd | 455     | y      | 65       | 555.0  |
| aaaafff  | 123     | f      | 23       | 455.0  |
| nenene   | 123     | j      | 22       | 888.0  |
| pacho    | 777     | l      | 44       | 888.0  |
| aa       | 12      | a      | 12       | 222.0  |
| oso      | 666     | l      | 99       | 777.0  |
| ola      | 555     | l      | 55       | 888.0  |
| pacho2   | 555     | y      | 77       | 999.0  |
| manaña   | 444     | t      | 77       | 99.0   |
| uno      | 888     | f      | 66       | 888.0  |
| DOS      | 777     | u      | 55       | 999.0  |
| tres     | 888     | h      | 66       | 888.0  |
| cuatro   | 666     | j      | 99       | 999.0  |
| cinco    | 999     | j      | 88       | 777.0  |
| jij      | 77      | j      | 888      | 99.0   |
| seis     | 888     | 9      | 88       | 111.0  |
| siete    | 444     | s      | 33       | 222.0  |
| ocho     | 222     | j      | 88       | 222.0  |
| nuevo    | 444     | h      | 99       | 777.0  |

## Configuration Associator

Se muestra cuando se arrastra un objeto *Associator* al *Work Area* y se hace clic derecho sobre él y se selecciona del pop menú la opción *Configure*.

Figura C. 32 Configuración objeto Associator

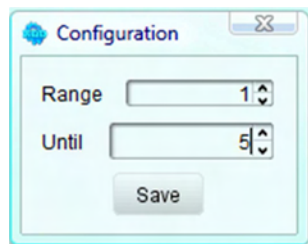


El rango indica el número de atributos que se tendrán en cuenta al aplicar el operador *Associator* sobre la tabla. Por defecto se cargará con el mínimo y máximo número de columnas seleccionadas en el objeto *Repository*.

## Configuration Assorow

Se muestra cuando se arrastra un objeto *Assorow* al *Work Area* y se hace clic derecho sobre él y se selecciona del pop menú la opción *Configure*.

Figura C. 33 Configuración objeto Assorow



El rango indica el número de atributos que se tendrán en cuenta al aplicar el operador *Assorow* sobre la tabla. Por defecto se cargará con el mínimo y máximo número de columnas seleccionadas en el objeto *Repository*.

## Configuration Equikeep

Se muestra cuando se arrastra un objeto *Equikeep* al *Work Area* y se hace clic derecho sobre él y se selecciona del pop menú la opción *Configure*.

Figura C. 34 Configuración EquiKeep



| Item     | Values               |
|----------|----------------------|
| cuartos  | <input type="text"/> |
| quintas  | <input type="text"/> |
| sextas   | <input type="text"/> |
| septimas | <input type="text"/> |
| octaba   | <input type="text"/> |

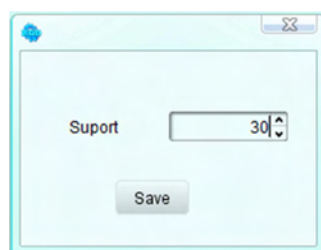
Save

Permite establecer los valores para la ejecución del operador Equikeep.

### Configuration EquipAsso

Se muestra cuando se arrastra un objeto *EquipAsso* al *Work Area* y se hace clic derecho sobre él y se selecciona del pop menú la opción *Configure*.

Figura C. 35 configuración EquipAsso



Suport

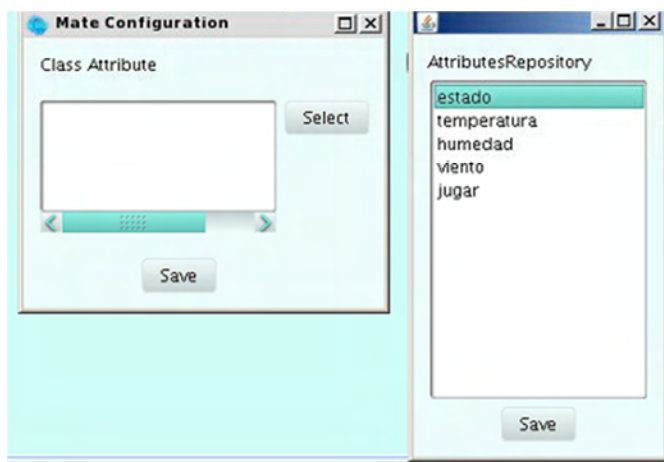
Save

Permite establecer el soporte para la ejecución del algoritmo EquipAsso.

## Configuration Mate

Se muestra cuando se arrastra un objeto *Mate* al *Work Area* y se hace clic derecho sobre él y se selecciona del pop menú la opción *Configure*.

