UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS.
FACULTAD MATEMÁTICA, FÍSICA Y COMPUTACIÓN
LICENCIATURA EN CIENCIA DE LA COMPUTACIÓN

TRABAJO DE DIPLOMA

Asistente para la creación de consultas en ambientes heterogéneos

Autores Javier Rodriguez Ortiz
Yoan Carlos Carralero

Tutores Dr. C. -Ing. Inty Sáez Mosquera
Ing. Adrian Gonzalez Oliva

Año 50 de la Revolución

Curso 2008-2009

DICTAMEN

Hago constar que el presente trabajo fue realizado en la Universidad Central “Marta Abreu” de Las Villas como parte de la culminación de los estudios de la especialidad de Ciencia de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

Firma de los autores

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____	_____	_____
Firma del tutor	Firma del tutor	Firma del jefe del Laboratorio

Exergo

Para mí los grandes no son grandes si no son buenos

Samuel Smiles

DEDICATORIA

A mis padres por confiar siempre en mí y ayudarme a esperar siete largos años para lograr este sueño.

Javier Rodríguez Ortiz.

A Juan y Cacha, ¡mis súper abuelos!

Yoan Carlos

AGRADECIMIENTOS

A mis tutores Inty y Adrián por guiarnos hacia la meta y hablar nuestro mismo idioma.

A mi tía Vivian que aunque no este entre nosotros hoy se hubiese sentido muy feliz en este momento.

A mis familias de Ciego, La Habana y Camagüey por ser tan comprensivos siempre.

A mis amigos de siempre Raya, Alexander y Maykel por compartir tantos buenos y malos momentos conmigo.

A mis amigos de Ciego por estar siempre de mi lado y confiar en mí.

A Maritza por esperar todo este tiempo por mí.

A mis muchos amigos y amigas de noches y días, a todos los que me tendieron su mano alguna vez y empinaron su codo conmigo de buena voluntad, mencionarlos llevaría otra tesis.

A los profesores de estos años que me educaron como profesional.

A la UCLV por ser mi hogar en estos siete años.

Y a todos los que de una forma u otra han hecho posible la realización de este Trabajo de Diploma.

Javier Rodríguez Ortiz

AGRADECIMIENTOS

A los tutores Inty y Adrián por guiarme y brindarme la posibilidad de hacer un trabajo serio.

A mi mamá, mis hermanas, mi tía Erlys y Jerez por confiar en mí cuando yo dudaba: al final parece que sabían en que confiaban.

A mis hermanos Ariel, Iris y Lisbel por poyarme, compartir conmigo las alegrías y los enojos de estos 5 años y sobre todo por hacerme sentir que aún con mis defectos puedo tener verdaderos amigos.

A Dado y su familia por toda la ayuda y comprensión que recibí de ellos.

A todos mis compañeros de año porque sin ellos el camino hasta aquí habría sido mas difícil.

A Yoandy (el chino) por complementar mis conocimientos y por su valiosa ayuda.

A mi ex tutor, oponente y amigo Abel por incluirme en el grupo de base de datos en el CEI.

Yoan Carlos

Índice

LISTADO DE FIGURAS

ABSTRACT

This investigation summarizes the design and the implementation of Module to construct and to solve databases queries. There does a deep analysis of requests, which helps to understand the databases model diagrammatically, as well as the Query Wizard implementation that allow information handling by users. On the other hand, information comes from a Query Manager which is capable to obtaining them from different databases. Module are also capable to showing the whole view of databases structure, in addition to their skills for built queries over one or several Databases, whenever they are in the same RDBMS. Manager also has the responsibility for query running, get the result and sending to the Wizard making the information available for the end users.

INTRODUCCIÓN

Los sistemas de información han evolucionado "incansablemente" desde los mismos inicios de la informática. En los años sesenta, consistían en aplicaciones hechas a la medida ejecutándose sobre un sistema operativo de funcionalidad muy limitada. En ambos casos, la evolución ha tomado caminos diferentes pero complementarios. Los sistemas operativos ofrecen cada vez más funcionalidades, por una parte, por otra se han desarrollado aplicaciones genéricas y reutilizables que implementan funcionalidades que antes residían por completo en aplicaciones hechas a la medida(Hirschheim et al., 2000; King et al., 2006).

Los actuales sistemas de información pueden ser entendidos como un conjunto muy diverso de aplicaciones y subsistemas, que deben ser integrados y coordinados para resolver necesidades particulares en las organizaciones. Las consecuencias directas de la descentralización y desarticulación de las aplicaciones hechas a la medida, así como las ventajas de la reutilización y el aumento de las funcionalidades "cobran" un alto precio en integración y coordinación (Hirsh et al., 2000; Soh et al., 2000).

La diversidad de sistemas, subsistemas y aplicaciones legales que persisten en las organizaciones generó un problema adicional: la heterogeneidad. De tipo estructural (diversidad de formatos en los que los datos son expresados) y semántica (diversidad de significados), este "flagelo" ha conseguido reducir a prácticamente nulos los esfuerzos de las organizaciones para estar mejor informadas (Cuellar Almeida et al., 2004; Hey, 2004).

El acceso a los datos relevantes para la toma de decisiones, su correcta identificación, extracción, transformación, procesamiento y posterior utilización dependen en gran medida, de la capacidad de la organización de relacionar correcta y oportunamente (capital relacional) la información para la toma de decisiones. Adicionalmente, la nueva información generada debe ser organizada, resumida, y estructurada (capital estructural) y pasar a formar parte de las reservas de conocimiento de la organización, más allá de la experticia de los recursos humanos involucrados en este proceso (capital intelectual) (Anderson, 2003; McNamara et al., 2004).

Estos tres niveles (capital relacional, estructural e intelectual) guardan una interesante similitud con el modelo de desarrollo de aplicaciones en tres capas (three-tier application), elemento

común en la mayoría de los sistemas de información modernos. Dividido en tres capas: acceso físico a los datos, lógica de negocios y lógica de presentación, este modelo divide el problema con el objetivo de lidiar con menores niveles de heterogeneidad.

El desarrollo de aplicaciones sobre Web ha producido un último y reciente impulso evolutivo en los sistemas de información. Desde las aplicaciones hechas a la medida, la desarticulación de funcionalidades y su reutilización, hasta la concepción de sistemas de información como servicios, la evolución continúa (Zimanyi et al., 2004; Hauch, 2004; Fernández et al., 2005). Sin embargo, un importante y creciente número de investigaciones en la última década han puesto, tanto mayor énfasis en los aspectos tecnológicos como en los organizativos (Cali et al., 2000; Pawlowski et al., 2001; Partridge, 2002; Patel-Schneider et al., 2002; Leymann et al., 2002; Sáez Mosquera et al., 2003; Wehr, 2003; Rouse, 2004; Hauch, 2004). La proliferación de patrones, buenas prácticas y estándares para la integración empresarial, resumen en buena medida sus principales contribuciones (Duchesi et al., 1998; Kobryn, 1999; Rada et al., 2000; George, 2000; Aalst et al., 2002; Ruijgrok et al., 2003).

La principal limitación de las soluciones comerciales basadas en Web, radica en el reducido número de formatos de representación de datos con los que son capaces de interactuar. En la mayoría de los casos prácticos, las soluciones hacen un supuesto muy fuerte para la amplia diversidad de organizaciones: toda la información que se va a gestionar, será almacenada a partir del momento en que son desplegadas. Equipadas con la casi totalidad de las funcionalidades necesarias para manejar los datos que almacenan, este tipo de soluciones representan apenas una solución de compromiso entre el enfoque centrado en datos (dominante antes de los 90) y el enfoque orientado a procesos (de los 90 en adelante) (George S. Nezlek, 1999; Piccinelli et al., 2001; McGuinness et al., 2003; McCarthy, 2003).

Una propuesta coherente con los años de evolución de los sistemas de información, debe conducir a una forma alternativa de pensar en el problema de la heterogeneidad de las fuentes[1], el acceso a la información, así como su posterior procesamiento, organización, estructuración, almacenamiento y gestión. Los esquemas de integración que desarrollaron el

[1] Aunque se reconocen perfectamente las diferencias, tanto de naturaleza como contenido e implicaciones de los dos tipos de heterogeneidades, el enunciado está enfocado a los formatos de representación de los datos. Su semántica merece mayores consideraciones.

concepto de Wrapper[2], eliminan la necesidad de migración (esquemas y datos) y respetan la autonomía de las fuentes durante el proceso de integración; bajo este enfoque resultan convenientemente combinados, tanto el enfoque centrado en los datos como el orientado a procesos (Vdovjak et al., 2002; Kokkoras et al., 2004).

Una parte importante de los datos empresariales se encuentran en gestores de bases de datos relacionales. El reconocimiento de la necesidad de relacionarlos para conservar parte del contexto en el cual tienen valor de uso y capacidad expresiva, por una parte, por otra la necesidad de representar su significado (semántica), condiciono de muy variadas maneras el auge y evolución de estos gestores (McClure, 1997; DBDS, 2001; Lin, 2003; Ling et al., 2005).

Sin embargo, las relaciones almacenadas en los gestores no son las únicas con importancia para la toma de decisiones. Continua y repetidamente la organización formulan preguntas que crean nuevas relaciones que no fueron tenidas en cuenta en los momentos iníciales. De hecho, los modelos relacionales no están sustentados en la respuesta a las preguntas de gestión, sino en la modelación de las entidades y sus interrelaciones; las que en determinados momentos, correctamente combinadas, adquieren la capacidad de responder este tipo de preguntas.

La respuesta a estas preguntas representa el tipo de información que tiene valor para la toma de decisiones. Los encargados de la toma de decisiones constantemente están formulando preguntas de este tipo, obteniendo respuestas y en base a las respuestas obtenidas, formulan un nuevo grupo de preguntas. Con el tiempo y en un ambiente estable, con un personal estable, un capital estable, un mercado estable y formas y procederes decisionales establecidos (condición de tesis de la organización), las preguntas relacionadas con la toma de decisiones adquieren cierta taxonomía (Kerschberg et al., 2005; decision-making-confidence.com, 2006; Foss et al., 2006).

