
**UNIVERSIDAD CATÓLICA DEL NORTE**

FACULTAD DE INGENIERÍA Y CIENCIAS GEOLÓGICAS

MAGÍSTER EN INGENIERÍA INFORMÁTICA

# EVALUACIÓN EMPÍRICA DEL ACOPLAMIENTO DE ALGORITMOS DE MINERÍA DE DATOS A UN SISTEMA GESTOR DE BASE DE DATOS

Propuesta de Tesis

**MARIO MISAEL CASTILLO MOREIRA**

Profesor guía: Ph.D. Claudio Meneses Villegas

**Antofagasta, Chile**

**Enero de 2014**

# ÍNDICE

**RESUMEN**

Las investigaciones en Descubrimiento de Conocimiento en Bases de Datos (DCBD), se centraron inicialmente en definir modelos de descubrimiento de patrones y desarrollar algoritmos para éstos. Investigaciones posteriores se han focalizado en el problema de integrar DCBD con Sistemas Gestores de Bases de Datos (SGBD), produciendo como resultado el desarrollo de sistemas y herramientas de Descubrimiento de Conocimiento cuyas arquitecturas se pueden clasificar en tres categorías:

- Arquitectura débilmente acoplada. Los algoritmos de Minería de Datos se encuentran en un entorno distinto al del SGBD, por este motivo el SGBD no tiene ninguna participación en el proceso de DCBD.
- Arquitectura medianamente acoplada. Parte de los algoritmos de Minería de Datos se encuentran en el SGBD, gracias a esto la carga de trabajo la comparten el Motor de Base de Datos y la aplicación que contiene la otra parte del algoritmo.
- Arquitectura fuertemente acoplada. Esta arquitectura tiene dos enfoques:
    - Extender el leguaje SQL. Los investigadores se han centrado más en este enfoque, robusteciendo al lenguaje SQL para que soporte la tarea de Minar Datos.
    - Migrar todo el algoritmo de Minería de Datos al SGBD. De esta manera toda la carga de trabajo de la Minería de Datos la tiene el SGBD.

Pese a plantear la forma de como acoplar la terea de Minería de Datos a un SGBD, no existen investigaciones que comparen cada tipo de arquitectura.

La presente propuesta de tesis busca evaluar los tres enfoques, y en base a métricas, tales como el rendimiento (tiempo, espacio) y facilidad de uso, indicar cuál es la arquitectura más apropiada (beneficiosa).

**Palabras clave –** DCBD; SGBD; Minería de Datos; SQL.

## 1. INTRODUCCIÓN

Los Sistemas Gestores de Base de Datos (SGBD) con el pasar del tiempo se han convertido en uno de los pilares tecnológicos de las empresas, esto debido a su excelente desempeño en el manejo de los datos (transacciones), pero se ven limitados al no lograr convertir estos datos en conocimiento, conocimiento que es valioso para la empresa y puede ayudar de gran manera a la toma de decisiones. Después de manipular estos datos, los SGBD son capaces de convertir estos datos en información, esto mediante la creación de reportes utilizando el Lenguaje de Consulta Estructurado (Structured Query Language, SQL).

Los investigadores han propuesto extender el SQL y así darle al SGBD la habilidad de generar conocimiento. Otros investigadores han logrado incrustar al SGBD esta habilidad mediante procedimientos almacenados y funciones definidas por el usuario, con lo cual no se afecta a la naturaleza del SQL.

## 2. MARCO TEÓRICO

### 2.1 Sistemas Gestores de Bases de Datos

Antes de definir lo que son los Sistemas Gestores de Bases de Datos (SGBD), consideremos dos conceptos básicos:

1. Una Base de Datos es el lugar donde se encuentran almacenados nuestros datos.
2. Un SGBD es quien nos permite acceder (Insertar, Actualizar, Eliminar y Desplegar) a nuestros datos.

#### 2.1.1 Definición de un SGBD

Un SGBD es una colección de programas que permite a los usuarios crear y mantener una base de datos. El SGBD es un sistema de software de propósito general que facilita los procesos de definición, construcción, manipulación y compartición de bases de datos entre varios usuarios y aplicaciones[1].

La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben proporcionar la fiabilidad de la información almacenada, a pesar de las caídas del sistema o los intentos de acceso sin autorización. Si los datos van a ser compartidos entre diversos usuarios, el sistema debe evitar posibles resultados anómalos[2].

Denominaremos Sistema de Bases de Datos a la combinación de Base de Datos y software SGBD[1]. La Figura 2.1 muestra algunos de los conceptos que se han explicado, cómo los usuarios realizan consultas a la base de datos y el SABD se encarga de procesar dichas consultas, accediendo a los datos almacenados.

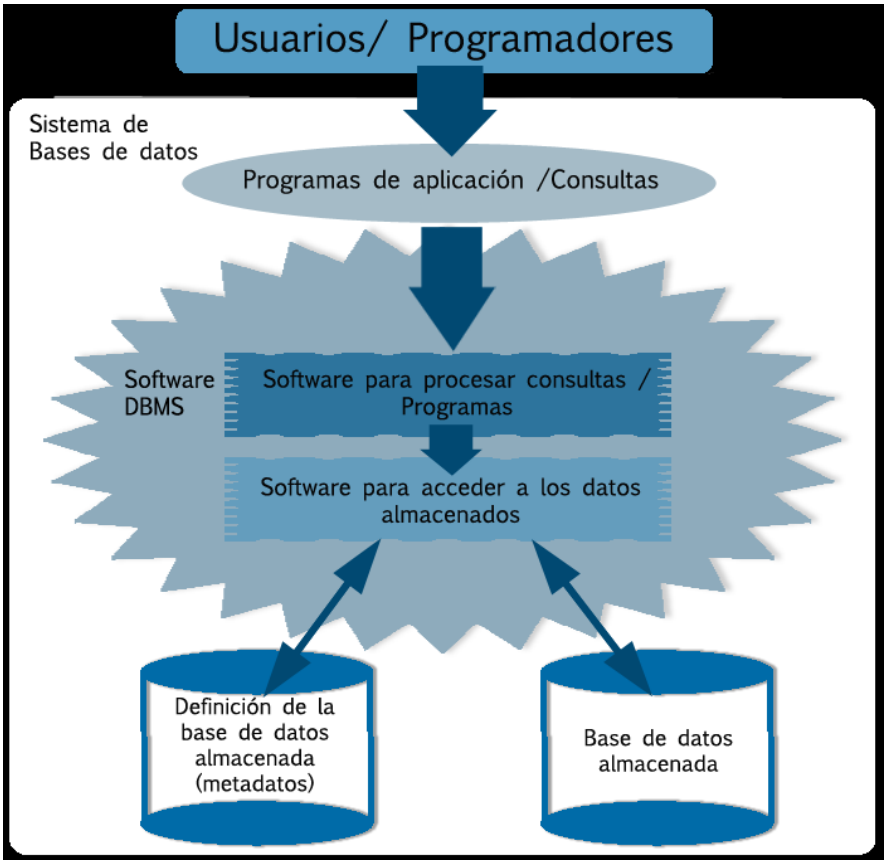



Figura 2.1 – Entorno de un Sistema de Bases de Datos simplificado[1].

### 2.1.2 Bases de Datos Objeto – Relacional

El modelo de Bases de Datos Objeto – Relacional fue creado como respuesta a las capacidades conflictivas de los modelos Relacionales y Orientado a Objetos. Esencialmente las capacidades que posee el modelo Orientado a Objetos ya se encuentran incluidos en las Bases de Datos Relacionales, sin embargo las capacidades de una Base de Datos Relacional no están incorporadas en las Bases de Datos Orientadas al Objeto. El modelo Objeto – Relacional intenta incluir los aspectos más eficientes del modelo Orientado a Objetos en la estructura del modelo Relacional, con variados grados de éxito[3].

Algunos SGBD como Oracle, añadieron algunas capacidades Orientadas a Objetos a sus productos, resultando en bases de datos híbrida Objeto – Relacionales. El modelo Objeto – Relacional extiende al modelo Relacional agregando algunos tipos de datos complejos y métodos. En vez de atomicidad y valores simples en los atributos que requiere el modelo Relacional, este modelo permite atributos estructurados y tienen conjuntos de arreglos de valores. También permite métodos y herencia. El lenguaje SQL fue extendido en 1999 para crear y manipular los tipos de datos más complejos que soporta este modelo, como la especificación de tipos de datos abstractos (denominados TDA o tipos definidos por el usuario)[1].

### 2.1.3 Lenguaje de Consulta Estructurado

El nacimiento del Lenguaje de Consulta Estructurado (SQL) data de 1970 cuando E. F. Codd publica su artículo "Un modelo de datos relacional para grandes bancos de datos compartidos"[4]. Ese artículo dictaría las directrices de las Bases de Datos Relacionales. Apenas dos años después IBM (para quien trabajaba Codd) utiliza las directrices de Codd para crear el Standard English Query Language (Lenguaje Estándar Inglés para Consultas) al que se le llamó SEQUEL. Más adelante se le asignaron las siglas SQL (Standard Query

Language, lenguaje estándar de consulta) aunque en inglés se siguen pronunciando SEQUEL. En español se le llama esecuele.

Poco después se convertía en un estándar en el mundo de las bases de datos avalado por los organismos ISO y ANSI (el primer estándar es del año 1982). Aún hoy sigue siendo uno de los estándares más importantes de la industria informática.

Los estándares más seguidos son los de los años 1992 y 1999 (el último estándar). Sobre estos dos estándares giran estos apuntes.

#### 2.1.3.1 Objetivos del Lenguaje de Consulta Estructurado

SQL pretende ser un lenguaje que simula su escritura en lenguaje normal, de ahí que se le considera un lenguaje de cuarta generación. Consta de palabras especiales y de expresiones.

Se trata de un lenguaje que intenta agrupar todas las funciones que se le pueden pedir a una base de datos.

#### 2.1.3.2 Modos de Ejecución del SQL

Podemos mencionar cuatro posibles Modos de Ejecución del SQL[5]:

A. Ejecución Directa, SQL Interactivo. Las instrucciones SQL se introducen a través de un cliente que está directamente conectado al servidor SQL; por lo que las instrucciones se traducen sin intermediarios y los resultados se muestran en el cliente.

