UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO UAEM TEXCOCO

INTEGRACIÓN DE CONSULTAS A BASES DE DATOS HETEROGÉNEAS

TESIS

PARA OBTENER EL TITULO DE:

LICENCIADA EN INFORMÁTICA ADMINISTRATIVA

PRESENTA

NOEMÍ MARTÍNEZ RAMÍREZ

DIRECTOR

DR. EN CIENCIAS ADRIAN TRUEBA ESPINOSA

REVISORES

DR. EN CIENCIAS JAIR CERVANTES CANALES
M. EN CIENCIAS ÁNGEL RAFAEL QUINTOS RAMÍREZ
M. EN CIENCIAS JOSÉ SERGIO RUIZ CASTILLA

TEXCOCO, ESTADO DE MÉXICO, MARZO DE 2012

Texcoco, México a 12 de Marzo de 2012.

M. EN C. JUAN MANUEL MUÑOZ ARAUJO
SUBDIRECTOR ACADÉMICO DEL
CENTRO UNIVERSITARIO UAEM
TEXCOCO.
PRESENTE:

COPIA

AT' M.EN F. GUADALUPE LIZETH ARCE CHÁVEZ
RESPONSABLE DEL DEPARTAMENTO DE TITULACIÓN.

Con base en las revisiones efectuadas al trabajo escrito titulado "Integración de consultas de Bases de Datos Heterogéneas" que para obtener el título de Licenciado en Informática Administrativa presenta la sustentante C. Noemí Martínez Ramírez, con número de cuenta 0721239 respectivamente, se concluye que cumple con los requisitos teórico- metodológicos necesarios para su aprobación, pudiendo continuar con la etapa de impresion del trabajo escrito.

ATENTAMENTE

DR. JAIR CERVANTES CANALES
REVISOR

M. EN C.C. ANGEL RAFAEL QUINTOS RAMIREZ
REVISOR

M. EN C.A. JOSE SERGIO RUIZ CASTILLA
REVISOR

TITULACION
RECIBIDO
12 03 12
TEXCOCO, Méx., a

DR. EN CIENCIAS ADRIAN TRUEBA ESPINOSA
DIRECTOR

c. c. p. Noemí Martínez Ramírez
c. c. p. Dr. en Ciencias Adrian Trueba Espinosa- Coordinador de Investigación y Estudios Avanzados
c. c. p. Titulación

Agradecimientos

A Dios:

Por cuidar de mis seres queridos y darme la oportunidad de tener un logro más en mi vida.

A mis abuelos:

Juana Portuguez Cabrera y José Guadalupe Ramírez Ayala

Por el apoyo incondicional y confianza para cumplir una meta más en mi vida, esforzándose por brindarme un mejor futuro.

A mis padres:

Ma. Edith Ramírez Portuguez y Vicente Martínez Salazar

Por apoyo y confianza depositada en mi en cumplir una meta.

A mis hermanos:

Ismael Martínez Ramírez y Jesús Guadalupe Martínez Ramírez

Por los consejos brindados y el apoyo incondicional en todos los momentos que lo he necesitado, gracias por estar conmigo.

A mi Director:

Dr. Adrian Trueba Espinosa

Por el apoyo incondicional durante el desarrollo de la Tesis, por su paciencia, dedicación, y la confianza depositada para el desarrollo del proyecto además de ayudarme a realizar un logro más en mi vida.

A mis revisores:

Dr. en Ciencias Jair Cervantes Canales, M. en Ciencias Ángel Rafael Quintos Ramírez y M. en Ciencias José Sergio Ruiz Castilla

Por el apoyo brindado y el conocimiento brindado en el desarrollo de la tesis.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

I. INTRODUCCIÓN

Muchas empresas privadas y públicas cuentan con Sistemas de Información, que cuando se usan adecuadamente logran proporcionar información oportuna y accesible, permitiendo el crecimiento organizacional al momento de proporcionar información. La información obtenida ayuda a gerentes o directivos a la toma de decisiones.

Una parte esencial en los sistemas de información son las Bases de Datos (BD), en ellas se almacenan los datos de la empresa u organización. Estas BD no se pensó fueran compartidas a futuro, por lo que, están diseñadas de forma diferente. Cada empresa u organización cuenta con su propia BD, almacenadas en diferentes manejadores de BD. Por otro lado, la globalización mundial ha creado la necesidad de intercambiar o compartir información dentro de la misma empresa o con otras. De los comentarios anteriores se desprende la necesidad de consultar BD con diferentes esquemas y manejadores de BD con ubicación geográfica diferente. Algunos autores han trabajado en resolver esta problemática como Enríquez (2010) que menciona “esto implica que la interoperabilidad entre diferentes sistemas de información ha sido uno de los aspectos más críticos en algunas organizaciones, y como consecuencia los cambios organizacionales que sufren las empresas modernas, alianzas estratégicas, compartimiento de información, y absorción de pequeñas y medianas industrias por grandes corporativos”.

En la última década esta preocupación se vio incrementada con la proliferación de diferentes bases de datos, con diferentes modelos de datos que corren en diferentes plataformas. En estos ambientes, los usuarios están limitados en el acceso uniforme de la información. Los sistemas de bases de datos heterogéneas son sistemas computacionales que hacen disponible la información de diversas fuentes, y donde esas fuentes de información pueden ser heterogéneas, distribuidas y autónomas. En este trabajo se presenta una alternativa para la consulta a Bases de Datos Heterogéneas (HDBS)

II. PLANTEAMIENTO DEL PROBLEMA

Hay muchos datos que se pueden compartir para realizar estudios de diferentes índoles, sin embargo, muchos de ellos no pueden ser consultados al mismo tiempo, esto debido a que los datos se encuentran almacenados en diferentes manejadores de BD. Tales como Oracle, MySql, SQL Server y PostgreSQL.

Por otro lado, la falta de la estandarización en la estructura de las consultas por los diferentes manejadores de bases de datos ocasiona pérdida de tiempo para realizar consultas para la toma de decisiones en el momento requerido, ya que se tiene que consultar manejador por manejador lo cual ocasiona altos costos generados por la consulta a cada manejador ya que se requiere de una persona que realice dichas consultas o de la capacitación para el manejo de cada una de estas.

III. JUSTIFICACIÓN

Los costos que se ocasionan por las consultas a BD en diferentes sistemas electrónicos son altos. Con la implementación de un sistema de consultas automatizada, donde independientemente de los tipos de manejadores de bases de datos (los considerados en este trabajo), se tendrá un ahorro en tiempo tanto en la consulta, como en la toma de decisiones, además de la disminución de costos en la contratación de personal para la consulta de cada uno de los manejadores de datos o de la capacitación para el manejo de estos.

La estandarización y automatización de las consultas, proporcionará a los usuarios eficiencia en las consultas requeridas en cualquier tipo de BD que se requieran cruzar, para extraer información y con ello tener los elementos necesarios para una adecuada toma de decisiones.

IV. OBJETIVOS

4.1 OBJETIVO GENERAL

Implementar un sistema para la automatización de consultas de forma sincronizada a Bases de Datos: Oracle, MySql, SQL Server y PostgreSQL para obtener datos en conjunto.

4.2 OBJETIVOS PARTICULARES

- Estructurar y ejecutar consultas en SQL a las BD de Oracle, MySql, SQL Server y PostgreSQL con el mismo fin.

- Identificar las variantes estructurales de las consultas SQL de los diferentes manejadores de bases de datos, para conocer las diferencias.

- Planteamiento de algoritmos para estructurar consultas sincronizadas a las diferentes bases de datos.

- Programar algoritmos de estandarización y automatización para las consultas en lenguaje JAVA.

- Implantación del Sistema para la automatización de las consultas a los diferentes manejadores de BD de manera sincrónica.

V. MARCO TEÓRICO

5.1 UML.

5.1.1 Antecedentes.

El UML (Unified Modeling Language o Lenguaje de Modelado Unificado) surgió a finales de la década de 80´s y principios de los 90´s. El UML unifica, sobre todo, los métodos de Booch, Rumbaugh (OMT) y Jacobson (padre de los casos de uso) también conocidos como los 3 amigos, los creadores del UML llegaron a la conclusión que el UML es un lenguaje de modelado y no un método. Debido que el lenguaje de modelado solo se necesita comprenderlo mientras que en el método se requieren analizar los pasos (Fowler, 1999). UML es una norma desarrollada bajo los auspicios del grupo de Administración de Objetos (Object Management Group, OMG) para la creación de especificaciones de diferentes componentes de los Sistemas de Software, UML se ha convertido en el estándar industrial para el modelado de Sistema de Información Silberschatz et al., (2006). Además Simon (2007) señala que UML consta, principalmente, de un lenguaje gráfico para representar los conceptos requeridos en el desarrollo de un Sistema de Información Orientado a Objetos.

5.1.2 ¿Qué es UML?.

Benet(2003) dice UML es un modelo para la construcción de software Orientado a Objetos que ha sido propuesto como estándar de ISO por el OMG, consta de un conjunto de tipo de diagramas interrelacionados, dentro de los cuales se utilizan elementos de modelo, que sirven para descubrir distintos aspectos de la estructura y la dinámica del software. Asimismo Cavero (2005) comenta “UML es una notación, o más bien un conjunto de notaciones, de naturaleza diagramática y esencialmente visual, concebidas para la especificación de aspectos estructurales y dinámicos en Sistemas Orientados a Objetos”.

Los diagramas de UML están formados por cuatro elementos que son iconos, símbolos bidimensionales, caminos y cadenas (Simon, 2007). Algunos de los diagramas que conforman el UML se mencionan a continuación:

- Diagramas de Clase: son parecidos a los diagramas Entidad-Relación.
- Diagramas de Caso de Uso: muestran la interacción entre los usuarios y el sistema, especial los pasos de las tareas que llevan a cabo los usuarios (como retirar dinero o matricularse en una asignatura).
- Diagramas de Actividad: describen el flujo de tareas entre los diferentes componentes del sistema.
- Diagramas de Implementación: muestran los componentes del sistema y sus interconexiones, tanto en el nivel de los componentes de software como en el de hardware.

5.1.3 Modelo UML.

Para describir el proceso de modelado UML, el concepto de modelo es la abstracción de un sistema (ente global) o de un subsistema (parte del sistema con elementos relacionados) desde una determinada perspectiva o vista. Además que el modelo es completo y coherente al nivel de abstracción que haya elegido. En estos diferentes modelos se presentan diferentes vistas de un sistema y un diagrama que es la representación grafica de un conjunto de elementos en el modelo de un sistema.

Estos diferentes modelos presentan diferentes vistas del sistema. Los creadores de UML sugieren cinco vistas para el empleo de UML: la vista de diseño, la vista de procesos, la vista de implementación y la vista de despliegue. La elección de los diagramas a utilizar para el modelado depende de la naturaleza y de la complejidad del sistema que se está modelando.

UML proporciona dicha notación para el modelado de sistemas y subsistemas (ver Figura1) además de modelos que emplea una extensión de notación para paquetes UML. Los paquetes son una forma de organizar los elementos de modelo y de agruparlos Simon (2007).

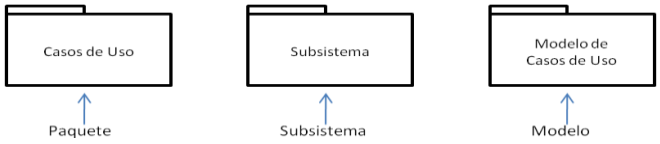

Figura 1 Notación UML para un Sistema que contiene subsistemas (Simon, 2007).

5.1.4 Modelado de los Requisitos.

5.1.4.1 Lista de Requisitos.

- Actor: usuario externo al sistema puede desempeñar diferentes funciones en la relación con el sistema. Una pareja (usuario, función) constituye un actor específico designado en UML únicamente por el nombre de la función. Existen dos tipos de actores: actores primarios en los cuales su objetivo del caso de uso es esencial y los actores secundarios que interactúan con el caso de uso, pero cuyo objetivo no es esencial (ver Figura 2).

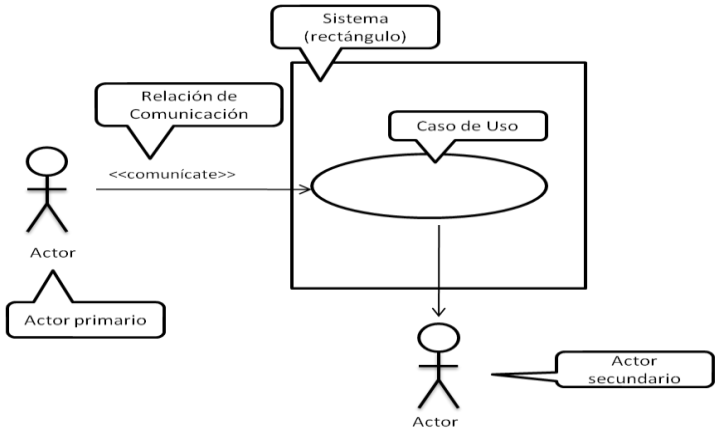

Figura 2 Actores primarios y secundarios de Caso de Uso (Laurent, 2005).

- Escenario: es la instancia de un caso de uso en la se fijan todas las condiciones relativas a los diferentes eventos. En un caso de uso le corresponde varios escenarios y al igual que a las clases, albergan los aspectos comunes de las instancias, los casos de uso describen de manera común el conjunto de escenarios utilizando derivaciones condicionales para representar las diferentes alternativas.

- Relación de comunicación: es la relación que vincula a un actor con un caso de uso a esto se le denomina relación de comunicación. Esta relación da soporte a diferentes modelos (ver Figura 3).

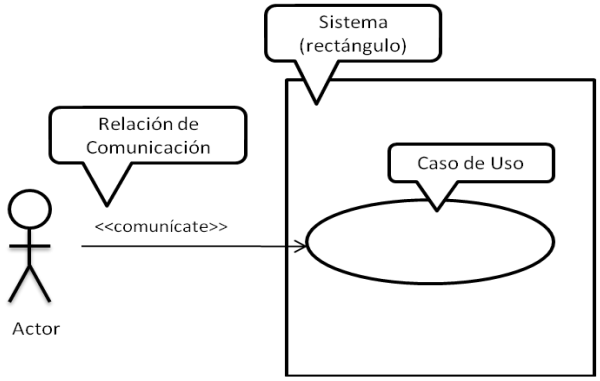

Figura 3 Sistema de Caso de Uso (Elaboración propia, 2012).

- Diagrama de los Casos de Uso: muestra los casos de uso representados en forma de elipses y a los actores en forma de personajes. También indica las relaciones de comunicación que los vinculan (ver Figura 4).

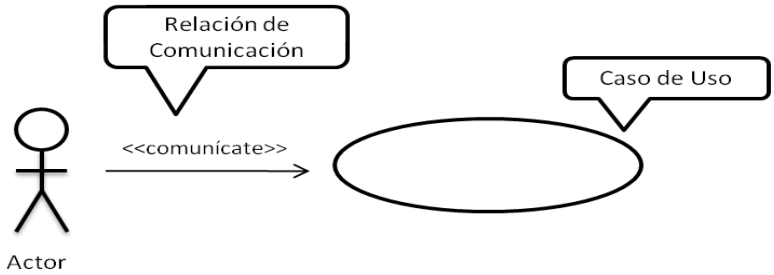

Figura 4 Diagrama de Caso de Uso (Laurent, 2005).

5.1.4.2 Casos de Uso.

Los Casos de Uso son descripciones de la finalidad de sistemas desde la perspectiva del usuario. Los diagramas de caso de uso se utilizan para mostrar la funcionalidad que el sistema ofrece y que los usuarios se comunicarán con el sistema para utilizar esa funcionalidad Simon (2007).

Los diagramas de Caso de Uso ofrecen una estrategia completa para el desarrollo de sistemas de software orientado a objetos, pero los diagramas de caso de uso son el punto inicial para la mayor parte de su estrategia.

5.1.4.2.1 Objetivo de Casos de Uso.

El principal objetivo de los Casos de Uso es una detalla descripción que ofrece la interacción entre los usuarios del sistema, denominados actores, y las funciones de alto nivel contenidas en el sistema, los casos de uso. Las descripciones pueden ser en forma de resumen o detalladas, en donde la interacción entre actor y casos de uso se describe paso a paso.

5.1.4.2.2 Relaciones entre Casos de Uso.

- Relación de Inclusión: sirve para enriquecer un caso de uso con otro, mediante una inclusión imperativa y por lo tanto es sistemático. Esta relación no responde a un actor primario, por lo tanto este tipo de caso es subfuncional (ver Figura 5).

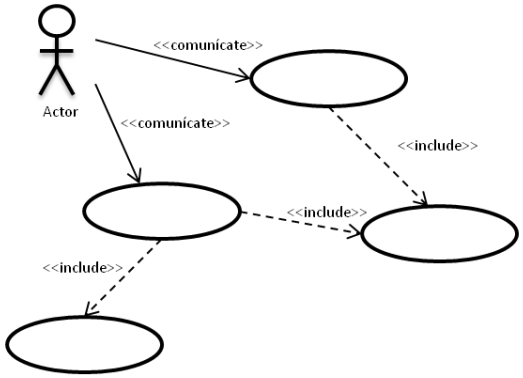

Figura 5 Descomposición de Caso de Uso por Inclusión (Laurent, 2005).

- Relación de Extensión: enriquece un caso de uso mediante un caso de uso de subfunción, este enriquecimiento es análogo y es opcional. Además hace una serie de puntos concretos y previstos en el momento del diseño, llamados puntos de extensión (ver Figura 6).

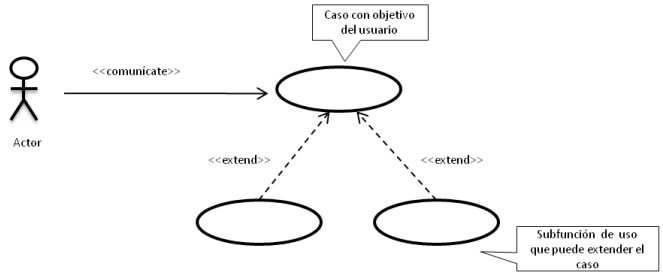

Figura 6 Extensión del Caso de Uso (Laurent, 2005).

5.1.4.2.3 Especialización y generalización de Casos de Uso.

Las clases de objetos son posibles, de misma forma que es posible especializar un caso de uso. De esta manera se obtiene un subcaso de uso. El subcaso hereda el comportamiento y la relaciones de comunicación, inclusión y extensión del súper-caso de uso. En muchas ocasiones el súper-caso de uso es abstracto (corresponde a un compartimiento parcial completado en el subcaso). Los subcasos de uso tienen el mismo nivel que sus súper-casos como se muestra en la Figura 7 Laurent (2005).

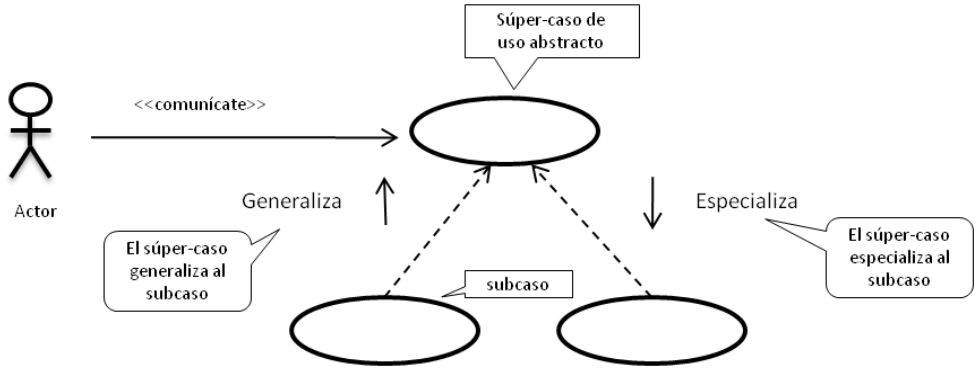

Figura 7 Especialización de un Caso de Uso (Laurent, 2005).

5.2 Bases de Datos.

Antes de entrar al tema de Bases de Datos es necesario mencionar otros conceptos para que la definición de Bases de Datos sea más entendible y términos básicos que se mencionaran dentro de este tema.

En la definición de Bases de Datos (BD) se menciona el concepto de dato que significa simplemente “hecho”, entidades independientes sin evaluar. Los datos pueden ser numéricos o no numéricos (por ejemplo, alfanuméricos o simbólicos). Mientras que información es un conjunto ordenado de datos los cuales pueden recuperarse de acuerdo con la necesidad del usuario.

Por otro lado, se encuentra el concepto de campo que es la unidad más pequeña a la cual uno puede referirse en un programa de computadora. De dicho concepto se define a registro como un conjunto de campos con relación entre sí. Ahora bien a un archivo se define como una colección de registros del mismo tipo Tsai (1990).

5.2.1 Definición de Bases de Datos.

Tsai(1990) menciona una Base de Datos (BD) “es una colección de archivos interrelacionados creados con un Sistema de Manejo de Bases de Datos (DSMS)”. El contenido de una BD se obtiene combinando datos de todas las diferentes fuentes en una organización, de tal manera que los datos estén disponibles para todos los usuarios, y los datos redundantes puedan eliminarse, o al menos minimizarse. Los datos almacenados en una BD se encuentran de forma física en una disposición distinta a la perspectiva lógica, además de que los usuarios pueden tener acceso a los datos.

5.2.2 Entidades, Relaciones y Atributos.

En el tema de bases de datos se especifican varios conceptos para el entendimiento de este. La mayoría de los Sistemas de Administración de Bases de Datos (DBMS) comerciales, las tablas almacenan un conjunto de entidades. Las tablas son arreglos de datos en dos dimensiones, además de estar formadas por un encabezado (el primer renglón) y un cuerpo (los otros renglones) que son la muestra del contenido. Las relaciones son las conexiones entre tablas. Mientras

que las entidades son un conjunto de datos del mismo tema, y se puede acceder de forma conjunta, la entidad puede representar una persona, lugar, evento o casa. El atributo es una propiedad de la entidad o relación, cada tributo tiene un tipo de dato que define el valor y las operaciones permitidas, el atributo es sinónimo de campo o columna. Michael (2007).

5.2.3 Características de una Base de Datos.

- Persistente. Los datos almacenados residen en un almacenamiento estable, tal como un disco magnético. El almacenamiento y el mantenimiento de los datos son costosos, por lo que solo se almacenan los datos relevantes para la toma de decisiones.
- Compartir. La BD puede tener diferentes usos y usuarios, además de proporcionar memoria común para la realización de varias funciones en una organización.
- Interrelación. Los datos almacenados como unidades separadas se pueden conectar para mostrar un cuadro completo. Este concepto demuestra las entidades y las relaciones.

5.3 Sistema de Manejador de Base de Datos.

5.3.1 ¿Qué es un Sistema de Manejador Base de Datos?.

Tsai (1990) dice el Sistema de Manejo de Bases de Datos (DBMS) es una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de alguna tarea específica (ver Figura 8).

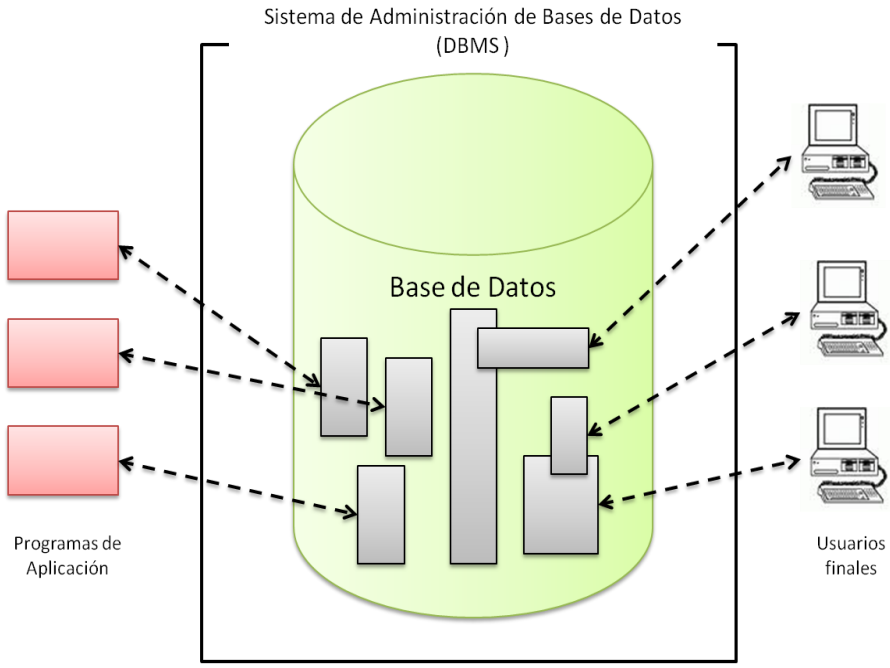

Figura 8 Imagen Simplificada de un Sistema de Bases de Datos (Date, 2001).

5.3.2 Fundamentos del Sistema de Manejador de Bases de Datos.

Las funciones principales de un DBMS son:

- Crear y organizar la Base de Datos.
- Establecer y mantener las trayectorias de acceso a la Base de Datos, de tal manera que los datos en cualquier parte de la base puedan acceder rápidamente.
- Manejar los datos de acuerdo con las peticiones de los usuarios.
- Mantener la integridad y seguridad de los datos.
- Registrar el uso de las bases de datos.

El DMBS interpreta y procesa las peticiones del usuario para recobrar información de la base. Además sirve de interface entre las peticiones del usuario y la BD. Las

preguntas a la base pueden tener distintas formas, pueden teclearse directamente desde la terminal, o codificarse como programas en lenguajes de alto nivel y presentarse para procesamiento interactivo o por lotes.

5.3.3 Ventajas de un Sistema Manejador de Bases de Datos.

Ramakrishnan (2007) dice el empleo de un DBMS para gestionar los datos tiene muchas ventajas:

- Independencia con respecto a los datos. Ofrece una vista abstracta de los datos, ocultando los detalles de representación y almacenamiento de los datos.
- Acceso eficiente a los datos. Emplean técnicas sofisticadas para almacenar y recuperar los datos de manera eficiente, al igual que el almacenamiento en dispositivos externos.
- Integridad y seguridad de los datos. El acceso a los datos se controla por las restricciones de integridad, además del control de acceso a la información almacenada.
- Administración de los datos. Cuando existen diversos usuarios en un DBMS, y los datos se comparten la centralización de la administración de esos datos ofrece una mejora significativa, además de cuidar la redundancia de los datos y el mejoramiento de almacenamiento de los datos, para obtener una mejor recuperación de datos.
- Acceso concurrente y recuperación en caso de fallo. Los DBMS están programados de tal manera que los accesos concurrentes a los datos que hacen creer al usuario que solo él lo está ocupando, además de proteger a los usuarios de fallos en el sistema.
- Reducción del tiempo de desarrollo de las aplicaciones. El DBMS soporta muchas funciones importantes y aplicaciones que accedan a los datos. Estas aplicaciones cuentan con interfaces de alto nivel, facilitando el fácil acceso a los datos, las aplicaciones suelen ser robustas e independientes debido a su manejo de diferentes funciones.