Esta organización *ad hoc* de preguntas y respuesta, establece un conocimiento tácito en la organización. El primer problema para utilizar, desarrollar y potenciar este conocimiento reside en su marcado carácter implícito (no formalmente definido). Cuando las condiciones de estabilidad cambian y la organización se mueve a escenarios de mayor incertidumbre, la

[2] **Wrapper**: encapsulan las funcionalidades necesarias para acceder e intercambiar datos en entornos heterogéneos: múltiples fuentes y diversidad de formatos.

reevaluación y posterior reconstrucción de sus procedimientos decisionales, tiene que esperar largos tiempos para establecerse nuevamente.

La forma de estructurar este conocimiento para compartirlo y más aún, hacer que forme parte de la organización y no de individuos con determinadas aptitudes y actitudes representa, a juicio de este autor, el segundo y más desafiante problema. La temporalidad de la información (dimensión temporal), las necesidades cambiantes de los entornos en los que son demandas (dimensión contextual), y la combinación de estas dos dimensiones con la perspectiva del analista o encargado de la toma de decisiones, condiciona una tercera dimensión importante: la dimensión cognitiva-deductiva[3].

No ha sido posible hasta el momento en la literatura nacional e internacional consultada, encontrar soluciones que combinen las soluciones tecnológicas (patrones, estándares) y las organizativas (buenas prácticas, modelos de referencia), y permitan la identificación y enmarcación de la dimensión cognitiva-deductiva. En esta dimensión, el encargado de la toma de decisiones (en general, usuario[4] con necesidades de información) debería ser liberado de la responsabilidad de memorizar los contextos y los "espacios temporales" en los que generalmente consume información, para poder concentrarse en identificar nuevas informaciones (nuevas formas de relacionarlas) que aumenten y complementen su conocimiento. La intercepción de las tres dimensiones, define el espacio en el que el usuario es más competente (Perales et al., 2001; Pérez-Acosta, 2002; Universidad de Costa Rica, 2004; Rodríguez Palmero, 2004).

El proceso de obtención de información, al igual que el relacionado con la adquisición de nuevo conocimiento, es un proceso que cumple la regularidad de producir-consumir-producir, tanto información como conocimiento. Los niveles táctico y estratégico de gestión empresarial, así como la interfaz entre ellos, tienen la particularidad de requerir de horizontes de tiempos más

[3] El debate sobre la epistemología del conocimiento, está fuera del alcance e intenciones de este documento. Sin embargo, la relación entre lo cognitivo y lo deductivo en el contexto abordado está ampliamente justificado. El sujeto cognoscente parte de un conocimiento y relacionando nuevas informaciones, deduce nuevo conocimiento aun cuando el resultado de su proceso cognitivo no sea el esperado.

[4] En lo adelante, a menos que se aclare explícitamente, al hacer referencia a la persona que interactúa con el sistema informático de información de la organización, se le denominará usuario.

largos que los que regularmente involucra el nivel operativo. Esta característica resulta útil, toda vez que amplia el área efectiva de la dimensión cognitiva-deductiva.

Coincidiendo con Sáez Mosquera (2008) en su Tesis Doctoral, el tradicional soporte informativo a la toma de decisiones debe ser cambiado por la asistencia decisional (Sáez Mosquera, 2008). El encargado de la toma de decisiones precisa de mecanismos que le permitan reinsertarse rápidamente en la dimensión tempo-contextual, a fin de situarse en la dimensión cognitiva-deductiva en la que es más competente y eficaz. Tales mecanismos tienen más que ver con la asistencia, que con la capacidad y aptitud de un sistema de información de responder preguntas de gestión formuladas sobre sus fuentes de información.

Además de las heterogeneidades de tipo estructural y semántica comentadas anteriormente, al proceso de formular preguntas y reutilizar las respuestas para formular nuevas se incorporan las diferencias de implementación de los diferentes gestores de bases de datos. Estas diferencias tienen incidencia directa no solo en los problemas relacionados con la integración, sino que además reducen la capacidad de la organización de formular preguntas sobre orígenes de información: bases de datos que residen en diferentes gestores (entornos corporativos heterogéneos).

Cualquier intento de capitalizar el conocimiento tácito que reside en las preguntas de gestión que son formuladas en entornos corporativos heterogéneos induce al usuario a ser, además de un experto en su área de competencia, capaz de reconocer y entender las diferencias entre los diferentes dialectos del lenguaje de consultas (SQL) que implementa cada gestor. Para garantizar la aptitud más que la capacidad de asistencia de un sistema de información decisional se hace necesario liberar al usuario de la responsabilidad de tratar con esta fuente de heterogeneidad, que poco aporta a los propósitos informativos.

Adicionalmente, se requiere un protocolo que utilizando un estándar de facto como XML, permita utilizar de formas muy diversas, la información recuperada. Finalmente, es necesario adicionar la información relacionada con la propia actividad informativa de los usuarios, a fin de poder anticipar las necesidades de información lo que permitiría su reinserción rápida en la dimensión tempo-contextual (dimensión cognitiva-deductiva).

Los aspectos antes comentados constituyen una apretada síntesis de la situación problema que motivó la presente investigación, y cuya intención fundamental es dar respuesta a las interrogantes científicas siguientes:

1. ¿Cuáles son las principales diferencias en los dialectos de SQL en los principales gestores comerciales?
2. ¿Qué implicaciones tienen desde la perspectiva de la capacidad de generar mejores respuestas a las preguntas formuladas?
3. ¿Qué mecanismos pueden diseñarse para hacer transparente al usuario estas diferencias en un ambiente corporativo heterogéneo?
4. ¿Cuál o cuáles tecnologías resultan más pertinentes para desarrollar e implementar los mecanismos diseñados?
5. ¿Cómo pueden evaluarse los resultados obtenidos de la implementación de los mecanismos diseñados?

Como **objetivo general** de la investigación se definió: Diseñar mecanismos de asistencia que permitan reducir la heterogeneidad entre los diferentes dialectos de SQL en los gestores comerciales más populares, desde el punto de vista del usuario. Para su cumplimiento, el objetivo fue desagregado en **objetivos específicos**, que son:

1. Caracterizar las principales diferencias entre los diferentes dialectos de SQL.
2. Diseñar mecanismos para hacer transparente las diferencias encontradas en los gestores comerciales más populares, contribuyendo de esta forma a reducir la heterogeneidad desde la perspectiva del usuario.
3. Seleccionar aquellas tecnologías más pertinentes para el diseño e implementación de un prototipo de prueba.
4. Diseñar un caso de uso que sirva a los propósitos de evaluación de las capacidades de los mecanismos diseñados

Para su presentación, el informe de investigación se ha estructurado de la forma siguiente. El primer capítulo está dedicado a dar respuesta a las preguntas 1 y 2 completamente, así como parcialmente a las preguntas 3 y 4. La respuesta a estas últimas dos preguntas en este capítulo,

identifica la brecha o “gap” existente entre las soluciones existentes, por una parte y por la otra, identifica aquellas tecnologías que sin constituir una elección final, permiten la implementación de un prototipo para evaluar los mecanismos diseñados.

El segundo capítulo complementa la respuesta a la pregunta 3. En este capítulo se propone a partir de su diseño, un mecanismo que permite reducir la heterogeneidad que incorporan los diferentes dialectos presenten en los gestores comerciales más populares. Aspectos como el tratamiento de todos los tipos de JOIN no quedaron resueltos, fundamentalmente por que la intención de los autores es presentar mecanismos, que si bien no pueden ser interpretados como una solución completa, contienen la suficiente generalidad y flexibilidad para incorporar progresivamente mayores consideraciones.

El tercer y último capítulo se dedicó a presentar el prototipo desarrollado para validar las capacidades de los mecanismos diseñados. Este capítulo presenta paso a paso cómo diseñar y ejecutar consultas cuyas fuentes de datos residen en diferentes gestores. Es importante salvar la importante confusión que puede derivarse de la expresión anterior. No fue objetivo de esta investigación diseñar consultas que involucren tablas ubicadas en gestores diferentes. Para probar la capacidad de la solución de hacer transparente las diferencias entre los diferentes dialectos de SQL (con respecto al estándar) la estrategia utilizada consistió en formular consultas utilizando el estándar y ejecutarlas sobre los gestores, con independencia del dialecto de SQL que implementa.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capitulo se aborda el marco teórico-referencial del estado del arte de los sistemas de información basados en Web. Se trata además el tema de la heterogeneidad en las fuentes de datos y se hace un análisis de las soluciones existentes: sus limitaciones y alcance.

1.1. Heterogeneidad estructural

La definición de Base de Datos como: “un conjunto de datos que pertenece al mismo contexto, almacenados sistemáticamente para su posterior uso”, ha sido ampliamente superada en nuestros días. Ciertamente, uno de los objetivos básicos es el de almacenar los datos para su posterior uso, pero no es menos cierto que la expresión “el mismo contexto” ha quedado como una definición obsoleta. Aunque en una empresa los datos tengan una cierta pertenencia al mismo contexto, en la actualidad, la realidad es que los tipos de datos que manejan las organizaciones son muy heterogéneos.

La información corporativa está compuesta por múltiples tipos de datos que provienen de diversos departamentos, tienen diferentes formatos y requieren distintos procesos. Pero, finalmente, cuando la empresa es capaz de reunir adecuadamente esta información y explotarla puede pasar a un tipo de análisis que hace unos pocos años era impensable.

El advenimiento de nuevos estándares de datos con el incremento de las capacidades de almacenamiento de los ordenadores, hacen que las organizaciones se enfrenten continuamente a escenarios informativos cada vez más diversos. La no existencia de un formato estándar de datos, da al traste con los esfuerzos de estandarización de la información almacenada.

Los modelos relacionales de bases de datos, tradicionalmente han lidiado con esquemas fijos para las entidades modeladas. En la actualidad es posible distinguir entre dos tipos de información bastante distantes en cuanto a métricas de similitud se refiere: datos estructurados y datos pobremente estructurados. Este diapasón tan amplio, al combinarse con un comportamiento similar presentado en los procesos de toma de decisiones, adiciona un “hándicap” al diseño de sistemas de soporte para la toma de decisiones, incrementando aún más la necesidad de la integración de la información.