   Normalmente es un modo de trabajo incómodo, pero permite tener acceso a todas las capacidades del lenguaje SQL de la base de datos a la que estamos conectados.

B. Ejecución Incrustada o Embebida. Las instrucciones SQL se colocan como parte del código de otro lenguaje que se considera anfitrión (C, Java, Pascal, Visual Basic, etc.). Al compilar el código se utiliza un precompilador de la propia base de datos para traducir el SQL y conectar

la aplicación resultado con la base de datos a través de un software adaptador (driver) como JDBC u ODBC por ejemplo.

C. Ejecución a través de Clientes Gráficos. Se trata de software que permite conectar a la base de datos a través de un cliente. El software permite manejar de forma gráfica la base de datos y las acciones realizadas son traducidas a SQL y enviadas al servidor. Los resultados recibidos vuelven a ser traducidos de forma gráfica para un manejo más cómodo.
D. Ejecución Dinámica. Se trata de SQL incrustado en módulos especiales que pueden ser invocados una y otra vez desde distintas aplicaciones.

#### 2.1.3.3 Código SQL

Podemos dividirlos en cinco grandes grupos[5]:

A. Comandos. Las distintas instrucciones que se pueden realizar desde SQL son:
    a. SELECT. Se trata del comando que permite realizar consultas sobre los datos de la base de datos. Obtiene datos de la base de datos.
    b. DML. Modifica filas (registros) de la base de datos. Lo forman las instrucciones INSERT, UPDATE, MERGE y DELETE.
    c. DDL. Permiten modificar la estructura de las tablas de la base de datos. Lo forman las instrucciones CREATE, ALTER, DROP, RENAME y TRUNCATE.
    d. Instrucciones de transferencia. Administran las modificaciones creadas por las instrucciones DML. Lo forman las instrucciones ROLLBACK, COMMIT y SAVEPOINT.
    e. DCL. Administran los derechos y restricciones de los usuarios. Lo forman las instrucciones GRANT y REVOKE.
B. Cláusulas. Son palabras especiales que permiten modificar el funcionamiento de un comando (WHERE, ORDER BY, etc.).

C. Operadores. Permiten crear expresiones complejas. Pueden ser aritméticos o lógicos.
D. Funciones. Para conseguir valores complejos (SUM (), DATE (), etc.).
E. Constantes. Valores literales para las consultas, números, textos, caracteres.

## 2.2 Descubrimiento de Conocimiento en Bases de Datos

Knowledge Discovery in Databases (KDD), en español Descubrimiento de Conocimiento en Bases de Datos (DCBD), es un proceso, el cual tiene en una de sus etapas a la Minería de Datos. Se debe tomar nota ya que en la literatura algunos autores consideran a la Minería de Datos como todo el proceso de DCBD.

### 2.2.1 Definición

El DCBD puede definirse como un proceso no trivial que busca identificar patrones válidos, novedosos, potencialmente útiles y en última instancia comprensibles a partir de los datos [6]. El proceso de DCBD está constituido por una serie de etapas tal y como se muestra en la figura 2.2.

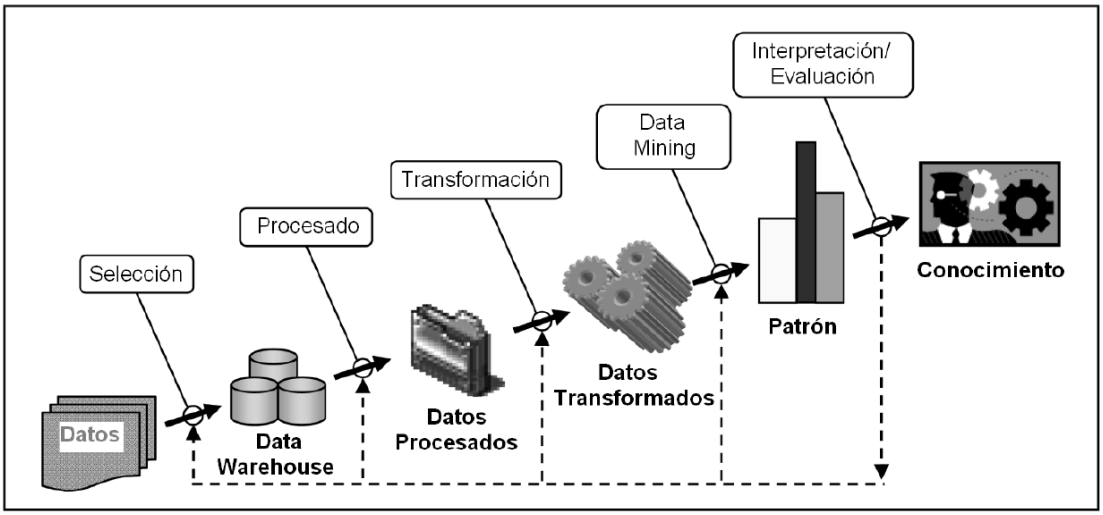



Figura 2.2 – Etapas del proceso de DCBD (adaptado de [6])

Como paso previo al proceso de DCBD es necesario entender al dominio de la aplicación y los objetivos del usuario final. Todo ello requiere cierta interacción entre el usuario y el ingeniero de Minería de Datos para que este último conozca las partes que son susceptibles de un procesado automático y las que no, los objetivos, los criterios de rendimiento exigibles, para qué se usaran los resultados que se obtengan, etc.

### 2.2.2 Fases del proceso de DCBD

A continuación se explica cada una de las etapas que constituyen el proceso de DCBD:

A. Selección. En esta etapa se elige el conjunto de datos objetivo sobre los que se realizará el análisis. Una consideración importante en esta etapa es que los datos pueden proceder de diferentes fuentes, por lo que necesitan ser unificadas.
B. Preproceso. El objetivo de esta etapa es asegurar la calidad de los datos a analizar ya que de ello depende, en gran medida, la calidad del conocimiento descubierto. En esta fase se incluyen tareas como el filtrado de individuos atípicos (datos que no se ajustan al comportamiento general de los datos), la eliminación de ruido, el manejo de valores ausentes o la normalización de los datos.
C. Transformación y reducción de los datos. Esta etapa consiste en modificar la estructura de los datos con el objetivo de facilitar el análisis de los mismos. Eso incluye, entre otras consideraciones, la transformación del esquema original de los datos a otros esquemas (tabla única, esquema desnormalizado, etc.), la proyección de los datos sobre espacios de búsqueda de menor dimensionalidad en los que es más sencillo trabajar o la discretización de datos continuos. Este paso resulta fundamental dentro del proceso global, ya que requiere un buen conocimiento del problema y una buena intuición que, con frecuencia,

marcan la diferencia entre el éxito o fracaso en el descubrimiento de conocimiento.

D. Minería de Datos. Esta es la etapa en la que se analizan los datos mediante un conjunto de técnicas y herramientas, y se extrae información oculta en ellos. La etapa de Minería de Datos se divide, a su vez, en otros tres pasos:
    a. Determinación del tipo de problema que se necesita resolver.
    b. Elección del algoritmo de minería de datos más adecuado para el problema en cuestión.
    c. Búsqueda de conocimiento con una determinada representación del mismo. El éxito de esta etapa depende, en gran medida, de la correcta realización de los pasos previos.
E. Interpretación/Evaluación. En esta última etapa se evalúa y se interpreta el conocimiento extraído en la fase anterior, atendiendo a tres criterios fundamentales: precisión, claridad e interés.

Tras la etapa de interpretación y evaluación del conocimiento extraído es posible que se necesite retroceder a etapas anteriores y repetir el proceso o parte de él. Además, la interacción entre los ingenieros de Minería de Datos y los expertos en el dominio de aplicación suele ser necesaria.

Una vez se descubre y consolida el conocimiento, éste es incorporado al sistema en cuestión o, simplemente, es documentado y enviado a la parte interesada. Además, los modelos generados han de ser reevaluados, reentrenados y reconstruidos cada cierto tiempo para actualizarse según las nuevas tendencias que puedan aparecer en los datos.

### 2.2.3 Técnicas de Minería de Datos

La Minería de Datos (Data Mining) es una etapa del proceso de DCBD en la que se pueden aplicar diferentes técnicas para resolver una gran variedad de

problemas. A continuación se describen los diferentes tipos de problemas que es posible abordar usando técnicas de Minería de Datos.

#### 2.2.3.1 Clasificación

Las técnicas de Clasificación permiten clasificar a un individuo nuevo dentro de un conjunto predefinido de clases. Existen diferentes enfoques dentro de las técnicas de clasificación. Todos ellos, sin embargo, construyen un modelo de clasificación a partir de un conjunto de individuos de entrenamiento de los que se conocen ciertos atributos, incluyendo la clase a la que pertenecen. Ante un nuevo individuo sin clasificar, el modelo de clasificación determinará su clase haciendo uso del valor conocido de sus atributos. A continuación se presentan las técnicas de clasificación más importantes.

A. Árboles de decisión. Existe un grupo de técnicas de clasificación que se basan en la construcción iterativa de un árbol de decisión (AD), también llamado árbol de clasificación. Para clasificar individuos, los nodos de los árboles de decisión almacenan una condición sobre un determinado atributo del individuo a clasificar, mientras que las ramas son las que determinan a qué nodo del nivel inmediatamente inferior descender dependiendo de que se cumpla o no dicha condición. Este proceso se realiza hasta alcanzar un nodo hoja, que almacena la clase en la que es clasificado el individuo.

   Los principales objetivos que se persiguen al crear un AD son [40]:

   - Clasificar correctamente la mayor cantidad de objetos del conjunto de entrenamiento.
   - Generalizar, durante la construcción del árbol, el conjunto de entrenamiento a fin de que nuevos objetos sean clasificados con el mayor porcentaje de aciertos posible.
   - Si el conjunto de datos es dinámico, la estructura del AD debe poder actualizarse fácilmente.