5.3.4 Herramientas de DBMS.

El éxito de los DBMS reside en mantener la seguridad e integridad de los datos. Lógicamente tiene que proporcionar herramientas a los distintos usuarios. Entre las herramientas que proporciona están:

- Herramientas para la creación y especificación de los datos. Así como de la estructura de la base de datos.
- Herramientas para administrar y crear la estructura física requerida en las unidades de almacenamiento.
- Herramientas para la manipulación de los datos de las bases de datos para añadir, modificar, suprimir o consultar datos.
- Herramientas de recuperación en caso de desastre.
- Herramientas para la creación de copias de seguridad.
- Herramientas para la gestión de la comunicación de la base de datos.
- Herramientas para la creación de aplicaciones que utilicen esquemas externos de los datos.
- Herramientas de instalación de la base de datos.
- Herramientas para la exportación e importación de datos.

5.3.5 Niveles de Abstracción.

Los DBMS describen tres niveles de abstracción (ver Figura 9), la descripción de las bases de datos consta de un esquema en cada uno de esos tres niveles de abstracción.

Los fabricantes de DBMS soportan las órdenes de SQL para la descripción de aspectos del esquema físico, pero esas órdenes no forman parte de la norma de lenguaje de SQL. La información sobre los esquemas conceptual, externo y físico se guarda en los catálogos del sistema.

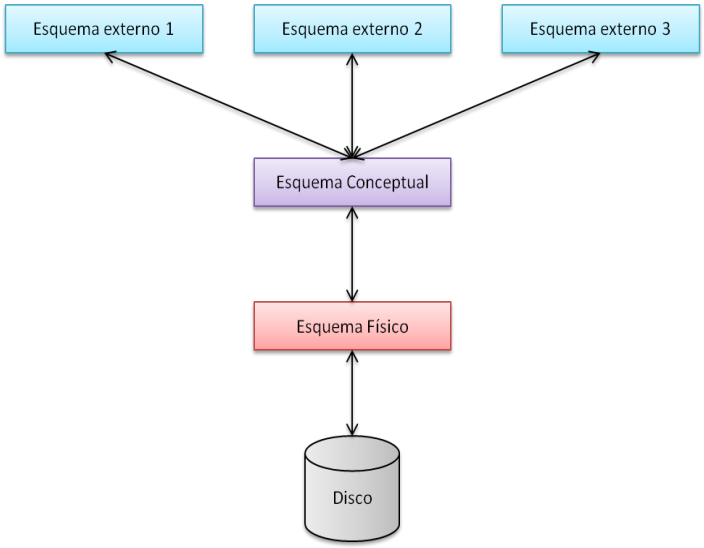

Figura 9 Niveles de Abstracción de DBMS (Ramakrishnan, 2007).

- Esquema conceptual

También es conocido como esquema lógico, se describen los datos almacenados en términos del modelo de datos del DBMS. En un DBMS relacional, el esquema conceptual describe todas las relaciones almacenadas en la base de datos. La elección de las relaciones, y de los campos de cada relación, no resulta siempre evidentes, y el proceso de búsqueda de un buen esquema conceptual se denomina diseño conceptual de bases de datos.

- Esquema físico

En este esquema se especifican los detallen de adicionales de almacenamiento, en pocas palabras este esquema resume el modo en que las relaciones descritas en el esquema conceptual se guardan realmente en dispositivos de almacenamiento secundario como en discos o cintas. La organización de los archivos, las relaciones de los datos y las estructuras de los datos auxiliares denominadas índices, son para acelerar las operaciones de recuperación. Las decisiones sobre el esquema físico se basa en conocer el modo en que se suele tener acceso a los datos, a este proceso se le llama diseño de bases de datos.

- Esquema externo

El esquema externo permite personalizar y autorizar el acceso a los datos a los usuarios y grupos de ellos. Cualquier base de datos tiene un esquema conceptual y físico, por lo que se tiene almacenado las relaciones, pero puede tener diferentes esquemas externos. Este esquema consta de un conjunto de una o varias vistas y relaciones del esquema conceptual, por otro lado el diseño de las vistas están diseñadas de acuerdo a las necesidades del usuario final.

5.4 Sistema de Bases de Datos.

5.4.1 Definición de Sistemas de Bases de Datos.

Date (2001) escribe un Sistema de Bases de Datos es básicamente un sistema computarizado para guardar registros, es decir, es un sistema computarizado cuya finalidad general es almacenar información y permitir a los usuarios recuperar y actualizar esa información con base en peticiones. La información en cuestión puede ser cualquier cosa que sea de importación para el individuo u organización (ver Figura 11). Mientras tanto Tsai (1990) dice que un sistema de bases de datos es un sistema computarizado de información para el manejo de datos por medio de paquetes de software llamados sistemas de manejo de bases de datos (DSMS).

5.4.2 Componentes de un Sistema de Bases de Datos.

Los componentes principales de un sistema de bases de datos son el hardware, el software DBMS y los datos a manejar, los cuales se describirán a continuación:

- Hardware

Los volúmenes de almacenamiento secundario, principalmente discos magnéticos, los cuales se emplean para contener los datos almacenados, junto con los dispositivos asociados de E/S (por ejemplo unidades de disco), los controladores de dispositivos, etc.

Los procesadores de hardware y la memoria principal asociada usados para apoyar la ejecución del software del sistema de bases de datos.

- Software

Es una capa de software que se encarga de administrar la base de datos o el servidor de bases de datos. Todas las solicitudes de acceso a la base de datos son manejadas por el DBMS, además de agregar y eliminar archivos (o tablas), recuperar y almacenar datos desde y en distintos archivos. También ofrece a los usuarios finales una percepción grafica de la BD con la que se está trabajando.

- Datos

En un sistema de bases de datos los datos deben ser integrados y compartidos. La integridad en los datos es la unificación de estos en varios archivos que serán distintos y su redundancia es eliminada de forma parcial. Mientras que los datos compartidos son los datos a los que un usuario puede tener acceso, pero los fines de uso de estos datos pueden ser diferentes, esto dependerá de los niveles de seguridad en la BD.

- Usuario

Existen tres tipos de usuarios los cuales se mencionaran a continuación:

- ❖ Programadores de aplicaciones son los responsables de escribir los programas de aplicación de base de datos en algunos lenguajes de programación, estos lenguajes permiten el acceso a la BD por medio de sentencias de selección, el propósito de estas sentencias es el acceso a la BD.
- ❖ Usuarios finales son los que interactúan con el sistema desde estaciones de trabajo o terminales en línea. Este usuario tiene acceso a BD a través de aplicaciones en línea.
- ❖ Administrador de base de datos o DBA es la persona de salvaguardar los datos de un usuario u organización (la información es lo más valiosos en una organización), además que decidirá el lugar de almacenamiento de los datos y establecer las políticas para mantener y manejar los datos almacenados.

5.4.3 Propósito de los Sistemas de Bases de Datos.

Permiten a los usuarios manipular la información, el sistema tienen varios programas de aplicación que gestionan los archivos, incluyendo programas.

Los sistemas operativos convencionales soportan este sistema de procesamiento de archivos típico. El sistema almacena los registros permanentes en varios archivos y necesita diferentes programas de aplicación para extraer y añadir a los archivos correspondientes.

Al almacenar la información de una organización en un sistema de procesamiento de archivos tiene una serie de inconvenientes importantes, los cuales se describirán a continuación:

- Redundancia e inconsistencia de los datos. La redundancia conduce a costes de almacenamiento y de acceso más elevados. A su vez surge la inconsistencia de los datos, es decir puede que las diferentes copias de los mismos datos no coincidan.
- Dificultad en el acceso a los datos. Son las peticiones que no son previstas por el diseñador y programador de la aplicación, y como consecuencia surge la dificultad de acceso.
- Aislamiento de datos. Como los datos están dispersos en varios archivos, y los archivos pueden estar en diferentes formatos, es difícil escribir nuevos programas de aplicación para recuperar los datos correspondientes.
- Problemas de Integridad. El almacenamiento debe de satisfacer ciertos tipos de restricciones de consistencia. Los problemas de restricciones se solucionan con ayuda de los desarrolladores del sistema por medio de código con las aplicaciones adecuadas.
- Problemas de atomicidad. Son aplicaciones encargadas de asegurar, si se produce algún fallo inesperado, esta aplicación permite que los datos se restauren al estado antes de la falla.
- Anomalías en el acceso concurrente. Consiste en aumentar el rendimiento global del sistema y obtener una respuesta más rápida, algunos sistemas permiten tener varios usuarios actualizando datos simultáneamente.
- Problemas de seguridad. Cumplir con normas de seguridad de acuerdo a las necesidades de cada usuario, las restricciones de acceso a la información dependen de cada organización de acuerdo a las políticas y normas internas de cada una de ellas.

5.4.4 Modelos de Datos.

En la estructura de las BD se encuentra el modelado de datos, que es una colección de herramientas conceptuales para describir los datos, las relaciones, la semántica y las restricciones de consistencia. Estos modelos de datos ofrecen un modo de escribir el diseño de las BD en nivel físico, lógico y de vista. Los modelos de datos se clasifican en cuatro categorías, las cuales se explican a continuación:

- Modelo Relacional. Usa una colección de tablas para representar tanto los datos como sus relaciones. Cada tabla tiene varias columnas, y cada columna tiene un nombre único. El modelo relacional es un ejemplo de un modelo basado en registros, además de ser el modelo de datos relacional más ampliamente usado.
- Modelo Entidad-Relacional (E-R). Se basa en una percepción del mundo real que consiste en una colección de objetos básicos, a los cuales se denominan entidades y a las relaciones entre ellos. La entidad es una cosa u objeto del mundo real que es distinguible de otros objetos. Este modelo es utilizado en el diseño de BD.
- Modelo de Datos Orientado a Objetos. Es considerado como una extensión del modelo E-R con los conceptos de encapsulamiento, los métodos (funciones) y la identidad de los objetos.
- Modelo de Datos Semiestructurado. Permite la especificación de datos donde los elementos de datos individuales del mismo tipo pueden tener diferentes conjuntos de atributos, esta es la diferencia de los demás modelos. El lenguaje de marcas extensible (XML, eXtensible Markup Language) es empleado para representar los datos semiestructurados.

5.4.5 Lenguaje de Manipulación de Datos.

El lenguaje de Manipulación de Datos (LMD) es el lenguaje que permite a los usuarios tener acceso a los datos organizados mediante el modelo de datos correspondiente o manipularlos. Los tipos de acceso son los siguientes:

- La recuperación de la información almacenada en la BD.
- La inserción de información nueva en la BD.
- El borrado de la información de la BD.
- La modificación de la información almacenada en la BD.

Existen fundamentalmente dos tipos:

- Los LMDs procedimentales que necesitan que el usuario especifique que datos se necesitan y como obtener esos datos.
- Los LMDs declarativos o LMDs no procedimentales, son los que necesitan que el usuario especifique los datos para así obtener esos datos.

En el Sistema de Bases de Datos se tiene que determinar un medio eficiente de acceso a los datos, para esto se requiere de una consulta. La consulta es una instrucción que solicita que se recupere la información. Por medio LMDs se recupera la información, denominándolo lenguaje de consulta o lenguaje de manipulación de datos, aunque este término sea técnicamente incorrecto. El lenguaje de consulta más ampliamente usado es el SQL.

5.4.6 Lenguaje de definición de datos.

El lenguaje de definición de datos (LDD) son los esquemas de BD que especifican el conjunto de definiciones expresadas, además de especificar más propiedades de datos. La estructura de almacenamiento y los métodos de acceso usados por los sistemas de bases de datos se especifican mediante un conjunto de instrucciones en un tipo especial de LDD al cual se denomina lenguaje de almacenamiento y definición de datos. Estas instrucciones definen los detalles de implementación de los esquemas de las BD, que están ocultas de los usuarios.
Los valores de los datos deben de satisfacer ciertas restricciones de consistencia. El LDD proporciona facilidades para especificar tales restricciones.

5.5 Modelo Entidad Relación.

El modelo Entidad Relación (E-R) lo introdujo P. P. Chen en 1976 y se ha aceptado mucho en el diseño de las BD. Ayuda al analista con tres conceptos semánticos principales: entidades, las relaciones y atributos. Los diseñadores que usan el modelo E-R en el análisis de datos deben describir la empresa en términos de:

- Entidades, que son objetos distintos en una empresa del usuario.
- Relaciones, que son interacciones significativas entre los objetos.
- Atributos, que describen las entidades y las relaciones. Cada atributo se asocia con un conjunto de valor (dominio) y puede adquirir un valor de este conjunto.

El modelo E-R es el metamodelo, entidades, relaciones y atributos son las abstracciones que proporciona el metamodelo. Estas abstracciones se utilizan para la construcción de modelos a nivel de empresa, de los sistemas del usuario y estos, a su vez conducen a modelos a nivel objeto, que muestran ocurrencias de objetos individuales Hawryszkiewycz (1994).

5.5.1 ¿Qué es el Modelo Entidad Relación?.

El modelo de datos entidad-relación está basado en una percepción del mundo real que consiste en un conjunto de objetos básicos, denominados entidades, y de las relaciones entre esos objetos. Una entidad es una cosa u objeto del mundo real que es distinguible de otros objetos. Mientras que las entidades se describen en las bases de datos mediante un conjunto de atributos. Por otro lado, la relación es una asociación entre varias entidades, el conjunto de entidades del mismo tipo, y el conjunto de todas las relaciones del mismo tipo se denominan conjunto de entidades y conjunto de relaciones respectivamente (Silberschatz, 2006).

5.5.2 Restricciones.

En el desarrollo del modelo E-R se definen ciertas restricciones a la que el contenido de la base de datos se debe adaptar.

5.5.2.1 Correspondencia de cardinalidad.

La correspondencia de cardinalidades, o la razón de cardinalidad, expresa el número de entidades a las que otra entidad se puede asociar mediante un conjunto de relaciones. La correspondencia de cardinalidad es útil para describir conjuntos de relaciones binarias, aunque pueda contribuir a la descripción de conjuntos de relaciones que impliquen más de dos conjuntos de entidades.

Para un conjunto de relaciones binarias R entre los conjuntos de entidades A y B, la correspondencia de cardinalidad es la siguiente:

- Uno a uno. Cada entidad de A se asocia, a lo sumo, con una entidad de B, y cada entidad de B se asocia, a lo sumo, con una entidad de A (ver Figura 10).
- Uno a varios. Cada entidad de A se asocia con cualquier numero (cero o más) de entidades de B. Cada entidad de B, sin embargo, se puede asociar, a lo sumo, con una entidad A (ver Figura 10).

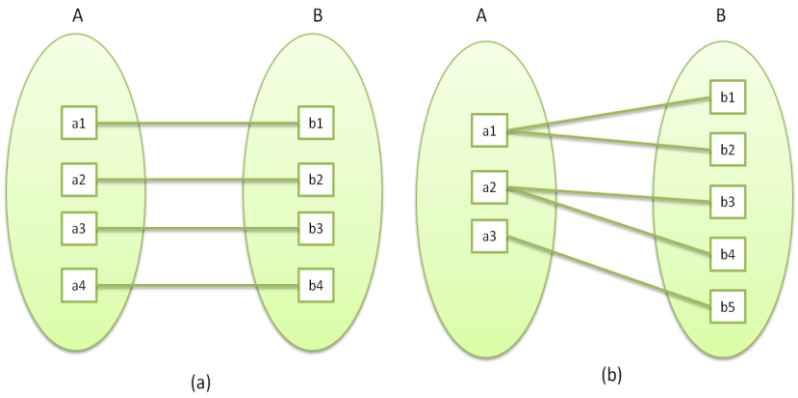

Figura 10 Correspondencia de cardinalidades. (a) Uno a uno. (b) Uno a varios. (Silberschatz, 2006).

- Varios a uno. Cada entidad de A se asocia, a lo sumo, con una entidad de B. Cada entidad de B, sin embargo, se puede asociar con cualquier numero (cero o más) de entidades de A (ver Figura 11).
- Varios a varios. Cada entidad de A se asocia con cualquier numero (cero o más) de entidades de B, y cada entidad de B se asocia con cualquier numero (cero o más) de entidades de A (ver Figura 11).

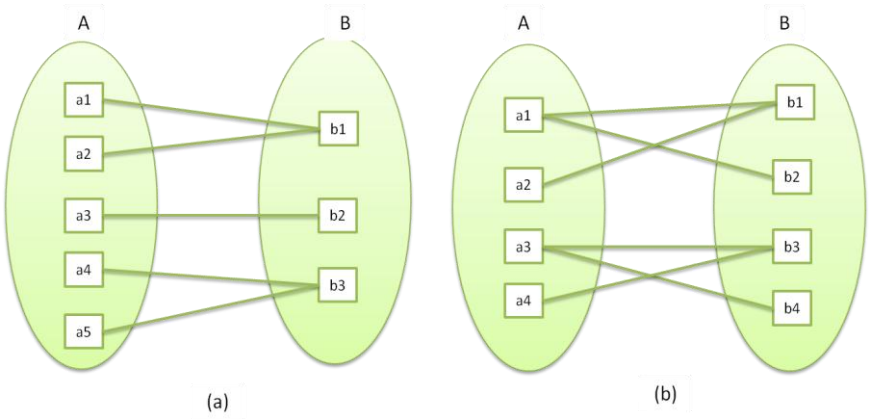

Figura 11 Correspondencia de cardinalidades. (a) Varios a uno. (b) Varios a varios.(Silberschatz, 2006).

5.5.2.2 Simbología de Cardinalidad.

No todos los programadores y analistas utilizan la misma simbología de cardinalidad. Unos utilizan simplemente números y letras para mostrar la proporcionalidad de relación y otros símbolos geográficos como se muestra en la Figura 12 (González, 1999).

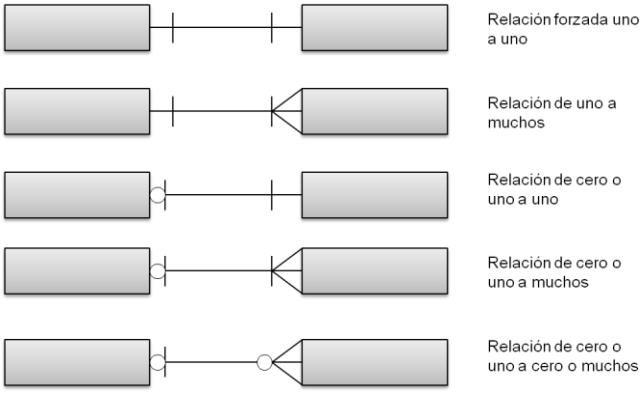

Figura 12 Simbología geométrica de cardinalidad (González, 1990).

5.5.3 Clasificación en el modelo Entidad Relación.

La clasificación del Modelo Entidad Relación se realiza por medio de jerarquías de generalización, la especificación de las restricciones de carnalidad para las jerarquías de generalización, y al mismo tiempo clasificaciones de niveles más complejos.

5.5.3.1 Jerarquía de generalización.

La jerarquía de generalización es un conjunto de tipos de entidad ordenadas en una estructura jerárquica para mostrar sus atributos similares, cada subtipo del tipo de entidad hijo contiene un subconjunto de entidades de su supertipo o tipo de entidad padre.

La herencia respalda la característica de compartir elementos entre un supertipo y sus subtipos, además de ser una característica del modelado de datos que permite compartir los atributos entre un supertipo y un subtipo. El subtipo hereda atributos de su supertipo (ver Figura 13).

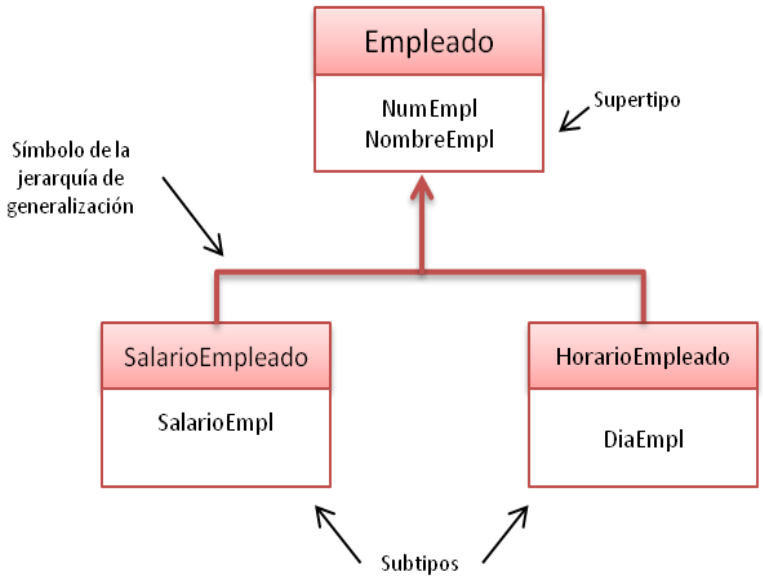

Figura 13 Jerarquía de generalización. Ejemplo de empleados (Michael, 2007).

5.5.3.2 Restricciones de separación e integridad.

Las restricciones de separación son los subtipos en una jerarquía de generalización no tienen ninguna entidad en común, y las restricciones de integridad significan que cada entidad de un supertipo debe ser una entidad en uno de los subtipos en la jerarquía de generalización (ver Figura 14).

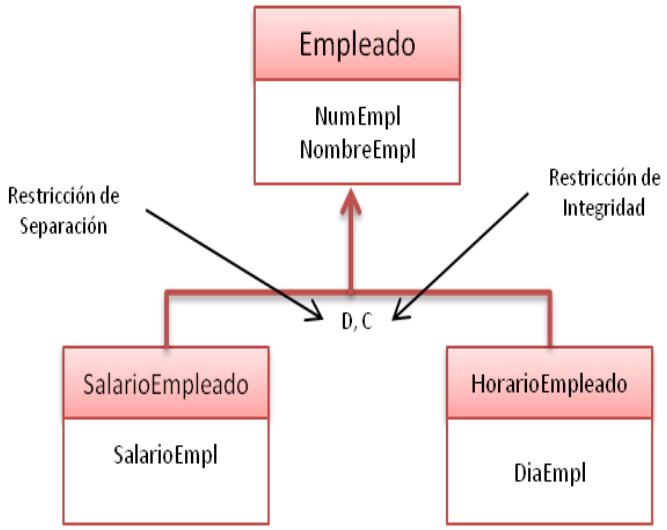

Figura 14 Restricciones de separación e integridad (Michael, 2007).

5.5.4 Reglas de Consistencia e Integridad.

Cuadro 1 Reglas de Consistencia e Integridad (Michael, 2007).

Tipo de Regla	Descripción
Integridad	1. Regla de la llave primaria: todos los tipos de entidad tienen una llave primaria (directa, prestada o heredada).
	2. Regla de denominación: todos los tipos de entidad, relaciones y atributos tienen nombre.
	3. Regla de cardinalidad: está dada para los dos tipos de entidad de una relación.
	4. Regla de participación de la entidad: todos los tipos de entidad participan en al menos una relación, excepto los de una jerarquía de generalización.

Consistencia	1. Regla de nombres de entidad: los nombres de tipo de entidad son únicos.
	2. Regla de nombre de atributo: los nombres de atributos son únicos dentro de los tipos de entidad y relaciones.
	3. Regla de nombre de atributos heredados: los nombres de los atributos de un subtipo no coinciden con los nombres de los atributos heredados (directos o indirectos).
	4. Regla de tipo de conexión relación/entidad: todas las relaciones conectan dos tipos de entidad (no necesariamente distintos).
	5. Regla de conexión relación/relación: las relaciones no se conectan con otras relaciones.
	6. Regla de entidad débil: las entidades débiles tienen al menos una relación identificable.
	7. Regla de relación identificable: para cada relación identificable, al menos uno de los tipos de entidad participante debe ser débil.
	8. Regla de cardinalidad de la identificación de dependencias: para cada relación identificable, la cardinalidad mínima y máxima debe ser 1 en el sentido del tipo de entidad hijo (entidad débil) al tipo de entidad padre.
	9. Regla de la llave foránea redundante: las llaves foráneas redundantes no se usan.

5.5.5 Diagrama de ER.

Los diagramas de E- R expresan gráficamente la estructura lógica general de las bases de datos. Los diagramas E-R son sencillos y claros. Los diagramas constan de los siguientes componentes principales:

- Rectángulos, que representan los conjuntos de entidades.
- Elipses, que representan atributos.
- Rombos, que representan conjuntos de relaciones.
- Líneas, que unen los atributos con los conjuntos de entidades y los conjuntos de entidades con los conjuntos de relaciones.
- Elipses dobles, que representan atributos multivalorados.
- Elipses discontinuas, que denotan atributos derivados.
- Líneas dobles, que indican participación total de una entidad en un conjunto de relaciones.
- Rectángulos dobles, que representan conjuntos de entidades débiles.

- Línea dirigida, si va desde el conjunto de relaciones, es una relación de uno a uno o varios a uno
- Línea no dirigida, si va desde el conjunto de relaciones, es una relación de varios a varios.

5.5.6 Diseño de Bases de Datos.

El proceso de diseño se bases de datos se puede dividir en seis etapas. El modelo E-R es muy relevante en los tres primeros pasos (ver Cuadro 2).

Cuadro 2 Diseño de Bases de Datos (Elaboración propia, 2012).

Etapa	Modelo	Nivel de abstracción	Descripción
Análisis de requisitos	Modelo Entidad-Relación	Esquema conceptual	Comprensión de los datos que se van a almacenar en la base de datos, además de averiguar las necesidades de los usuarios.
Diseño conceptual de bases de datos		Esquema conceptual	Desarrollar una descripción de alto nivel de los datos que se van a guardar en la base de datos, junto con las restricciones a los datos.
Diseño lógico de bases de datos		Esquema conceptual	Escoger un DBMS para implementar el diseño de base de datos y transformar el diseño conceptual de la base de datos en un esquema de base de datos del modelo de datos del DBMS.
Refinamiento de los esquemas		Esquema físico	Análisis del conjunto de relaciones del esquema relacional de la base de datos para identificar posibles problemas y refinarlo.
Diseño físico de bases de datos		Esquema físico	Refinamiento del diseño de la base de datos, para garantizar el cumplimiento de criterios de rendimiento.
Diseño de aplicaciones y de la seguridad		Esquema externo	Se identifican las entidades y los procesos relacionados con la aplicación, se describe el papel a desarrollar cada entidad, así como el acceso a los datos.

5.6 Modelo de Bases de Datos Relacionales.

El modelo de datos relacional fue introducido por Tedd Codd de IBM Research en 1970 mediante un articulo clásico (Coddm, 1970), y pronto acaparó gran atención debido a su simplicidad y los fundamentos matemáticos. El modelo utiliza el concepto de relación matemática (tiene una apariencia similar a una tabla de valores) como su bloque de construcción básico, y tiene sus bases teóricas en la teoría de conjuntos y la lógica de predicados de primer día.

Los modelos de datos que precedieron al modelo relacional son los modelos jerárquicos y en red. Se propusieron en los años sesenta y se implementaron en los primeros DBMS, durante los años setenta y ochenta. Debido a su importancia histórica y a la gran base de usuarios existente para los DBMS (Ramez, 2002).