Muy a pesar de las controversiales discusiones académicas relacionadas con los estándares para la representación de la información, no podemos pasar por alto que XML (eXtensible Markup Language) está ganando silenciosamente la batalla de los estándares para representar y compartir la información. Como consecuencia directa de tal éxito, los vendedores de gestores de datos están incorporando tecnologías en sus gestores que soportan XML y sus extensiones.

Adicionalmente, el problema de la integración se hace aún más complejo con el surgimiento y desarrollo de los nuevos formatos y nuevas entidades informativas en las organizaciones, tanto dentro de sus fronteras como fuera de ellas, esta provocando lo que muchos autores han convenido en llamar ¨ el efecto de la torre de Babel de las organizaciones¨ (Cuellar Almeida et al., 2004; Fuß et al., 2004; Ling et al., 2005; King et al., 2006). La diversidad de significados de los tipos de datos, genera una dificultad adicional para extraer información útil de los datos almacenados, que cumple con el principio de incertidumbre enunciado en la física [3]. En efecto, la heterogeneidad de los datos tiene dos dimensiones importantes: la heterogeneidad **estructural** y la **semántica**.

Por consiguiente, las bases de datos ya no son, como “antaño”, la aplicación maestra. Hoy día las ¨estrellas¨ son las diversas aplicaciones que manejan, filtran y seleccionan los datos. La capacidad de realizar análisis inteligente y unificado, con todos los datos de una empresa, puede suponer la diferencia entre planificar la evolución y anticiparse a los cambios o simplemente reaccionar ante ellos. Lo cual supone la diferencia entre sobrevivir y ganar a la competencia o ser “desbancado” por ella (respuesta proactiva versus reactiva).

Teniendo en cuenta que sigue aumentando la tendencia a almacenar mayores volúmenes de información (toda vez que los precios de los dispositivos de almacenamiento disminuyen en similar proporción al aumento de sus capacidades) y que con ello florecerán nuevas formas de representación de datos, cabe preguntarse: ¿Podrá alguna herramienta, en el futuro, eliminar de raíz el “punzante” problema de la heterogeneidad estructural?

1.2. Heterogeneidad en los gestores relacionales

Los Sistemas de Gestión de Base de Datos (SGBD) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Los SGBD pueden alojarse (y normalmente así lo hacen para mejorar el rendimiento) en una localización diferente a la que ejecuta las aplicaciones. De hecho, las aplicaciones modernas se programan de forma que se puede utilizar esta característica de distribución física, aunque a la hora de instalar la aplicación no se utilice y se ubique todo el software en la misma máquina. Esto ha dado lugar a diferentes configuraciones de la arquitectura de las aplicaciones, todas ellas conocidas como arquitecturas multi-capa. La Figura 1 muestra arquitecturas típicas de dos, tres y cuatro capas.

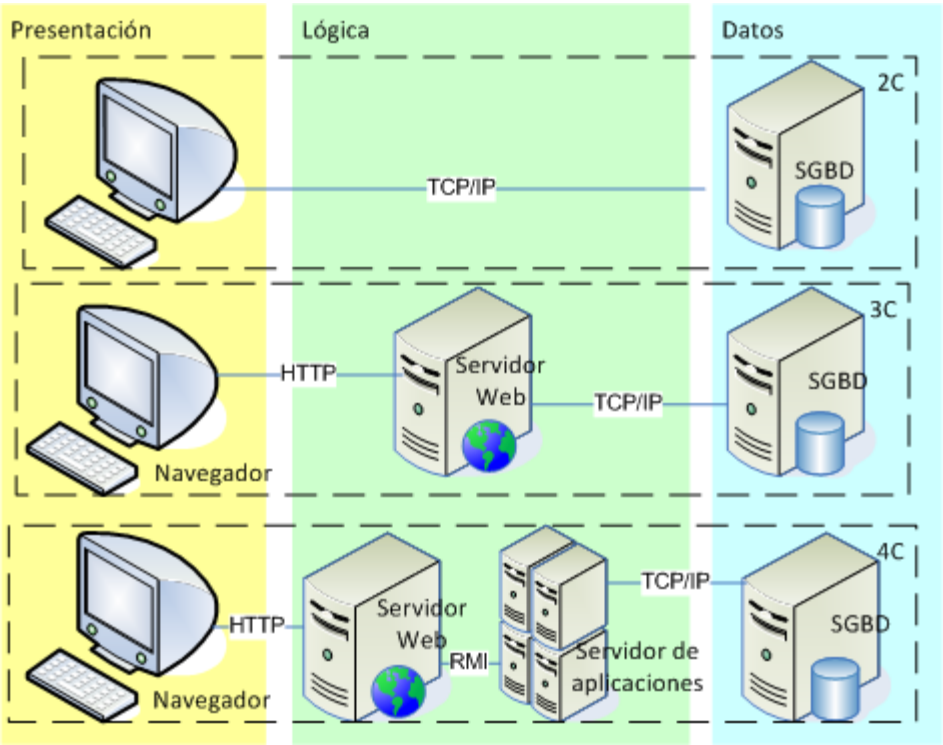


Figura 1. Aplicaciones en tres capas: datos, lógica y presentación.

En 1985 el Dr. Edgar Frank Codd publicó 12 reglas para evaluar si un SGBD puede considerarse un **SGBDR** (en inglés: **R**elational **D**ata**B**ase **M**anagement **S**ystem), o dicho más concisamente, si un sistema de bases de datos puede considerarse o no relacional:

1. Regla de la información.
2. Regla del acceso garantizado.
3. Regla del tratamiento sistemático de valores nulos.
4. Regla de diccionario dinámico en línea basado en el modelo relacional.

5. Regla del sub-lenguaje de datos integral.
6. Regla de actualización de vistas.
7. Regla de inserción, actualización y borrado de alto nivel.
8. Regla de independencia física de datos.
9. Regla de independencia lógica de datos.
10. Regla de independencia de integridad.
11. Regla de independencia de distribución.
12. Regla de no subversión.

La tendencia a la centralización de las fuentes de datos, discutidas en el epígrafe anterior, ha sido uno de los catalizadores del crecimiento en el mercado de las bases de datos relacionales, así como de los **gestores relacionales** con cada vez más prestaciones en el manejo de grandes volúmenes de información.

Microsoft® SQL Server, Oracle®, Firebird®, Informix®, MySQL® y PostgreSQL®, por solo mencionar algunos, son SGBDR multi usuarios que por sus buenas prestaciones han ganado gran popularidad en el mercado. Así mismo, desde que en los últimos tiempos el software libre se ha consolidado como alternativa, técnicamente viable y económicamente sostenible, al software comercial, es cada vez más común que se empleen indistintamente, dentro de una misma organización, tanto SGBDR de tipo software distribuido bajo la licencia GNU como propietario, por lo que en un ambiente empresarial nos encontramos con una muy marcada heterogeneidad de los SGBDR.

1.3. Diferencias en la implementación de SQL en los gestores de bases de datos

Una de las principales consecuencias, de este problema de la heterogeneidad de los SGBDR es la incompatibilidad entre los lenguajes de consulta de los diferentes fabricantes. Con el objetivo de potenciar la interoperabilidad de todos los productos que usan los distintos lenguajes de consulta existente, es que estos últimos se han estandarizado.

Independientemente de la existencia de los estándares, para el SQL por ejemplo el estándar SQL3 (SQL 2003), que es el lenguaje más ampliamente usado en la actualidad; sin embargo, normalmente los fabricantes construyen "dialectos" para sus productos.

El estándar SQL 2003 hace algunas modificaciones a todas las partes de SQL 1999 y oficialmente introduce algunas nuevas características tales como:

- Las características para el trabajo con XML
- Ventana de funciones
- Generador de secuencia, lo que permite secuencias estandarizadas
- Dos nuevos tipos de columna: valores auto-generados (auto-generated values) y la identidad de columnas (identity-columns)
- La sentencia MERGE
- Ampliaciones de la sentencia CREATE TABLE, para permitir "CREATE TABLE AS" y "CREATE TABLE LIKE"
- Eliminación de los mal implementados tipos de datos "BIT" y "BIT VARYING"

En este epígrafe abordaremos algunas de las diferencias de los dialectos de SQL que usan los gestores MySQL 5.0.18, SQL Server 2005, PostgreSQL 8.3.3 y Oracle 10 con respecto al estándar SQL: 2003. La última revisión al estándar se realizó en el año 2008 y se utilizó como referencia el borrador de la ISO (JTC, 2006).

La cláusula **Select** : el ordenamiento del conjunto de resultados.

Estándar SQL:2003	La norma establece que las relaciones son desordenadas, pero los conjuntos de resultados pueden ser ordenados cuando se devuelven al usuario, a través de un cursor: **DECLARE** cursorname **CURSOR FOR** **SELECT** Field1, Field2, …, Field*n* **FROM** Table1, Table2, …, Table*n* **WHERE** Condition **ORDER BY** column_name1,column_name2 El estándar no especifica cómo deben ser ordenados los valores NULL en

	comparación con los valores no NULL, salvo que cualquiera de los dos valores NULL se consideren igualmente ordenados, ni que los valores NULL deban ordenarse encima o debajo de los valores no NULL. Sin embargo, el SGBD opcionalmente puede permitir al usuario especificar si los valores NULL deben ordenarse primeros o últimos: **ORDER BY** ... **NULLS FIRST** or... **ORDER BY** ... **NULLS LASTS**
PostgreSQL®	Así como en las definiciones del cursor, permite **ORDER BY** en otros contextos. Por defecto, valores NULL se consideran **superiores** a cualquier valor no NULL, sin embargo este comportamiento puede ser modificado mediante la adición de la expresión **NULLS LAST** a la cláusula **ORDER BY**.
Microsoft® SQL Server	Así como en las definiciones del cursor, permite **ORDER BY** en otros contextos. Los valores NULL se consideran **inferiores** a cualquier valor no NULL.
MySQL®	Así como en las definiciones del cursor, permite **ORDER BY** en otros contextos. Los valores NULL se consideran **inferiores** a cualquier valor no NULL, excepto si un - (menos) se añade el carácter antes de que el nombre de columna y se cambia **ASC** a **DESC** viceversa.
Oracle®	Así como en las definiciones del cursor, permite **ORDER BY** en otros contextos. Por defecto, valores NULL se consideran **superiores** a cualquier valor no NULL, sin embargo este comportamiento puede ser modificado mediante la adición de la expresión **NULLS LAST** a la cláusula **ORDER BY**.