Un algoritmo de generación de árboles de decisión consta de dos etapas: la primera es la etapa de inducción del árbol y la segunda es la etapa de clasificación. En la primera etapa se construye el árbol de decisión a partir del conjunto de entrenamiento, comúnmente cada nodo interno del árbol se compone de un atributo de prueba y la porción del conjunto de entrenamiento presente en el nodo es dividida de acuerdo a los valores que pueda tomar ese atributo. La construcción del árbol inicia generando su nodo raíz, eligiendo un atributo de prueba y particionado el conjunto de entrenamiento en dos o más subconjuntos, para cada partición se genera un nuevo nodo y así sucesivamente. Cuando en un nodo se tienen objetos de más de una clase se genera un nodo interno, cuando contiene objetos de una clase solamente, se forma una hoja a la cual se le asigna la etiqueta de la clase. En la segunda etapa del algoritmo, cada objeto nuevo es clasificado por el árbol construido, se recorre el árbol desde el nodo raíz hasta una hoja, a partir de la cual se determina la pertenencia del objeto a alguna clase. El camino a seguir en el árbol lo determinan las decisiones tomadas en cada nodo interno, de acuerdo al atributo de prueba presente en él.

Existen diversos criterios de selección de atributos que se pueden utilizar para determinar el o los atributos para caracterizar a los nodos internos del AD. Entre los criterios más usados se pueden encontrar:

- Ganancia de Información [41]. La Ganancia de Información se define a partir de la Entropía. Dada una colección S de objetos, con c clases, la entropía de S se mide como:

  $$Entropía\ (S) = \sum_{i=1}^{c} -pi\ log_2\ pi \qquad (2.1)$$

  Donde $pi$ es la proporción de ejemplos en $S$ que pertenecen a la clase $i$. Usando la entropía, se define la Ganancia de Información, de un atributo $X$ en un conjunto de datos $S$, como:

$$Ganancia(S, X) = Entropía(S) - \sum_{v \in Valores(X)} \frac{|Sv|}{|S|} Entropía(S_v) \qquad (2.2)$$

Donde $Valores(X)$ es el conjunto de todos los posibles valores que puede tomar el atributo $X$ y $S_v$ es el subconjunto de $S$, donde los objetos toman el valor $v$ en el atributo $X$. El atributo a elegir será aquel que proporcione el mayor valor de ganancia.

- Proporción de la Ganancia de Información [3]. Utiliza la medida anterior (2) pero se le aplica una normalización, la cual ajusta la ganancia obtenida por cada atributo. Por lo tanto se expresa como:

  $$Gain(S,X) = Ganancia(S/X) / \left(-\sum_{v \in Valores(X)} \frac{|S_v|}{|S|} log_2 \left(\frac{|S_v|}{|S|}\right)\right) \quad (2.3)$$

  El atributo a elegir será aquel que maximice (3).

- Índice de diversidad de Gini [43]. Esta medida se calcula para cada dupla atributo-valor. Sea $CE_x$ el conjunto de entrenamiento descrito solamente por el atributo que será evaluado, $CE_x$ descrito por $c$ clases, el índice de Gini se define como:

  $$I_{gini}(CE_x) = 1 - \sum_{j=1}^{c} pj(X,v)^2 \quad (2.4)$$

  Donde $pj$ es el número de objetos de la clase $j$ que cumplen con la condición del valor $v$ en el atributo $X$ dividido entre el número de objetos de la clase $j$. La dupla atributo-valor elegida para caracterizar a los nodos del árbol, será aquella que presente menor valor con el Índice de Gini.

En la mayoría de los algoritmos de generación de AD, el procedimiento general de inducción del árbol está constituido por dos fases: la fase de entrenamiento (construcción del AD) y la fase de poda del árbol generado.

La poda busca un balance entre el aumento del error de clasificación y la reducción del tamaño del árbol. Además, el proceso de poda ayuda a reducir el sobreajuste que tenga el árbol, ya que el árbol completo puede clasificar perfectamente el conjunto de entrenamiento, pero puede no ser

tan efectivo como un conjunto de prueba (conjunto de objetos no utilizados en la fase de construcción).

Los procesos de poda se basan en el error de clasificación del AD construido sobre un conjunto de prueba.

El procedimiento general simplifica el árbol descartando uno o más subárboles y remplazándolos por hojas, la clase asignada a la nueva hoja se encuentra examinando los objetos de entrenamiento asociados al nodo, así la clase asignada será la de mayor frecuencia.

Los subárboles a reemplazar son elegidos examinando cada nodo interno del árbol completo, reemplazando el subárbol por una hoja y calculando el error de clasificación que genera el árbol podado sobre un conjunto de prueba. Ejemplos de estas técnicas de poda [44] son:

- Poda de error reducido. Empieza a recorrer el árbol construido de las hojas hacia la raíz y en cada nodo interno se calcula el error de clasificación, posteriormente se reemplaza el nodo por una hoja y se calcula el error de clasificación con el árbol podado, si este error es menor que el calculado con el árbol completo, se reemplaza el nodo por la hoja, de lo contrario, se queda el árbol como estaba. Este proceso usa un conjunto de prueba para calcular los errores de clasificación.
- Poda costo-complejidad. Este proceso se realiza en dos pasos. En el primer paso se genera, a partir del árbol completo, una familia de árboles denotada por $T = \{T_0, \dots, T_k\}$, donde $T_{i+1}$ es obtenido podando el árbol $T_i$. $T_0$ es el árbol completo y $T_k$ es un árbol representado solo por el nodo raíz. Para construir el árbol $T_1$, se calcula el parámetro $\alpha$ para cada nodo interno $t$ de $T_0$, utilizando la ecuación (5). Aquellos nodos internos que tengan el mínimo valor de $\alpha$ serán reemplazados por hojas, la clase asignada a esa nueva hoja será la de mayor frecuencia en los objetos asociados a

ese nodo. Este procedimiento se efectúa hasta que se construya el árbol $T_k$. $\alpha$ se define como:

$$\alpha = \frac{e(t) - \sum_{l \in L_t} e(l)}{n(|L_t| - 1)} \quad (2.5)$$

Donde, $t$ es el nodo interno a reemplazar, $n$ es el número de objetos, del conjunto utilizado para obtener el error de clasificación, que pasan por el nodo $t$ en el recorrido que hacen del árbol, $e(t)$es el número de errores si $t$ es reemplazado por una hoja y $L_t$ es el conjunto de hojas que tiene el subárbol con raíz $t$.

El segundo paso es encontrar el árbol de la familia $T$ con el menor error de clasificación, utilizando un conjunto de prueba o realizando validación cruzada con el conjunto de entrenamiento, este árbol elegido caracterizara al conjunto de entrenamiento.

B. Clasificación Bayesiana. La clasificación Bayesiana es una técnica de aprendizaje probabilística que calcula hipótesis probabilísticas explícitas. En este caso, las hipótesis que se barajan son las de pertenencia del individuo sin clasificar a cada una de las clases. Aquella hipótesis con mayor probabilidad será la que determine la clase del individuo que se desea clasificar.

   Para construir el modelo probabilístico, se procesa cada individuo del conjunto de entrenamiento de manera incremental, de tal forma que a medida que se procesan los individuos puede incrementar o decrementar la probabilidad de las hipótesis. Para calcular la probabilidad de cada una de las hipótesis se utiliza el Teorema de Bayes, según el cual dado un conjunto de datos D, la probabilidad a posteriori de una hipótesis h se calcula según la ecuación 2.6.

$$P(h|D) = \frac{P(D|h) * P(h)}{P(D)} \quad (2.6)$$

   Esta técnica necesita conocer las probabilidades a priori, lo que permite incluir conocimiento previo del dominio.

C. Redes Neuronales Artificiales. Las redes de neuronas artificiales son una técnica que modela computacionalmente el aprendizaje humano llevado a cabo a través de las neuronas del cerebro. Una red neuronal se compone de unidades llamadas neuronas conectadas entre sí. Cada neurona recibe una serie de entradas y proporciona una salida, que será la entrada de la siguiente neurona a la que está conectada. Las conexiones entre las neuronas tienen un peso que se usa para ponderar el valor que cada neurona transmite. Al recibir las entradas, cada neurona calcula la suma ponderada de sus entradas y, en la mayoría de los casos, aplica una función de activación para transformar el valor obtenido en una salida válida. Las redes de neuronas se organizan en capas. Todas las redes neuronales poseen una capa de entrada y una capa de salida y pueden tener las capas intermedias (u ocultas) que se desee, dando lugar a redes de muy diferentes arquitecturas, como muestra la figura 2.3.

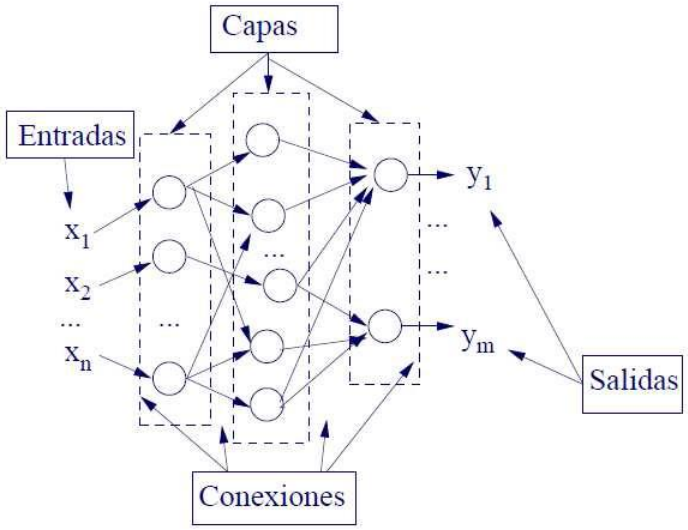



Figura 2.3 – Estructura de las Redes Neuronales Artificiales.

Inicialmente la red se ha de entrenar para calcular el peso de cada conexión. Para ello, se suministra a la red cada individuo del conjunto de

entrenamiento (para los cuales la salida es conocida) y se ajustan los pesos de acuerdo a las diferencias encontradas entre la salida obtenida y la esperada. De esta forma, si las diferencias son grandes se modifica el modelo de forma significativa y, según van siendo menores, se converge a un modelo final estable.

El Perceptrón Simple es una de las redes de neuronas más sencillas que se emplea en Minería de Datos para resolver problemas de clasificación. En este tipo de red, que tiene dos capas, la entrada x (atributos conocidos) se transforma en una salida y (variable respuesta, es decir, clase a la que se asigna el individuo), mediante una función no lineal.

En el Perceptrón, como el resto de redes neuronales, el objetivo de la red durante la fase de entrenamiento es aprender cuáles son los valores más adecuados para los pesos. Para realizar este proceso de aprendizaje existen múltiples algoritmos, pero el más empleado, al menos en Minería de Datos, es el algoritmo Backpropagation. La principal limitación del Perceptrón Simple es que sólo permite resolver problemas de clasificación binarios (dos clases). Este inconveniente se resuelve gracias al Perceptrón Multicapa, que es una red neuronal formada por múltiples capas que no son linealmente separables.