5.6.1 Objetivo del Modelo de Datos Relacional.

El objetivo principal del modelo de datos relacional es facilitar que la base de datos sea percibida o vista por el usuario como una estructura lógica que consiste en un conjunto de relaciones y no como una estructura física de implementación. Esto ayuda a conseguir un alto grado de independencia de los datos.

Un objetivo adicional del modelo es conseguir que esta estructura lógica con la que se percibe la base de datos sea simple y uniforma, con el fin de proporcionar simplicidad y uniformidad, toda la información se representa de una manera: mediante valores explícitos que contienen las relaciones

5.6.2 ¿Qué es el Modelo de Datos Relacional?.

El modelo relacional representa la base de datos como una colección de relaciones. En pocas palabras cada relación a una tabla de valores o, hasta cierto punto, a un fichero plano de registros.

Si se viera una relación como una tabla de valores, cada fila de la tabla representa, la colección de valores de datos relacionales entre sí.

En el modelo relacional, cada fila de la tabla representa un hecho que normalmente se corresponde con una entidad o vinculo del mundo real. El nombre

de la tabla y los nombres de las columnas ayudan a interpretar el significado de los valores que están en cada fila.

En la terminología utilizada en el modelo relacional, una fila es denominada tupla (ver Figura 15), una cabecera de columnas es un atributo y la tabla se denomina relación. El tipo de datos que describe los tipos de valores que pueden aparecer en cada columna se llama dominio (Ramez, 2002).

5.6.3 Definiciones del Modelo de datos Relacional.

5.6.3.1 Dominio.

Un dominio D es un conjunto de valores atómicos (cada valor del dominio es indivisible en lo que concierne al modelo relacional). El método de especificación de los dominios consiste en especificar un tipo de datos al cual pertenecen los valores que constituyen el dominio, además de especificar el nombre para el dominio que ayuda a interpretar los valores, también se debe de especificar el tipo de dato o formato para cada dominio. En pocas palabras el dominio debe tener un nombre, un tipo de datos y un formato. También puede incluir información adicional para interpretar los valores de un dominio. Se dice que D es el dominio de A_i y se denota por dom (A_i) (Ramez, 2002).

5.6.3.2 Tupla.

Las filas de una relación se conocen como tuplas. Se asume que no hay un orden preestablecido de las filas o tuplas de la relación y que dos tuplas no tienen idénticas conjuntos de valores (Hansen 1998).

Cada tupla (t) es una lista ordenada de n valores $t=< V_1, V_2,\ldots, V_n>$, en donde cada valor V_1, $1 \leq i \leq n$, es un elemento de dom (A_i), o bien un valor nulo especial. El *i*-esimo valor de la tupla t, que corresponde al atributo A_i, se referencia como $t[A_i]$. También se acostumbra a usar los términos intensión de una relación para el esquema R y extensión para el estado de una relación r(R) ver Figura 20 (Ramez, 2002).

5.6.3.3 Relaciones.

Las relaciones es un término matemático y representa una simple tabla de dos dimensiones, consistentes en filas y columnas de datos (Hansen, 1998).

El esquema de relación R, denotado por R (A_1, A_2,…, A_n), se compone de un nombre de relación R y una lista de atributos, A_1, A_2,…, A_n. Cada atributo A_i es el nombre de un papel desempeñado por algún dominio D en el esquema de relación R. Un esquema de relación sirve para describir una relación; R es el nombre de la relación. El grado de una relación es el número de atributos n de su esquema de relación ver Figura 20 (Ramez, 2002).

5.6.3.4 Valor nulo.

El valor nulo es al valor dado a un atributo en una tupla si el atributo es inaplicable o su valor es desconocido que puede ser reemplazado mas tarde. Un valor nulo no es un espacio en blanco o cero.

5.6.3.5 Claves.

Hansen (1998) menciona en cualquier conjunto de atributos que identifique únicamente a cada tupla en la relación se le llama superclave. Una clave de una relación es un conjunto minimal de tales atributos, es decir una clave es una superclave minimal (unívocamente). También la tupla puede ser descrita como un determinante funcional ya que determina unívocamente el valor del atributo.

Existen varios tipos de claves las cuales se mencionan a continuación:

- Clave compuesta. Contiene más de un atributo.
- Clave candidata. Cualquier conjunto de atributos que puede ser elegido como una clave de una relación.
- Clave primaria. La clave candidata elegida como una clave de la relación
- Clave externa (ajena). Es un conjunto de atributos en una relación que constituye una clave en alguna otra (o posiblemente la misma) relación: usada para indicar enlaces lógicos entre relaciones.
- Clave externa (recursiva). Una clave que referencia su propia relación.

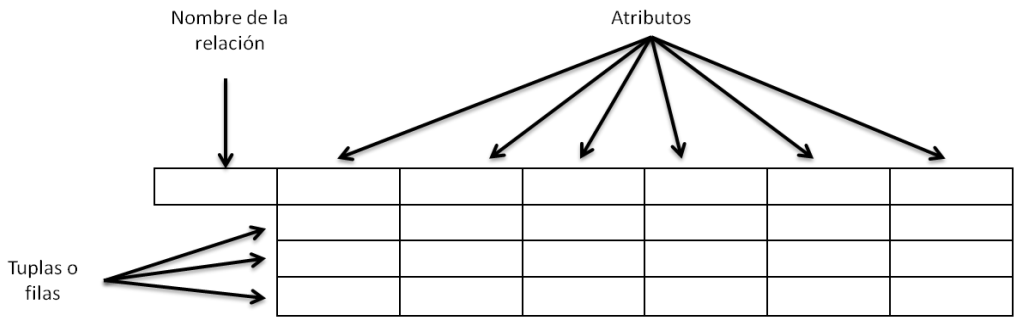

Figura 15 Los atributos y tuplas de una relación (Ramez, 2002).

5.6.4 Características de las relaciones.

- Orden de las tuplas en una relación. El ordenamiento de las tuplas no tienen un orden específico.
- Orden de los valores dentro de una tupla y definición alternativa de relación. A nivel lógico, el orden de los atributos y de sus valores no importantes en tanto se mantenga la correspondencia entre atributos y valores. La definición alternativa de relación que hace innecesario el ordenamiento de los valores de una tupla.
- Valores en las tuplas. Cada valor en una tupla es un valor atómico, es decir, no es divisible en componentes en lo respecta al modelo relacional básico
- Interpretación de una relación. El esquema de una relación puede interpretarse como un predicado, los valores de cada tupla se interpretan como valores que satisfacen el predicado. La interpretación es muy útil en el contexto de los lenguajes de programación lógica.

5.6.5 Propiedades de las relaciones.

Pons (2005) menciona de acuerdo a los conceptos definidos dentro del modelo de datos relacional, se consideran algunas propiedades las cuales son las siguientes:

- No hay orden en las tuplas
- No hay orden en los atributos
- No hay tuplas duplicadas
- El esquema de toda relación incluye una clave primaria

5.6.6 Restricciones relacionales.

5.6.6.1 Restricciones de dominio.

Las restricciones de dominio especifican que el valores de cada atributo A debe ser un valor atómico del dominio dom (A), los tipos de datos asociados a los dominios por lo regular incluyen los tipos de datos numéricos estándar de los numero enteros (entero corto o entero largo) y números reales (flotante)(Ramez, 2002).

5.6.7 Reglas de integridad

- Reglas específicas. Son aquellas que provienen de la semántica del atributo y son propias de cada base de datos concreta.

- Reglas genéricas. Se aplican a los atributos en función del papel que desempeñan en la estructura de la base de datos. Estas no son reglas sino metareglas, ya que son normas genéricas cuya aplicación en una base de datos concreta genera un conjunto de reglas de integridad. Son las siguientes (Pons, 2005):
  - Regla de integridad de la entidad. El atributo que es la clave de una fila no puede ser nulo (Hansen, 1998).

  - Regla de integridad referencial. El valor no nulo de una clave externa debe ser un valor real de la clave de otra relación (Hansen, 1998).

5.6.8 Normalización.

Mannino (2007) argumenta la normalización es el proceso de eliminación de redundancias en una tabla para que sea más fácil de modificar. Se han desarrollado un sinnúmero de formas normales para eliminar las redundancias. Una forma normal es una regla sobre las dependencias permisibles. Cada forma normal elimina cierto tipo de redundancias.

Asimismo Osorio (2008) dice la normalización es un concepto que hace referencia a las relaciones. Básicamente, el principio de normalización indica que las tablas

de las bases de datos eliminaran las incoherencias y redundancias y minimizaran la ineficacia.

En las bases de datos se describen como incoherentes cuando sus datos se introducen de forma incoherente o cuando los datos de una tabla no coinciden con los datos introducidos en otra tabla. Una base de datos ineficaz no permite aislar los datos exactos que desea. Una base de datos que almacene todos sus datos en una tabla obligan a pasar por innumerables campos simplemente para recuperar los datos. Si una base de datos se encuentra completamente normalizada almacena cada información en su propia tabla e identifica cada información con su propia clave principal.

Como conclusión del autor la normalización es una serie de reglas que involucra análisis y transformación de las estructuras de datos relacionales que exhiben propiedades únicas de consistencias, mínima redundancia y máxima estabilidad.

5.6.8.1 Reglas de Normalización.

Estas son las reglas que debe seguir el administrador de la base de datos para llevar a cabo la normalización de forma satisfactoria (Osorio, 1994):

- Regla 1: Unicidad de campo. Cada campo de una tabla debe contener un único tipo de información.
- Regla 2: Clave principal. Cada tabla de tener un único identificador, o clave principal, que este formado por uno o más campos de la tabla.
- Regla 3: Dependencias Funcional. Para cada valor único de la clave principal, los valores de las columnas de datos deben estar relacionados y debe describir completamente el contenido de la tabla.
- Regla 4: independencia de los campos. Debe ser posible realizar cambios en cualquier campo que no forme parte de la clave principal sin que para ello se vea afectado cualquier otro campo.

5.6.8.2 Primera Normalización.

La primera forma normal (1NF) prohíbe la anidación o repetición de grupos en las tablas. Una tabla que no esté en 1NF esta desnormalizada o sin normalizar. Para convertir una tabla desnormalizada en 1NF, reemplazar cada valor de un grupo repetido por una fila, en la fila nueva copiar las columnas que no se repiten (Mannino, 2007).

5.6.8.3 Segunda Normalización.

Una tabla esta en segunda normalización (2NF) si cada columna que no forma parte de la llave depende de todas las llaves candidatas, no de un subconjunto de cualquier llave candidata. Las llaves candidatas son el conjunto mínimo de la columna (s) con valores únicos en la tabla (Mannino, 2007).

5.6.8.4 Tercera Normalización.

Una tabla esta en tercera normalización (3NF) si esta en 2NF y cada columna que no forma parte de la llave depende solo de llaves candidatas, no de otras columnas que no forman parte de la llave (Mannino, 2007).

5.7 Base de Datos Heterogéneas.

El concepto de bases de datos heterogéneas surge a partir de las Bases de Datos Federadas. En 1990 Amit Sheth y James Larson propusieron una arquitectura de esquemas para los sistemas de bases de datos federadas muy general, que sirve como arquitectura de referencia. En las bases de datos federadas sirven para la nueva creación de una BD, integrándose todas las BD preexistentes realizando la integración de datos (Lozano, 2000).

5.7.1 ¿Qué es una Base de Datos Heterogénea?.

Enríquez (2010) menciona las HDBS consisten en la creación de modelos computacionales que ofrezcan una interfaz uniforme de consultas a datos recolectados y almacenados en múltiples BD. Por otro lado Pastor (2000), argumenta una base de datos distribuida heterogénea permite integrar en una sola BD global un conjunto de BD preexistentes gestionadas por diferentes Sistemas Manejadores de Bases de Datos (DBMS), ya sean relacionales o en red. Si se considera la proliferación de bases de datos que pueden cooperar entre sí, el dominio de aplicaciones potenciales de los sistemas heterogéneos es muy importante.

5.7.2 Problemática de las Bases de Datos Heterogéneas.

Uno de los principales problemas en estos sistemas es la interoperabilidad y se considera que es un punto clave para la integración de sistemas informáticos heterogéneos. En un enfoque habitual se propone la integración conceptual de los sistemas de bases de datos en un esquema global, para resolver la sintáctica y la heterogeneidad estructural.

Existen diferentes barreras en la integración de datos, pero principalmente la representación de la heterogeneidad, es decir, las diferencias en los modelos de datos, esquemas, convenciones de nomenclatura y los niveles de granularidad para representar datos que son conceptualmente similares (Enríquez, 2010).

Pastor (2000) comenta la complejidad que requiere la heterogeneidad proviene de las diferencias entre los modelos de datos que pueden soportar los DBMS locales.

El mayor problema radica en la conversión de los datos y de las consultas entre un modelo (y su lenguaje asociado) a otro. El problema es muy complejo, e incluso insoluble, debido principalmente al gran numero de conceptos presentes. Es casi imposible traducir una estructura de datos compleja, en otra estructura equivalente y compleja. La conversión de una estructura de datos o consulta de un modelo y su lenguaje, a una estructura de datos o consulta equivalente de acuerdo al modelo es realizado por un subsistema denominado traductor.

5.7.2.1 Principales conflictos de Heterogeneidad.

Enríquez (2010) menciona cuatro principales conflictos de heterogeneidad:

- Uso de modelos de datos o lenguajes de consultas diferentes; conflictos semánticos.
- Uso de terminología similar para referirse a dos conceptos semánticos diferentes, o viceversa, utilizar terminología diferente para referirse a un mismo concepto; conflictos descriptivos.
- Conflictos de nombres, alcance del dominio de los atributos, escala, limitaciones; conflictos estructurales.
- Utilizar diferentes constructores para representar las mismas entidades del mundo real.

5.7.3 Alternativas de Solución.

Hoy en día, la integración de fuentes de datos heterogéneas se ha convertido en un problema central de la informática moderna. Con las mejoras en las tecnologías de Internet, ha habido una mayor demanda sobre la integración de datos de fuentes diversas, especialmente en el comercio electrónico, donde las empresas necesitan para conectar sus sistemas en línea con sus proveedores. Se destacan la integración y consulta de datos en fuentes heterogéneas es un tema reciente en el campo de la investigación en BD, cuya finalidad es proporcionar a los usuarios un acceso más uniforme a múltiples fuentes de datos heterogéneas.

Las organizaciones utilizan diferentes DBMS por lo que buscan maneras de solucionar los problemas de integridad de datos (Enríquez, 2010) y algunas soluciones se mencionaran a continuación.

5.7.3.1 Propuestas de Software.

En el ambiente web la recuperación de información utiliza *wrappers*, son considerados como software que aceptan las consultas de usuarios de datos en la Web para así extraer la información relevante y retornar los resultados (Enríquez, 2010).

5.7.3.2 Propuestas de Arquitectura.

Otra forma es, una arquitectura de tres capas, siendo la primera el nivel de datos que administra las fuentes de datos, el segundo nivel es de administración que se encarga de llevar a cabo la integración de las fuentes de datos y por último el nivel de usuario que proporciona al usuario una interfaz grafica para obtener una sola representación virtual de los datos, se requieren agentes especializados para realizar funciones predefinidas que residen en los diferentes niveles (Enríquez, 2010).

5.7.3.3 Propuestas de Herramientas.

Existen varias herramientas que abordan el problema para la consulta de HDBS, que en primera instancia, es necesario conectarse a las diferentes bases de datos, identificar las estructuras para poder en un principio interactuar con todos los datos; este trabajo lo pueden realizar agentes computacionales. Según la *Foundation for Intelligent Physical* Agents (FIPA) un agente; “es una entidad que reside en entornos donde interpreta datos que reflejan eventos y ejecuta comandos que producen efectos en ese entorno”. Una definición más reciente, indica que es un sistema de hardware y/o software autónomo (independiente del usuario), que interactúa con su entorno (u otros agentes o humanos), guiado por uno o varios propósitos, su comportamiento es proactivo (reacciona a eventos y a veces se anticipa haciendo propuestas), adaptable (puede enfrentar situaciones

novedosas), sociable (se comunica, coopera y/o negocia), y su comportamiento es predecible en cierto contexto. Por otro lado se considera que, XML y los servicios web se han convertido en medios estandarizados para la publicación y la integración de datos en la Web, de manera complementaria Active XML (AXML), proporciona un marco de trabajo para la integración de datos y servicios punto a punto. Este trabajo plantea una alternativa en donde se puedan obtener datos de los manejadores de bases de datos Oracle, MySql, SQL Server y PostgreSQL, con el fin de realizar consultas genéricas en fuentes de datos heterogéneas basadas en el modelo de datos relacional, auxiliándose del uso de agentes computacionales.

La consulta de bases de datos heterogéneas por medio de agentes en los últimos años, ha sido atendida por varios investigadores. Los cuales han propuesto metodologías orientadas a agentes de software (AOSE). Una de ellas es Tropos, que abarca el proceso de desarrollo de software, y está basado en dos ideas clave; el concepto de agente y las nociones relacionadas con el. Así como las fases tempranas de análisis de requerimientos, que permite una comprensión profunda del entorno en donde el software debe funcionar, y el tipo de interacciones que deben tener lugar entre el software y agentes humanos. Esta metodología abarca cuatro fases: *Early requirements, Late requirements, Architectural design y Detailed design*. Otra aportación es *MaSE (Multi-agent systems Software Engineering)*, que se concibe como una abstracción del paradigma orientado a objetos, donde los agentes son especializaciones de objetos. En lugar de simples objetos, con métodos que pueden invocarse desde otros objetos, los agentes se coordinan unos con otros vía conversaciones y actúan proactivamente para alcanzar metas individuales y del sistema. También se encuentra la PROMETHEUS, que emplea la metodología de ingeniería de software para el diseño de agentes, así mismo, apoya el desarrollo de agentes inteligentes que utilizan metas, creencias, planes y eventos. Utilizando un proceso iterativo, consta de tres fases: Especificaciones del Sistema, Diseño de alto nivel o diseño arquitectural y Diseño detallado. Otra herramienta es ZEUS, que se basa en una metodología Multi – Agentes, que desde su aparición se ha convertido en

referencia para el desarrollo de SMA (Sistemas Multi – Agentes). Combina los distintos resultados de investigación en agentes (planificación, ontologías, asignación de responsabilidades, relaciones sociales entre agentes) en un sistema completamente funcional. Propone un desarrollo en cuatro etapas: análisis del dominio, diseño de los agentes, realización de los agentes y soporte en tiempo de ejecución. Por último, *DENODO Coorporation*, desarrollo un sistema mediador que integra datos estructurados y semi-estructurado. La plataforma sigue una arquitectura de mediación compuesta por la capa física que contiene envoltorios para diferentes fuentes de datos: relacionales, hojas de cálculo, documentos planos, entre otros, los contenedores son generados automáticamente por la herramienta, la capa lógica, es el núcleo de la plataforma y contiene el planificador de consultas y motor de ejecución, contiene una herramienta para administrar diccionarios de datos y modulo cache utilizando vistas materializadas. Además evita las consultas cuando estas pueden ser resueltas utilizando la cache, y cifrar datos cuando sea necesario (Enríquez, 2010).

5.8 SQL.

SQL es el lenguaje estándar para trabajar con bases de datos relacionales y es soportado prácticamente por todos los productos en el mercado. Originalmente, SQL fue desarrollado en IBM Research a principios de los años setenta, fue implementado por primera vez a gran escala en un prototipo de IBM llamado System R, y posteriormente en numerosos productos comerciales de IBM y de muchos otros fabricantes, el nombre oficial es Estándar Internacional del Lenguaje de Bases de Datos SQL en 1992 (Date, 2001).

5.8.1 ¿Qué es SQL?

Lenguaje de Consulta Estructurado o Structured Query Language (SQL) es un lenguaje de programación diseñado específicamente para el acceso a sistemas de administración a bases de datos relacionales (DBMSR). Actualmente los sistemas son de este tipo utilizando el lenguaje SQL, se puede decir sin ninguna duda, que este lenguaje es empleado mayoritariamente a sistemas existentes hoy en día e indiscutiblemente no tiene rival alguno. Este lenguaje es empleado en sistemas informáticos que van desde ordenadores personales muy básicos con apenas 64 MB de espacio en memoria central hasta los más potentes multiprocesadores y multicomputadores con decenas de procesadores superescalares de 64 bits.

El lenguaje SQL es un lenguaje de cuarta generación. Es decir, en este lenguaje se indica que información se desea obtener o procesar, pero no como se debe hacer. Es labor interna del sistema elegir la forma más eficiente de llevar a cabo la operación ordenada por el usuario.

5.8.2 Partes de SQL.

Aunque SQL es considerado como un sublenguaje por su naturaleza no procesa datos, no obstante es un lenguaje completo que permite crear, mantener, obtener y manipular objetos en BD. Un método común utilizado para clasificar las sentencias SQL es dividirlos de acuerdo a las funciones que realizan. Con base en este método, SQL se pueden separar en tres tipos de declaraciones: Lenguaje de definición de datos, Lenguaje de manipulación de datos y Lenguaje de control de datos (Oppel, 2008).

5.8.2.1 Lenguaje de definición de datos.

Lenguaje de definición de datos (en ingles Data Definition Language o DDL). Incluye aquellas sentencias que sirven para definir los datos o para modificar su definición, como por ejemplo la creación de tablas, índices, etc.

El lenguaje de definición de datos (DDL) permite trabajar con las estructuras de datos en vez de trabajar con los datos en sí mismos. Se manejan las instrucciones: CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX. DROP INDEX.

5.8.2.1.1 Instrucción CREATE.

León (1999) menciona, la forma general de la instrucción CREATE es la siguiente:

```
CREATE TABLE base-table-name
(columna1- definición,
[columna2- definición],
[columna-n- definición],
[llave-primaria- definición]);
```

Donde la columna - definición toma la forma de:

Columna- nombre data-type [NULL | NOT NULL [WITH DEFAULT | UNIQUE]]

Las palabras clave NULL y NOT NULL son opcionales. El valor por defecto de SQL/ 2 es permitir valores nulos, pero no, en todas las implementaciones.

Si se especifica la opción NULL, el DBMS inserta el valor NULL en la columna, si el usuario no especifica un valor.

Si se especifica el NOT NULL, la columna debe tener un valor. Si no se especifica el valor de la columna, el sistema rechaza la entrada y devolvera un mensaje de error.

Cuando se utiliza la opción NOT NULL, puede especificar la opción WITH DEFAULT o única opción. Si se especifica la opción WITH NO NULL DEFAULT, entonces el DBMS va a sustituir los valores por defecto (por ejemplo, 0 para los tipos de datos numéricos, espacios para datos de tipo carácter y así

sucesivamente). Si no es NULL se especifica UNIQUE, el DBMS asegura los valores de las columnas además de ser únicos, y no hay duplicados.

Ejemplo de sintaxis de CREATE TABLE:

```
CREATE TABLE libro
(id_clave     CHAR(10)    NOT NULL,
titulo        CHAR(30)    NOT NULL WITH DEFAULT,
autor         CHAR(30)    NOT NULL WITH DEFAULT,
editorial     CHAR(30)    NOT NULL WITH DEFAULT,
año           INTEGER     NOT NULL WITH DEFAULT,
PRIMARY KEY (id_clave));
```

La definición del tipo de dato, se tiene que especificar ver Cuadro 3.

Cuadro 3 Tipo de datos predefinidos (Heurtel, 2009).

Tipos de datos predefinidos	
Tipo de datos	Descripción
CHAR (longitud)	Cadenas de caracteres de longitud fija.
CHAR VARYING (longitud)	Cadena de caracteres de longitud variable.
BIT (longitud)	Cadena de bits de longitud fija.
BIT VARYING (longitud)	Cadena de bits de longitud variables.
NUMERIC (precisión, escala)	Número decimales con tantos dígitos como indique la precisión y tantos decimales como indique la escala.
DECIMAL (precisión, escala)	Número decimales con tantos dígitos como indique la precisión y tantos decimales como indique la escala.
INTEGER	Números enteros.
SMALLINT	Números enteros pequeños.
REAL	Números con coma flotante con precisión predefinida.
FLOAT (precisión)	Números con coma flotante con la precisión especifica.
DOUBLE PRECISION	Números con coma flotante con mas precisión predefinida que la del tipo REAL.
DATE	Fechas, están compuestas de: YEAR año, MONTH mes, DAY días.
TIME	Horas, están compuestas de HOUR hora, MINUT minutos, SECOND segundos.
TIMESTAMP	Fechas y horas, están compuestas de YEAR año, MONTH mes, DAY día, HOUR hora, MINUT minutos, SECOND segundos.

5.8.2.2 Lenguaje de manipulación de datos.

Lenguaje de manipulación de datos (en ingles *Data Manipulation Language o DML*). Incluye aquellas sentencias que sirven para manipular o procesar los datos, como por ejemplo la inserción, borrado, modificación o actualización de datos en las tablas.

El lenguaje de manipulación se concentra en las instrucciones SELECT, INSERT, DELETE y UPDATE, permiten trabajar con los registros o datos de las tablas.

5.8.2.2.1 Instrucción SELECT.

La instrucción SELECT lanza consultas, determinando cuales son los campos (columnas) que queremos leer y su organización.

La sintaxis general es:

```
SELECT <columnas>
FROM <tablas>
[WHERE <condiciones>]
[GROUP BY <columnas>]
[HAVING <condiciones>]
[ORDER BY <columnas>];
```

Solo son obligatorias las clausulas que no están entre corchetes. Por ejemplo para obtener el nombre y la dirección de todos los registros (o filas) de una tabla de clientes, la instrucción de consulta es:

```
SELECT nombre, dirección FROM clientes;
```

Los operadores de la condición WHERE se muestran en el Cuadro 4.

Cuadro 4 Operadores de condición WHERE (Heurtel, 2009).

Operador	Descripción
=	Igualdad
>	Estrictamente superior
>=	Superior o igual
<	Estrictamente inferior
<=	Inferior o igual
< > o !=	Diferente
BETWEEN, min, AND, máx	Superior o igual a min o igual a máx
IN (valor,..)	Igualdad con cualquier elemento de una lista
IS NULL, IS NOT NULL	Comprueba si una expresión es NULL o no
LIKE	Correspondencia con relación a un modelo

5.8.2.2.2 nstrucción INSERT.

La sentencia INSERT permite añadir filas en una tabla.

Sintaxis:

INSERT [IGNORE]

[INTO] nombre_tabla [(nombre_columna, ….)]

VALUES ({expresión | DEFAULT},…), (…),…

[ON DUPLICATE KEY UPDATE nombre_columna= expresión,..]

Las columnas afectadas por la inserción se especifican, mediante una lista de nombres de columnas tras el nombre de la tabla. En esta sintaxis, si la lista de las columnas está ausente, la sentencia afecta a todas las columnas de la tabla de manera predeterminada. Hay que precisar que no es obligatorio insertar un valor en todas las columnas de la tabla. Los valores de las columnas se especifican, mediante la clausula VALUES, debe de incluir una expresión para cada columna mencionada en la lista de columnas (en el orden correspondiente) (Heurtel, 2009).