Tabla 1. Cláusula SELECT en SQL estándar.

Limitar los conjuntos de resultados

Estándar SQL:2003	El estándar SQL proporciona tres formas de realizar un « límite simple »: • El uso de **FETCH FIRST** (desde el SQL: 2008) **SELECT ... FROM ... WHERE ... ORDER BY ... FETCH FIRST n ROWS ONLY** Puede escribirse **ROW** en lugar de **ROWS** • Uso de una **Window Function** (a partir de SQL: 2003) ROW_NUMBER() OVER: **SELECT** "*" **FROM** (**SELECT ROW_NUMBER**() **OVER** (**ORDER BY** key ASC) **AS** rownumber , columns **FROM** TableName) AS foo **WHERE** RowNumber <= n) • El uso de un **cursor:** o **DECLARE** Cursor-Name **CURSOR FOR** ... o **OPEN** Cursor-Name o **FETCH** ... o **CLOSE** Cursor-Name
PostgreSQL®	No admite **ROW_NUMBER()** ni **FETCH FIRST** Alternativa al uso de **ROW_NUMBER()** **SELECT** columns **FROM** TableName **ORDER BY** key **ASC LIMIT n** Hay que tener en cuenta que **LIMIT** cambia las semánticas de SELECT...FOR UPDATE.
Microsoft® SQL Server	Soporta **ROW_NUMBER()** (MSSQL desde 2005) y el cursor de los enfoques basados en estándares, no es compatible con **FETCH FIRST** MSSQL 2000 no soporta **ROW_NUMBER(),** en su lugar se usa la sintaxis: **SELECT TOP n columns FROM** tablename **ORDER BY** key **ASC**
MySQL	No es compatible con el estándar. Solución alternativa: **SELECT** columns **FROM** tablename **ORDER BY** key **ASC LIMIT n**
Oracle	Soporta **ROW_NUMBER(),** no admite **FETCH FIRST** **SELECT** "*" **FROM** (**SELECT ROW_NUMBER() OVER** (**ORDER BY** key **ASC**) AS rownumber , columns **FROM** tablename) **WHERE** rownumber <= n **FROM** **SELECT** "*"

	FROM (SELECT ROW_NUMBER() OVER (ORDER BY key ASC) AS rownumber , columns FROM tablename WHERE rownumber <= n))

Tabla 2. Limitar conjuntos de resultados.

La cláusula INSERT: varias filas a la vez.

Estándar SQL:2003	Permite la inserción de múltiples filas a la vez, ej INSERT INTO tablename VALUES (0,'foo') , (1,'bar') , (2,'baz');
PostgreSQL®	Soporta esta facilidad (desde la versión 8.2)
Microsoft® SQL Server	No permite esta característica.
MySQL®	Soporta esta característica conforme al estándar.
Oracle®	No permite esta característica.

Tabla 3. Cláusula INSERT en SQL estándar.

Tipos de Datos: el tipo BOOLEAN.

Estándar SQL:2003	El estándar establece que un dato **BOOLEAN** puede ser unos de los siguientes literales: • **TRUE** • **FALSE** • **UNKNOWN** o **NULL**
PostgreSQL®	Sigue el estándar
Microsoft® SQL Server	No soporta el tipo de dato **BOOLEAN**. Da la alternativa de usar el tipo de dato BIT el cual puede tener 0 o 1 como valores.
MySQL®	Ofrece un tipo de dato **BOOLEAN** no confirmado. El tipo de datos BOOLEAN de MySQL es uno de los alias de su tipo de dato **TINYINT**(1). Además MySQL acepta literales **TRUE** Y **FALSE** como alias de valores 1 o 0 respectivamente.
Oracle®	No soporta el tipo de dato **BOOLEAN**. Oracle recomienda **NUMBER**(1) como forma de almacenar datos boleanos.

Tabla 4. Tipos de datos. Datos BOOLEAN.

Fecha y tiempo: el tipo TIMESTAMP

Estándar SQL:2003	Parte de los requerimientos del núcleo establece que se almacenan: año, mes, día, hora, minuto y segundos. Ejemplos de **TIMESTAMP** • **TIMESTAMP** '2003-07-29 13:19:30' • **TIMESTAMP** '2003-07-29 13:19:30.5' Ejemplos de **TIMESTAMP** con TIME ZONE • **TIMESTAMP** '2003-07-29 13:19:30+02:00' • **TIMESTAMP** '2003-07-29 13:19:30.5+02:00'
PostgreSQL®	Sigue el estándar con una excepción: En algunos casos los **TIMESTAMP** con TIME ZONE (2003-08-23 01:02:03 +02:00) son tratados como **TIMESTAMP** sin TIME ZONE (la parte '+02:00' es descartada)
Microsoft® SQL Server	Hay que hacer notar que el vocabulario de **Microsoft® SQL Server** relativo al tiempo y a la fecha, tiende a confundir. Para **Microsoft SQL Server,** DATETIME es un tipo de dato concreto mientras que en el estándar, DATETIME es un término general
MySQL®	MySQL posee un tipo de dato **TIMESTAMP** que es bien diferente al TIMESTAMP estándar.
Oracle®	Sigue el estándar. Posee los tipos, **TIMESTAMP** y **TIMESTAMP** con TIME ZONE.

Tabla 5. Tipos de datos. Datos TIMESTAMP.

Manipulación de Metadatos: obtener una lista de bases de datos.

Estándar SQL:2003	Hasta donde se ha investigado no hemos encontrado que el estándar permita esta operación.
PostgreSQL®	Usando una sentencia SQL: **SELECT** datname **FROM** pg_catalog.pg_database
Microsoft® SQL Server	Mediante la expresión: EXEC SP_HELPDB
MySQL®	Mediante la expresión: SHOW DATABASES
Oracle®	En Oracle hay una relación uno a uno entre las bases de datos. La forma de obtener una lista de bases de datos varia de acuerdo al Sistema Operativo donde esté corriendo Oracle.

Tabla 6. Manipulación de meta-datos.

Obtener una lista de esquemas.

Estándar SQL:2003	**SELECT** SCHEMA_NAME **FROM** INFORMATION_SCHEMA.SCHEMATA
PostgreSQL®	Sigue el estándar
Microsoft® SQL Server	Sigue el estándar
MySQL®	No soporta esquemas
Oracle®	Aquí existen esquemas para todos los usuarios y no puede existir un esquema sin el usuario correspondiente. Consecuentemente una forma de obtener todos los esquemas seria interrogando la vista ALL_USERS: **SELECT** username **FROM** all_users

Tabla 6.Obtener listas de esquemas.

1.4. Soluciones existentes

Las capas de abstracción para el acceso a las bases de datos en los gestores, se agrupan bajo el nombre común de Databases Abstraction Layer (Capas de Abstracción). Disponibles para el lenguaje Php hay algunas soluciones que a su vez, son agrupadas bajo la denominación de Tool Command Language, Tcl (Lenguajes de herramientas de comandos). Una comparación en cuanto a los gestores que son soportados, puede verse en las Tabla 7y 8.

Paquete	**Descripción**
TDBC	Tcl Database Connectivity, BSD License
DIO	Apache License
tcldb	BSD License
XOSql	GPL License
TclODBC	BSD License (also applies to SnODBC)
nstcl-database	MIT/X11 License''
nsdbi	MPL/GPL License''
sqlite	Public Domain
MrPersister	MPL/LGPL

Tabla 7. Paquetes Tcl. Capas de abstracción disponibles para Php. Fuente: http://wiki.tcl.tk/14972

DB	TDBC	DIO	tcldb	XOSql	tclodbc	nstcl	nsdbi	sqlite3	MrPersister
MySQL	x	x	x	x	x	x	x		x
PostgreSQL		x	x	x	x	x	x		x
SQLite	x	x	x	x	x	x	x	x	x
ODBC	x		x	x	x	x			
DB2					x				x
Oracle		x	x	x	x	x			x
Sybase/MSSSQL					x	x			x
Solid					x	x			x
Perl DBI				x					
JDBC									x

Tabla 8. Gestores comerciales soportados. Fuente: http://wiki.tcl.tk/14972

Adicionalmente, los mecanismos que soportan los esquemas federados de datos constituyen una propuesta de solución al problema de la heterogeneidad introducida por los dialectos de SQL. Particularmente interesante por el nivel de modularidad e integración, así como por las capacidades de extensión la propuesta SQL Integrator de Novell® 2009, se comentará brevemente.

La arquitectura consta de los componentes siguientes:

Componente	Descripción
Componentes del cliente	
SQL 1: ODBC	Microsoft® ODBC API
SQL 1: JDBC	Java Soft Java SQL Classes
SQL 1: MGR	Diccionario de datos
Data Broker Components	
SQL 1: DRB	Data Request Broker
SQL 1: DD	Data Dictionary
Data Driver Components (local or remote Broker)	
SQL 1: Oracle	Oracel Database Connection
SQL 1: SysBase	Sysbase Database Connection
SQL 1: MS-SQL	Microsoft SQL Server DB Connection

SQL 1: BD2	IBM DB2 Database Connection
SQL 1: Informix	Informix Database Connection
SQL 1:INGRES	INGRES Database Connection
SQL 1: RDB	Oracle/RDB Database Connection
SQL 1: Files	Flat Files Database Connection
SQL 1: Remote	SQL Remote Integrator

Tabla 9. Componentes y descripción en SQL Integrator Solutions. Fuente: http://www.b2system.com

La arquitectura de la solución para un modelo en tres capas se muestra en la Figura 2. Obsérvese que para el caso de las solicitudes locales, la solución incorpora un gestor Oracle®, mientras que para el caso de clientes remotos en redes Windows® se propone la utilización de SQL Server©. Esta combinación tiene garantías de eficiencia, robustez y velocidad a expensas de los gestores involucrados. Desde el punto de vista del despliegue, la solución es muy pesada. Las labores de mantenimiento de datos podrían complicarse y requerir soluciones de transformación de datos (ETL), así como sería necesario diseñar estrategias de replicación para que en un ambiente distribuido, se mantengan las ventajas de la centralización que propone la solución.