Las redes de neuronas poseen una serie de características que las hacen muy interesantes, entre las cuales destacan las siguientes:

- La exactitud es generalmente alta.
- Son robustas, por lo que trabajan bien incluso cuando los datos contienen errores.
- La salida puede ser discreta, un valor real o un vector de valores reales.
- Realizan una evaluación rápida de la función aprendida.

### 2.2.3.2 Regresión

Las técnicas de Regresión tienen como objetivo predecir el valor (desconocido) de un atributo de un determinado individuo a partir de los valores (conocidos) de otros atributos de dicho individuo. Existen dos tipos de técnicas de regresión dependiendo del tipo de modelo de regresión generado: lineal y no lineal.

La regresión lineal es la forma más simple de regresión, ya que en ella se modelan los datos usando una línea recta [7]. En la regresión lineal se utiliza una variable aleatoria, y (llamada variable respuesta), que es función lineal de otras variables,$a_i (0 \leq i \leq k)$, (llamadas variables predictoras), según viene descrito en la ecuación 2.2 [8]

$$y = w_0 a_0 + w1 \;\; a_1 +, \dots, + w_k \tag{2.7}$$

La tarea fundamental de las técnicas de regresión consiste en obtener el valor de una serie de pesos $w_i (0 \leq i \leq k)$, a partir de los datos del conjunto de entrenamiento. Dichos pesos se obtienen realizando un proceso de minimización por mínimos cuadrados, de tal forma que, considerando que se tienen *n* individuos en el conjunto de entrenamiento, el objetivo es minimizar la expresión descrita en la ecuación 2.8, y así obtener los pesos.

$$\sum (y^{(i)} - \sum w_j {a_j}^{(i)})^2, (1 \leq i \leq n), (0 \leq j \leq k) \tag{2.8}$$

Una vez obtenido el modelo de regresión lineal (descrito en la ecuación 2.5) se puede utilizar para predecir, para un nuevo individuo, el valor del atributo desconocido, *y*, a partir de los valores de los atributos conocidos, $a_i$.

En algunas ocasiones, sin embargo, puede que las variables no presenten una dependencia lineal. En tales casos han de emplearse técnicas de regresión no lineal, en las cuales la relación entre variables puede ser, por ejemplo, polinómica, tal y como indica la ecuación 2.9 [7].

$$y = a + b_1 x + b_2 x^2 + b_3 x^3 \tag{2.9}$$

En este caso es necesario realizar un paso previo de conversión a la forma lineal, según lo establecido en la ecuación 2.10.

$$x_1 = x, x_2 = x^2, x_3 = x^3 \tag{2.10}$$

De esta forma se obtiene la ecuación 2.11.

$$y = a + b_1 x_1 + b_2 x_2 + b_3 x_3 \quad (2.11)$$

Sobre la ecuación 2.11 ya se pueden aplicar las técnicas de regresión lineal para predecir el valor del atributo desconocido *y*.

#### 2.2.3.3 Clustering

Las técnicas de Clustering pretenden dividir los objetos en grupos (denominados clusters), atendiendo a sus características y/o comportamientos. A diferencia de las técnicas de clasificación, el clustering es un proceso de aprendizaje no supervisado ya que las clases no están predefinidas sino que deben ser descubiertas dentro de los datos.

Los algoritmos de clustering poseen muchas y muy diversas aplicaciones en diferentes dominios. En algunas ocasiones, el proceso de clustering es un paso previo a la aplicación de otras técnicas de Minería de Datos. En concreto, algunas de las aplicaciones de las técnicas de clustering son las que se detallan a continuación:

- Marketing: descubrimiento de distintos grupos de clientes en la base de datos. Este conocimiento puede ser útil para la política publicitaria y las ofertas.
- Seguros: identificación de grupos de asegurados con características parecidas (número y tipo de siniestros, posesiones, etc.). Gracias a ello, es posible calcular los riesgos más fielmente.
- Planificación urbana: identificación de grupos de viviendas de acuerdo a su tipo, valor o situación geográfica.
- World Wide Web (WWW): clasificación de documentos, analizar ficheros log para descubrir patrones de acceso similares, etc.
- Diversos usos en ciencias: agrupación de genes con funciones asociadas, identificar filogenias de genes u organismos, etc.

Formalmente, el problema de clustering se define de la siguiente manera:

Dada una base de datos $D = \{t_1, t_2, \dots, t_n\}$ de registros y un valor *k*, se debe establecer una correspondencia $f: D \rightarrow \{C_1, C_2, \dots, C_n\}$ donde cada $t_i$ se asigna a un cluster $C_j, 1 \leq j \leq k$.

Para poder realizar esa partición de los registros de una base de datos, es necesario considerar una determinada medida de distancia (o similaridad) entre dos registros. Existe un gran número de medidas de distancia, aunque las más empleadas son la City-Block, la Euclídea y la Minkowski. Sean $t_i$ y $t_j$ dos registros con *p* atributos cada uno y sea $W_m$ el peso asignado al atributo *m*. La distancia $d_{ij}$ entre registros se calcula como sigue:

- City-Block: $d_{ij} = \sum_{m=1}^{p} W_m |x_{im} - x_{jm}|$ (2.12)
- Euclídea: $d_{ij} = \sqrt{\sum_{m=1}^{p} W_m (x_{im} - x_{jm})^2}$ (2.13)
- Minkowski: $d_{ij} = \sqrt{\sum_{m=1}^{p} (x_{im} - x_{jm})^{\alpha}}\ \alpha > 0$ (2.14)

El objetivo principal de toda técnica de clustering es realizar una partición de los datos de forma que los elementos que pertenecen a un mismo cluster sean muy similares entre sí y los elementos de clusters diferentes sean lo más diferentes posible. Para lograrlo es importante elegir la medida de distancia adecuada.

Adicionalmente, para que un método de clustering sea aplicado en un proyecto real de Minería de Datos, son deseables algunas propiedades de entre las que se destacan las siguientes:

- Que sea escalable, es decir, que funcione correctamente tanto al tratar con conjuntos de datos pequeños como grandes.
- Que posea la capacidad para operar con distintos tipos de variables.
- Que permita descubrir clusters con formas arbitrarias.
- Que sea capaz de tratar datos con ruido y objetos atípicos.
- Que sea insensible al orden de los registros de la base de datos.
- Que funcione correctamente cuando trabaja con registros de alta dimensionalidad (con muchos atributos).

- Que los resultados obtenidos sean interpretables.

El problema de clustering es probablemente el más estudiado dentro de Minería de Datos. Habitualmente las técnicas de clustering se agrupan según la filosofía de las mismas.

#### 2.2.3.4 Reglas de Asociación

Las reglas de asociación constituyen un mecanismo de representación del conocimiento simple y útil para caracterizar las regularidades que se pueden encontrar en grandes bases de datos.

La extracción de reglas de asociación se ha aplicado tradicionalmente a bases de datos transaccionales. En este tipo de bases de datos, una transacción $T$ es un conjunto de artículos o ítems, junto a un identificador único y algún que otro dato adicional (fecha y cliente, por ejemplo). Una transacción contiene un conjunto de ítems $I$ si $I$ está incluido en $T$. Un conjunto de items se denomina itemset, en general, o itemset de grado $k$ $k(k-itemset)$ cuando se especifica el número de ítems que incluye $(k)$.

Una regla de asociación es una implicación $X \Rightarrow Y$ en la cual $X$ e $Y$ son itemsets de intersección vacía (esto es, sin ítems en común). El significado intuitivo de una regla de asociación $X \Rightarrow Y$ es que las transacciones (o tuplas) que contienen a $X$ también tiendan a contener a $Y$.

La confianza (confidence) de una regla de asociación $X \Rightarrow Y$ es la proporción de las transacciones que, conteniendo a $X$, también incluyen a $Y$. El soporte (support) de la regla es la fracción de transacciones en la base de datos que contienen tanto a $X$ como a $Y$.

En la base de datos típica de un supermercado, por ejemplo, los items pueden ser cada uno de los productos o categorías de productos que el establecimiento en cuestión comercializa. Si analizamos la base de datos de transacciones del supermercado, podríamos encontrar que los jueves por la tarde se verifica la siguiente regla: $\{bebidas\} \Rightarrow \{panales\}$ con confianza 66% y soporte 2 %. Dicha

regla nos indica que el 2% de las cestas de la compra incluyen bebidas y panales los jueves por la tarde. Además, dos de cada tres clientes que se llevan bebidas también compran panales (quizá porque el fin de semana tienen que pasarlo en sus casas cuidando a sus bebes).

La extracción de reglas de asociación también puede realizarse en bases de datos relacionales. En ellas, un ítem es un par (atributo, valor). Además, se puede añadir una restricción adicional como consecuencia directa de la Primera Forma Normal, la cual establece que todo atributo de una relación debe contener únicamente valores atómicos. Por tanto, todos los ítems de un itemset deben corresponder a atributos diferentes.

Dada una base de datos concreta, el proceso habitualmente seguido consiste en extraer todas las reglas de asociación que tengan un soporte y una confianza por encima de unos umbrales establecidos por el usuario, MinSupport y MinConfidence respectivamente. Este problema de extracción de reglas de asociación se suele descomponer en dos subproblemas utilizando la estrategia ”Divide y Vencerás”:

- Encontrar todos los itemsets frecuentes [denominados frequent, covering o large itemsets en la literatura], que son aquellos itemsets cuyo soporte es mayor que un umbral de soporte mínimo establecido por el usuario. Este problema puede resolverse construyendo un conjunto candidato de itemsets potencialmente frecuentes e identificando, en dicho conjunto de candidatos, aquellos itemsets que realmente lo son. El tamaño de los itemsets considerados se va incrementando progresivamente hasta que no quedan itemsets frecuentes por descubrir.
- Generar las reglas de asociación que se derivan en los itemsets frecuentes. Si $X \cup Y$ y $X$ son itemsets frecuentes, la regla $X \Rightarrow Y$ se verifica si el cociente entre el soporte de $X \cup Y$ y el soporte de $X$ es, al menos, tan grande como el umbral de confianza mínima. La regla superara el umbral de soporte porque $X \cup Y$ es frecuente.