Ejemplo de sintaxis:

INSERT INTO libro

VALUES(5,´Manual SQL´, ´Juan Lopez´,´McGraw Hil´, 2000);

5.8.2.2.3 Instrucción DELETE.

La sentencia DELEGATE permite eliminar la fila siempre y cuando la fila cumpla condición del WHERE (León, 1999).

Sintaxis:

```
DELETE
FROM nombre_tabla
WHERE nombre_columna = dato
```

Ejemplo de la sentencia:

```
DELETE
FROM libro
WHERE autor =´Juan Lopez´;
```

5.8.2.2.4 Instrucción UPDATE

La sentencia SQL UPDATE permite modificar filas en una tabla (Heurtel 2009):

```
UPDATE nombre_tabla
SET nombre_columna = {expresión| DEFAULT} [,…]
[WHERE condicion]
[ORDER BY orden]
[LIMIT numero_de_filas];
```

Ejemplo de la sintaxis de la sentencia UPDATE:

```
UPDATE libro SET nivel=´avanzado´ WHERE id_coleccion=2;
```

5.8.2.3 Lenguaje de control de datos.

Lenguaje de control de datos (en ingles *Data Control Language*). Son los permisos de acceso a determinadas tablas por parte de determinados usuarios; política de seguridad y privacidad de los datos: organización de grupos de usuarios, etc. (González, 1999).

5.9 Lenguajes Manejadores de Bases de Datos.

5.9.1 MySQL.

MySQL es el más popular del mundo, y algunos podrían decir el mejor, además de código abierto para el manejo de base de datos. MySQL, es un competidor viable contra Oracle y Microsoft SQL Server.

MySQL fue creado y es mantenido por MySQLAB, una compañía con sede en Suecia. MySQL es un sistema de gestión de bases de datos (DBMS) para bases de datos relacionales (por lo tanto, MySQL es un RDBMS). Una base de datos es simplemente una colección de (a menudo relacionados entre sí) datos, ya sea texto, números, o archivos binarios almacenados organizados por el DBMS. Técnicamente MySQL es una aplicación que gestiona los archivos de base de datos, pero el término de base de datos se aplica por igual a los archivos de datos y el propio programa (Ullman, 2006).

5.9.1.1 Estructura Básica.

5.9.1.1.1 Instrucción CREATE.

La instrucción CREATE TABLE permite crear una tabla con el nombre dado.

Sintaxis básica:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nombre_tabla
[(crear_archivos_definicion,...)]
[tabla_opciones] [sentencia_select]
```

Ejemplo de la sintaxis:

```
CREATE TABLE clientes (nombre CHAR (20), apellidopat CHAR (20),
apellidomat CHAR (20), nombre_empresa CHAR (20));
```

5.9.1.1.2 Instrucción SELECT.

La instrucción SELECT se usa para recibir registros seleccionados desde una o más tablas.

Sintaxis:

```
SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE]
[SQL_CALC_FOUND_ROWS] expresion_select, ...
[INTO OUTFILE 'nombre_archivo' optiones_exportacion
| INTO DUMPFILE "nombre_archivo]
[FROM nombre_tabla
[WHERE condiciones_where]
[GROUP BY {nombre_columna | expr | position}
[ASC | DESC], ... [WITH ROLLUP]]
[HAVING condiciones_where]
[ORDER BY {nombre_columna | expr | position}
[ASC | DESC] , ...]
[LIMIT {[offset,] numero_filas| numero_filas OFFSET offset}]
[PROCEDURE nombre_procedimiento (lista_agumentos)]
[FOR UPDATE | LOCK IN SHARE MODE]]
```

Ejemplo de la sintaxis:

```
SELECT nombre, apellidopat, nombre_empresa
FROM clients
ORDER BY nombre DESC;
```

5.9.1.1.3 Instrucción INSERT.

La instrucción INSERT inserta nuevos registros en una tabla existente, los nuevos registros están basados en los valores explícitamente especificados.

Sintaxis:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] nombre_tabla[(nombre_columna,...)]
VALUES ({expresion  | DEFAULT},...),(...),...
[ ON DUPLICATE KEY UPDATE columna_nombre=expresion, ... ]
```

Ejemplo de la sintaxis:

```
INSERT INTO artículos_oficina (nombre, precio)
VALUES (escritorio, 3000.00);
```

5.9.1.1.4 Instrucción DELETE.

La instrucción DELETE borra los registros de una tabla siempre y cuando la condición del WHERE se cumpla.

Sintaxis:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM nombre_tabla
[WHERE condicion]
[ORDER BY ...]
[LIMITnumero_filas]
```

Ejemplo de sintaxis:

```
DELETE  FROM  clientes
WHERE nombre_empresa = ´Gas´;
```

5.9.1.1.5 Instrucción UPDATE.

La instrucción UPDATE actualiza las filas de registros de tabla existentes con nuevos valores.

Sintaxis:

```
UPDATE nombre_tabla
SET nombre_columna=expresion
[WHEREcondicion_where]
[ORDER BY ...]
[LIMITnumero_filas]
```

Ejemplo de sintaxis:

```
UPDATE artículos_oficina
SET nombre =´escritorio grande´
WHERE precio = ´3000.00´;
```

5.9.2 SQL Server.

SQL Server de Microsoft es un sistema de gestión de bases de datos relacionales que se instalan tanto en computadoras portátiles o sobremesa, así como en servidores corporativos o dispositivos de bolsillo, como en los PocketPC y los lectores de códigos de barras, con una versión basada en el sistema operativo PocketPC. SQL Server se desarrollo originalmente en los años ochenta del siglo

veinte en SyBase para sistemas UNIX y posteriormente Microsoft lo tradujo para sistemas Windows NT (Silberschatz, 2006).

En lo referente a la programación SQL Server propone un conjunto de herramientas y de funcionalidades que permiten codificar con mayor rapidez. SQL Server permite reducir la distancia existente entre los aspectos de programación y de administración de la base de datos y del servidor. Además de mejorar el rendimiento también permite gestionar todos los datos presentados en la empresa y con los cuales trabajan habitualmente los usuarios (Gabillaud, 2008).

5.9.2.1 Estructura Básica.

Puede utilizar Transact-SQL para comunicarse directamente con SQL Server. Transact-SQL le permite trabajar con los datos, crear y manipular objetos de las bases de datos, interactuar con procedimientos de almacenamiento que se compilan como declaraciones SQL estáticas. Cuatro son las bases de Transact-SQL que forman el corazón del lenguaje SQL:

- La declaración SELECT se usa para recuperar datos ya existentes.
- La declaración UPDATE se utiliza para editar datos ya existentes.
- La declaración INSERT se utiliza para crear datos nuevos.
- La declaración DELETE se utiliza para eliminar datos.

5.9.2.1.1 Instrucción SELECT.

La instrucción SELECT recupera filas y columnas de una tabla y muestra los resultados solicitados. También se puede utilizar para unir tablas o recuperar una serie de columnas de una o más tablas.

Sintaxis Básica:

```
SELECT
[ALL | DISTINCT]
FROM
[tabla_o_vista]
```

WHERE

[condiciones_de_busqueda]

Ejemplo de sintaxis:

SELECT Nombre, ApellidoPat , ApellidoMat, Edad

FROM estudiantes

WHERE edad > 18

Condiciones de WHERE:

Cuadro 5 Rango y listas (Gunderloy, 1999).

Condición de búsqueda	Ejemplo de la clausula WHERE
BETWEEN	WHERE edad BETWEEN 18 AND 20
NOT BETWEEN	WHERE edad NOT BETWEEN 18 AND 20
IN	WHERE ApellidoPat IN (´Arcos´, ´Guerrero´)
NOT IN	WHERE ApellidoPat NOT IN (´Arcos´, ´Guerrero´)
LIKE	WHERE nombre LIKE ´a%´
NOT LIKE	WHERE nombre NOT LIKE ´a%´
IS NULL	WHERE nombre IS NULL
IS NOT NULL	WHERE nombre IS NOT NULL
AND	WHERE nombre LIKE ´a%´ AND nombre LIKE ´f%´
OR	WHERE nombre LIKE ´a%´ OR nombre LIKE ´f%´

5.9.2.1.2 Instrucción INSERT.

Para realizar la inserción de datos en una fila nueva en una tabla se utiliza la petición INSERT, es una herramienta que se usa para mover datos de una declaración Transact-SQL a la tabla.

Sintaxis Básica:

INSERT [INTO]

{[base_datos.] [propietario.] nombre_tabla | nombre_vista}

[(lista_columna)]

{

DEFAUL VALUES

| VALUES({DEFAUL | expresion_constante } [,…n])

|declaración_seleccion

|declarcion_ejecucion

}

La opción INTO no añade ninguna función solo específica el tipo de petición.

Ejemplo de la sintaxis:

INSERT estudiantes

VALUES (´María´, ´Hernández´, ´Hernández´, ´19´)

5.9.2.1.3 Instrucción DELETE.

SQL Server dispone de las declaraciones Transact-SQL que son DELETE y TRUNCATE TABLE las cuales permite eliminar datos y tablas de las bases de datos respectivamente.

Sintaxis básica:

DELETE [FROM] {[datase.] [owner.] nombre_tabla | nombre_vista}

[FROM

{tabla_o_vista

|tabla_vista CROSS JOIN tabla_vista

tabla_vista JOIN tabla_vista

ON condiciones_busqueda

} [,…n]]

[WHERE condiciones_busqueda]

Ejemplo de la sintaxis:

DELETE FROM estudiantes

WHERE nombre= ´María´

Sintaxis básica de TRUNCATE TABLE:

TRUNCATE TABLE nombre_tabla

En la instrucción TRUNCATE TABLE se eliminan todas las filas de la tabla, además de deshacer la distribución de los datos que se encuentran almacenadas.

Ejemplo de la sintaxis:

```
TRUNCATE TABLE estudiantes
```

5.9.2.1.4 Instrucción UPDATE.

La sintaxis de la instrucción UPDATE es parecida a INSERT o DELETE.

Sintaxis básica:

```
UPDATE [FROM] {[datase.] [owner.] nombre_tabla | nombre_vista}
SET
{columna = {expresión | DEFAULT}
| @variable = expresion} [,…n]
[FROM
{ tabla_o_vista
| tabla_o_vistaCROSS JOIN tabla_o_vista
| tabla_o_vista JOIN tabla_o_vista
ON condiciones_busqueda
}[,..n] ]
```

Ejemplo de la sintaxis:

```
UPDATE estudiantes
SET nombre = ´Maria Luz´
WHERE edad = ´20´;
```

5.9.3 PostgreSQL.

PostgreSQL es una base de datos relacional con una larga historia. A finales de 1970 la Universidad de California en Berkeley, comenzó el desarrollo de los antepasados de PostgreSQL, una base de datos relacional conocida como Ingres. Relational Technologiesse convirtió en un Producto comercial. Más tarde fue adquirida por Computer Associates. Alrededor de 1986, Michael Stonebraker de la Universidad de Berkeley llevó a un equipo que añade características orientadas a

objetos a la base de Ingre, la nueva versión se conoce como PostgreSQL (Douglas, 2003).

Momjian (2001) menciona “PostgreSQL es sin duda el más potente sistema de código abierto de base de datos relacional, además de cumplir con los estándares para ser usados en empresas u organizaciones”.

5.9.3.1 Estructura básica.

5.9.3.1.1 Instrucción SELECT.

La instrucción SELECT recupera los registros desde una tabla o vista.
Sintaxis:

```
SELECT  [ ALL | DISTINCT [ ON ( expresion [, ...] ) ] ]
expression [ AS nombre ] [, ...]
[ INTO [ TEMPORARY | TEMP ] [ TABLE ] nueva_tabla ]
[ FROM tabla [ alias ] [, ...] ]
[ WHERE condición]
[ GROUP BY columna [, ...] ]
[ HAVING condición [, ...] ]
[ { UNION [ ALL ] | INTERSECT | EXCEPT } select ]
[ ORDER BY columna [ ASC | DESC | USING operator ] [, ...] ]
[ FOR UPDATE [ OF class_name [, ...] ] ]
LIMIT { count | ALL } [ { OFFSET | , } start ]
```

Ejemplo de la sintaxis:

```
SELECT DISTINCT ON (Nombre) Nombre, ApellidoPat, Edad
FROM Estudiantes
ORDER BY Edad DESC;
```

5.9.3.1.2 Instrucción INSERT.

La instrucción INSERT, inserta filas nuevas en una nueva tabla.

Sintaxis:

```
INSERT INTO tabla [ ( columna [, ...] ) ]
    { VALUES ( expresión [, ...] ) | SELECT query }
```

El query es una consulta valida, de la instrucción SELECT.

Ejemplo de la sintaxis:

```
INSERT INTO Estudiantes
VALUES (´Juan´, ´´Ramos´, ´Ramos´, ´23´, ´Aprobado´, ´9.0´);
```

5.9.3.1.3 Instrucción DELETE.

La instrucción DELETE, permite borrar filas de una tabla.

Sintaxis:

```
DELETE FROM tabla [ WHERE condición ]
```

Ejemplo de la sintaxis:

```
DELETE FROM Estudiantes  WHERE  nombre = ´Juan´;
```

5.9.3.1.4 Instrucción UPDATE.

La instrucción UPDATE substituye valores de columnas en una tabla.

Sintaxis:

```
UPDATE tabla SET columna = expresión [, ...]
[ FROM lista ]
[ WHERE condición ]
```

Ejemplo de la sintaxis:

```
UPDATE Estudiantes
SET Status = ´Aprobado´
WHERE Promedio >= ´6.0´;
```

5.9.4 Oracle.

Cuando Oracle se fundó en 1977 como Software Development Laboratories por Larry Ellison, Bob Miner y Ed Oates no había productos de bases de datos relacionales comerciales. La compañía, cuyo nombre cambio posteriormente a Oracle, se estableció para construir un sistema de gestión de bases de datos como producto comercial y fue la primera en lanzarlo al mercado. Oracle ofrece herramientas de inteligencia, de negocio, de consulta y análisis, productos de minería de datos y un servidor de aplicaciones con una integración con el servidor de bases de datos (Silberschatz, 2006).

También el conjunto de herramientas de las tecnologías desarrolladas por *Oracle Corporation*, que resulta de gran utilidad para los profesionales de sistemas, tanto los que estén vinculados a la tecnología de Oracle como los que no han entrado en contacto con ella. Además de privilegiar las BD, pilar fundamental de esta tecnología; a las herramientas *Developer*, para la construcción de aplicaciones para Internet y Cliente/Servidor; *Designer*, para desarrollos con altos estándares de calidad; y *Discoverer*, en la implementación de *Datawarehousing* y *DataMarts*, y de la arquitectura Internet con Oracle Portal, y la construcción de sitios para la Web; JDeveloper, para desarrollos de aplicaciones en JAVA; e IAS (Internet Application Server), servidor de Internet para *e-business* (López 2001).

5.9.4.1 Estructura Básica.

5.9.4.1.1 Instrucción CREATE.

Con la instrucción CREATE TABLE permite crear nuevas tablas dentro de una base de datos.

Sintaxis:

```
CREATE TABLE [esquema.] nombre_Tabla
(nombreDeLaColumna1 tipo_Datos [, ...]);
```

Ejemplo de la sintaxis:

CREATE TABLE proveedores (nombre varchar2 (25));

5.9.4.1.2 Instrucción SELECT.

La instrucción SELECT, permite obtener datos de ciertas columnas de una tabla (proyección), obtener registros (filas) de una tabla de acuerdo con ciertos criterios (selección) y mezclar datos de tablas diferentes (asociación, join).

Sintaxis:

SELECT * | {[DISTINCT] columna | expresión [[AS] alias], ...}
FROM tabla;

Ejemplo de la sintaxis:

SELECT nombre, empresa FROM proveedores;

5.9.4.1.3 Instrucción INSERT.

Para la inserción de datos es la instrucción INSERT.

Sintaxis básica:

INSERT INTO tabla [(columna1 [, columna2...])]
VALUES (valor1 [,valor2]); Ejemplo de la sintaxis:

Ejemplo de la sintaxis:

INSERT INTO proveedores (nombre, empresa)
VALUES (´Ramón LP´,´Gas´);

5.9.4.1.4 Instrucción DELETE.

Se utiliza esta instrucción para borrar los registros (fila) de la tabla.

Sintaxis:

```
DELETE [FROM] tabla
[WHERE condición]
```

Ejemplo de la sintaxis:

```
DELETE FROM empleados
WHERE sección =23;
```

5.9.4.1.5 Instrucción UPDATE.

La modificación de los datos de los registros lo implementa la instrucción UPDATE.

Sintaxis:

```
UPDATE tabla
SET columna1=valor1 [,columna2=valor2...]
[WHERE condición]
```

Ejemplo de la sintaxis:

```
UPDATE clientes SET provincia =´Oeste´
WHERE provincia =´Sur´;
```

5.9.5 Características generales de los Lenguajes Manejadores de Bases de Datos.

Cuadro 6 Reglas o Normas específicas de la plataforma los identificadores objeto regular (no incluye identificadores entre comillas) (Kline, 2008).

Características	Plataforma	Especificaciones
Identificador puede contener	MySQL	Cualquier número, carácter o símbolo. No puede ser compuesto en su totalidad de los números.
	Oracle	Cualquier número o carácter, y el subrayado (_), signo de almohadilla (#), y el signo de dólar ($) símbolos (aunque los dos últimos son en absoluto). Enlaces de base de datos también puede contener un punto (.).
	PostgreSQL	Cualquier número o carácter, y el subrayado (_) símbolo.
	SQLServer	Cualquier número o carácter, y el guión bajo (_), a cantar (@), la libra cantar (#), y el dólar símbolos cantar ($).
Identificador debe comenzar con	MySQL	Una letra o número. no puede ser compuesto en su totalidad de los números.
	Oracle	Una letra
	PostgreSQL	Una letra o un guión bajo (_).
	SQL Server	Una letra, guión bajo (_), arroba (@), signo de ot almohadilla (#).
Identificador no puede contener	MySQL	Punto (.), barra (/), o de cualquier ASCII (0) o ASCII (255) carácter. Las comillas simples (") sólo se permiten en los identificadores de cotización. Identificadores no debe terminar con los caracteres de espacio.
	Oracle	Espacios, comillas dobles (""), o caracteres especiales.
	PostgreSQL	Comillas dobles ("").
	SQL Server	Espacios o caracteres especiales.
Permite identificadores entre comillas	MySQL	Si
	Oracle	Si
	PostgreSQL	Si
	SQL Server	Si
Identificador puede ser reservada	MySQL	No, a menos que como un identificador entre comillas
	Oracle	No, a menos que como un identificador entre comillas
	PostgreSQL	No, a menos que como un identificador entre comillas
	SQL Server	No, a menos que como un identificador entre comillas
Esquema de direccionamiento	MySQL	Database.object.
	Oracle	Schema.object.
	PostgreSQL	Database.schema.object.
	SQL Server	Server.database.schema.object.
Identificador debe ser único	MySQL	Si
	Oracle	Si
	PostgreSQL	Si
	SQL Server	Si
Entre mayúsculas y minúsculas	MySQL	Sólo si sistema de ficheros subyacente es sensible a mayúsculas (por ejemplo, Mac OS o Unix). Triggrts, grupos de archivo de registro, y los espacios de tablas son siempre mayúsculas y minúsculas.
	Oracle	No por defecto, pero se puede cambiar.
	PostgreSQL	No.
	SQL Server	No por defecto, pero se puede cambiar.
Otras reglas	MySQL	No puede contener sólo números
	Oracle	Enlaces de bases de datos están limitados a 128 bytes, y no puede ser citado identificadores
	PostgreSQL	Ninguno
	SQL Server	Microsoft suele utilizar entre paréntesis en vez de comillas dobles para los identificadores entre comillas

VI. Metodología

6.1 Pasos Metodológicos.

Paso 1. Determinar las diferencias que hay entre las estructuras de la escritura de las consultas SQL de los manejadores de las bases de datos Oracle, PostgreSQL, SQLServer y MySQL.

Paso 2: Analizar las diferencias entre las estructuras de consultas para cada Estructura del lenguaje SQL de cada manejador de bases de datos.

Paso 3: Diseñar la aplicación mediante casos de uso.

Paso 4. Plantear los algoritmos que permitan una estandarización y automatización de consultas de BD para cada Lenguaje Manejador de BD y consultas a HDBS.

Paso 5. Programar los algoritmos planteados en JAVA.

Paso 6. Implementar la aplicación con las rutinas de JAVA.

Paso 7. Validación del Sistema Manejador de Bases de Datos.

6.1.1 Desarrollar de Pasos Metodológicos.

6.1.1.1 Paso 1. Determinar las diferencias que hay entre las estructuras de la escritura de las consultas SQL en los manejadores de las bases de datos Oracle, PostgreSQL, SQLServer y MySQL.

Para la integración de consultas a Bases de Datos Heterogéneas se realizo un análisis de las estructuras de consultas de cada manejador, permitiendo visualizar

la problemática de integración e interoperabilidad al momento de realizar la consulta.

Por medio de los manuales de usuario se obtuvieron las estructuras de las consultas más utilizadas, colocándose en Cuadro 7.

Cuadro 7 Comparación de estructuras de sentencias (Elaboración propia, 2012).

MySQL
CONSULTAS BASICAS
select * from articulo
select descripcion, precio from articulo
select articulo.clavearticulo, articulo.precio, compras.fecha from articulo, compras
select articulo.clavearticulo, articulo.precio, fecha from articulo, compras
select articulo.* from articulo
CONSULTAS INTERMEDIAS
select * from articulo where descripcion='sacas'
select * from articulo where descripcion='sacas' or descripcion ='regla' or descripcion='lapiz'
select clavearticulo, descripcion from articulo where precio >=3.00
select * from articulo where precio >=3.00 and precio<=10.00
select * from articulo where precio between 3.00 and 10.00
select * from compras where fecha= '2012-02-02'
SELECT * FROM compras WHERE fecha BETWEEN '2000-01-01' AND '2012-02-02'
SELECT * FROM articulo WHERE descripcion LIKE 'es*'
SELECT * FROM articulo WHERE descripcion LIKE 'sa%'
SELECT * FROM articulo WHERE descripcion LIKE '%es'
SELECT * FROM articulo WHERE descripcion LIKE '%sa%'
SELECT * FROM articulo WHERE descripcion IN ('lapiz', 'sacas')
SELECT descripcion,precio FROM articulo ORDER BY descripcion
SELECT * FROM articulo ORDER BY descripcion DESC
SELECT * FROM articulo ORDER BY descripcion, precio DESC
SELECT SUM(precio) FROM articulo
SELECT SUM(precio- existencias) AS saldo FROM articulo
SELECT SUM(no_compra) AS tl FROM compras WHERE fecha=CURDATE()
SELECT AVG(precio) FROM articulo
SELECT AVG(precio-existencias) AS saldo_medio FROM articulo
SELECT AVG(precio) AS media FROM articulo WHERE fecha=CURDATE()
SELECT MIN(precio) AS minimo FROM articulo
SELECT MAX(precio) AS maximo FROM articulo
SELECT MAX(clavearticulo) AS maximo FROM compras WHERE fecha= CURDATE()
SELECT COUNT(*) AS total FROM articulo
SELECT COUNT(*) AS total FROM compras WHERE fecha=CURDATE()
SELECT COUNT(*) AS num_pagos FROM articulo WHERE precio=5.50
SELECT SUM(precio) AS total, AVG(precio) AS media, COUNT(*) AS registros, MAX(precio) AS maximo, MIN(precio) AS minimo FROM articulo
SELECT DISTINCT descripcion FROM articulo
SELECT precio, count(precio) AS num_pedidos, SUM(precio) AS cantidad FROM articulo GROUP BY precio
SELECT precio, count(precio) AS num_pedidos FROM articulo GROUP BY precio
CONSULTAS AVANZADAS
SELECT * FROM articulo INNER JOIN compras ON articulo.clavearticulo =compras.clavearticulo
SELECT descripcion FROM articulo INNER JOIN compras ON articulo.clavearticulo =compras.clavearticulo WHERE fecha>'31/01/11'
SELECT descripcion, compras.* FROM articulo INNER JOIN compras ON articulo.clavearticulo = compras.clavearticulo ORDER BY fecha
SELECT * FROM articulo LEFT JOIN compras ON articulo.clavearticulo =compras.clavearticulo
SELECT * FROM articulo WHERE clavearticulo = (SELECT clavearticulo FROM movimientos)
SELECT clavearticulo, descripcion FROM articulo WHERE clavearticulo = (SELECT clavearticulo FROM movimientos)
SELECT clavearticulo, descripcion FROM articulo WHERE EXISTS (SELECT clavearticulo FROM movimientos)
SELECT clavearticulo, descripcion FROM articulo WHERE clavearticulo IN (SELECT clavearticulo FROM movimientos)
SELECT clavearticulo, descripcion FROM articulo WHERE clavearticulo IN (SELECT DISTINCT clavearticulo FROM movimientos)
SELECT clavearticulo, fecha FROM compras WHERE clavearticulo = ANY (SELECT clavearticulo FROM articulo WHERE clavearticulo =1)

Continua Cuadro 7 Comparación de estructuras de sentencias (Elaboración propia, 2012).

Postgresql
CONSULTAS BASICAS
select * from articulo
select descripcion, precio from articulo
select articulo.claveArticulo,articulo.precio, compras.fecha from articulo, compras
select articulo.claveArticulo,articulo.precio, fecha from articulo, compras
select articulo.* from articulo
CONSULTAS INTERMEDIAS
select * from articulo where descripcion='sacas'
select * from articulo where descripcion='sacas' or descripcion ='regla' or descripcion='lapiz'
select claveArticulo, descripcion from articulo where precio >='3.00'
select * from articulo where precio >='3.00' and precio<='10.00 '
select * from articulo where precio between '3.00' and '10.00 '
select * from compras where fecha= '02-02-12'
SELECT * FROM compras WHERE fecha BETWEEN '18/11/11' AND '31/01/12'
SELECT * FROM articulo WHERE descripcion LIKE 'es*'
SELECT * FROM articulo WHERE descripcion LIKE 'sa%'
SELECT * FROM articulo WHERE descripcion LIKE '%es'
SELECT * FROM articulo WHERE descripcion LIKE '%sa%'
SELECT * FROM articulo WHERE descripcion IN ('lapiz', 'sacas')
SELECT descripcion,precio FROM articulo ORDER BY descripcion
SELECT * FROM articulo ORDER BY descripcion DESC
SELECT * FROM articulo ORDER BY descripcion, precio DESC
SELECT SUM(precio) FROM articulo
SELECT SUM (precio-precio_uni) AS saldo FROM articulo
SELECT * FROM compras WHERE fecha=current_date
SELECT AVG(existencias) FROM articulo
SELECT AVG(existencias-existencias_an) AS saldo_medio FROM articulo
SELECT AVG(total_prod) AS media FROM compras WHERE fecha=current_date
SELECT MIN(precio) AS minimo FROM articulo
SELECT MAX(precio) AS maximo FROM articulo
SELECT MAX(total_prod) AS maximo FROM compras WHERE fecha=current_date
SELECT COUNT(*) AS total FROM articulo
SELECT COUNT(*) AS total FROM compras WHERE fecha=current_date
SELECT COUNT(*) AS num_pagos FROM articulo WHERE precio='5.50'
SELECT SUM(precio) AS total, AVG(existencias) AS media, COUNT(*) AS registros, MAX(precio) AS maximo, MIN(precio) AS minimo FROM articulo
SELECT DISTINCT descripcion FROM articulo
SELECT precio, count(precio) AS num_pedidos, SUM(precio) AS cantidad FROM articulo GROUP BY precio
SELECT precio, count(precio) AS num_pedidos FROM articulo GROUP BY precio
CONSULTAS AVANZADAS
SELECT * FROM articulo INNER JOIN compras ON articulo.claveArticulo =compras.claveArticulo
SELECT descripcion FROM articulo INNER JOIN compras ON articulo.claveArticulo=compras.claveArticulo WHERE fecha>'31/01/11'
SELECT descripcion, compras.* FROM articulo INNER JOIN compras ON articulo.claveArticulo= compras.claveArticulo ORDER BY fecha
SELECT * FROM articulo LEFT JOIN compras ON articulo.claveArticulo =compras.claveArticulo
SELECT * FROM vehiculo WHERE idvehiculo IN (SELECT idvehiculo FROM renta)
SELECT idvehiculo, modelo FROM vehiculo WHERE idvehiculo=(SELECT idvehiculo FROM renta)
SELECT idvehiculo, modelo FROM vehiculo WHERE EXISTS (SELECT idvehiculo FROM renta)
SELECT idvehiculo, modelo FROM vehiculo WHERE idvehiculo IN (SELECT idvehiculo FROM renta)
SELECT idvehiculo, modelo FROM vehiculo WHERE idvehiculo IN (SELECT DISTINCT idvehiculo FROM renta)
SELECT idvehiculo, modelo FROM vehiculo WHERE idvehiculo = ANY (SELECT DISTINCT idvehiculo FROM renta WHERE idvehiculo =1)

Continua Cuadro 7 Comparación de estructuras de sentencias (Elaboración propia, 2012).

Oracle
CONSULTAS BASICAS
select * from articulo
select descripcion, precio from articulo
select articulo.clavearticulo, articulo.precio, compras.fecha from articulo, compras
select articulo.clavearticulo, articulo.precio, fecha from articulo, compras
select articulo.* from articulo
CONSULTAS INTERMEDIAS
select * from articulo where descripcion='sacas'
select * from articulo where descripcion='sacas' or descripcion ='regla' or descripcion='lapiz'
select clavearticulo, descripcion from articulo where precio >=3.00
select * from articulo where precio >=3.00 and precio<=10.00
select * from articulo where precio between 3.00 and 10.00
SELECT * FROM compras WHERE fecha BETWEEN '18/11/11' AND '01/01/12'
SELECT * FROM articulo WHERE descripcion LIKE 'es*'
SELECT * FROM articulo WHERE descripcion LIKE 'sa%'
SELECT * FROM articulo WHERE descripcion LIKE '%as'
SELECT * FROM articulo WHERE descripcion LIKE '%sa%'
SELECT * FROM articulo WHERE descripcion IN ('goma', 'sacas')
SELECT descripcion,precio FROM articulo ORDER BY descripcion
SELECT * FROM articulo ORDER BY descripcion DESC
SELECT * FROM articulo ORDER BY descripcion, precio DESC
SELECT SUM(precio) FROM articulo
SELECT SUM (precio- existencias) AS saldo FROM articulo
SELECT SUM(precio) AS total FROM articulo WHERE fecha= (select sysdate from dual)
SELECT AVG(precio) FROM articulo
SELECT AVG(precio-existencias) AS saldo_medio FROM articulo
SELECT AVG(precio) AS media FROM articulo WHERE fecha= (select sysdate from dual)
SELECT MIN(precio) AS minimo FROM articulo
SELECT MAX(precio) AS maximo FROM articulo
SELECT MAX(clavearticulo) AS maximo FROM compras WHERE fecha= (select sysdate from dual)
SELECT COUNT(*) AS total FROM articulo
SELECT COUNT(*) AS total FROM compras WHERE fecha= (select sysdate from dual)
SELECT COUNT(*) AS num_pagos FROM articulo WHERE precio='2.50'
SELECT SUM(precio) AS total, AVG(precio) AS media, COUNT(*) AS registros, MAX(precio) AS maximo, MIN(precio) AS minimo FROM articulo
SELECT DISTINCT descripcion FROM articulo
SELECT precio, count(precio) AS num_pedidos, SUM(precio) AS cantidad FROM articulo GROUP BY precio
SELECT precio, count(precio) AS num_pedidos FROM articulo GROUP BY precio
CONSULTAS AVANZADAS
SELECT * FROM articulo INNER JOIN compras ON articulo.clavearticulo =compras.clavearticulo
SELECT descripcion FROM articulo INNER JOIN compras ON articulo.clavearticulo =compras.clavearticulo WHERE fecha>'31/01/11'
SELECT descripcion, compras.* FROM articulo INNER JOIN compras ON articulo.clavearticulo = compras.clavearticulo ORDER BY fecha
SELECT * FROM articulo LEFT JOIN compras ON articulo.clavearticulo =compras.clavearticulo
SELECT * FROM articulo WHERE clavearticulo IN (SELECT clavearticulo FROM compras)
SELECT clavearticulo, descripcion FROM articulo WHERE clavearticulo= (SELECT clavearticulo FROM compras)
SELECT clavearticulo, descripcion FROM articulo WHERE EXISTS (SELECT clavearticulo FROM compras)
SELECT clavearticulo, descripcion FROM articulo WHERE clavearticulo IN (SELECT clavearticulo FROM compras)
SELECT clavearticulo, descripcion FROM articulo WHERE clavearticulo IN (SELECT DISTINCT clavearticulo FROM compras)
SELECT clavearticulo, fecha FROM compras WHERE clavearticulo = ANY (SELECT clavearticulo FROM articulo WHERE clavearticulo =1)

Continua Cuadro 7 Comparación de estructuras de sentencias (Elaboración propia, 2012).

SQLServer
CONSULTAS BASICAS
select * from clientes
select contrasenia, nombre from clientes
select clientes.contrasenia, clientes.nombre, cuenta.idCuenta from clientes, cuenta
select clientes.contrasenia, clientes.nombre, idCuenta from clientes, cuenta
select clientes.* from clientes
CONSULTAS INTERMEDIAS
select * from clientes where nombre='maria'
select * from clientes where nombre='maria' or nombre ='juan'
select contrasenia, nombre, numero from clientes where numero >=3.00
select * from clientes where numero >=3.00 and numero<=10.00
select * from clientes where numero between 3.00 and 10.00
select * from operacionescuenta where fechaOperacion= '2012-04-22'
select * from operacionescuenta where fechaOperacion between '2012-04-22' and '2012-04-24'
SELECT * FROM clientes WHERE nombre LIKE 'j*'
SELECT * FROM clientes WHERE nombre LIKE 'j%'
SELECT * FROM clientes WHERE nombre LIKE '%ia'
SELECT * FROM clientes WHERE nombre LIKE '%ia%'
SELECT * FROM clientes WHERE nombre IN('maria', 'juan')
SELECT contrasenia, nombre FROM clientes ORDER BY nombre
SELECT * FROM clientes ORDER BY nombre DESC
SELECT * FROM clientes ORDER BY nombre,numero DESC
SELECT SUM(numero) FROM clientes
SELECT SUM(numero) AS saldo FROM clientes
SELECT * FROM operacionescuenta WHERE fechaOperacion= (SELECT CONVERT (char(10), getdate(), 102))
SELECT AVG(balance) FROM cuenta
SELECT AVG(balance-idClientes) AS saldo_medio FROM cuenta
SELECT AVG(cantidad) AS media FROM operacionescuenta WHERE fechaOperacion=(SELECT CONVERT (char(10), getdate(), 102))
SELECT MIN(cantidad) AS minimo FROM operacionescuenta
SELECT MAX(cantidad) AS maximo FROM operacionescuenta
SELECT MAX(cantidad) AS maximo FROM operacionescuenta where fechaOperacion=(SELECT CONVERT (char(10), getdate(), 102))
SELECT COUNT(*) AS totalclientes FROM clientes
SELECT COUNT(*) AS totalclientes FROM operacionescuenta where fechaOperacion = (SELECT CONVERT (char(10), getdate(), 102))
SELECT COUNT(*) AS total_mes FROM cuenta WHERE balance=4.50
SELECT SUM(balance) AS cantidad, AVG(balance) AS media, COUNT(*) AS registros, MAX(balance) AS maximo, MIN(balance) AS minimo FROM cuenta
SELECT DISTINCT nombre FROM clientes
SELECT balance, count(idCuenta) AS cuentas, SUM(balance) AS cantidad FROM cuenta GROUP BY balance
SELECT balance, count(balance) AS total_bala FROM cuenta GROUP BY balance
CONSULTAS AVANZADAS
SELECT * FROM clientes INNER JOIN cuenta ON clientes.idClientes=cuenta.IdClientes
SELECT cantidad FROM operacionescuenta INNER JOIN cuenta ON operacionescuenta.idCuenta=cuenta.idCuenta where fechaOperacion = '2012-03-04'
SELECT cantidad, cuenta.* FROM operacionescuenta INNER JOIN cuenta ON operacionescuenta.idCuenta=cuenta.idCuenta ORDER BY fechaOperacion
SELECT * FROM cuenta LEFT JOIN operacionescuenta ON cuenta.idCuenta = operacionescuenta.idCuenta
SELECT * FROM clientes WHERE idClientes = (SELECT balance FROM cuenta WHERE clientes.idClientes = cuenta.idClientes)
SELECT idClientes, nombre FROM clientes WHERE idClientes = (SELECT balance FROM cuenta WHERE clientes.idClientes = cuenta.idClientes)
SELECT idClientes, nombre FROM clientes WHERE EXISTS (SELECT balance FROM cuenta WHERE clientes.idClientes = cuenta.idClientes)
SELECT idClientes, nombre FROM clientes WHERE idClientes IN (SELECT balance FROM cuenta WHERE clientes.idClientes = cuenta.idClientes)
SELECT idClientes, nombre FROM clientes WHERE idClientes IN (SELECT DISTINCT balance FROM cuenta WHERE clientes.idClientes = cuenta.idClientes)
SELECT idClientes, nombre FROM clientes WHERE idClientes = ANY (SELECT idClientes FROM cuenta WHERE idClientes = 1)

6.1.1.2 Paso 2: Analizar las diferencias entre las estructuras de consultas para cada Estructura del lenguaje SQL de cada manejador de BD.

El cuadro comparativo permitió detectar las consultas que tenían cambios en su escritura sombreándose de color morado, por otro lado existen consultas que no se ejecutaron y estas se subrayaron de color verde para así, poder plantear los algoritmos de estandarización y automatización de consultas (ver Cuadro 8).

Cuadro 8 Identificación de diferencias de las estructuras de sentencias (Elaboración propia, 2012).

MySQL	OBSERVACIONES
CONSULTAS BASICAS	
select * from articulo	
select descripcion, precio from articulo	
select articulo.clavearticulo, articulo.precio, compras.fecha from articulo, compras	
select articulo.clavearticulo, articulo.precio, fecha from articulo, compras	
select articulo.* from articulo	
CONSULTAS INTERMEDIAS	
select * from articulo where descripcion='sacas'	
select * from articulo where descripcion='sacas' or descripcion ='regla' or descripcion='lapiz'	
select clavearticulo, descripcion from articulo where precio >=3.00	
select * from articulo where precio >=3.00 and precio<=10.00	
select * from articulo where precio between 3.00 and 10.00	
select * from compras where fecha= '2012-02-02'	Estructura para conusltar la fecha (DD/MM/YYYY)
SELECT * FROM compras WHERE fecha BETWEEN '2000-01-01' AND '2012-02-02'	