Adicionalmente, la solución casi fuerza la utilización de aplicaciones “duras” para las interfaces con los clientes. En el caso del servidor de Microsoft, las aplicaciones basadas en Web (característica que garantiza el acceso fácil y global) tienen probados problemas de eficiencia en lo que a recuperación y presentación de los datos recuperados se refiere.

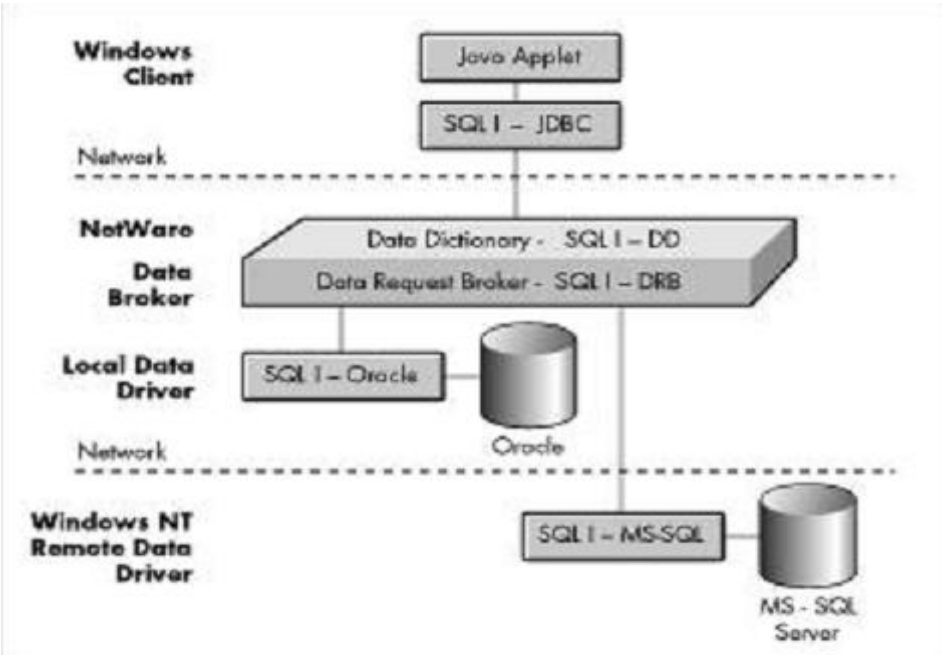


Figura 2. Arquitectura en tres capas SQL Integrator. Fuente: http://www.b2system.com

La Figura 3 presenta la arquitectura general para la solución, cuando son tenidas en cuenta todos los componentes listados en la Tabla 9, para todos los gestores considerados. Nótese que está solución es similar a los modelos basados en Wrapper. Cada uno de los gestores que serán soportados, requieren un componente que se adiciona a la arquitectura. Se trata de una solución que utiliza el concepto de Mediador.

Este tipo de solución aunque han estado siendo utilizadas con éxito en muchas aplicaciones comerciales, presenta algunas importantes desventajas. La principal desventaja proviene del hecho de ser necesario un componente (Wrapper) por cada fuente heterogénea incorporada. Las soluciones que utilizan capas de abstracción, encapsulan estas funcionalidades dejando a los diseñadores de aplicaciones en libertad para ocuparse de otros aspectos funcionales más importantes.

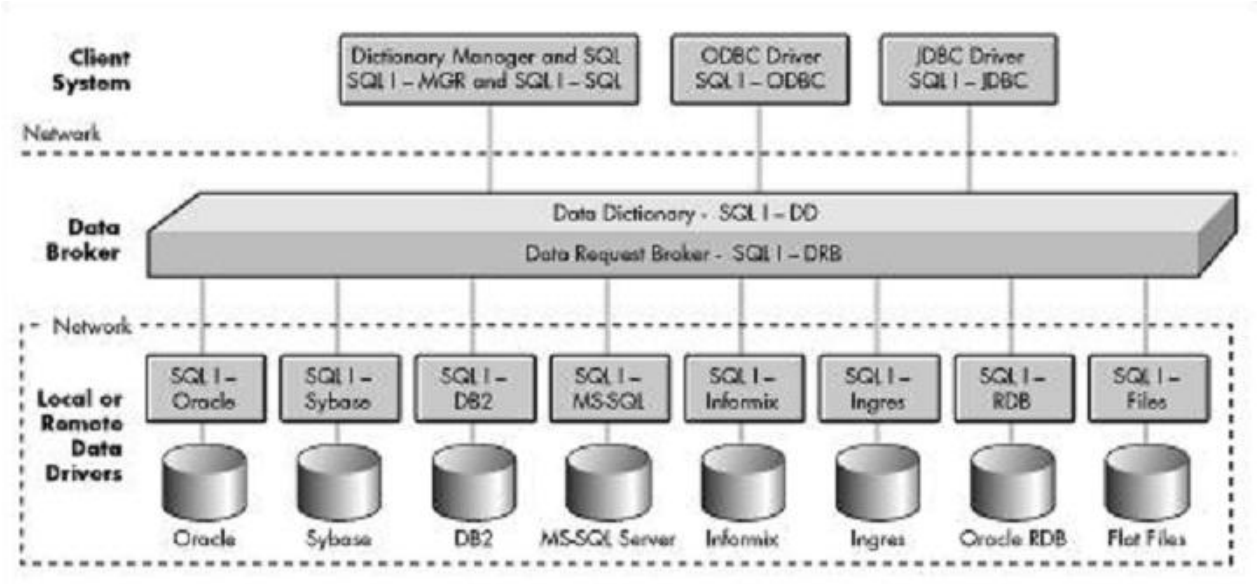


Figura 3. Arquitectura general de SQL Integrator. Fuente: http://www.b2system.com

1.5. Conclusiones parciales

1. Las soluciones encontradas en la literatura consultada representan soluciones de compromiso que hacen sacrificios de unas funcionalidades a favor de otras.
2. Las soluciones que utilizan el concepto de esquemas de mediación involucran mayor responsabilidad por parte de los desarrolladores de aplicaciones y los hacen más dependientes de detalles implementativos que aquellas que utilizan capas de abstracción.
3. Las soluciones que utilizan servidores de bases de datos propietarios como Oracle® y SQL Server® garantizan eficiencia y robustez pero sacrifican velocidad y acceso global a las soluciones.
4. La utilización de capas de abstracción para el acceso a bases de datos representan una solución que ofrece eficiencia y robustez comparable con las soluciones que utilizan servidores propietarios y la elección de una de ellas en particular, debe responder a criterios de diseño al ser todas comparables.

CAPÍTULO 2: DISEÑO DE LA SOLUCIÓN

2.1. Estrategia para el diseño del asistente de consulta basado en Web

Para lograr una implementación efectiva se decidió usar el Proceso Unificado de Rational (RUP), que plantea un Ciclo de Vida del Software dividido en fases con relación al tiempo, estas son: Concepción, Elaboración, Construcción y Transición. Teniendo en cuenta este ciclo de vida podemos distribuir las disciplinas de obtención de requerimientos, análisis, diseño, implementación y prueba, por cada fase de forma tal que se garantice un desarrollo correcto del trabajo.

Según RUP el software debe ser:

- Centrado en la Arquitectura
- Iterativo e Incremental
- Guiado por Casos de Uso

Además, RUP establece el modelado basado en notación UML (Lenguaje Unificado del Modelado) y Programación Orientada a Objetos (POO). (Jacobson, 2004a, Jacobson, 2004b).

La notación UML (versión 2.0), es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software, captura decisiones y conocimientos sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener y controlar la información de tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. UML se ha convertido en un estándar para la notación en el desarrollo Orientado a Objeto.

La Programación Orientada a Objetos, técnica utilizada para el diseño e implementación del Módulo Asistente de Consultas, constituye una singular forma de pensar acerca de los problemas, empleando modelos que se han organizado tomando como base el mundo real, útiles para analizar requisitos, comunicarse con expertos de aplicaciones, modelar empresas, preparar documentación, diseñar programas y bases de datos.

Las herramientas de desarrollo empleadas fueron:

- Visual Paradigm 3.0 para el modelado de todos los diagramas correspondientes a las fases de análisis y diseño.
- Zend Studio como herramienta para la implementación del software.
- Uniserver 4.1 (Mona) para ejecutar los servidores Apache y MySQL para correr la aplicación.

Zend Studio o Zend Development Environment es un completo entorno de desarrollo integrado para el lenguaje de programación PHP. Este IDE esta programado en Java, algunas de sus características principales son:

- No requiere la instalación previa de PHP ni del entorno de ejecución de Java.
- Soporte para PHP 4 y PHP 5.
- Resaltado de sintaxis, autocompletado de código, ayuda de código y lista de parámetros de funciones y métodos de clase.

Visual Paradigm 3.0 para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste, es muy importante cuanto se desarrolla una aplicación seguir el proceso de ingeniería correcto, siempre el producto final contará con una mayor calidad si así se hace. Este software permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

Uniform Server es un paquete que combina el servidor Apache, la base de datos MySQL y el lenguaje de programación PHP. Este paquete está configurado para un entorno de producción. Es la manera más pequeña para correr un servidor WAMP (Apache, MySQL y PHP en Windows). No es necesaria ninguna instalación ni modificación al registro de Windows, sólo se deben descomprimir los archivos incluidos y el servidor está listo para correr. A pesar de los pocos bytes que ocupa, es funcional y tiene un completo panel de control. Este mini servidor puede ser de mucha utilidad para hacer pruebas en PCs que no tienen un servidor instalado.