El descubrimiento de los itemsets frecuentes es la parte del proceso de extracción de reglas de asociación más costosa computacionalmente, mientras que la generación de las reglas de asociación a partir de los itemsets frecuentes es casi inmediata. Por este motivo la mayor parte de los algoritmos de extracción de reglas de asociación han centrado su diseño en la enumeración eficientemente de todos los itemsets frecuentes presentes en una base de datos. Como ejemplo tenemos al algoritmo A priori [45], el cual consiste en que cada subconjunto de un itemset frecuente debe ser también un itemset frecuente. Podemos crear itemsets frecuentes iterativamente, tomando los itemsets frecuentes de tamaño n y extendiéndolos a itemsets frecuentes de tamaño n+1. El algoritmo A priori contempla los siguientes pasos:

1. Se calcula el soporte de cada ítem de forma individual, de modo que se determinan los 1-itesmsets frecuentes.
2. En cada paso que sigue, se utilizan los itemsets frecuentes generados en los pasos anteriores para generar nuevos itemsets (itemsets candidatos).
3. Se calcula el soporte de cada itemsets candidatos y se determinan los itemsets frecuentes.
4. El proceso sigue hasta que no exista nuevos itemsets frecuentes.

## 2.3 Integración del proceso de Descubrir Conocimiento a los Sistemas Gestores de Bases de Datos

A continuación se mostraran los tres tipos de formas de integrar este proceso a un Motor de Base de Datos.

### 2.3.1 Arquitectura Débilmente Acoplada

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 éste se hace a partir de una interfaz cuya

función, en la mayoría de los casos, se limita a los comandos "leer de " y "escribir en" [9].

En una arquitectura débilmente acoplada, los procesos de Minería de Datos corren en un espacio de direccionamiento diferente al del SGBD [10], [11]. Mientras el SGBD provee el almacenamiento persistente, la mayor parte del procesamiento de datos se realiza en herramientas y aplicaciones por fuera del motor del SGBD, como se puede ver en la Figura 2.4.

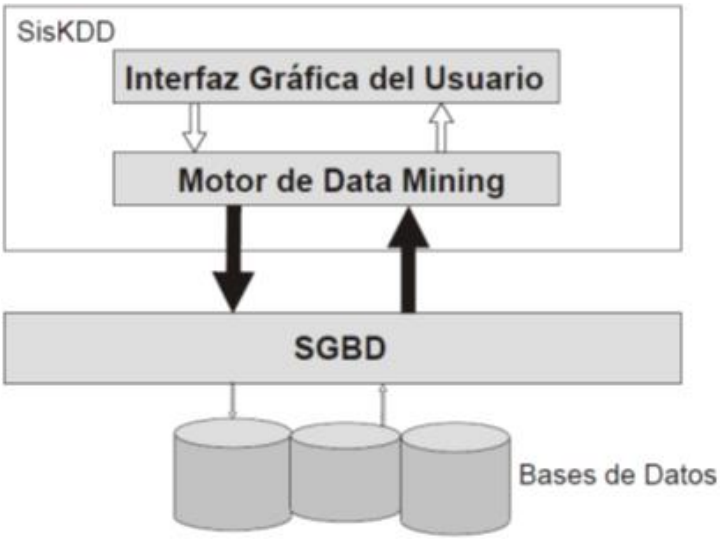



Figura 2.4 Arquitectura Débilmente Acoplada [12].

### 2.3.2 Arquitectura Medianamente Acoplada

Una arquitectura es medianamente acoplada cuando ciertas tareas y algoritmos de Minería de Datos se encuentran formando parte del SGBD mediante procedimientos almacenados o funciones definidas por el usuario [49].

En una arquitectura medianamente acoplada las principales tareas de Minería de Datos se descomponen en subtareas, algunas de las cuales se mueven al SGBD como consultas SQL en procedimientos almacenados o funciones definidas por el usuario, como se ve en la Figura 2.5.

La principal ventaja de esta arquitectura es que tiene en cuenta las capacidades de escalabilidad, administración y manipulación de datos del SGBD. Su mayor reto es obtener un buen rendimiento ya que el optimizador de consultas del SGBD no cuenta con nuevas estrategias de búsqueda y transformación que le permitan generar eficientes planes de ejecución, para las consultas que involucren Minería de Datos. Este problema se puede solucionar, en algún grado, implementando algoritmos ingeniosos, pero que incrementan la complejidad y el costo de desarrollo de estos sistemas [14].

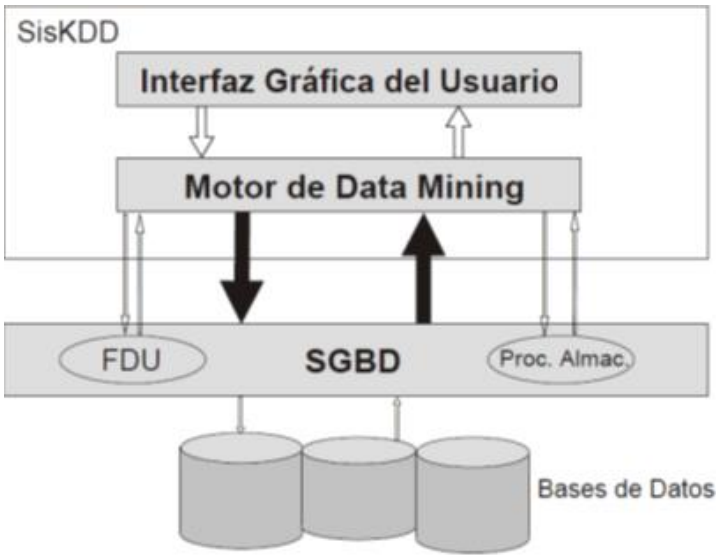



Figura 2.5 Arquitectura Medianamente Acoplada [12].

En la arquitectura medianamente acoplada a través de procedimientos almacenados [10], [11], [14], los algoritmos de descubrimiento son encapsulados como procedimientos almacenados que corren en el mismo espacio de direccionamiento del SGBD. Un procedimiento almacenado es un conjunto nombrado de instrucciones y lógica de procedimientos de SQL compilado, verificado y almacenado en la base de datos. El motor de Minería de Datos invoca al procedimiento almacenado y le transmite los parámetros requeridos para hacer una tarea. Un solo mensaje desencadena la ejecución de

un conjunto de instrucciones SQL almacenadas. La ventaja de este método es que brinda mejor desempeño, ya que ejecutan todas las conexiones, aplicaciones y la base de datos en el mismo espacio de direccionamiento, además, un procedimiento almacenado recibe igual trato que cualquier otro objeto de la base de datos y es registrado en su catálogo.

En la arquitectura medianamente acoplada por medio de funciones definidas por el usuario (FDUs) [10], [11], [14], las funciones son definidas e implementadas en un lenguaje de programación de propósito general. El ejecutable de una FDU se almacena en el SGBD y éste puede acceder e invocar la función en el momento que ésta sea referenciada en una instrucción SQL. Los algoritmos de Minería de Datos se expresan como una colección de funciones definidas por el usuario. La mayoría del procesamiento se realiza en la FDU y el SGBD se usa principalmente para proveer tuplas hacia las FDUs. Las FDUs, al igual que los procedimientos almacenados, corren en el mismo espacio de direccionamiento que el SGBD. La principal ventaja de las FDUs sobre los procedimientos almacenados es su tiempo de ejecución ya que pasar tuplas a un procedimiento almacenado es más lento que hacia una FDU, debido a que en un procedimiento almacenado se deben ejecutar una serie de instrucciones SQL y en una FDU no.

El resto del procesamiento es el mismo. La principal desventaja es el costo de desarrollo ya que los algoritmos de minería de datos tienen que ser escritos como FDUs.

### 2.3.3 Arquitectura Fuertemente Acoplada

Una arquitectura es fuertemente acoplada cuando la totalidad del algoritmo de descubrimiento de patrones forman parte del SGBD como una operación primitiva [12], [13], dotándolo de las capacidades de descubrimiento de conocimiento y posibilitándolo para desarrollar aplicaciones de este tipo, como se observa en la Figura 2.6.

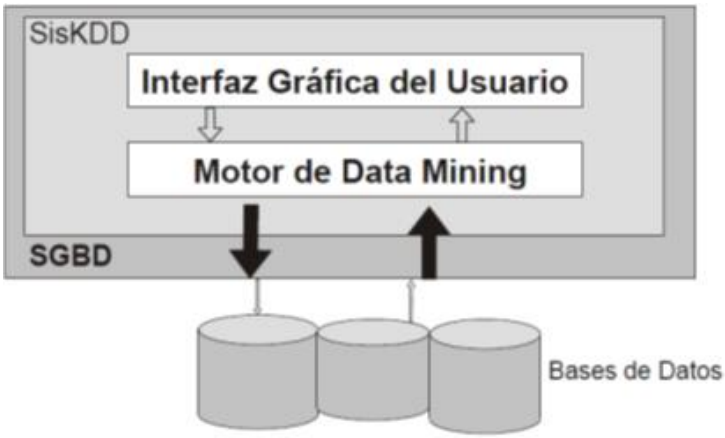



Figura 2.6 Arquitectura Fuertemente Acoplada [12].

En una arquitectura fuertemente acoplada, un algoritmo completo de Minería de Datos se integra al motor de un SGBD como una primitiva. Debido a que todos los algoritmos son ejecutados conjuntamente con los datos en el SGBD, la ventaja potencial de este enfoque es que resuelve los problemas de escalabilidad y rendimiento de las otras arquitecturas. Tal vez la mayor limitación de este método es que los desarrolladores de aplicaciones y herramientas DCBD no están dispuestos a poner algoritmos completos en los sistemas de bases de datos por razones de competencia. Por otro lado, muchos algoritmos cambian frecuentemente con los resultados de las nuevas investigaciones, lo que dificulta su oportuna actualización [13].

También, una arquitectura está fuertemente acoplada cuando el algoritmo completo de Minería de Datos se integra al motor de un SGBD mediante procedimientos almacenados o funciones definidas por el usuario [46], [47], [48], [50], [51], a diferencia del enfoque medianamente acoplado que solo parte del algoritmo de Minería de Datos se encuentra en el SGBD.