SELECT * FROM articulo WHERE descripcion LIKE 'es*'	Esta consulta no funciona con *
SELECT * FROM articulo WHERE descripcion LIKE 'sa%'	
SELECT * FROM articulo WHERE descripcion LIKE '%es'	
SELECT * FROM articulo WHERE descripcion LIKE '%sa%'	
SELECT * FROM articulo WHERE descripcion IN ('lapiz', 'sacas')	
SELECT descripcion,precio FROM articulo ORDER BY descripcion	
SELECT * FROM articulo ORDER BY descripcion DESC	
SELECT * FROM articulo ORDER BY descripcion, precio DESC	
SELECT SUM(precio) FROM articulo	
SELECT SUM(precio- existencias) AS saldo FROM articulo	
SELECT SUM(no_compra) AS tl FROM compras WHERE fecha=CURDATE()	Para esta consulta se utiliza CURDATE() ya que solo muestra la fecha (DD/MM/YYYY)
SELECT AVG(precio) FROM articulo	
SELECT AVG(precio-existencias) AS saldo_medio FROM articulo	
SELECT AVG(precio) AS media FROM articulo WHERE fecha=CURDATE()	
SELECT MIN(precio) AS minimo FROM articulo	
SELECT MAX(precio) AS maximo FROM articulo	
SELECT MAX(clavearticulo) AS maximo FROM compras WHERE fecha= CURDATE()	Para esta consulta se utiliza CURDATE() ya que solo muestra la fecha (DD/MM/YYYY)
SELECT COUNT(*) AS total FROM articulo	
SELECT COUNT(*) AS total FROM compras WHERE fecha=CURDATE()	Para esta consulta se utiliza CURDATE() ya que solo muestra
SELECT COUNT(*) AS num_pagos FROM articulo WHERE precio=5.50	
SELECT SUM(precio) AS total, AVG(precio) AS media, COUNT(*) AS registros, MAX(precio) AS maximo, MIN(precio) AS minimo FROM articulo	
SELECT DISTINCT descripcion FROM articulo	
SELECT precio, count(precio) AS num_pedidos, SUM(precio) AS cantidad FROM articulo GROUP BY precio	
SELECT precio, count(precio) AS num_pedidos FROM articulo GROUP BY precio	
CONSULTAS AVANZADAS	
SELECT * FROM articulo INNER JOIN compras ON articulo.clavearticulo =compras.clavearticulo	
SELECT descripcion FROM articulo INNER JOIN compras ON articulo.clavearticulo =compras.clavearticulo WHERE fecha>'31/01/11'	
SELECT descripcion, compras.* FROM articulo INNER JOIN compras ON articulo.clavearticulo = compras.clavearticulo ORDER BY fecha	
SELECT * FROM articulo LEFT JOIN compras ON articulo.clavearticulo =compras.clavearticulo	
SELECT * FROM articulo WHERE clavearticulo = (SELECT clavearticulo FROM movimientos)	
SELECT clavearticulo, descripcion FROM articulo WHERE clavearticulo = (SELECT clavearticulo FROM movimientos)	
SELECT clavearticulo, descripcion FROM articulo WHERE EXISTS (SELECT clavearticulo FROM movimientos)	No funciona la consulta
SELECT clavearticulo, descripcion FROM articulo WHERE clavearticulo IN (SELECT clavearticulo FROM movimientos)	
SELECT clavearticulo, descripcion FROM articulo WHERE clavearticulo IN (SELECT DISTINCT clavearticulo FROM movimientos)	
SELECT clavearticulo, fecha FROM compras WHERE clavearticulo = ANY (SELECT clavearticulo FROM articulo WHERE clavearticulo =1)	

Continua Cuadro 8 Identificación de diferencias de las estructuras de sentencias (Elaboración propia, 2012).

Postgresql	OBSERVACIONES
CONSULTAS BASICAS	
select * from articulo	
select descripcion, precio from articulo	
select articulo.claveArticulo,articulo.precio, compras.fecha from articulo, compras	
select articulo.claveArticulo,articulo.precio, fecha from articulo, compras	
select articulo.* from articulo	
CONSULTAS INTERMEDIAS	
select * from articulo where descripcion='sacas'	
select * from articulo where descripcion='sacas' or descripcion ='regla' or descripcion='lapiz'	
select claveArticulo, descripcion from articulo where precio >='3.00'	A diferencia de MySQL en Postresql el numero va entre ' ' o sin ' '
select * from articulo where precio >='3.00' and precio<='10.00 '	
select * from articulo where precio between '3.00' and '10.00 '	
select * from compras where fecha= '02-02-12'	A diferencia de MySQL en Postresqlel formato de la fecha es 'DD/MM/YY'
SELECT * FROM compras WHERE fecha BETWEEN '18/11/11' AND '31/01/12'	
SELECT * FROM articulo WHERE descripcion LIKE 'es*'	No funciona la consulta
SELECT * FROM articulo WHERE descripcion LIKE 'sa%'	
SELECT * FROM articulo WHERE descripcion LIKE '%es'	
SELECT * FROM articulo WHERE descripcion LIKE '%sa%'	
SELECT * FROM articulo WHERE descripcion IN ('lapiz', 'sacas')	
SELECT descripcion,precio FROM articulo ORDER BY descripcion	
SELECT * FROM articulo ORDER BY descripcion DESC	
SELECT * FROM articulo ORDER BY descripcion, precio DESC	
SELECT SUM(precio) FROM articulo	
SELECT SUM (precio-precio_uni) AS saldo FROM articulo	
SELECT * FROM compras WHERE fecha=current_date	Para sacar la fecha en Postgresql es current_date
SELECT AVG(existencias) FROM articulo	
SELECT AVG(existencias-existencias_an) AS saldo_medio FROM articulo	Para el AVG los atributos deben tener el tipo de dato (integer) para que se realice
SELECT AVG(total_prod) AS media FROM compras WHERE fecha=current_date	Para sacar la fecha en Postgresql es current_date
SELECT MIN(precio) AS minimo FROM articulo	
SELECT MAX(precio) AS maximo FROM articulo	
SELECT MAX(total_prod) AS maximo FROM compras WHERE fecha=current_date	Para sacar la fecha en Postgresql es current_date
SELECT COUNT(*) AS total FROM articulo	
SELECT COUNT(*) AS total FROM compras WHERE fecha=current_date	Para sacar la fecha en Postgresql es current_date
SELECT COUNT(*) AS num_pagos FROM articulo WHERE precio='5.50'	
SELECT SUM(precio) AS total, AVG(existencias) AS media, COUNT(*) AS registros, MAX(precio) AS maximo, MIN(precio) AS minimo FROM articulo	
SELECT DISTINCT descripcion FROM articulo	
SELECT precio, count(precio) AS num_pedidos, SUM(precio) AS cantidad FROM articulo GROUP BY precio	
SELECT precio, count(precio) AS num_pedidos FROM articulo GROUP BY precio	
CONSULTAS AVANZADAS	
SELECT * FROM articulo INNER JOIN compras ON articulo.claveArticulo =compras.claveArticulo	
SELECT descripcion FROM articulo INNER JOIN compras ON articulo.claveArticulo=compras.claveArticulo WHERE fecha>'31/01/11'	El formato de la fecha es 'DD/MM/YY'
SELECT descripcion, compras.* FROM articulo INNER JOIN compras ON articulo.claveArticulo= compras.claveArticulo ORDER BY fecha	
SELECT * FROM articulo LEFT JOIN compras ON articulo.claveArticulo =compras.claveArticulo	
SELECT * FROM vehiculo WHERE idvehiculo IN (SELECT idvehiculo FROM renta)	
SELECT idvehiculo, modelo FROM vehiculo WHERE idvehiculo=(SELECT idvehiculo FROM renta)	No funciona la consulta
SELECT idvehiculo, modelo FROM vehiculo WHERE EXISTS (SELECT idvehiculo FROM renta)	
SELECT idvehiculo, modelo FROM vehiculo WHERE idvehiculo IN (SELECT idvehiculo FROM renta)	
SELECT idvehiculo, modelo FROM vehiculo WHERE idvehiculo IN (SELECT DISTINCT idvehiculo FROM renta)	
SELECT idvehiculo, modelo FROM vehiculo WHERE idvehiculo = ANY (SELECT DISTINCT idvehiculo FROM renta WHERE idvehiculo =1)	

Continua Cuadro 8 Identificación de diferencias de las estructuras de sentencias (Elaboración propia, 2012).

Oracle	OBSERVACIONES
CONSULTAS BASICAS	
select * from articulo	
select descripcion, precio from articulo	
select articulo.clavearticulo, articulo.precio, compras.fecha from articulo, compras	
select articulo.clavearticulo, articulo.precio, fecha from articulo, compras	
select articulo.* from articulo	
CONSULTAS INTERMEDIAS	
select * from articulo where descripcion='sacas'	
select * from articulo where descripcion='sacas' or descripcion ='regla' or descripcion='lapiz'	
select clavearticulo, descripcion from articulo where precio >=3.00	
select * from articulo where precio >=3.00 and precio<=10.00	
select * from articulo where precio between 3.00 and 10.00	
SELECT * FROM compras WHERE fecha BETWEEN '18/11/11' AND '01/01/12'	A diferencia de MySQL y Postgresql en Oracle la fecha tiene este formato al momento de hacer la consulta 'dd-mm-yyyy'
SELECT * FROM articulo WHERE descripcion LIKE 'es*'	No se puede realizar
SELECT * FROM articulo WHERE descripcion LIKE 'sa%'	
SELECT * FROM articulo WHERE descripcion LIKE '%as'	
SELECT * FROM articulo WHERE descripcion LIKE '%sa%'	
SELECT * FROM articulo WHERE descripcion IN ('goma', 'sacas')	
SELECT descripcion,precio FROM articulo ORDER BY descripcion	
SELECT * FROM articulo ORDER BY descripcion DESC	
SELECT * FROM articulo ORDER BY descripcion, precio DESC	
SELECT SUM(precio) FROM articulo	
SELECT SUM (precio- existencias) AS saldo FROM articulo	
SELECT SUM(precio) AS total FROM articulo WHERE fecha= (select sysdate from dual)	Para sacar la fecha en Postgresql es (select sysdate from dual)
SELECT AVG(precio) FROM articulo	
SELECT AVG(precio-existencias) AS saldo_medio FROM articulo	
SELECT AVG(precio) AS media FROM articulo WHERE fecha= (select sysdate from dual)	Para sacar la fecha en Postgresql es (select sysdate from dual)
SELECT MIN(precio) AS minimo FROM articulo	
SELECT MAX(precio) AS maximo FROM articulo	
SELECT MAX(clavearticulo) AS maximo FROM compras WHERE fecha= (select sysdate from dual)	Para sacar la fecha en Postgresql es (select sysdate from dual)
SELECT COUNT(*) AS total FROM articulo	
SELECT COUNT(*) AS total FROM compras WHERE fecha= (select sysdate from dual)	Para sacar la fecha en Postgresql es (select sysdate from dual)
SELECT COUNT(*) AS num_pagos FROM articulo WHERE precio='2.50'	A diferencia de MySQL el numero va entre " o sin "
SELECT SUM(precio) AS total, AVG(precio) AS media, COUNT(*) AS registros, MAX(precio) AS maximo, MIN(precio) AS minimo FROM articulo	
SELECT DISTINCT descripcion FROM articulo	
SELECT precio, count(precio) AS num_pedidos, SUM(precio) AS cantidad FROM articulo GROUP BY precio	
SELECT precio, count(precio) AS num_pedidos FROM articulo GROUP BY precio	
CONSULTAS AVANZADAS	
SELECT * FROM articulo INNER JOIN compras ON articulo.clavearticulo =compras.clavearticulo	
SELECT descripcion FROM articulo INNER JOIN compras ON articulo.clavearticulo =compras.clavearticulo WHERE fecha>'31/01/11'	A diferencia de MySQL y Postgresql en Oracle la fecha tiene este formato al momento de hacer la consulta 'dd-mm-yyyy'
SELECT descripcion, compras.* FROM articulo INNER JOIN compras ON articulo.clavearticulo = compras.clavearticulo ORDER BY fecha	
SELECT * FROM articulo LEFT JOIN compras ON articulo.clavearticulo =compras.clavearticulo	
SELECT * FROM articulo WHERE clavearticulo IN (SELECT clavearticulo FROM compras)	
SELECT clavearticulo, descripcion FROM articulo WHERE clavearticulo= (SELECT clavearticulo FROM compras)	No se puede realizar
SELECT clavearticulo, descripcion FROM articulo WHERE EXISTS (SELECT clavearticulo FROM compras)	No se puede realizar
SELECT clavearticulo, descripcion FROM articulo WHERE clavearticulo IN (SELECT clavearticulo FROM compras)	
SELECT clavearticulo, descripcion FROM articulo WHERE clavearticulo IN (SELECT DISTINCT clavearticulo FROM compras)	
SELECT clavearticulo, fecha FROM compras WHERE clavearticulo = ANY (SELECT clavearticulo FROM articulo WHERE clavearticulo =1)	

Continua Cuadro 8 Identificación de diferencias de las estructuras de sentencias (Elaboración propia, 2012).

SQLServer	OBSERVACIONES
CONSULTAS BASICAS	
select * from clientes	
select contrasenia, nombre from clientes	
select clientes.contrasenia, clientes.nombre, cuenta.idCuenta from clientes, cuenta	
select clientes.contrasenia, clientes.nombre, idCuenta from clientes, cuenta	
select clientes.* from clientes	
CONSULTAS INTERMEDIAS	
select * from clientes where nombre='maria'	
select * from clientes where nombre='maria' or nombre ='juan'	
select contrasenia, nombre, numero from clientes where numero >=3.00	
select * from clientes where numero >=3.00 and numero<=10.00	
select * from clientes where numero between 3.00 and 10.00	
select * from operacionescuenta where fechaOperacion= '2012-04-22'	
select * from operacionescuenta where fechaOperacion between '2012-04-22' and '2012-04-24'	
SELECT * FROM clientes WHERE nombre LIKE 'j*'	No se puede realizar
SELECT * FROM clientes WHERE nombre LIKE 'j%'	
SELECT * FROM clientes WHERE nombre LIKE '%ia'	
SELECT * FROM clientes WHERE nombre LIKE '%ia%'	
SELECT * FROM clientes WHERE nombre IN('maria', 'juan')	
SELECT contrasenia, nombre FROM clientes ORDER BY nombre	
SELECT * FROM clientes ORDER BY nombre DESC	
SELECT * FROM clientes ORDER BY nombre,numero DESC	
SELECT SUM(numero) FROM clientes	
SELECT SUM(numero) AS saldo FROM clientes	
SELECT * FROM operacionescuenta WHERE fechaOperacion= (SELECT CONVERT (char(10), getdate(), 102))	La fecha se saca con subconsulta, que realiza una conversion
SELECT AVG(balance) FROM cuenta	
SELECT AVG(balance-idClientes) AS saldo_medio FROM cuenta	
SELECT AVG(cantidad) AS media FROM operacionescuenta WHERE fechaOperacion=(SELECT CONVERT (char(10), getdate(), 102))	La fecha se saca con subconsulta, que realiza una conversion
SELECT MIN(cantidad) AS minimo FROM operacionescuenta	
SELECT MAX(cantidad) AS maximo FROM operacionescuenta	
SELECT MAX(cantidad) AS maximo FROM operacionescuenta where fechaOperacion=(SELECT CONVERT (char(10), getdate(), 102))	La fecha se saca con subconsulta, que realiza una conversion
SELECT COUNT(*) AS totalclientes FROM clientes	
SELECT COUNT(*) AS totalclientes FROM operacionescuenta where fechaOperacion = (SELECT CONVERT (char(10), getdate(), 102))	La fecha se saca con subconsulta, que realiza una conversion
SELECT COUNT(*) AS total_mes FROM cuenta WHERE balance=4.50	
SELECT SUM(balance) AS cantidad, AVG(balance) AS media, COUNT(*) AS registros, MAX(balance) AS maximo, MIN(balance) AS minimo FROM cuenta	
SELECT DISTINCT nombre FROM clientes	
SELECT balance, count(idCuenta) AS cuentas, SUM(balance) AS cantidad FROM cuenta GROUP BY balance	
SELECT balance, count(balance) AS total_bala FROM cuenta GROUP BY balance	
CONSULTAS AVANZADAS	
SELECT * FROM clientes INNER JOIN cuenta ON clientes.idClientes=cuenta.IdClientes	
SELECT cantidad FROM operacionescuenta INNER JOIN cuenta ON operacionescuenta.idCuenta=cuenta.idCuenta where fechaOperacion = '2012-03-04'	
SELECT cantidad, cuenta.* FROM operacionescuenta INNER JOIN cuenta ON operacionescuenta.idCuenta=cuenta.idCuenta ORDER BY fechaOperacion	
SELECT * FROM cuenta LEFT JOIN operacionescuenta ON cuenta.idCuenta = operacionescuenta.idCuenta	
SELECT * FROM clientes WHERE idClientes = (SELECT balance FROM cuenta WHERE clientes.idClientes = cuenta.idClientes)	
SELECT idClientes, nombre FROM clientes WHERE idClientes = (SELECT balance FROM cuenta WHERE clientes.idClientes = cuenta.idClientes)	No funciona
SELECT idClientes, nombre FROM clientes WHERE EXISTS (SELECT balance FROM cuenta WHERE clientes.idClientes = cuenta.idClientes)	Utiliza el operador EXISTS
SELECT idClientes, nombre FROM clientes WHERE idClientes IN (SELECT balance FROM cuenta WHERE clientes.idClientes = cuenta.idClientes)	No funciona
SELECT idClientes, nombre FROM clientes WHERE idClientes IN (SELECT DISTINCT balance FROM cuenta WHERE clientes.idClientes = cuenta.idClientes)	La consulta se realiza pero no arroja resultados
SELECT idClientes, nombre FROM clientes WHERE idClientes = ANY (SELECT idClientes FROM cuenta WHERE idClientes = 1)	

6.1.1.3 Paso 3: Diseñar la aplicación mediante casos de uso.

6.1.1.3.1 Caso de uso Conexión a una BD.

Para realizar la conexión a una BD se especifican los requerimientos de usuario, contraseña, BD, servidor y puerto, además de seleccionar omitir la conexión para que esta se realice. Este proceso se realiza cada vez que se requiera conectarse a una BD diferente (ver Figura 16).

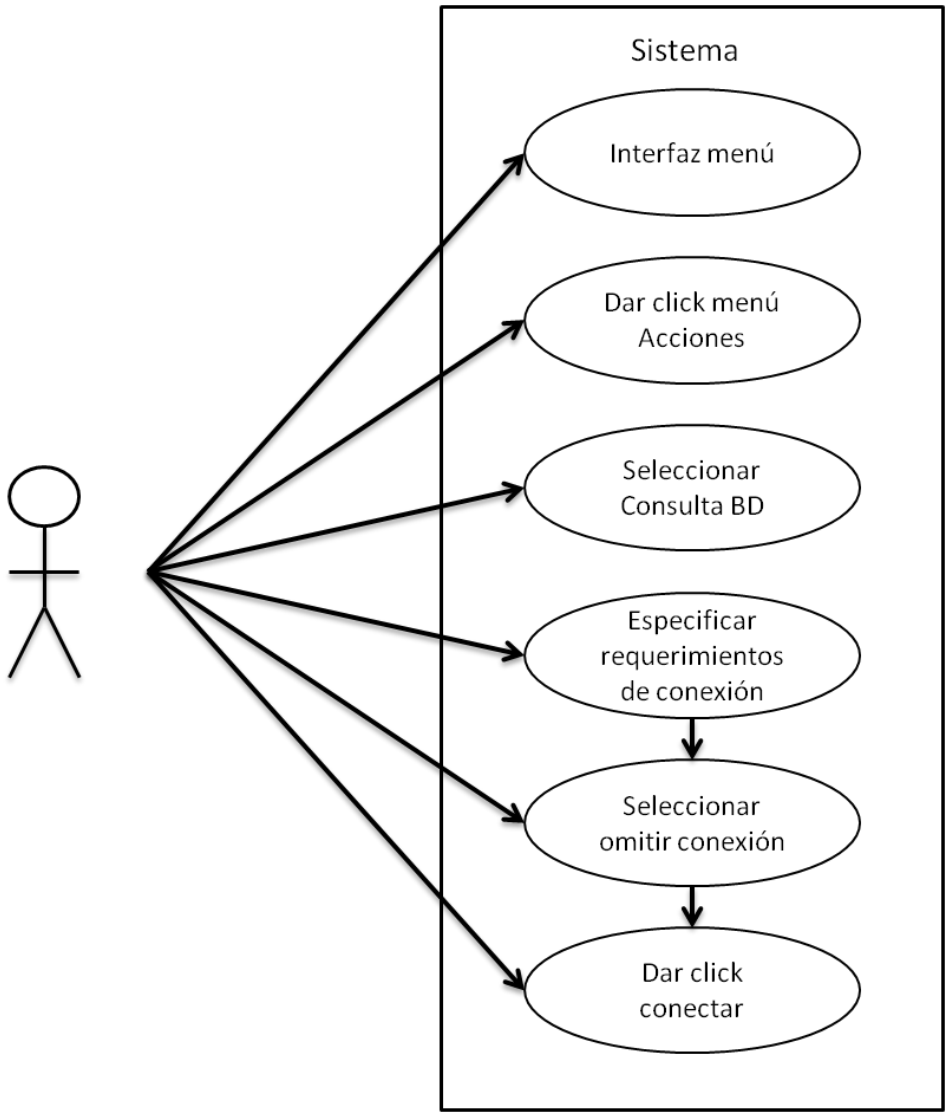

Figura 16 Caso de uso Conexión a BD (Elaboración propia, 2012).

6.1.1.3.2 Caso de uso. Conexión a dos o más BD de diferentes manejadores.

Al conectarse a una BD, si los datos aportados por el usuario son correctos se procederá a realizar la conexión, como resultado se visualizara la interfaz Consulta BD, donde se muestran las tablas, atributos y tipos de dato. El usuario para realizar otra conexión tendrá que dar click en el botón cerrar, para luego proceder a seleccionar el menú Acciones (ver Figura 17).

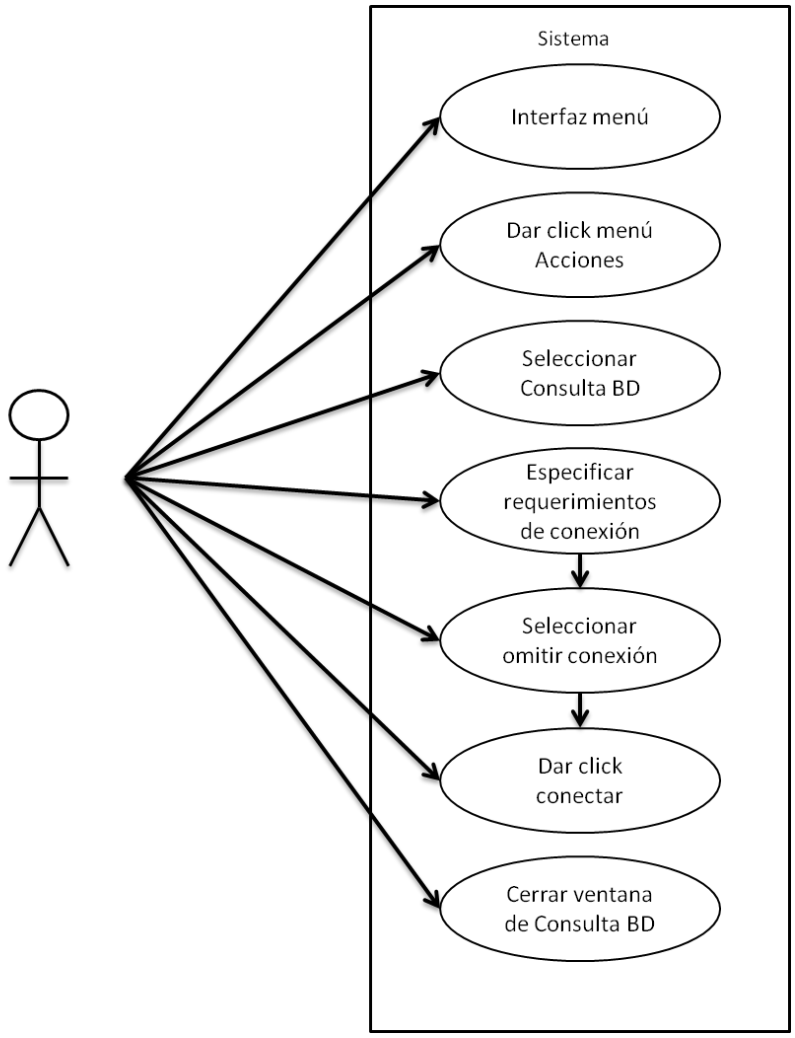

Figura 17 Caso de uso Conexión a dos o más BD de diferentes manejadores (Elaboración propia, 2012).

6.1.1.3.3 Caso de uso. Consulta a BD.

La interfaz Consulta BD permite realizar las consultas solicitadas para cada Lenguaje Manejador de BD, el usuario debe saber las estructuras básicas, intermedias y avanzadas de consulta del Lenguaje MySQL (ver Figura 18).

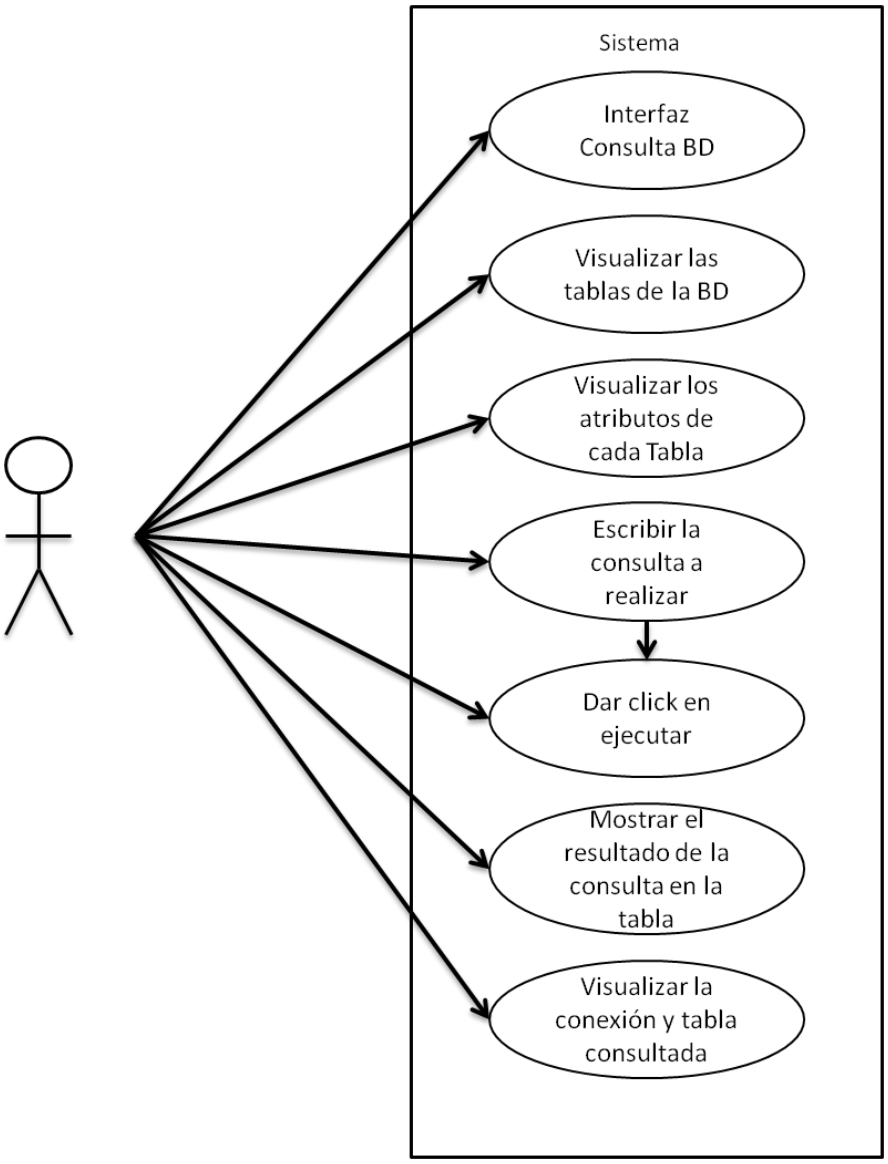

Figura 18 Caso de uso Consulta a BD (Elaboración propia, 2012).

6.1.1.3.4 Caso de uso. Consulta a HDBS.

La interfaz Consulta BD permite realizar consultas heterogéneas mientras tanto las conexiones se realicen previamente por el usuario, antes de realizar la consulta el usuario procederá a seleccionar la tabla que dese consultar, el usuario debe tener los conocimiento de las estructuras de consultas y subconsultas en Lenguaje MySQL (ver Figura 19).

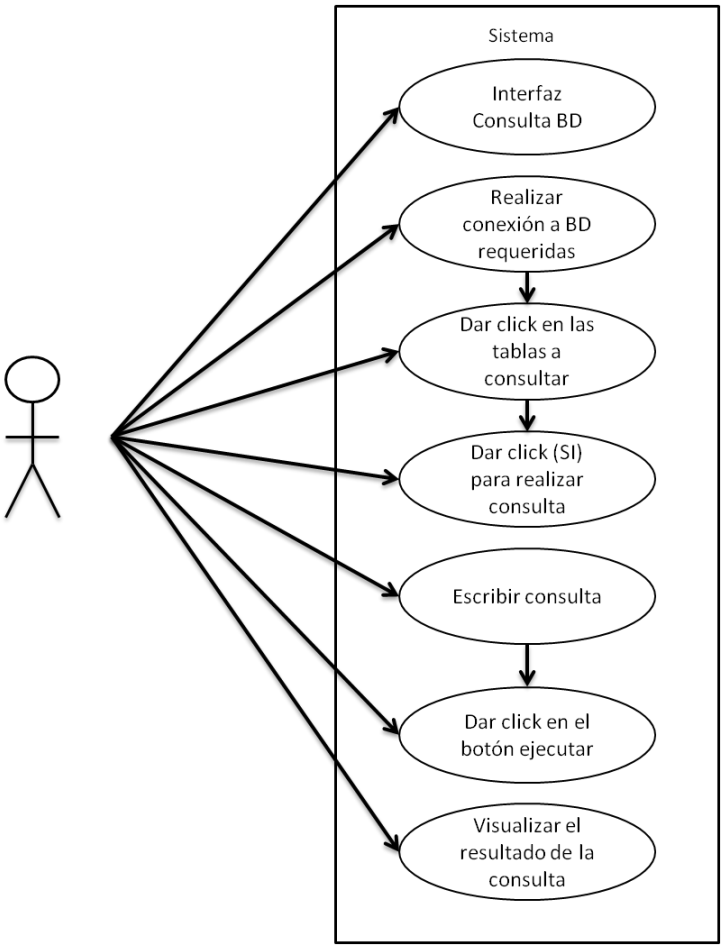

Figura 19 Caso de uso Consulta a HDBS (Elaboración propia, 2012).

6.1.1.3.5 Diseño de interfaces del DBMS.

Para el diseño de las interfaces del sistema se consideraron las necesidades del usuario final, por lo que se diseñaron bocetos de fácil manejo y compresión de manera amigable y de sencilla ejecución, cubriendo así las necesidades básicas.

6.1.1.4 Paso 4. Plantear algoritmos que permitan una estandarización y automatización de consultas de BD para cada Lenguaje Manejador de BD y consultas a HDBS.

Para la realizar el planteamiento de los algoritmos, se tomaron en cuenta las diferencias de estructura de consultas entre los Lenguajes Manejadores de BD (Cuadro 8), para así poder, desarrollar un algoritmo que permita realizar los cambios pertinentes sin que intervenga el usuario.
El algoritmo se divide en dos fases en la primera fase se hacen consultas a bases de datos especificas sin considerar las bases de datos heterogéneas, en la segunda fase se consideran las consultas a bases de datos heterogéneas.

Pasos para hacer consultas de la primera fase:

a) Se deben conectar las cuatro bases de datos.
b) Se identifican los nombres de tablas, atributos y tipo de datos.
c) Escribir la consulta en MySQL.
d) Por ser la consulta una cadena, se hace una división de sub cadenas.
e) Identificar las sub cadenas con los nombres de las tablas y atributos.
f) Con esos nombres se hace la comparación con los datos que se obtuvieron del inciso b) y se determina a que base de datos se debe hacer la consulta.
g) Se realiza la transformación a la sintaxis y semántica de acuerdo a como lo requiere el SMBD ya determinado.
h) Después de la transformación se ejecuta la consulta a la base que ya se determinó.

A continuación se muestra una ejecución Figura 20.

El algoritmo para hacer las consultas de la segunda fase se describe a continuación:

a) Se deben conectar las cuatro bases de datos.
b) Se identifican los nombres de tablas, atributos y tipo de datos.

c) Escribir la consulta en MySQL para un Query a bases de datos heterogéneas.
d) Por ser la consulta una cadena, se hace una división de sub cadenas.
e) Identificar las sub cadenas con los nombres de las tablas y atributos.
f) Se plantean consultas para bases de datos especificas a partir del conocimiento que se tiene de la identificación de la bases de datos y atributo.
g) De las consultas que se tienen se reestructura la consulta con las tablas que se tienen en memoria.
h) Se ejecuta la consulta.

En la Figura 21 se ejecuta un ejemplo de una consulta realizada