Para implementar nuestro asistente para hacer consultas se trato de hacer un diseño similar a la vista de diseño de MsAccess, con el objetivo de facilitar al usuario la creación de la consulta ya que este software es uno de los más populares mundialmente y es de fácil manejo para los usuarios.

2.2. Estructuración y acceso a las distintas fuentes de información

Un punto fundamental en el diseño de este problema fue decidir la estrategia a seguir para lograr la interacción de nuestro sistema con los diferentes SGBDR que existen en las organizaciones.

El lenguaje de programación PHP es ideal para el desarrollo de una herramienta Web como esta que se propone, sin embargo, las funciones de acceso a base de datos en PHP no están estandarizadas. Teniendo en cuenta que el sistema que se propone actúa en un ambiente de base de datos heterogéneo, se requiere del uso de una biblioteca que oculte las diferencias entre cada API (encapsular las diferencias) para que se pueda cambiar fácilmente de base de datos.

Entre las diferentes bibliotecas disponibles para PHP, que permiten el acceso a múltiples bases de datos, se eligió AdoDB para usarla como capa de abstracción por reunir algunas características importantes:

- Es *open source* y posee una amplia comunidad de desarrolladores.
- Posee un diseño de clases eficiente que se traduce en mayor velocidad de ejecución de las consultas.
- Provee *drivers* que permiten la conexión a casi la totalidad de los SGBDR existentes.
- Marca una diferencia con otras bibliotecas al poseer *driver* para *Active Directory.*

Luego con el uso de esta biblioteca, la conexión a las diferentes fuentes de información queda homogeneizada y se reduce a dos pasos:

- Crear una instancia de la clase **ADONewConnection ( )** pasándole como parámetro el *driver* correspondiente.
- Llamar al método **Connect ()** pasándole los parámetros de conexión.

Ej. : `$conn1 = ADONewConnection ('driver');`

```
$conn1->PConnect($server, $userid, $password, $database);
```

El parámetro “driver” es el nombre de del driver correspondiente al gestor al cual se va a realizar la conexión. Por ejemplo para el SGBDR MySQL se usa el parámetro mysql y para SQL Server el usado es mssql.

Las fuentes de información disponibles, sobre las que se realizarán las consultas, quedarán reflejadas en el fichero XML ¨ gestor-ip.conf.XML ¨. Este tendrá la siguiente estructura:

```
<?xml version="1.0?>
<entradas>
<item id=¨id_asigando_al_servidor¨>
      <host>número_ip</host>
      <driver>driver_requerido</driver>
      <user>nombre_usuario</user>
      <password>contraseña</password>
</item>
</entradas>
```

Como la administración de este fichero se deja en manos del administrador del sistema, se expone un ejemplo a continuación.

En una organización que tiene dos gestores de bases de datos, uno SQL Server y otro My SQL. Ambos gestores ubicados en ordenadores diferentes dentro de la misma red: 10.12.xx.xx y 10.12.xx.xx los números IP, respectivamente. A SQL Server se puede acceder con nombre de usuario ¨xxx¨ y contraseña ¨ok¨ y a MySQL con nombre de usuario ¨yyy¨ y contraseña ¨ok¨. Dadas estas condiciones el fichero ¨ *gestor-ip.conf.XML* ¨ quedaría:

```
<?xml version="1.0"?>
<entradas>
      <item id="1">
            <host>10.12.20.10</host>
            <driver>mssql</driver>
            <user>xxx</user>
            <password>ok</password>
      </item>
```

```
<item id="2">
      <host>10.12.20.15</host>
      <driver>mysql</driver>
      <user>yyy</user>
      <password>ok</password>
</item>
</entradas>
```

Luego cada etiqueta <item> del fichero, contiene la ubicación y la forma de acceder a las fuentes de información disponibles y haciendo uso de esta estructuración se obtendrán los datos que necesitará el usuario para la elaboración de las consultas.

2.3. Estrategia para expresar las sentencias SQL en XML

Para lograr una eficiente comunicación entre capas es necesario decidir una estrategia con la cual deben intercambiar la información de forma tal que los datos puedan ser comprendidos por el manejador (Manager), quien será quien dé la solución final a la consulta que formulara el usuario.

Sin duda alguna la solución a esta problemática en una correcta estructuración de la información. Las herramientas para crear documentos XML (y sus especificaciones) se basan en estándares abiertos. Es decir: el mercado no tiene ningún tipo de control sobre ellos y, por lo tanto, no puede imponer dependencia alguna hacia una determinada marca o sistema operativo. SQL es un lenguaje asentado y normalizado en tanto XML presenta la información de forma diferente (pero no opuesta) al modelo relacional: los datos forman un árbol jerárquico de elementos y atributos. Lograr un correcto lenguaje de interrogación compatible con XML facilitaría el proceso de la realización de las consultas.

Para lograr una eficiente comunicación entre el asistente y el manager se definió el formato XML de la siguiente forma para cuando se seleccione una consulta simple (un solo elemento):

```
<?xml version="1.0"?>
  <query>
    <select>*</select>
    <from>DB.Table.Field</from>
    <where>Condition</where>
    <order_by>Ascending</order_by>
    <group_by>DB.Table.Field</group_by>
    <having>Condition</having>
  </query>
</params>
```

Cuando la consulta incluya más de un elemento en cualquiera de sus campos (siempre debe tener alguno en el Select) se acordó separar cada elemento por coma (,) de forma que al integrar nuevamente la consulta para ser ejecutada se puedan diferenciar que campos corresponden a cada columna. Un ejemplo de una consulta realizada involucrando varios campos quedaría estructurado como se presenta a continuación.

```
<?xml version="1.0"?>
<params>
  <id_manager>2</id_manager>
  <nombre>ejemplo</nombre>
  <id_query>425463414</id_query>
  <descripcion>ejemplo</descripcion>
  <query>
    <select>tempdb.spt_monitor.name,tempdb.spt_monitor.phyname</select>
    <from>tempdb</from>
    <where>name=x,phyname=y</where>
    <order_by>ASC(tempdb.spt_monitor.name),DESC(tempdb.spt_monitor.phyname)</order_by>
    <group_by></group_by>
    <having></having>
  </query>
</params>
```

Las fuentes de información (SGBDR) disponibles han sido etiquetadas con un valor único que las representa. De este modo la etiqueta <**id_manager**> representa el valor asignado al gestor donde están hospedadas las bases de datos que involucra la consulta. La etiqueta <nombre> y la etiqueta <**descripción**> contienen respectivamente, el nombre y la descripción que el usuario ha dado a la consulta.

Teniendo en cuenta que el nombre de dos consultas completamente diferentes podrían tener el mismo nombre, se decidió adicionar a una etiqueta <**id_query**> cuyo valor es el identificador único asociado a la consulta.

Como se puede apreciar XML es un formato que puede ser de muy conveniente uso en este trabajo ya que permitirá un intercambio fluido entre las capas (Wizard y Manager) dado que se manejaran metadatos de las fuentes de información y será necesario lograr una comunicación eficiente. Por lo tanto podemos afirmar que será una herramienta muy útil para lograr una buena compresión entre las distintas capas.

2.4. Diseño de las pruebas

El Módulo de Consultas tiene dos casos de uso fundamentales:

- Diseñar Consulta.
- Resolver Consulta.

Ellos se relacionan con otros casos de uso secundarios hasta mostrar paso por paso cómo el sistema interactúa con los actores. En el primer caso de uso el actor es el usuario con necesidad de información el cual crea las consultas que serán resueltas por el segundo actor (el sistema).

El primer Caso de Uso es donde el usuario selecciona los elementos que van a estar en su consulta, estos serán todos los disponibles de la base de datos que este seleccionada en ese momento. Puede crear consultas simples o que involucren varios campos, estos pueden ser de bases de datos diferentes pero siempre del mismo gestor.

Solucionar Consulta es donde se ejecutará la consulta diseñada en el caso de uso anterior y sus resultados serán estructurados de modo tal que sean inteligibles para el usuario que posteriormente los visualizará. El diagrama con estos casos de uso y otros secundarios, que son consecuencia directa de los primeros, aparece a continuación en la Figura 2.1.