## 3. PLANTEAMIENTO DEL PROBLEMA

El lenguaje SQL a pesar de su gran desempeño en el manejo de información se ve limitado a la hora de convertir esta información en conocimiento [48]. A

causa de esto, se han creado herramientas que pueden realizar esta tarea, pero estas herramientas trabajan en un entorno diferente al del SGBD (débilmente acoplada), lo cual significa que no se aprovecha al máximo las capacidades del Motor de Base de Datos [50]. Existen propuestas de investigadores de cómo se podría acoplar, parcial o totalmente, un algoritmo de Minería de Datos a un SGBD. No obstante no existen estudios que muestren los beneficios de utilizar un tipo de arquitectura u otro, desde un punto de vista de rendimiento y facilidad de uso.

### 3.1 Relevancia del problema

Desde el punto de vista tecnológico, el estudio de este problema es importante por dos razones:

- Conocer cómo los SGBD pueden soportar la tarea de Minar Datos, ya sea extendiendo el lenguaje SQL o utilizando funciones en las sentencias SQL.
- Aprovechar al máximo las capacidades de las arquitecturas de Bases de Datos para obtener mejor desempeño en la Minería de Datos.

Desde el punto de vista económico, las empresas aprovecharían esta funcionalidad de sus Motores de Bases de Datos, así no tendrían que optar por la compra de herramientas de terceros.

Desde el punto de vista científico, el estudio de este problema es importante por tres razones:

- Generar nuevo conocimiento respecto al rendimiento de algoritmos de Minería de Datos que son débil, mediana o fuertemente acoplados a un SGBD.
- Generar nuevo conocimiento respecto a la facilidad de uso de un enfoque de Minería de Datos que interactúa con un SGBD, lo cual tiene impacto en los costos de los proyectos que desarrollan soluciones de Minería de Datos.

- Estimular la investigación del comportamiento de los tres tipos de arquitectura en el análisis de grandes volúmenes de datos (bigdata).

### 3.2 Preguntas de investigación

Una vez planteada la problemática, surgen las siguientes preguntas de investigación:

1. ¿Es posible integrar algoritmos de Minería de Datos a un SGBD?

   Los Sistemas Gestores de Base de Datos tienen como principal tarea la manipulación (insertar, actualizar, eliminar) de datos, esta tarea la realizan mediante el lenguaje de consulta estructurado (SQL). Para visualizar estos datos en un reporte también se utiliza SQL, de esta forma los datos pasan a convertirse en información, no así en conocimiento, conocimiento que puede ser útil para la toma de decisiones.

2. ¿Tiene beneficios integrar algoritmos de Minería de Datos en un SGBD?

   Una de las etapas del proceso de Descubrir Conocimiento en una Base de Datos es la Minería de Datos, la cual es la más costosa (en relación a la escalabilidad y rendimiento) de llevar a cabo, esto debido a los algoritmos de Minería de Datos que se utilizan.

3. Respecto a integrar algoritmos de Minería de Datos a un SGBD ¿Cuál de las formas es conveniente realizarla?

   Que ventajas y desventajas tiene una forma de la otra, analizar el rendimiento, la manera en la que se lleva a cabo el proceso, la forma en que muestra los resultados.

## 4. OBJETIVOS

Definida la problemática existente, se plantean el objetivo general y objetivos específicos.

### 4.1 Objetivo general

Determinar empíricamente las ventajas y desventajas de acoplar algoritmos de Minería de Datos a un SGBD.

### 4.2 Objetivos específicos

1. Analizar el estado del arte del acoplamiento de algoritmos de Minería de Datos en un SGBD.
2. Seleccionar los materiales y métodos.
3. Implementar dos algoritmos de Minería de Datos en las tres arquitecturas (débil, mediana y fuertemente acoplada).
4. Determinar experimentalmente el comportamiento de las tres arquitecturas (débil, mediana y fuertemente acoplada).
5. Analizar los resultados experimentales obtenidos.
6. Difundir los resultados en medios científicos.

## 5. ANÁLISIS DEL ESTADO DEL ARTE

A continuación presentaremos un análisis del estado del arte respecto a las tres arquitecturas existentes:

### 5.1 Arquitectura Débilmente Acoplada

Esta arquitectura está presente en la mayoría de las herramientas de Minería de Datos [10]. Algunas como Alice [15], C5.0_RuleQuest [16], Qyield [17], CoverStory [18] ofrecen soporte únicamente en la etapa de minería de datos y requieren un pre y un pos procesamiento de los datos. Hay una gran cantidad de este tipo de herramientas [19], especialmente para clasificación apoyadas en árboles de decisión, redes neuronales y aprendizaje basado en ejemplos. El usuario de este tipo de herramientas puede integrarlas a otros módulos como parte de una aplicación completa [20].

Otros ofrecen soporte en más de una etapa del proceso de Minería de Datos y una variedad de tareas de descubrimiento, típicamente, combinando clasificación, visualización, consulta y clustering, entre otras [20]. En este grupo están Clementine [21], DBMiner [22], [23], [24], DBLearn [25], Data Mine [9], IMACS [26], Intelligent Miner [27], Quest [28] entre otras. Una evaluación de un gran número de herramientas de este tipo se puede encontrar en [29]. La implementación de este tipo de arquitectura se hace a través de SQL embebido en el lenguaje anfitrión del motor de minería de datos [30], [14], [10], [11]. Los datos residen en el SGBD y son leídos registro por registro a través de ODBC, JDBC o de una interfaz de cursores SQL. La ventaja de esta arquitectura es su portabilidad.

Sus principales desventajas son la escalabilidad y el rendimiento. El problema de escalabilidad consiste en que las herramientas y aplicaciones bajo este tipo de arquitectura, cargan todo el conjunto de datos en memoria, lo que las limita para el manejo de grandes cantidades de datos.

El bajo rendimiento se debe a que los registros son copiados uno por uno del espacio de direccionamiento de la base de datos al espacio de direccionamiento de la aplicación de minería de datos [30], [14] y estas operaciones de entrada/salida, cuando se manejan grandes volúmenes de datos, son bastante costosas, a pesar de la optimización de lectura por bloques presente en muchos SGBD (Oracle, DB2, Informix, etc.) donde un bloque de tuplas puede ser leído al tiempo. Estos sistemas, a pesar de que son fáciles de integrar a cualquier SGBD, son ineficientes en términos computacionales. El problema radica en que los costosos algoritmos de descubrimiento de conocimiento se implementan por fuera del núcleo del SGBD, impidiendo cualquier aplicación de las técnicas de optimización de consultas por parte del optimizador del SGBD.

### 5.2 Arquitectura Medianamente Acoplada

Un análisis de este tipo de arquitectura se reporta en [10]. En este trabajo, el algoritmo Apriori [31], [32] se implementa de varias maneras en SQL, e.g. usando únicamente SQL-92 y usando SQL con extensiones objeto-relacional (OR) tales como BLOBS (binary large objects) y funciones definidas por el usuario. De acuerdo con los autores, las pruebas de rendimiento mostraron que usando únicamente SQL-92 el algoritmo Apriori fue demasiado lento. El rendimiento mejoró a niveles aceptables usando características OR, pero con una implementación demasiado compleja y a un alto costo de desarrollo.

### 5.3 Arquitectura Fuertemente Acoplada

Hay propuestas de investigación que discuten la manera cómo tales sistemas pueden ser implementados y las extensiones del lenguaje SQL necesarias para soportar estas operaciones de descubrimiento de patrones: DMQL [33], MSQL [34], MINE RULE [35], [36] y NonStop SQL/MX [13], [37]. Sin embargo, a la fecha no existe un consenso general sobre el conjunto de primitivas necesarias para soportar el descubrimiento de conocimiento en un SGBD.

Este punto no ha sido lo suficientemente investigado aún. La búsqueda de tales primitivas continuará en el futuro cercano y será un área activa de investigación [37], [13], [9]. Con relación a las propuestas de extensión del lenguaje SQL, Han et al. [33] proponen el DMQL (Data Mining Query Language), un lenguaje de consultas de minería de datos para bases de datos relacionales, el cual adopta una sintaxis SQL-like para facilitar el alto nivel de minería de datos y una natural integración con el lenguaje relacional SQL.

DMQL extiende el lenguaje SQL con una colección de operadores para generalización, reglas caracterizantes, reglas discriminantes, reglas de clasificación y reglas de asociación con varias clases de umbrales (thresholds). DMQL puede servir como un lenguaje de desarrollo para implementaciones de sistemas de minería de datos.

Imielinski et al. [34], proponen el M-SQL, un lenguaje de consulta que extiende el SQL con un conjunto pequeño de primitivas para minería de datos en un operador especial unificado, el operador MINE. MINE genera y recupera todo un conjunto de reglas que cumplen con el soporte y la confianza establecidas por el usuario. M-SQL se puede incrustar en el lenguaje anfitrion C++ (de manera similar como SQL es embebido en C) para proveer APIs (Application Programming Interface) para aplicaciones de descubrimiento de conocimiento.

La principal ventaja de estas dos propuestas, DMQL y M-SQL, es que extienden el lenguaje SQL con nuevos operadores para poder expresar operaciones de Minería de Datos con una sintaxis SQL. La desventaja de estas propuestas es la arquitectura bajo la cual estos lenguajes fueron implementados.

El lenguaje DMQL está implementado en el sistema DBMiner [22], [23], [24], un sistema para Minería de Datos débilmente acoplado con SGBD relacional. La arquitectura de DBMiner consta de una interfaz gráfica de usuario (GUI), un motor de descubrimiento y un módulo de comunicación de datos. El motor de descubrimiento contiene módulos para el análisis de consultas, generalización y descubrimiento de patrones. El módulo de comunicación de datos permite el intercambio de datos entre el motor de descubrimiento y el servidor SQL [22].

De igual forma, M-SQL hace parte del prototipo Data Mine [34], un sistema para Minería de Datos que, de acuerdo a sus autores [34], es débilmente acoplado con un SGBD. La arquitectura de Data Mine consta de un motor de minería de datos, una interfaz gráfica de usuario y una interfaz de comunicación de datos. Data Mine puede leer datos a partir de archivos ASCII o desde una base de datos directamente. El SGBD usado actualmente es Sybase, pero el diseño asegura un acoplamiento débil entre la base de datos y el motor de minería de datos, lo que hace posible el uso de otros sistemas gestores de bases de datos. Data Mine provee además, una interfaz de programación de aplicaciones para Minería de Datos.