```
Conexion a la base de datos de trabajo
Conectado a: postgres
Base de Datos: renta_vehiculos
postgres
Condicion true
drop TABLE if exists cliente;
CREATE TABLE cliente (rfc varchar(20), nombre varchar(50), calle varchar(100), notarjeta int)
INSERT INTO cliente (rfc, nombre, calle, notarjeta) VALUES (?, ?, ?, ?);
CREATE TABLE clientes (idClientes int, contrasenia char(10), nombre varchar(50),
           numeroTelefonico varchar(45), calle varchar(45), colonia varchar(45), numero int, rfc varchar(1))
INSERT INTO clientes (idClientes, contrasenia, nombre, numeroTelefonico, calle, colonia, numero, rfc)
           VALUES (?, ?, ?, ?, ?, ?, ?, ?);

ORACION  REALIZADA : select cliente.nombre , clientes.nombre , clientes.calle
                     from cliente , clientes where cliente.nombre= clientes.nombre
```

Figura 20 Ejecución del Algoritmo de consultas a BD (Elaboración propia, 2012).

```
Conectado a: postgres
Base de Datos: renta_vehiculos
postgres
Condicion true
Oracion de postgresql 1: select marca , modelo  from renta where  fechaentrada= curdate()

subcadena -->select
subcadena -->marca
subcadena -->,
subcadena -->modelo
subcadena -->
subcadena -->from
subcadena -->renta
subcadena -->where
subcadena -->
subcadena -->fechaentrada=
subcadena -->curdate()
nombre tabla   -->renta
Son iguales   -->renta con  renta
ORACION  REALIZADA : select marca , modelo  from renta where  fechaentrada= current_date
```

Figura 21 Ejecución del Algoritmo de consultas a HDBS (Elaboración propia, 2012).

La figura 22 muestra esquemáticamente el algoritmo que se describe en la fase uno.

.

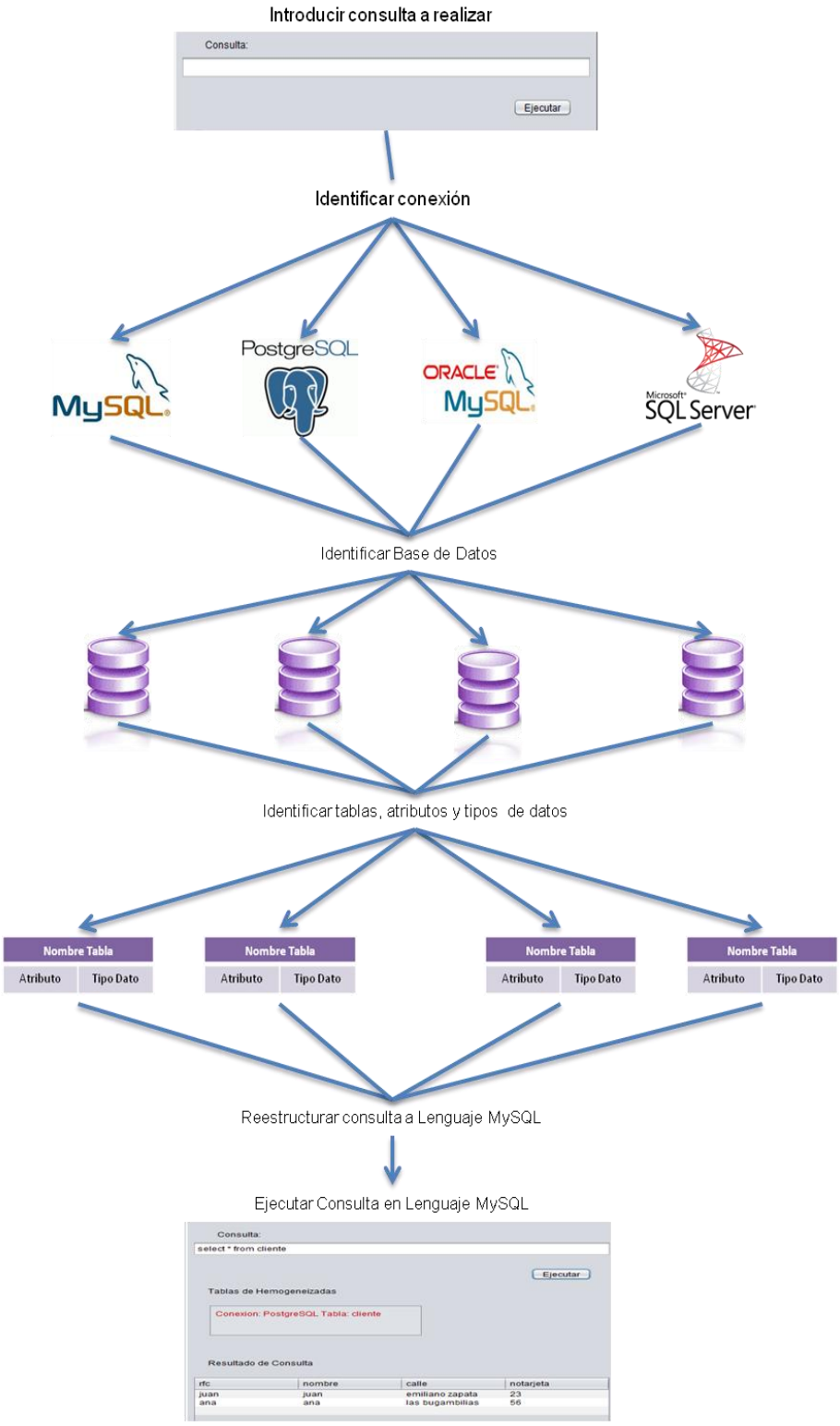

Figura 22 Algoritmo de consulta a BD (Elaboración propia, 2012).

Mientras la Figura 23 muestra la fase dos, las consultas se pueden escribir seleccionando las tablas y atributos para plantear el QUERY a consultas a bases de datos heterogéneas.

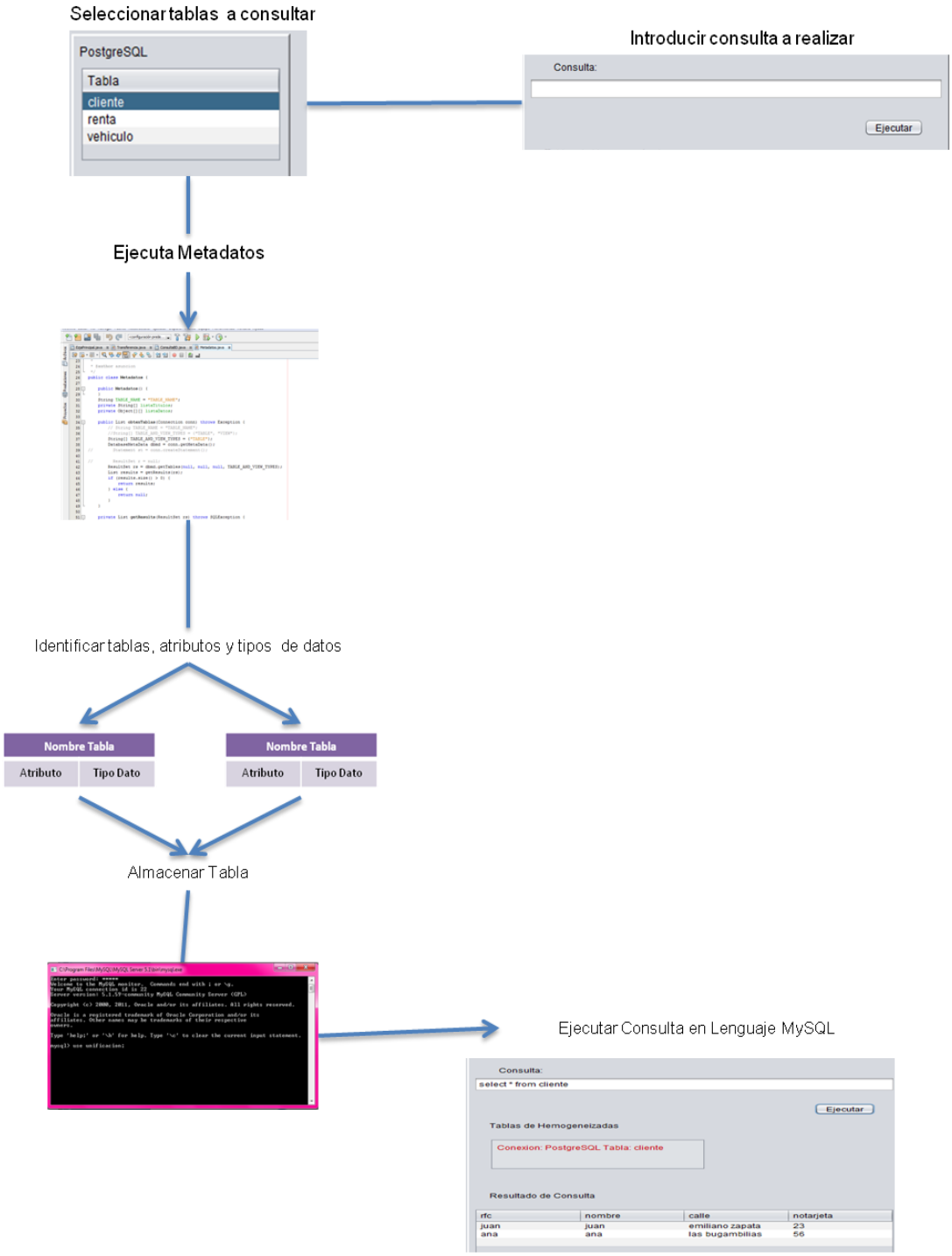

Figura 23 Algoritmo de consulta a HDBS (Elaboración propia, 2012).

6.1.1.5 Paso 5. Programar los algoritmos planteados en JAVA.

Para desarrollar y programar estos algoritmos se utilizo el lenguaje de programación de JAVA bajo la plataforma de NetBeans IDE 7.0.1 (ver Figura 24).

Figura 24 Netbeans IDE 7.0.0. (Netbeans IDE 7.0.1).

Netbeans proporciona un entorno de desarrollo integrado para el lenguaje de programación java. Esta plataforma cuenta con numerosos módulos (clases java) que permiten el desarrollo de aplicaciones con las APIs de Netbeans y un archivo especial (manifest file). Las aplicaciones desarrolladas a partir de módulos pueden ser extendidas agregando nuevos módulos, debido a que los módulos se desarrollan de forma independiente (ver Figura 25).

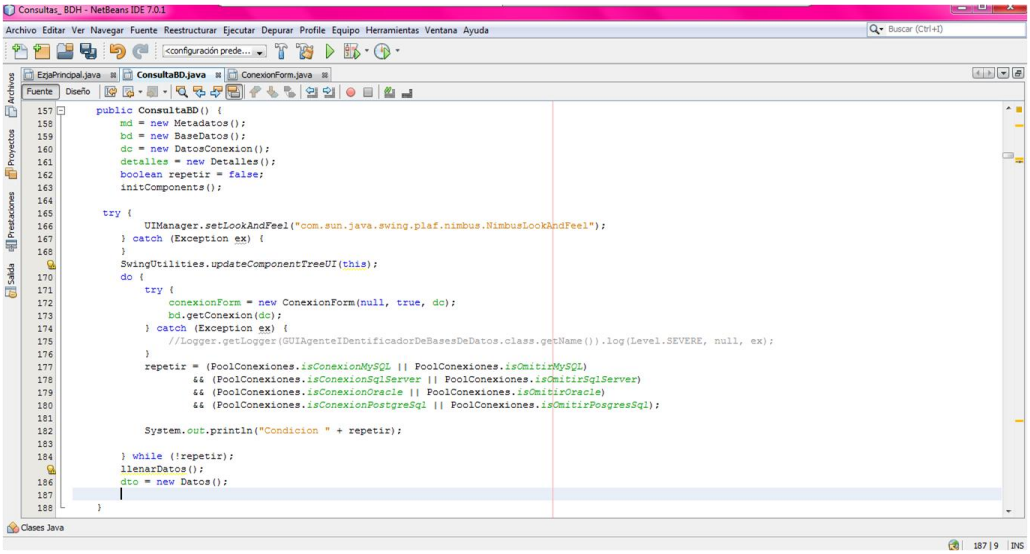

Figura 25 Parte del código de identificación de conexión a BD (Elaboración propia, 2012).

6.1.1.6 Paso 6: Implementar de los algoritmos en lenguaje de programación JAVA.

El desarrollo del sistema manejador de BD se creó paqueterías funcionales para la ejecución de los algoritmos plateados (ver Figura 26).

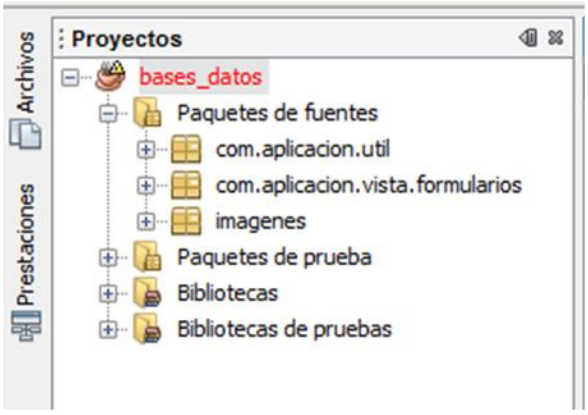

Figura 26 Paquetería (Elaboración propia utilizando netbeans, 2012).

- Paquete com.aplicacion.util: contiene los algoritmos de conexión a las BD de los diferentes manejadores de BD (ver Figura 27).

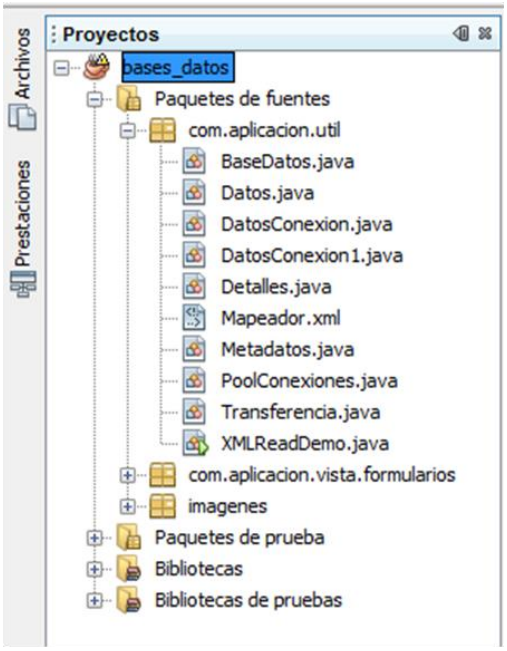

Figura 27 Paquete com.aplicacion.util (Elaboración propia utilizando netbeans, 2012).

- Paquete com.aplicacion.vistas.formularios: contiene el diseño de las interfaces (ver Figura 28).

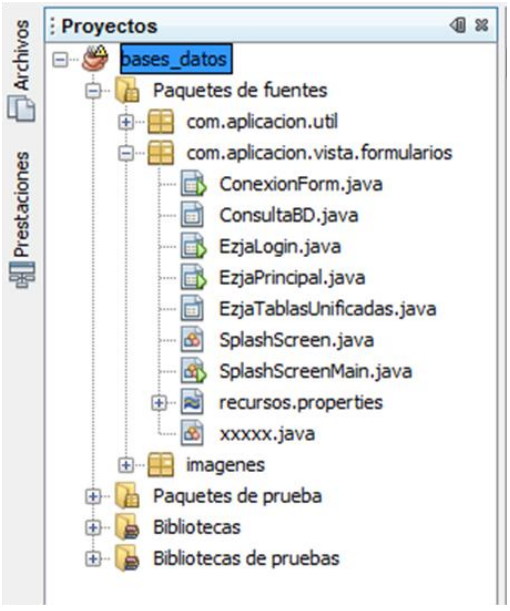

Figura 28 Paquete com.aplicacion.vistas.formularios (Elaboración propia utilizando netbeans, 2012).

- Paquete Bibliotecas: contiene los drivers para realizar las conexiones a los diferentes manejadores de BD (ver Figura 29).

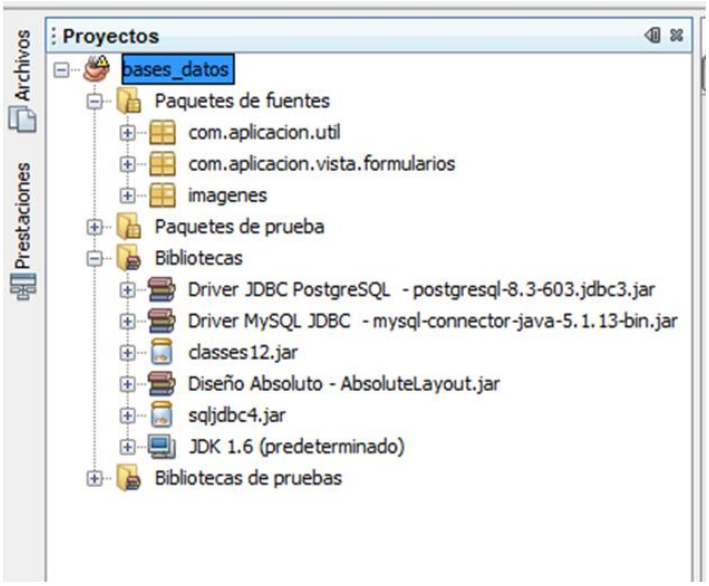

Figura 29 Paquete bibliotecas (Elaboración propia utilizando netbeans, 2012).

6.1.1.7 Paso 7: Validación del Sistema Manejador de Bases de Datos.

Para realizar la validación del Sistema Manejador de BD, se utilizaron las consultas más frecuentes de nivel básico, intermedio y avanzado. Con la realización de dichas consultas se verificara el nivel de satisfacción de ejecución de la consulta.

Para corroborar el cumplimiento de la ejecución se realizaron consultas individuales en cada manejador probando los datos arrojados por la consulta realizada.

VII. Resultados

Se programaron los algoritmos planteados para el desarrollo de la aplicación del DBMS, ofrece la estandarización y automatización de consultas en lenguaje MySQL de diferentes manejadores de BD, permitiendo la realización de consultas en una sola interfaz.

Para demostrar la funcionalidad de la aplicación se mostrarán imágenes de la interfaz diseñada para la realización de consultas. Mostrando de forma general su funcionalidad, adames de una barra de menú para un fácil manejo para el usuario (ver Figura 30).

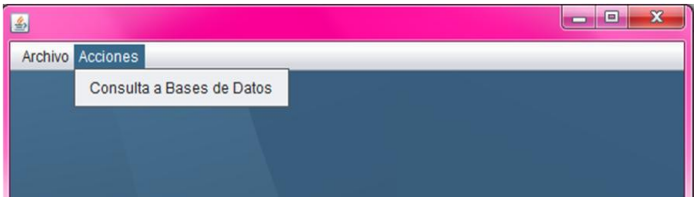

Figura 30 Interfaz. Barra de Menú (Elaboración propia, 2012).

El proceso de conexión a BD consiste en capturar los datos la BD que desea consultar el usuario (ver Figura 31).

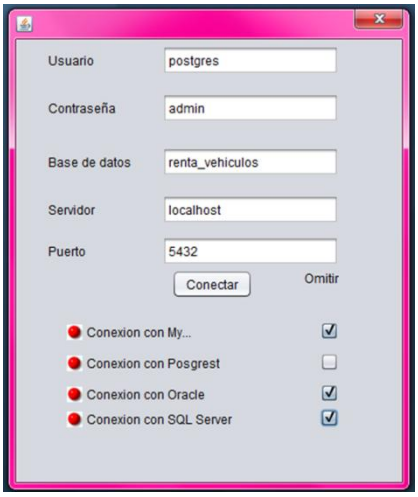

Figura 31 Interfaz. Conexión a BD (Elaboración propia, 2012).

La interfaz de conexión verificara los datos capturados por el usuario, al momento de realizar esta validación aparecerá la interfaz de Consulta BD la cual permite la interacción con los datos de la BD (ver Figura 32.).

La interfaz de Consulta BD proporciona los nombres de las tablas contenidas en la BD al que se conecto, además de esquemas de las estructuras de las tablas con los nombres de los atributos y tipos de datos (ver Figura 32). El botón ejecutar, realizará la consulta escrita por el usuario por lo que primero se tiene que ingresar la sentencia, la tabla de resultados mostrará la consulta siempre y cuando la tabla se encuentre en la BD (ver Figura 33.).

Para realizar una consulta a HDBS, el usuario tiene que conectarse a las BD que requiera para la consulta (ver Figura 34), después selecciona y dará click en el mensaje ” Desea trabajar con esta tabla” , si este paso es omitido la consulta no se realizará (ver Figura 35) en las tablas a consultar, posteriormente el usuario escribirá la consulta y dará click en ejecutar, el resultado de la consulta se mostrará en la tabla de resultados (ver Figura 36).

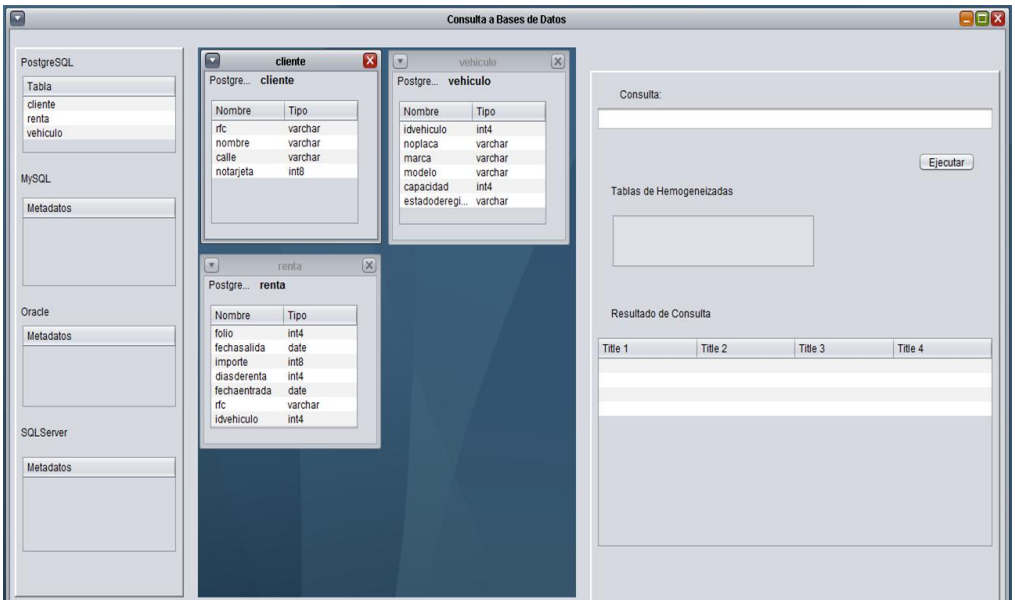

Figura 32 Interfaz. Consulta BD (Elaboración propia, 2012).

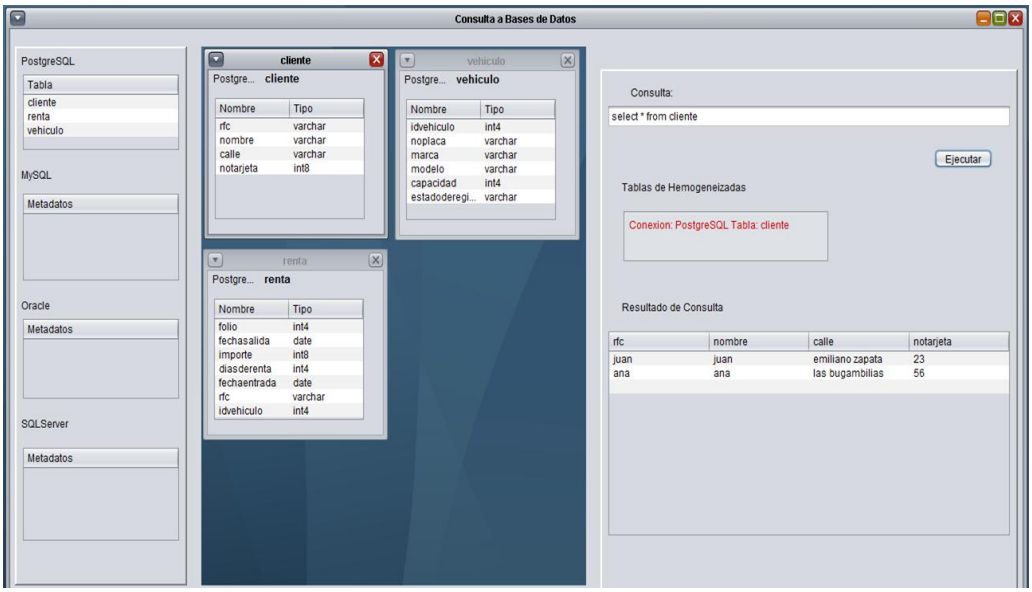

Figura 33 Interfaz. Ejecutar Consulta (Elaboración propia, 2012).

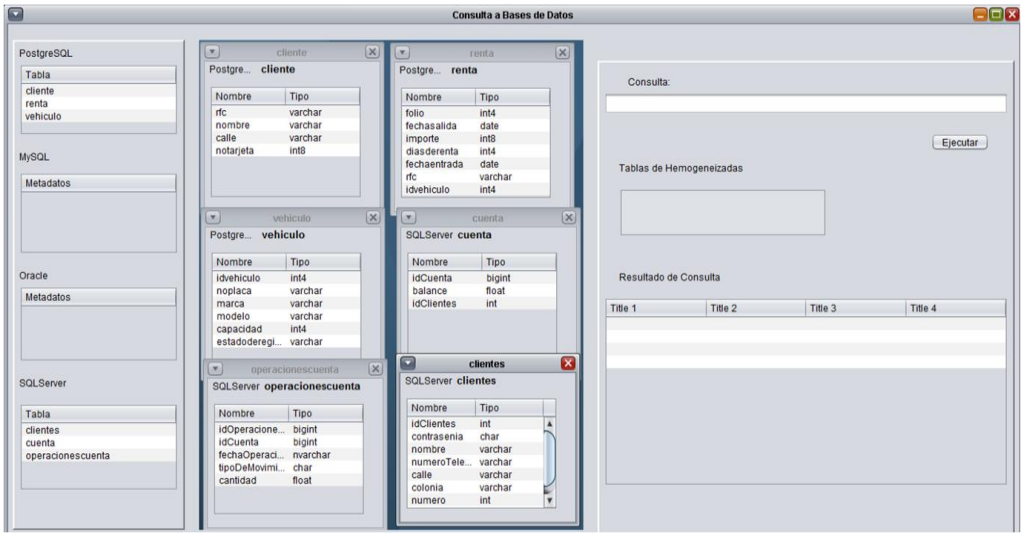

Figura 34 Interfaz. Visualización de dos BD conectadas el mismo tiempo (Elaboración propia, 2012).

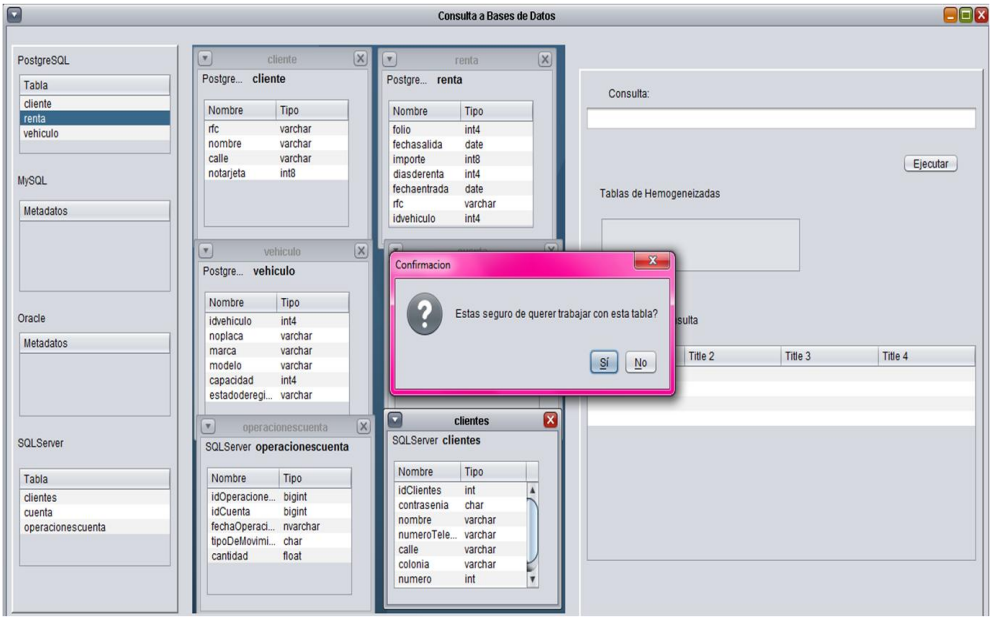

Figura 35 Interfaz. Visualización de tablas seleccionadas para consulta de HDBS (Elaboración propia, 2012).

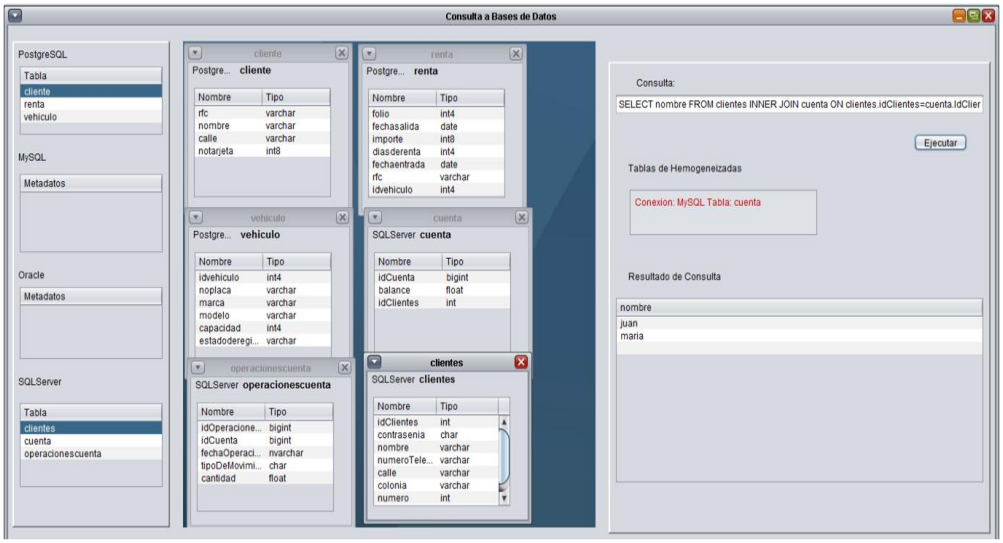

Figura 36 Interfaz. Visualización de consulta realizada a una HDBS Heterogéneas (Elaboración propia, 2012).

El desarrollo de los algoritmos se contemplaron las consultas más frecuentes, sin embargo existen consultas que no se consideraron en el desarrollo del sistema es por eso que se elaboraron estadísticas de funcionalidad obteniendo las siguientes gráficas.

Las gráficas están dividas por lenguaje y a su vez clasificadas por niveles, para realizar esta clasificación se tomo en cuenta el nivel de complejidad de los operadores en las estructuras de consultas, y como resultado la clasificación de consultas de nivel básico son sentencias que no ocupan operadores, las consultas de nivel intermedio son las sentencias que ocupan los operadores WHERE, ORDER BY, DESC, ASC, SUM, AVG, MAX, MIN, COUNT y DISTINCT, mientras que el nivel avanzado son las sentencias con operadores INNER JOIN, ON, IN, EXISTS y LEFT JOIN . Del Lenguaje MySQL no se realizaron gráficas ya que las consultas están basadas en este lenguaje por lo que es innecesario.

- Estadísticas del Lenguaje PostgreSQL: Consultas nivel Básico (ver Figura 37).

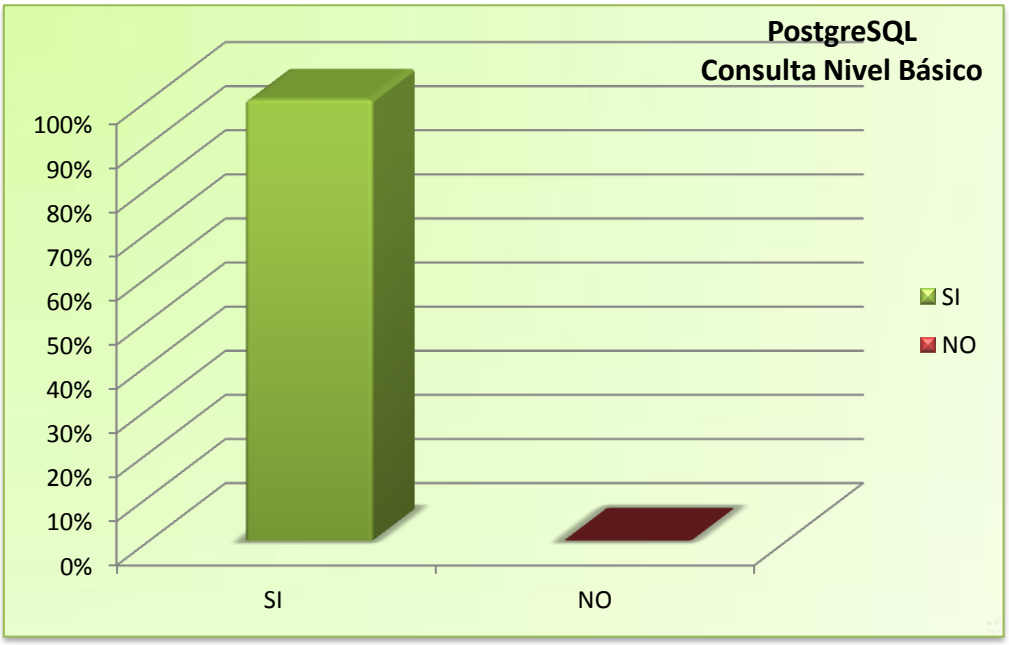

Figura 37 Gráfica de funcionalidad de PostgreSQL de consultas nivel básico (Elaboración propia, 2012).

- Estadísticas del Lenguaje PostgreSQL: Consultas nivel Intermedio (ver Figura 38).

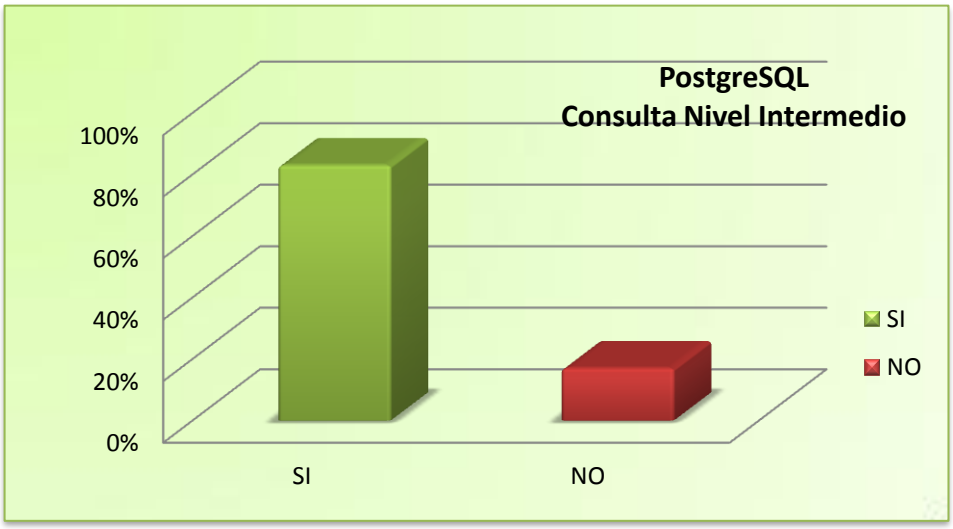

Figura 38 Gráfica de funcionalidad de PostgreSQL de consultas nivel intermedio (Elaboración propia, 2012).

- Estadísticas del Lenguaje PostgreSQL: Consultas nivel Avanzado (ver Figura 39).

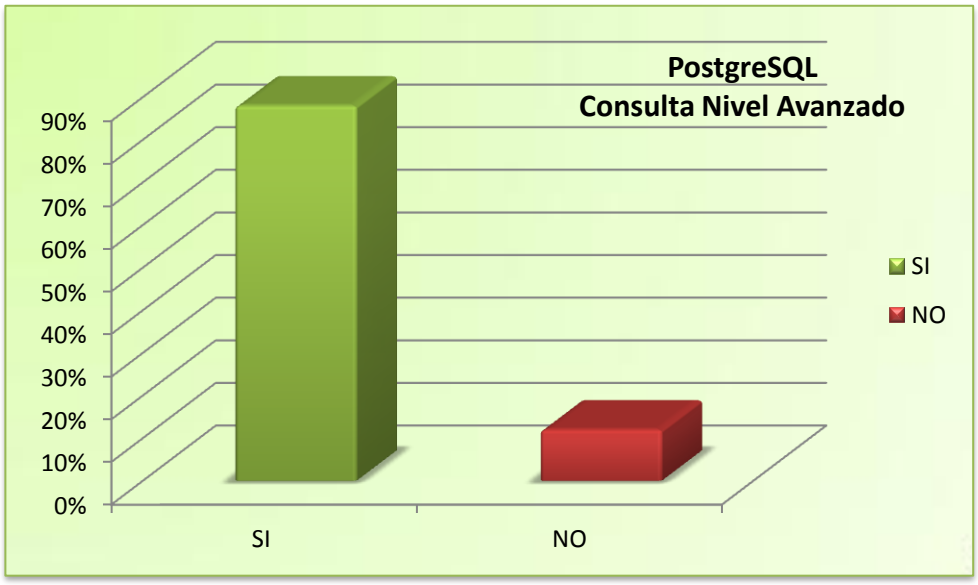

Figura 39 Gráfica de funcionalidad de PostgreSQL de consultas nivel avanzado (Elaboración propia, 2012).

- Estadísticas del Lenguaje Oracle: Consultas nivel Básico (ver Figura 40).

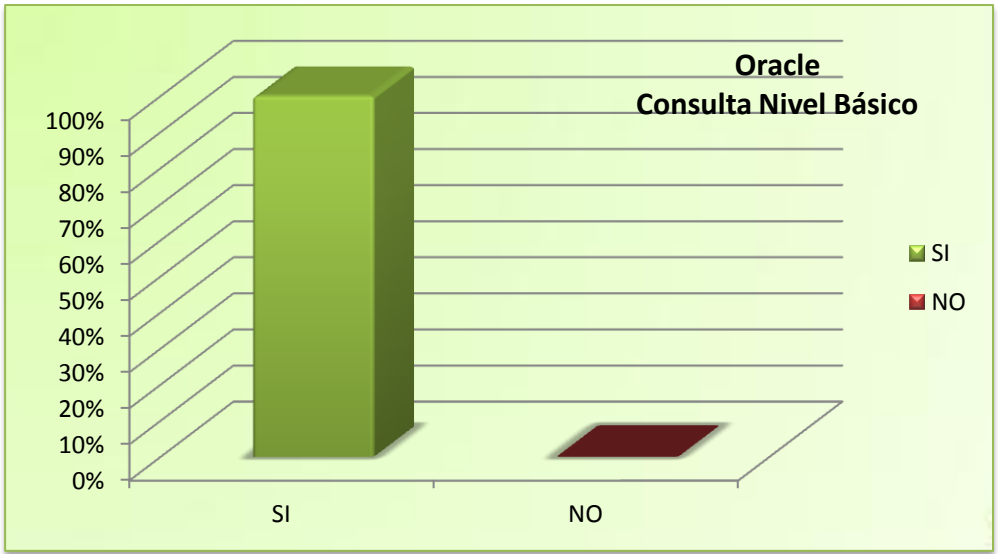

Figura 40 Gráfica de funcionalidad de Oracle de consultas nivel básico (Elaboración propia, 2012).

- Estadísticas del Lenguaje Oracle: Consultas nivel Intermedio (ver Figura 41).

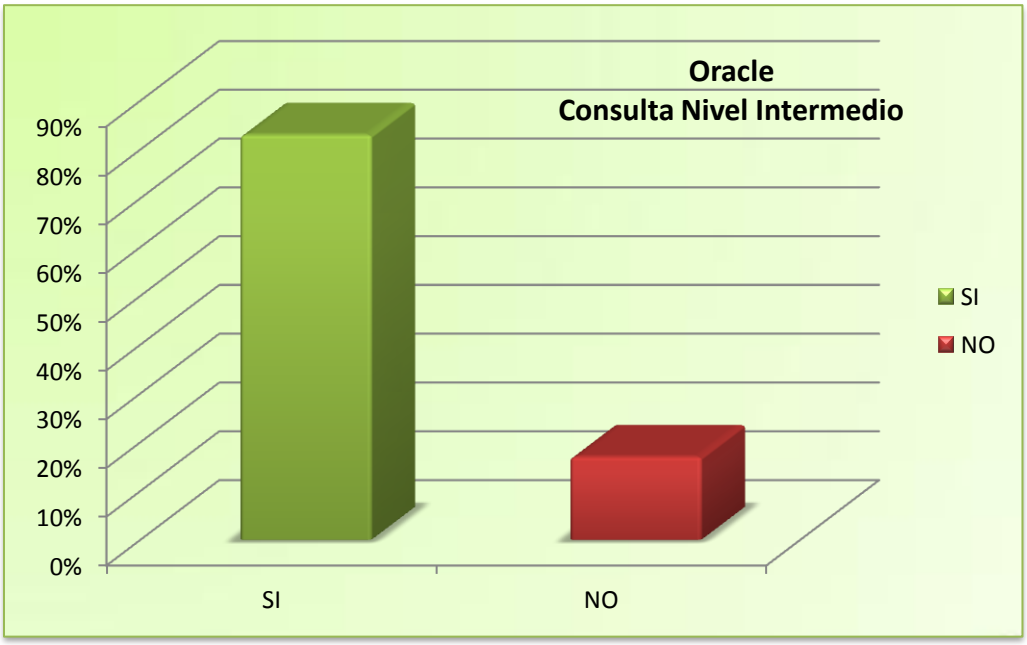

Figura 41 Gráfica de funcionalidad de Oracle de consultas nivel intermedio (Elaboración propia, 2012).

- Estadísticas del Lenguaje Oracle: Consultas nivel Avanzado (ver Figura 42).

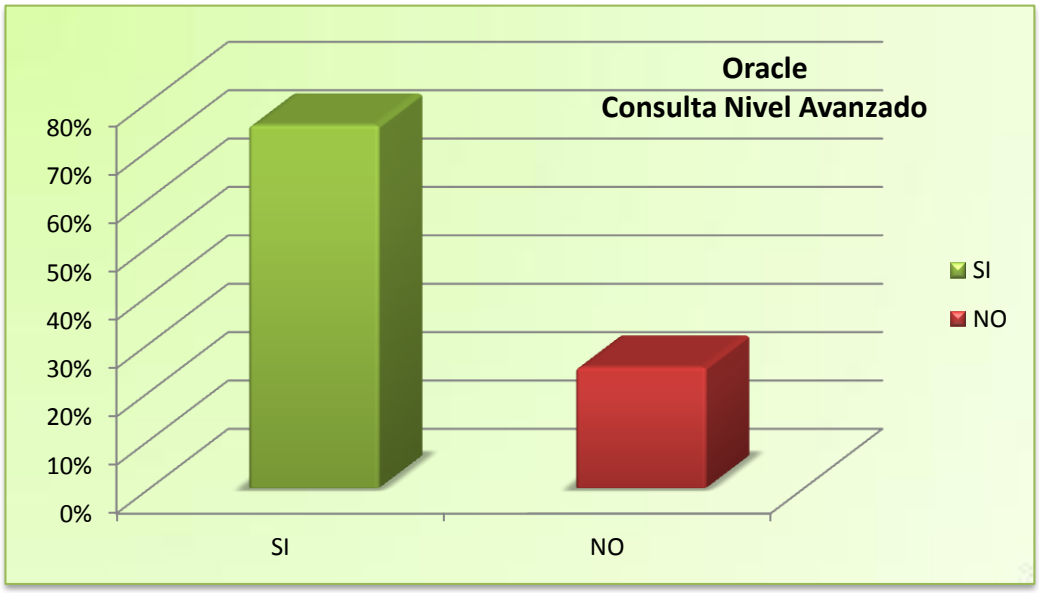

Figura 42 Gráfica de funcionalidad de Oracle de consultas nivel avanzado (Elaboración propia, 2012).

- Estadísticas del Lenguaje SQLServer: Consultas nivel Básico (ver Figura 43).

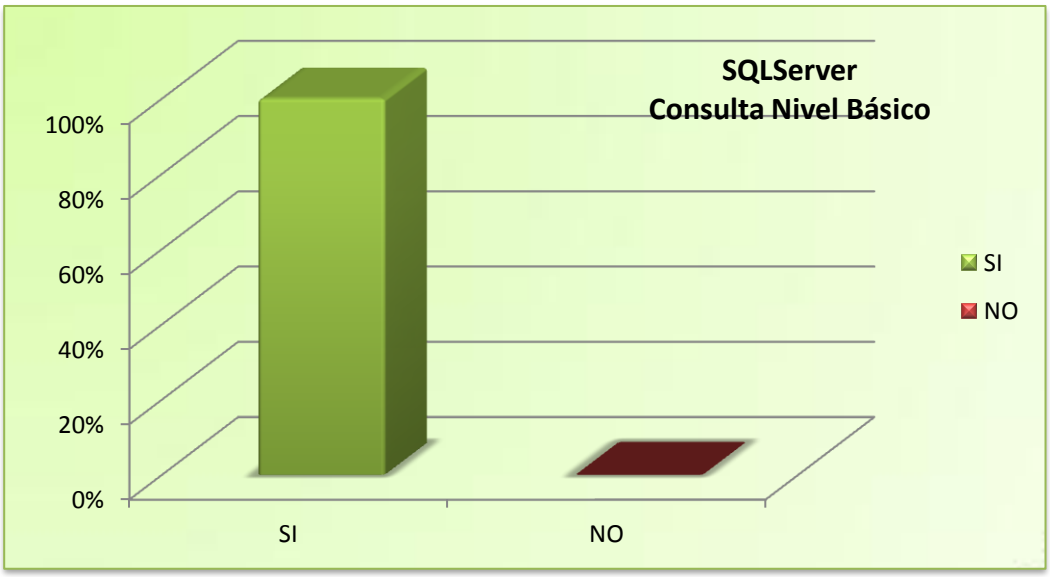

Figura 43 Gráfica de funcionalidad de SQLServer consultas nivel básico (Elaboración propia, 2012).

- Estadísticas del Lenguaje SQLServer: Consultas nivel Intermedio (ver Figura 44).

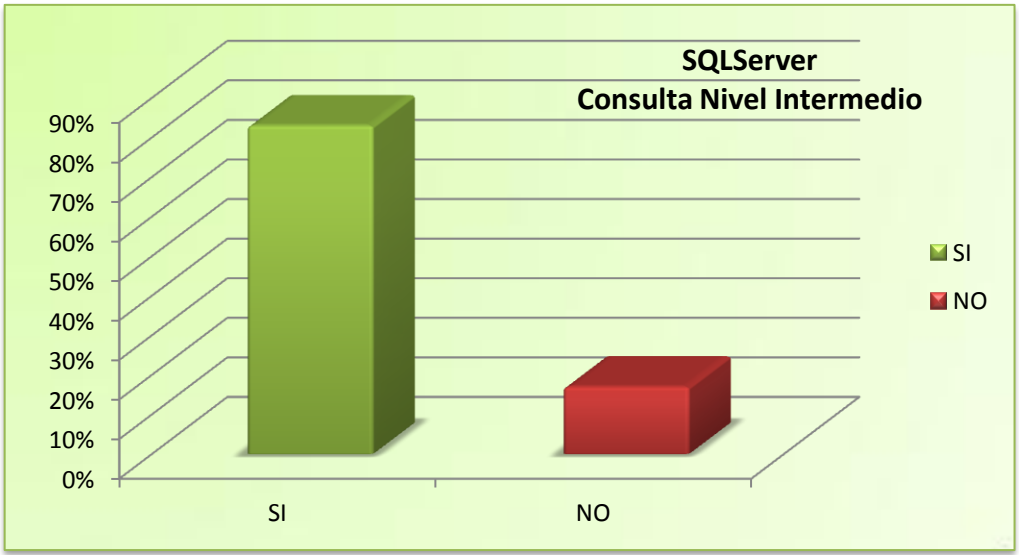

Figura 44 Gráfica de funcionalidad de SQLServer consultas nivel intermedio (Elaboración propia, 2012).