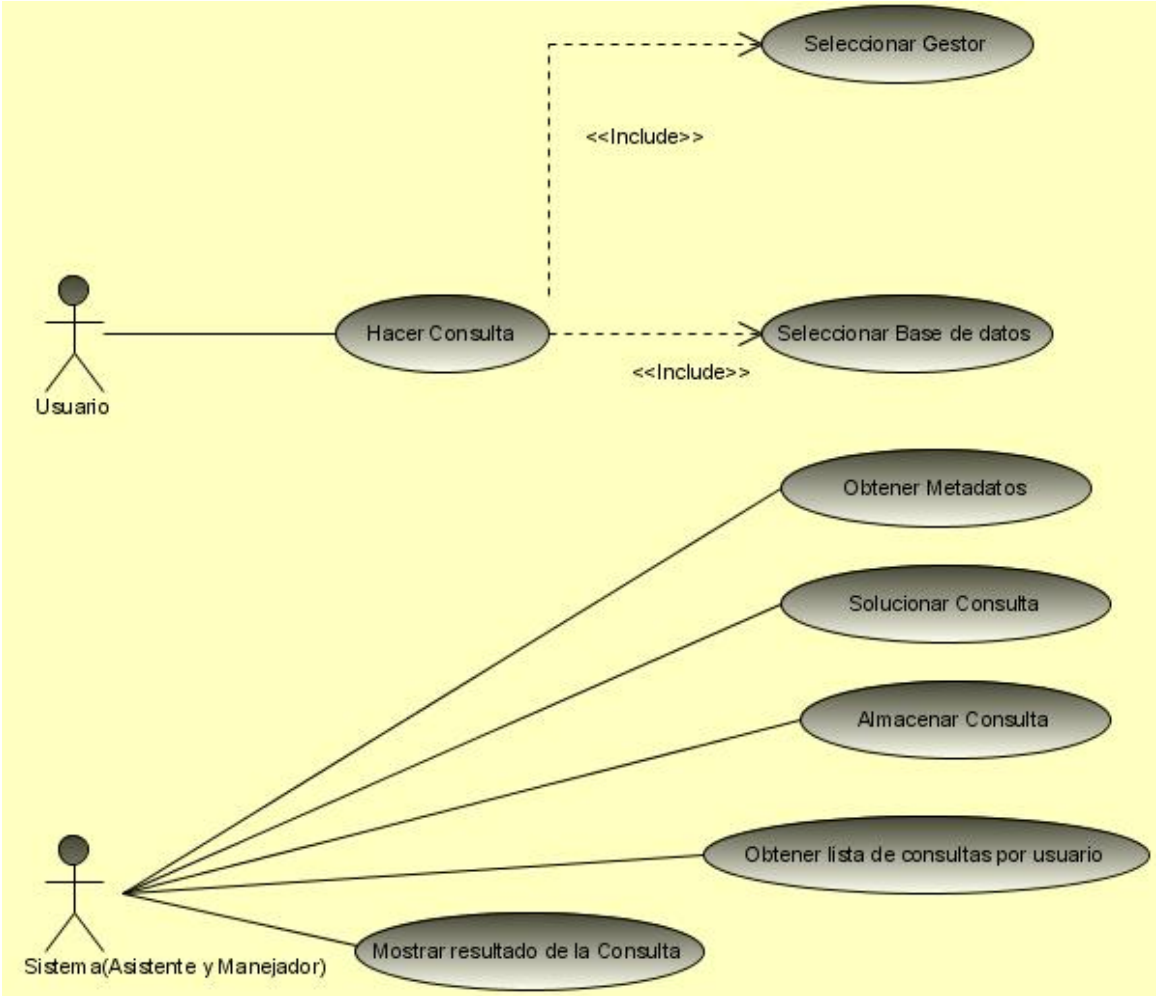


Figura 3. Diagrama de Caso de Uso del sistema. Fuente: Elaboración propia.

Para una mejor comprensión del role que cumplen cada uno de estos casos de uso que complementan a los principales que fueron descritos anteriormente presentamos una breve descripción de cada uno en las siguientes tablas:

Nombre del caso de uso	**Seleccionar Gestor**
Actor	Usuario.
Precondiciones	El asistente debe cargar los metadatos del XML enviado por el manager.
Descripción	El usuario selecciona el gestor en el cual trabajará.
Poscondiciones	Queda seleccionado el gestor en el cual se trabajará

Tabla 11. Caso de uso Seleccionar Gestor.

Nombre del caso de uso	Seleccionar Base de datos
Actor	Usuario.
Precondiciones	El asistente debe cargar los metadatos del XML enviado por el manager.
Descripción	El usuario selecciona la base de datos en la cual trabajará.
Poscondiciones	Queda seleccionada la base de datos en la cual se trabajará.

Tabla 12. Caso de uso Seleccionar base de datos

Nombre del caso de uso	Obtener metadatos.
Actor	Sistema.
Precondiciones	El usuario accede al Asistente de Consultas
Descripción	Cuando el usuario entra al Asistente este automáticamente le muestra los datos que tendrá disponibles para la realización de su consulta.
Poscondiciones	Quedan visualizados los datos disponible sobre los que se realizaran las consultas

Tabla 13. Caso de uso Obtener metadatos.

Nombre del caso de uso	Mostrar resultado de la Consulta.
Actor	Sistema.
Precondiciones	El manejador envía en XML con la solución de la consulta.
Descripción	Muestra los resultados de la consulta hecha por el usuario y le da la opción de guardarla o ir hacia el asistente sin salvar este resultado.
Poscondiciones	Envía mensaje al manejador para que guarde la consulta y vuelve hacia el asistente o no hace nada y vuelve al asistente dependiendo la opción que sea escogida.

Tabla 14. Caso de uso Mostrar resultado de la Consulta.

Nombre del caso de uso	Almacenar consulta.
Actor	Sistema.
Precondiciones	El usuario ordena que se guarde la consulta.
Descripción	La consulta realizada exitosamente es guardada en la base de datos del sistema.
Poscondiciones	La base de datos se actualiza.

Tabla 15. Caso de uso Almacenar consulta.

Después de planteado este modelo, se hizo un análisis de cómo se podía crear una jerarquía de clases para realizar cada caso de uso. La arquitectura lógica del sistema es capturada en el diagrama de clases que contiene las clases y las relaciones.

Para hacer el mismo se necesitó pensar en como sería la interfaz con el usuario y como ver además el mecanismo que le daría solución final a la consulta creada. En el diagrama que se expone continuación se puede apreciar claramente las clases empleadas en el asistente (color gris) y las clases del manager (color azul). (Figura 2.2).

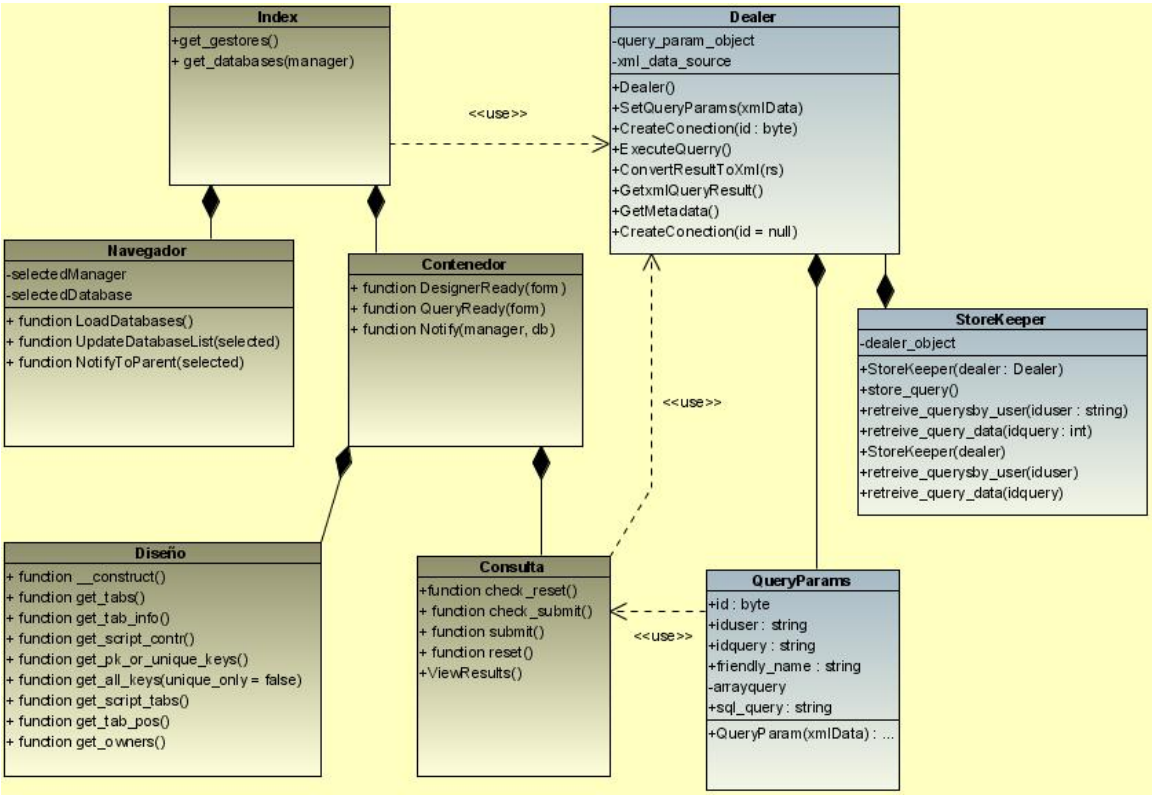


Figura 4. Diagrama de Clases del Sistema. Fuente: Elaboración propia.

Como se puede apreciar en el diseño la clase **Index** contiene en ella a todas las demás que conforman el asistente. Es la encargada de obtener los metadatos servidos desde el manager mediante los métodos **get_gestores** y **get_databases**, y "distribuirlos" entre todas las demás clases para que los manipulen.

De la clase **Index** reciben directamente los datos las clases **Navigator** y **Container**. La primera es la encargada de que el usuario seleccione sobre cual gestor y base de datos trabajará y esta lo

“notificará” a las demás clases para que se actualicen. A su vez la segunda es la encargada de servirles los datos a las clases **Designer** y **Query.**

La clase **Designer** cumple el role de la visualización de los datos, estos serán mostrados en forma de tablas con toda la información de los nombres de los campos y los tipos de estos, además de relaciones en caso de que las tengan. Se actualiza si el usuario cambia de base de datos o gestor y muestra las nuevas tablas que estén dentro de la nueva selección.

No por ser la última mencionada la clase **Query** es menos importante, todo lo contrario pues la misma es la que provee al usuario del constructor de consultas que le permite formular la misma y enviarla hacia el manager para que este haga su ejecución. Cuando se obtiene la solución de la consulta muestra la misma mediante la función **View_Results**, en forma de tabla, además notifica al manager si el usuario decidió guardar la estructura de la consulta al ver si son correctos los resultados que obtuvo al ejecutarla.

La clase **QueryParams** se ha concebido con el fin de que una instancia suya contenga (además de la sentencia SQL que es la consulta en sí) toda la información necesaria para poder solucionar una consulta recibida.

Cuando el usuario concluye la elaboración de una consulta, esta se le envía a la clase **Dealer** creándose un objeto de este tipo. Mediante el método ***SetQueyParams*** crea un objeto de tipo **QueryParams** que será atributo de la clase. Otro atributo de la clase **Dealer** es el fichero gestor-***ip.conf.XML*** (que ya se vio anteriormente). Así una vez instanciado un objeto **Dealer** crea la conexión con el gestor cuyas bases de dato involucra la consulta (***CreateConection()***); la ejecuta (***ExecuteQuery())***, y finalmente convierte el resultado a XML (***ConvertResultToXML()***).

Por otro lado la clase **StoreKeeper** es la encargada de almacenar las consultas exitosas. Estas consultas que el usuario decide guardar porque los resultados obtenidos mediante ellas son correctos, tendrán en el futuro un valor inestimable. En futuras versiones mejoradas del sistema, aprovechando métodos ya implementados en la clase **StoreKeeper** y las consultas almacenadas, se podrá asistir al usuario en el proceso de formulación de consultas.