Meo et al. [35], proponen un modelo de operador unificado SQL-like (i.e. con una sintáxis parecida al SQL) para encontrar reglas de asociación en datos agrupados por diferentes atributos. El operador MINE RULE se diseña como una extensión del lenguaje SQL para obtener diferentes tipos de reglas de asociación: reglas de asociación simples, con condicionales, con clustering, con generalización, con jerarquías. El operador MINE RULE produce una nueva tabla donde cada tupla corresponde a una regla descubierta.

Los autores proponen una semántica formal para el operador MINE RULE. La semántica se describe por medio de una extensión del álgebra relacional con nuevos operadores: Group by ( G ), Unnest (h ), Extended ( E ), Substitute ( å ), Rename ( r ), Powerset ( R ), que permiten transformar una relación con el fin de descubrir reglas de asociación [35], [36]. Además, proponen una arquitectura, que a juicio de los autores, está fuertemente acoplada con SQL server, para soportar el operador MINE RULE [37].

El sistema se divide en tres módulos fundamentales: La interfaz del usuario; el kernel de Minería de Datos, que ejecuta la extracción de las reglas y el SGBD, que almacena los datos origen, las reglas que se obtienen y los resultados intermedios. El kernel, a su vez, está compuesto por el translator, que se encarga del análisis sintáctico de una orden MINE RULE y genera un conjunto de programas SQL para ser utilizados posteriormente; el preprocesador, que a partir de los datos iniciales, genera un conjunto de tablas codificadas (a cada item candidato a aparecer en las reglas se le asocia un identificador único), que son almacenadas nuevamente en el SGBD; el core operator, compuesto por un conjunto de algoritmos para reglas de asociación, produce, a partir de las tablas codificadas, preparadas por el preprocesador, un conjunto de reglas codificadas, i.e. reglas de asociación cuyos elementos están codificados, que son almacenadas en forma de tablas en el SGBD; el postprocesador, decodifica las reglas y las entrega en una forma entendible para el usuario.

Según los autores, la implementación del operador MINE RULE no puede ser completamente relacional (i.e. que todo se exprese con operadores SQL) aún en el contexto de una arquitectura fuertemente acoplada. La parte del núcleo de Minería de Datos es responsabilidad de un procedimiento no-SQL, llamado core operator el cual recibe los datos recuperados por el SQL server, extrae las reglas y las retorna en forma de una relación SQL3 hacia el SGBD.

Muchas características del operador MINE RULE (como la evaluación de subconsultas, agrupamiento, etc.) son mejor ejecutados en SQL, por fuera del core operator [38]. Por esta razón, en esta arquitectura existe una frontera entre los procesos relacionales y de Minería de Datos, dependiendo si los procesos los puede realizar el SQL server o el core operator.

Las ventajas de esta propuesta son: el poder expresivo del operador MINE RULE para formular cualquier tarea sobre reglas de asociación; la definición de una sintaxis del operador y la extensión del álgebra relacional para soportar formalmente al operador. La desventaja es la arquitectura en la cual se ha implementado el operador MINE RULE, que a pesar que los autores consideran fuertemente acoplada con un SGBD, no lo es en su totalidad, debido a que los algoritmos de Minería de Datos forman parte de un módulo independiente llamado kernel por encima del SGBD [38]. En este módulo se realiza todo el proceso de descubrimiento de reglas. El SGBD se utiliza únicamente para almacenar los datos iniciales, las reglas de salida, los resultados intermedios y para realizar ciertas tareas del operador MINE RULE (como evaluación de subconsultas, agrupamientos, etc.), que según los autores, es más eficiente hacerlo por fuera del core operator.

Además, no se reporta nada acerca de la optimización de consultas realizadas con el operador MINE RULE ni tampoco sobre el rendimiento de esta arquitectura.

Clear et al. [13], [37] proponen e implementan un conjunto de primitivas para soportar el proceso de DCBD al interior del motor de NonStop SQL/MX, un

nuevo SGBD objeto-relacional, paralelo de la Division Tandem de Compaq. Estas primitivas tales como transposition, vertical partitioning, round-robin, horizontal partitioning, sequence functions y sampling, extienden el motor del SGBD y se expresan a través de una interfaz SQL, junto con otras características de alto rendimiento del motor SQL/MX, permiten realizar algunas tareas básicas de descubrimiento de conocimiento de manera eficiente y escalable.

La primitiva transposition, implementada como TRANSPOSE en la cláusula select, resume varias cláusulas group by en una, permitiendo obtener múltiples resultados, tales como contar la frecuencia por cada atributo y el cruce de tablas, en una sola búsqueda sobre una tabla [37].

Vertical partitioning permite particionar verticalmente una tabla por un atributo o grupo de atributos y almacenar el resultado en archivos separados con el fin de proveer un acceso rápido a únicamente los atributos que se necesitan. Round-robin permite expandir los datos equitativamente a través de múltiples particiones horizontales. Horizontal partitioning permite realizar particiones horizontales sin requerir un conocimiento detallado de los datos o de la definición de una llave de particionamiento. Según los autores, estas tres estrategias de particionamiento de tablas son muy útiles en la paralelización de una consulta.

SQL/MX soporta un número de funciones de secuencia (sequence functions) que se pueden usar, para especificar en forma concisa, consultas que involucren secuencias de datos en el tiempo.

La primitiva sampling permite generar rápidamente respuestas por muestreo. Ofrece tres tipos de muestreo: randómico, los primeros n registros y periódico. Por cada tipo, se puede especificar el tamaño del muestreo indicando el porcentaje de los datos para el muestreo o con un número determinado de registros.

Esta propuesta es sin duda la única que se puede considerar fuertemente acoplada al integrar al interior del motor NonStop SQL/MX un conjunto de primitivas que permite soportar, de manera eficiente y escalable, algunas tareas básicas de descubrimiento de conocimiento. El hecho de ser NonStop SQL/MX un motor paralelo hace que esta arquitectura sea muy específica, ya que otros motores no paralelos seguramente no pueden aprovechar estas ventajas.

Uno de los proyectos de investigación que Microsoft actualmente adelanta es el de Minería de Datos [39], cuyo objetivo es proveer a la industria del software de estándares para Minería de Datos. Entre las metas de este proyecto están: desarrollar métodos computacionales eficientes para la extracción de patrones, limpieza y reducción de datos en grandes bases de datos; construir un sistema escalable y eficiente que esté fuertemente integrado con un sistema de bases de datos relacional/OLAP, usando métodos de bases de datos, estadísticas, complejidad de algoritmos y optimización. Éste es un proyecto a largo plazo.

A corto plazo, el objetivo será automatizar el proceso de Minería de Datos sobre Bodegas de Datos e incluye entre otras áreas la integración de Minería de Datos con SGBD. En esta área, en colaboración con el Grupo de SQL Server, se han identificado las interfaces que permitan la integración de Minería de Datos con SQL Server.

El resultado es la definición de la especificación OLE-DB para DM (OLE-DB para Minería de Datos), una extensión de OLE DB (un conjunto de interfaces para acceder a un diverso rango de tipos de datos, localizados en una variedad de medios de almacenamiento de datos) que introduce una interfaz común para Minería de Datos. OLE-DB para DM permitirá que los desarrolladores de bases de datos fácilmente accedan y exitosamente apliquen la tecnología de Minería de Datos en sus aplicaciones. El trabajo futuro será el de hacer posible la integración fuerte entre las consultas de las bases de datos y la Minería de Datos.

## 6. HIPÓTESIS

- El acoplar mediana o totalmente un algoritmo de Minería de Datos a un SGBD presenta ventajas en rendimiento respecto a la arquitectura débilmente acoplada.
- El acoplar mediana o totalmente un algoritmo de Minería de Datos a un SGBD presenta desventajas en facilidad de uso respecto a la arquitectura débilmente acoplada.

Producto de la evaluación empírica y en base a las métricas definidas, tales como el rendimiento (tiempo, espacio) y facilidad de uso, se lograra concluir cuál es la mejor forma de acoplar algoritmos de Minería de Datos a un SGBD, indicando las ventajas y desventajas de cada arquitectura (débil, mediana y fuertemente).

## 7. METODOLOGÍA

En este apartado se presentan las fases que conforman la metodología de trabajo con sus respectivos pasos.

A. Implementación de algoritmos en base a la selección y análisis de los materiales y métodos. En esta fase se busca analizar la información recopilada en la literatura respecto al acoplamiento de algoritmos de Minería de Datos a un SGBD.

   En base al análisis, se seleccionan los materiales y métodos que serán utilizados para el trabajo de investigación.

   Esta fase toma en cuenta:

   - Identificación de las arquitecturas. Producto del análisis de la literatura, se pueden definir tres arquitecturas de acoplamiento, débil, mediana y fuertemente.
   - Identificación de los problemas que se presentan en cada una de las arquitecturas. Otro producto del análisis de la literatura, es resaltar los problemas encontrados en cada tipo de arquitectura

de acoplamiento. También muestra la no existencia de un estudio en paralelo de las tres arquitecturas.

- Selección de los materiales y métodos. En este proceso se seleccionaran los algoritmos, SGBD, conjunto de datos, métricas y otro tipo de herramientas que serán útiles para dar solución al problema identificado.
- Implementación de algoritmos de Minería de Datos en las arquitecturas identificadas. Después del proceso de selección, se procede a implementar los algoritmos en los tres tipos de arquitectura (débil, mediana y fuertemente).

B. Experimentación y análisis de resultados. En esta fase se ponen en marcha los algoritmos implementados en los tres tipos de arquitectura con los distintos conjuntos de datos (ya sean distintos en tamaño o tipo), de esta manera podemos determinar el comportamiento de cada tipo de arquitectura (débil, mediana y fuertemente).