- Estadísticas del Lenguaje SQLServer: Consultas nivel Avanzado (ver Figura 45).

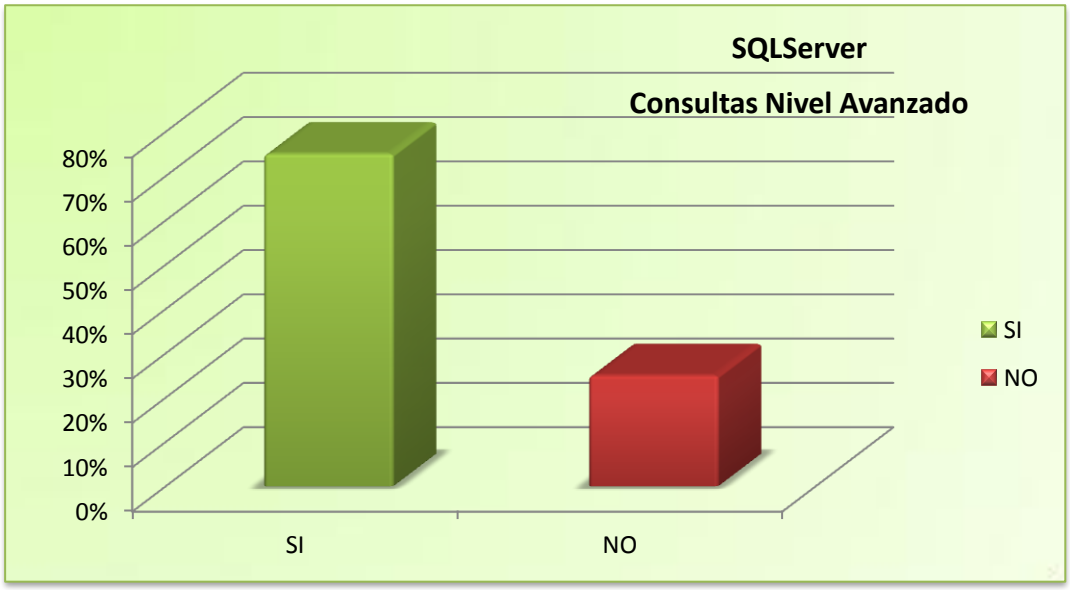

Figura 45 Gráfica de funcionalidad de SQLServer consultas nivel avanzado (Elaboración propia, 2012).

VIII. Discusión

Para la realización del sistema propuesto fue necesario analizar, identificar y conocer las diferencias entre las estructuras de consultas de cada manejador de BD, para así cubrir las principales necesidades del usuario al momento de ejecutar la sentencia.

Algunas de las problemáticas detectadas fueron; cuando se realizan consultas con fecha (obtenerla del sistema por comandos del Lenguaje de BD), la sentencia se redacta de manera diferente como en PostgreSQL, Oracle y SQLSever, además el formato de la fecha en una consulta determinada y el uso de comillas simples (‘’). Otra problemática fueron los problemas de interoperabilidad e integridad de los datos al momento de realizar consultas a HDBS, además que las estructuras de subconsultas ya que son diferentes para cada manejador y que algunos operadores no funcionan entre los Lenguajes de BD. Es por eso que el planteamiento del algoritmo debe cumplir con estos requerimientos para efectuar las consultas que dese el usuario.

Antes de realizar las consultas a HDBS, se ejecutaron consultas en estructura del Lenguaje de MySQL en los demás manejadores para idear un algoritmo de estandarización y automatización de consultas, hasta este punto se han cumplido los tres primeros objetivos particulares planteados.

Para la codificación de los algoritmos plateados para estandarización y automatización de consultas en lenguaje MySQL. Se utilizó la plataforma de Netbeans Lenguaje JAVA para programar por medio de paquetes facilitando el desarrollo del sistema. Se eligió el modelo en espiral para el desarrollo del sistema debido que permite crear sistemas con un ciclo de vida indefinido en base a los requerimientos y opiniones del usuario.

Para tener una mejor perspectiva de la funcionalidad del DBMS programado se realizaron estadísticas de funcionalidad las consultas se dividieron en básicas, intermedias y avanzadas, tomando en cuenta las condiciones, operadores utilizados y el número de tablas que intervienen en la consulta. Las pruebas se resumen en el Cuadro 9 y gráficamente en la Figura 44 esto con el fin de tener una mejor visualización de la funcionalidad de los algoritmos.

NIVELES DE CONSULTA		EJECUCION	
		SI	NO
PostgreSQL	Básicas	100%	0%
	Intermedias	83%	17%
	Avanzadas	88%	12%
Oracle	Básicas	100%	0%
	Intermedias	83%	17%
	Avanzadas	75%	25%
SQLServer	Básicas	100%	0%
	Intermedias	83%	17%
	Avanzadas	75%	25%

Cuadro 9 Funcionalidad de PostgreSQL, SQLServer y Oracle (Elaboración propia, 2012).

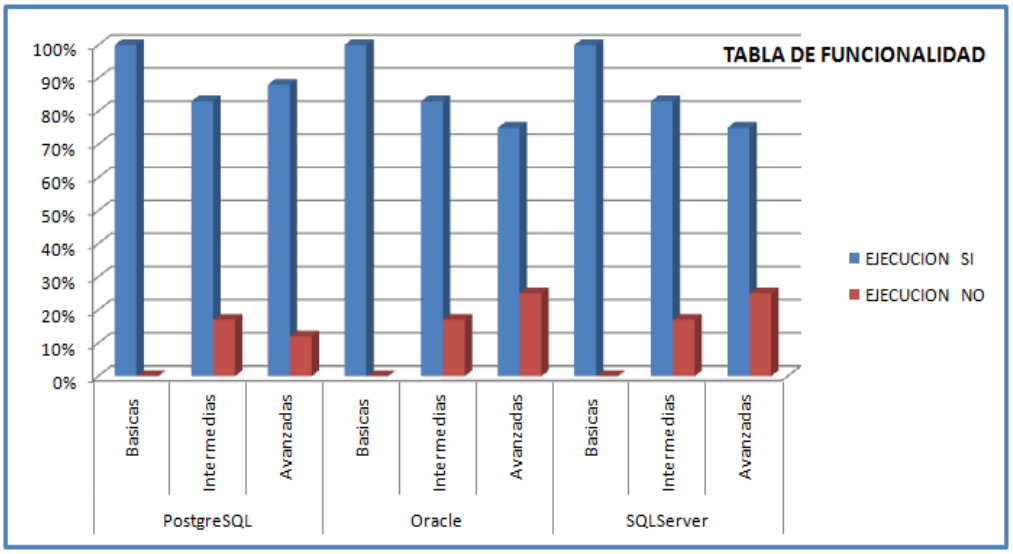

Figura 46 Gráfica de funcionalidad de PostgreSQL, SQLServer y Oracle (Elaboración propia, 2012).

Como se puede ver a nivel básico de los lenguajes PostgreSQL, Oracle y SQLServer se cumple el 100% de las consultas ejecutadas, estos resultados son alentadores para señalar que el planteamiento es aceptable.

Con respecto a las consultas con nivel intermedio, más del 80% de consultas son ejecutadas con éxito en los lenguajes PostgreSQL, Oracle y SQLServer, estos resultados aunque son aceptables implicará que se debe poner una especial atención, ya que son las que más se emplean en las consultas requeridas por los usuarios.

Con respecto al nivel avanzado el lenguaje PostgreSQL cumple con más del 80% en ejecución de consultas, por otro lado Oracle y SQLServer cumplen con más del 70% de la ejecución.

Al no cumplir con el cien por ciento de las consultas en los tres niveles implica que se tenga que revisar por qué no se pudieron realizar al 100%. De acuerdo a la experiencia que se adquirió para realizar este trabajo se estima que hay que revisar otras palabras reservadas del lenguaje SQL que posiblemente estén influyendo en los resultados obtenidos, además de analizar otros aspectos de semántica y sintaxis para la escritura de la consulta. Sin embargo con los resultados obtenidos se considera que esta etapa de la investigación tiene resultados aceptables y se tienen avances para que otros compañeros sigan con los trabajos de desarrollo de afinación y mejoramiento de algoritmos.

Logrando cumplir los objetivos particulares planteados, con el desarrollo del sistema DMBS a HDBS con un algoritmo funcional para la mayoría de los casos en las estructuras de consulta de los diferentes manejadores de Bases de Datos.

IX. Conclusiones

- El sistema DBMS desarrollado con los algoritmos propuestos reduce los gastos de operación por la contratación de personal especializado.

- La Estandarización y Automatización de consultas SQL en los Manejadores de bases de datos PostgreSQL, Oracle, SQLServer y MySQL si es posible con algoritmos especializados a pesar que no se pudo lograr el 100% de las consultas.

- Las consultas a bases de datos heterogéneas con la metodología planteada es operable y debe ser afinada para llegar a un 100% de satisfacción.

- El sistema permite realizar consultas en tiempo real.

- Se disminuye el tiempo de consulta a realizar, debido a la estandarización y automatización propuesta.

X. Recomendaciones

- Se recomienda profundizar la investigación de manejo de operadores y estructura de consultas a BD Y HDBS para desarrollar más el algoritmo que permitan aumentar el porcentaje de buena funcionalidad.

- La persona que retome el desarrollo del sistema deberá realizar más evaluaciones y pruebas de consultas, así como automatizar los procesos para un fácil manejo del usuario.

XI. Referencias Bibliográficas

- Alfredo Weitzenfeld (2005), Ingeniería de Software Orientada a Objetos con UML Java e Internet , Thomson Editores.
- Benet Campderrich Falqueras (2003), Ingeniería del software Biblioteca multimedia: Informatica, Editorial UOC.
- Fowler, M., Kend, A.S., González, R.J., (1999), “UML gota a gota”, México, Pearson Educacion, S.A.
- Ian Sommerville (2006), Ingeniería de Software, Pearson Addison Wesley, 7ª Edición.
- Jose Maria Cavero Barca (2005), Aspectos filosóficos, psicológicos y metodológicos de la informática Volumen 8 de Ciencias experimentales y tecnología, Liberia-Editorial Dykinson.
- Laurent Debrauwer, FienVan Der Heyde (2005), UML 2: iniciación, ejemplos y ejercicios corregidos Colección Recursos informáticos Recursos Informáticos, Ediciones ENI.
- Silberschatz, Korth , Sudarshan (2006), Fundamentos de Bases de Datos, Madrid, 5ta Edición, Mac Graw Hill.
- Simon Bennett, Steve McRobb y Ray Farmer (2007), Análisis y diseño orientado a objetos de sistemas: Usando UML, Madrid, 3ra Edición, Mc Graw Hill.
- Sonia Jaramillo Valbuena, Sergio Augusto Cardona Torres, Dumar Antonio Villa Zapata (2008), Programación Avanzada en Java, ELIZCOM S.A.S
- Alice Y. H. Tsai (1990), Sistema de Bases de Datos Administracion y Uso, Prentice Hall
- Michael V. Mannino (2007), Administración de Bases de Datos Diseño y Desarrollo de aplicaciones, 3ra Edición, Mc Graw Hill
- I. T. Hawryszkiewycz (1994), Análisis y Diseño de Bases de Datos, 1ra reimpresión, Megabyte Noriega Editores
- Ramakrishnan, Gehrke (2007), Sistema de Gestión de Bases de Datos, 3era edición Madrid, Mc Graw Hill

- Ramez A. Elmasri, Shamkant B. Navathe (2002), Fundamentos de Sistemas de Bases de Datos, 3ra edición, Pearson Addison Wesley, Madrid
- Pons Capote Capote (2005), Introducción a las bases de datos: el modelo relacional,reimpresión, Editorial Paraninfo
- Costal Costa Dolors (2009), El modelo relacional y el algebra relacional, Editorial UOC.
- Nevado Cabello Ma. Victoria (2010), Introducción a las bases de datos relacionales, Editorial Visión Libros.
- Hansen Gary W. y Hansen James V. (1998), Diseño y administración de qbases de datos, 2da edición, Prentice Hall
- Fray León Osorio Rivera (), Bases de Datos relacionales: teoría y práctica, ITM
- Enríquez Zarate José Asunción (2010),"Avances en Sistemas Inteligentes en México" Ed. Miguel González Mendoza y Oscar Herrera Alcántara.
- Pastor Lopez Oscar, Blesa Pons Pedro (2000), Gestión de Bases de Datos, Ed. Universidad Politécnica de Valencia.
- Lozano Pérez María Dolores (2000), Ingeniería del Software y bases de datos: tendencias actuales, Universidad de Castilla La Mancha, Isidro Ramos Salavert, María Dolores Lozano Pérez.
- Date C. J. (2001), Introducción a los Sistemas de Bases de Datos, 7a edición, Pearson Prentice Hall, México.
- Quintana G., Marques M., Aliaga J. L. y Aramburu M. J. (2008), Aprende SQL, Universidad Jaume I.
- González Alfons (1999), SQL Server: programación y administración, Alfaomega ra-ma.
- Oppel Andy, Sheldon Robert (2008), SQL: a beginner´s guide, 3ra edición ilustrada, McGraw Hill Profesional.
- Heurtel Olivier (2009), PHP y MySQL Domine el desarrollo de un sitio Web dinámico e interactivo, Ediciones ENI
- León (1999), SQL, Tata McGraw-Hill Education.

- Ullman Larry (2006),MySQL Visual quickstart guide, 2da edición, Peachpit Press
- Douglas Korry, Doluglas Susan (2003), PostgreSQL: a comprehensive guide to building, programming, and administering PostgreSQL databases, version ilustrada, Sams Publishing.
- Momjian Bruce (2001), PostgreSQL: introduction and concepts, edicion ilustrada, Addison Wesley.
- Visser Susan M. y Wong Bill (2003), Sams teach yourself DB2 Universal Database in 21 days, 2da edición ilustrada, Sams Publishing
- Gabillaud Jerome (2008), SQL Server 2008 SQL Transact SQL, Ediciones ENI.
- López Javier (2001), Oracle: Fundamentos Para El Desarrollo De Aplicaciones Web, Edición ilustrada, MP Ediciones
- Kline Kevin E., Kline Daniel, Brand Hunt (2008), SQL in Nutshell, 3ra edición ilustrada, O´Reily Medina, Inc
- Gunderloy Mike, Chipman Mary (1999), Microsoft SQL Server 7, Unica edición, Ediciones Anaya Multimedia S.A, Madrid.