Figura C. 77 Configuración Mate

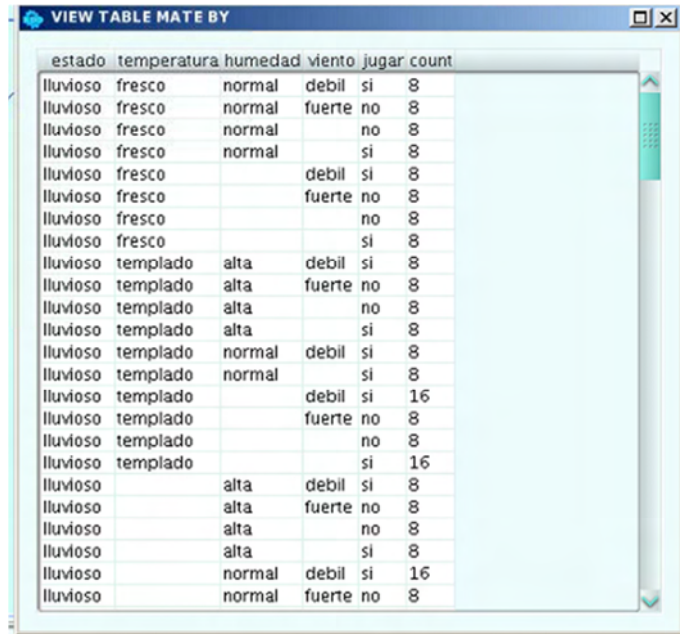


Al hacer clic en el botón *Select* se muestra una ventana con los atributos de la tabla seleccionada. Con clic seleccione el atributo que servirá como atributo Clase para la ejecución del operador *Mate*. Haga clic en *Save* para cerrar la ventana y guardar los cambios.

## Table View

Se muestra cuando se arrastra un objeto *Table View* al *Work Area* y se hace clic derecho sobre él y se selecciona del pop menú la opción *View*.

Figura C. 78 Table View



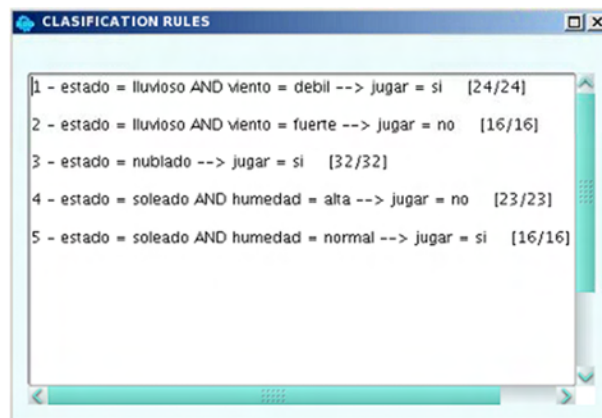
| estado   | temperatura | humedad | viento | jugar | count |
|----------|-------------|---------|--------|-------|-------|
| lluvioso | fresco      | normal  | debil  | si    | 8     |
| lluvioso | fresco      | normal  | fuerte | no    | 8     |
| lluvioso | fresco      | normal  |        | no    | 8     |
| lluvioso | fresco      | normal  |        | si    | 8     |
| lluvioso | fresco      |         | debil  | si    | 8     |
| lluvioso | fresco      |         | fuerte | no    | 8     |
| lluvioso | fresco      |         |        | no    | 8     |
| lluvioso | fresco      |         |        | si    | 8     |
| lluvioso | templado    | alta    | debil  | si    | 8     |
| lluvioso | templado    | alta    | fuerte | no    | 8     |
| lluvioso | templado    | alta    |        | no    | 8     |
| lluvioso | templado    | alta    |        | si    | 8     |
| lluvioso | templado    | normal  | debil  | si    | 8     |
| lluvioso | templado    | normal  |        | si    | 8     |
| lluvioso | templado    |         | debil  | si    | 16    |
| lluvioso | templado    |         | fuerte | no    | 8     |
| lluvioso | templado    |         |        | no    | 8     |
| lluvioso | templado    |         |        | si    | 16    |
| lluvioso |             | alta    | debil  | si    | 8     |
| lluvioso |             | alta    | fuerte | no    | 8     |
| lluvioso |             | alta    |        | no    | 8     |
| lluvioso |             | alta    |        | si    | 8     |
| lluvioso |             | normal  | debil  | si    | 16    |
| lluvioso |             | normal  | fuerte | no    | 8     |

Permite visualizar los datos resultantes de la ejecución de operadores y de otros objetos KDD a los que se puede conectar.

### Rules View

Se muestra cuando se arrastra un objeto *Rules View* al *Work Area* y se hace clic derecho sobre él y se selecciona del pop menú la opción *View*.

Figura C. 79 Rules View



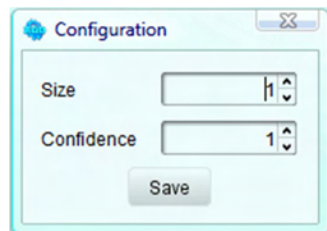
|  |         |
|--|---------|
| 1 - estado = lluvioso AND viento = debil --> jugar = si  | [24/24] |
| 2 - estado = lluvioso AND viento = fuerte --> jugar = no | [16/16] |
| 3 - estado = nublado --> jugar = si                      | [32/32] |
| 4 - estado = soleado AND humedad = alta --> jugar = no   | [23/23] |
| 5 - estado = soleado AND humedad = normal --> jugar = si | [16/16] |

Este objeto muestra las reglas resultantes de la tarea de clasificación. Debe estar conectado a un objeto *Classification Rules*.

### Configuration Associator Rules

Se muestra cuando se arrastra un objeto *Associator Rules* al *Work Area* y se hace clic derecho sobre él y se selecciona del pop menú la opción *Configure*.

Figura C. 36 Configuración Associator Rules



Permite configurar las opciones para la generación de reglas de asociación. Se establece el *Size* o tamaño de las reglas y la confianza o *confidence* de las mismas.