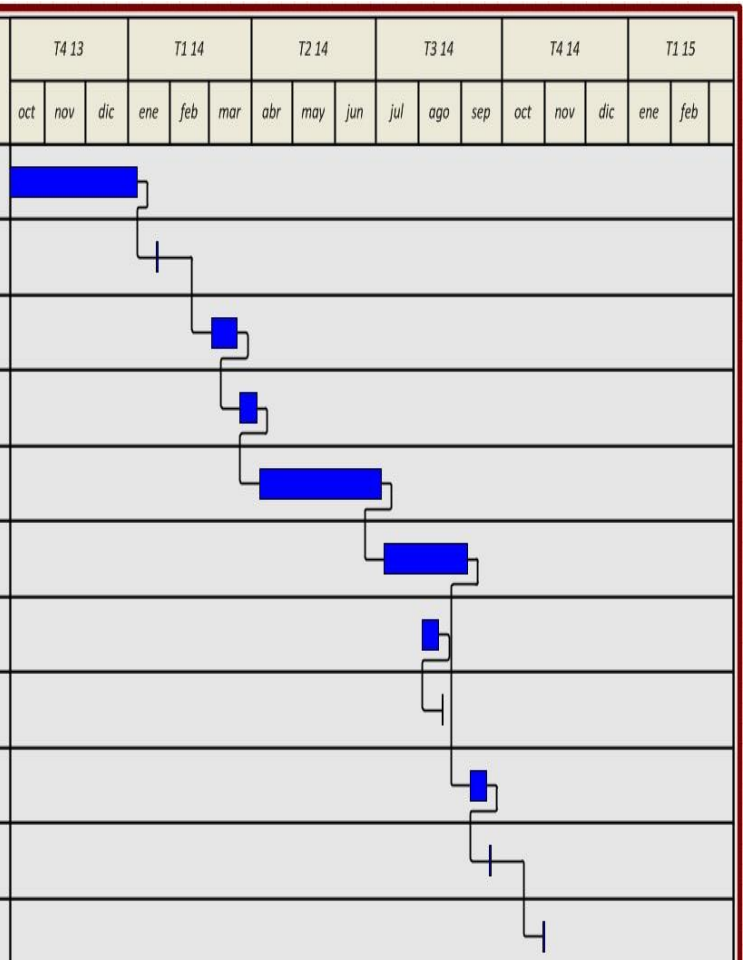
   Luego de haber realizado los experimentos se procede a analizarlos. Este análisis contemplara el rendimiento (tiempo, espacio), facilidad de implementación, y cuan comprensible es el resultado que se obtiene de cada tipo de arquitectura.

C. Publicación. En esta fase se pretende hacer público los resultados de la investigación, ya sea difundiendo los resultados en revistas, congresos o seminarios.

## 8. PLAN DE TRABAJO

A continuación se muestra el diagrama de Gantt, el cual contiene el plan de trabajo.

| Id. | Nombre de tarea | Comienzo | Fin | Duración |
|---|---|---|---|---|
| 1 | Elaboración de la propuesta de tesis | 07-10-2013 | 07-01-2014 | 67d |
| 2 | Presentación de propuesta de tesis | 22-01-2014 | 22-01-2014 | 1d |
| 3 | Análisis del estado del arte | 03-03-2014 | 21-03-2014 | 15d |
| 4 | Selección de materiales y métodos | 24-03-2014 | 04-04-2014 | 10d |
| 5 | Implementación de algoritmos de Minería de Datos | 07-04-2014 | 04-07-2014 | 65d |
| 6 | Experimentación | 07-07-2014 | 05-09-2014 | 45d |
| 7 | Análisis de resultados (primera parte) | 04-08-2014 | 15-08-2014 | 10d |
| 8 | Difusión de resultados (primera parte) | 18-08-2014 | 18-08-2014 | 1d |
| 9 | Análisis de resultados (segunda parte) | 08-09-2014 | 19-09-2014 | 10d |
| 10 | Difusión de resultados (segunda parte) | 22-09-2014 | 22-09-2014 | 1d |
| 11 | Defensa de tesis | 31-10-2014 | 31-10-2014 | 1d |



## BIBLIOGRAFÍA


[1] R. Elmasri and S. Navathe, *Fundamentos de Sistemas de Bases de Datos*. 2007.

[2] A. Silberschatz, H. Korth, and S. Sudarshan, *Fundamentos de Bases de Datos*. 2002.

[3] G. Powell, *Beginning Database Design*. 2006.

[4] E. F. Codd, “A relational model of data for large shared data banks. 1970.,” *MD. Comput.*, vol. 15, no. 3, pp. 162–6, 1970.

[5] J. Sánchez, “Lenguaje SQL (I), DDL y DML,” 2008. [Online]. Available: http://www.cartagena99.com/recursos/programacion/apuntes/SQL(DDLyDML).pdf.

[6] U. Fayyad, G. Piatetsky-shapiro, and P. Smyth, “From Data Mining to Knowledge Discovery in Databases,” pp. 37–54, 1996.

[7] E. Y. Weka, “Técnicas de análisis de datos,” 2006.

[8] P. Machine and L. Tools, *Data Mining: Practical Machine Learning Tools and Techniques*.

[9] Imielnski T., Mannila, H., A Database Perspective on Knowledge Discovery, Communications of the ACM, Vol 39, No. 11, November, 1996.

[10] Sarawagi, S., Thomas S., Agrawal R., Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications, ACM SIGMOD, 1998.

[11] 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.

[12] Timarán, R., Nuevas primitivas SQL para el Descubrimiento de Conocimiento en Arquitecturas Fuertemente Acopladas con un SGBD, propuesta investigación doctoral, Doctorado en Ingeniería, Universidad del Valle, C a l i , C o l o m b i a , 2 0 0 1 ,ritimar@borabora.univalle.edu.co.

[13] Clear, J., Dunn, D., Harvey, B., Heytens, M., Lohman, P., Mehta, A., Melton, M., Rohrberg, L., Savasere, A., Wehrmeister, R., Xu, M., Large Scale Knowledge Discovery Using NonStop SQL/MX, Technical Report, Compaq Computer Corporation, Tandem Division, 1999.

[14] Agrawal R., Shim K., Developing Tightly-Coupled Applications on IBM DB2/CS Relational Database System: Methodology and Experience, Research Report, 1996.

[15] Isoft S.A. Alice, http://www.alicesoft.com/html/prod_alice.htm, 2013.

[16] RuleQuest Research Inc., C5.0, http://www.rulequest.com, 2013.

[17] Quadrillion Corp., Q-Yield, http://www.quadrillion.com/qyield.shtm, 2001.

[18] Matheus C., Chang P., Piatetsky-Shapiro G., Systems for Knowledge Discovery in Databases, IEEE Transactions on Knowledge and Data Engineering, Vol 5, No 6, 1993.

[19] Kdnuggets, http://www.kdnuggets.com/software, 2001.

[20] Piatetsky-Shapiro G., Brachman R., Khabaza T., An Overview of Issues in Developing Industrial Data Mining and Knoledge Discovery Applications, 2º Conference KDD y Data Mining, Portland, Oregon, 1996.

[21] SPSS, Clementine, http://www.spss.com/clementine, 2001.

[22] 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.

[23] 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.

[24] 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.

[25] Han J., Fu Y., Tang S., Advances of the DBLearn System for Knowledge Discovery in Large Databases, Int'l Joint Conference on Artificial Intelligence IJCAI, Montreal, Canada, 1995.

[26] Brachman R., Anand T., The Process of Knowledge Discovery in Databases: A First Sketch, Workshop on Knowledge Discovery in Databases, 1994.

[27] IBM Corporation, Intelligent Miner, http://www-4.ibm.com/software/data/iminer, 2001.

[28] Agrawal R., Mehta M., Shafer J., Srikant R., Arning A., Bollinger T., The Quest Data Mining System, 2º Conference KDD y Data Mining, Portland, Oregon, 1996.

[29] Goebel, M., Gruenwald, L., A Survey of Data Mining and Knowledge Discovery Software Tools, SIGKDD Explorations, Vol. 1, Issue 1, June, 1999.

[30] Agrawal R., Shim K., Developing Tightly-Coupled Data Mining Applications on a Relational Database System, The Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, 1996.

[31] Agrawal R., Srikant R., Fast Algorithms for Mining Association Rules, VLDB Conference, Santiago, Chile, 1994.

[32] 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.

[33] 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.

[34] Imielinski T., Virmani A., Abdulghani A., Data Mine: Application Programming Interface and Query Language for database Mining, 2º Conference KDD y Data Mining, Portland, Oregon, 1996.

[35] Meo R., Psaila G., Ceri S., A New SQL-like Operator for Mining Association Rules, VLDB Conference, Bombay, India, 1996.

[36] 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.

[37] 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.

[38] Meo R., Psaila G., Ceri S., A Tightly-Coupled th. Architecture for Data Mining, 14 International Conference on Data Engineering ICDE98, 1998.

[39] R e s e a r c h M i c r o s o f t ,http://research.microsoft.com/dmx/datamining, 2001.

[40] Safavian, S.R. and Landgrebe, D.: A survey of decision tree classifier methodology. IEEE Transactions on Systems, Man and Cybernetics. 21-3, 1991.

[41] Mitchell, T.: Machine Learning. McGraw Hill, 1997.

[42] Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, C.A., 1993.

[43] Kohavi, R. and Quinlan, J.R.: Decision-tree discovery. Will Klosgen and Jan M. Zytkow, editors. Oxford University Press, 2002.

[44] Kuncheva, L.: Combining Pattern Classifiers. John Wiley and Sons, 2004.

[45] Agrawal R., Imielinski T., and Swami A., Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, ACM SIGMOD International Conference on Management of Data. USA,1993.

[46] Oracle, Oracle Data Mining, 2013 http://www.oracle.com/technetwork/database/options/advanced-analytics/odm/index.html?ssSourceSiteId=otnru

[47] Microsoft, SQL Server 2012 Predictive Analytics, 2013 http://www.microsoft.com/en-us/sqlserver/solutions-technologies/business-intelligence/predictive-analytics.aspx

[48] Ordonez C., Garcia-Alvarado C., A Data Mining System Based on SQL Queries and UDFs for Relational Databases. ACM CIKM, Glasgow, Scotland, UK, 2011.

[49] Ordonez C., Mohanam N., Garcia-Alvarado C., Fast PCA Computation in a DBMS with Aggregate UDFs and LAPACK. ACM CIKM, Maui, HI, USA, 2012.

[50] Robles Y., Sotolongo A., INTEGRACIÓN DE LOS ALGORITMOS DE MINERÍA DE DATOS 1R, PRISM E ID3 A POSTGRESQL. JISTEM - Journal of Information Systems and Technology Management, Brazil, 2013.

[51] Ordonez C., Pitchaimalai S., One-pass Data Mining Algorithms in a DBMS with UDFs. ACM SIGMOD Athens, Greece, 2011.