Para crear una idea más exacta de cómo funciona “físicamente” este sistema se muestra un diagrama de despliegue para que se pueda comprender la configuración en funcionamiento del

sistema, incluyendo su hardware y software. Para nuestro sistema el diagrama de ejecución correspondiente es el de la Figura 4.

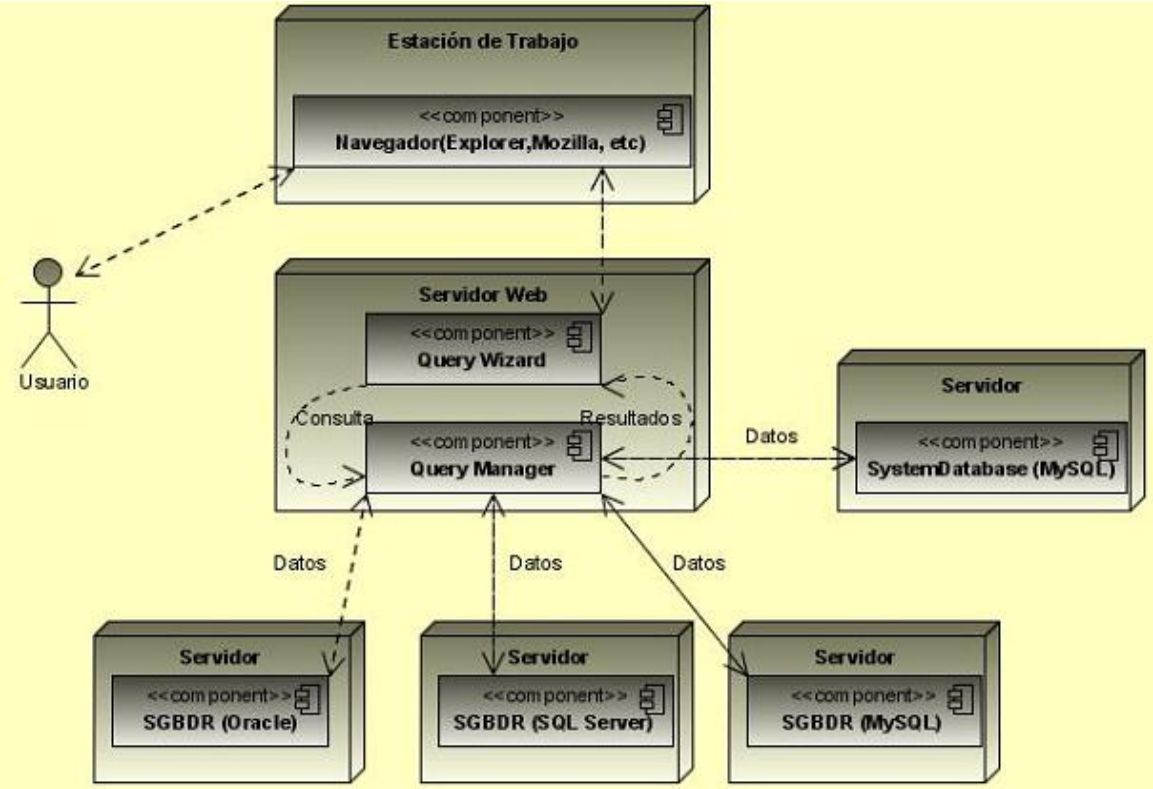


Figura 4. Diagrama de despliegue del sistema. Fuente: Elaboración propia.

El nodo **Estación de Trabajo** es desde donde el usuario hace uso del sistema utilizando en su PC cualquiera de los navegadores usuales como se especifica en el diagrama. Desde allí se conectará con el Servidor Web que es quien "hospeda" al asistente y al manager. A su vez este interactúa con los demás nodos (Servidores de Bases de datos) de los cuales extrae los datos con los cuales trabaja el usuario, como se puede apreciar son diferentes tipos de SGBDR. También el manager se comunica con el nodo **Servidor** (SystemDatabase), que es la Base de datos del sistema la cual almacena todas las consultas que se deseen guardar.

Para describir un poco más lo que se trata de explicar con el diagrama de despliegue presentado y ver a nivel de capas (Interfaz, Manager y Gestores) como es que fluye la relación entre ellos se presenta en el Anexo 1 el Diagrama de Secuencia correspondiente a la aplicación.

CAPÍTULO 3: EVALUACIÓN DE LA SOLUCIÓN

3.1. Presentación de los resultados del sistema.

En este capitulo se presenta el resultado final consistente en la implementación del módulo diseñado en el anterior capitulo. El resultado fue la herramienta Web que se muestra a continuación, para esto se detallará como es su funcionamiento e interrelación con el manager de consultas que aunque no tiene interfaz visual es vital a la hora del funcionamiento del sistema.

Siguiendo el diseño se comenzó a implementar la herramienta dividiéndola en dos grandes partes, la primera el asistente para hacer consultas (Query_Wizard) que es la interfaz que tendrá el usuario ante sí (Ver Anexo 1) y la segunda el manejador de consultas (Query_Manager) encargado de servir los metadatos al usuario y dar solución a las consultas. Como se ve ambos deben estar estrechamente relacionados entre ambos y la comunicación debe ser eficaz para posibilitar una solución correcta final. La interfaz del sistema trata de ser lo más amigable y fácil de usar posible, brindándole al usuario comodidades para el trabajo que realizará, en posteriores versiones se pretende mejorar el mismo incorporándole otras que por el momento no tiene, como por ejemplo el tratamiento de los **OUTER JOINS**, esta versión solo resulte los **INNER JOINS**.

A continuación se muestra el funcionamiento del módulo explicando detalladamente como trabajan cada una de las partes que lo componen, además de que puede ser este capítulo un documento de ayuda para una posterior utilización de este producto.

3.1 Selección de la Base de Datos (Navigator).

Cuando el usuario accede al modulo asistente para hacer consultas este carga los metadatos de los gestores a los cuales se tiene acceso en la sub-red. Estos son enviados desde el manager en formato XML, este trae codificado toda la información a la que se pudo acceder y sobre la cual será posible trabajar.

La estructura determinada para el XML se puede apreciar a continuación.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<schema>
    <id value >
        <database name >
            <table name >
                <column type primary_key foreing_key reference >
                </column>
            </table>
        </database>
    </id>
</schema>
```

Todo esto es realizado por el sistema, de forma tal que el usuario al acceder al asistente ya tiene "servidos" estos datos y solo tendrá que manipularlos para llegar al resultado final que desee.

Primeramente se selecciona el gestor en el cual se encuentran las bases de datos sobre las cuales desea conformar su consulta en "Seleccione del Gestor" que muestra todos los gestores que están disponibles para utilizar (Figura 5).

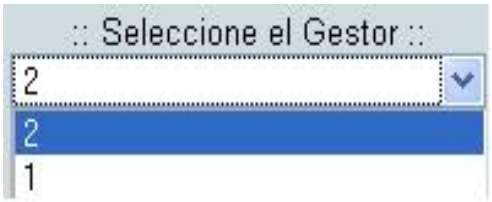


Figura 5. Selección del Gestor. Fuente: Query Wizard.

El usuario solo observa el ID del gestor pues no necesita conocer que SGBDR es realmente pues cuando realice la manipulación de los datos de cualquiera de ellos lo puede hacer con cualquiera de igual forma.

Después de elegir el gestor, en la lista desplegable "Seleccione la BD" se encuentran las bases de datos que contiene el gestor seleccionado en el paso anterior. Sobre las mismas podrá trabajar el usuario hasta que no se seleccione una nueva (Figura 6).

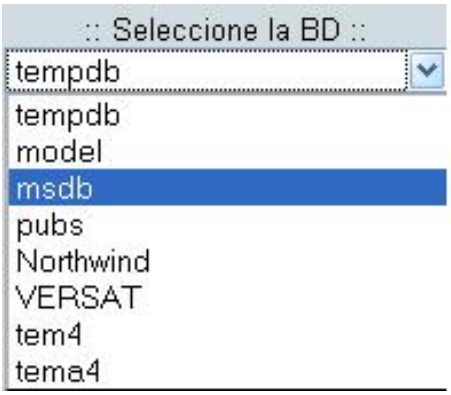


Figura 6. Selección de la base de datos. Fuente: Query Wizard.

3.2 Visualización de las Tablas (Nombre, Contenido y Relaciones).

Para facilitar la construcción final de la consulta se tiene la opción de visualizar una o varias tablas de la Base de Datos que está seleccionada en ese momento. El usuario podrá ver los nombres y tipos de los campos de tantas tablas como desee así como las relaciones entre ellas.

En la Ventana de Diseño se encontrará una lista con las tablas (Figura 7).

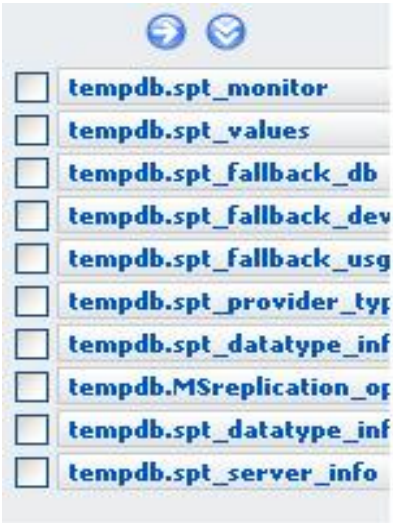


Figura 7. Lista de tablas en la base de datos seleccionada. Fuente: Query Wizard.

Si por alguna razón no se deseara tener esta lista desplegada en el meú que se encuentra encima de la misma se tiene la opción de ocultarla, esto se logra presionando el botón , presente en la parte izquierda de este.

Al marcar los botones de chequeo que se encuentran al lado izquierdo del nombre de las tablas, automáticamente en la vista de diseño aparecerá la tabla seleccionada (Figura 7). Si se desea que se muestren al unísono todas las tablas bastará con presionar el botón  que está en la parte superior del menú, para que aparezcan en la ventana.

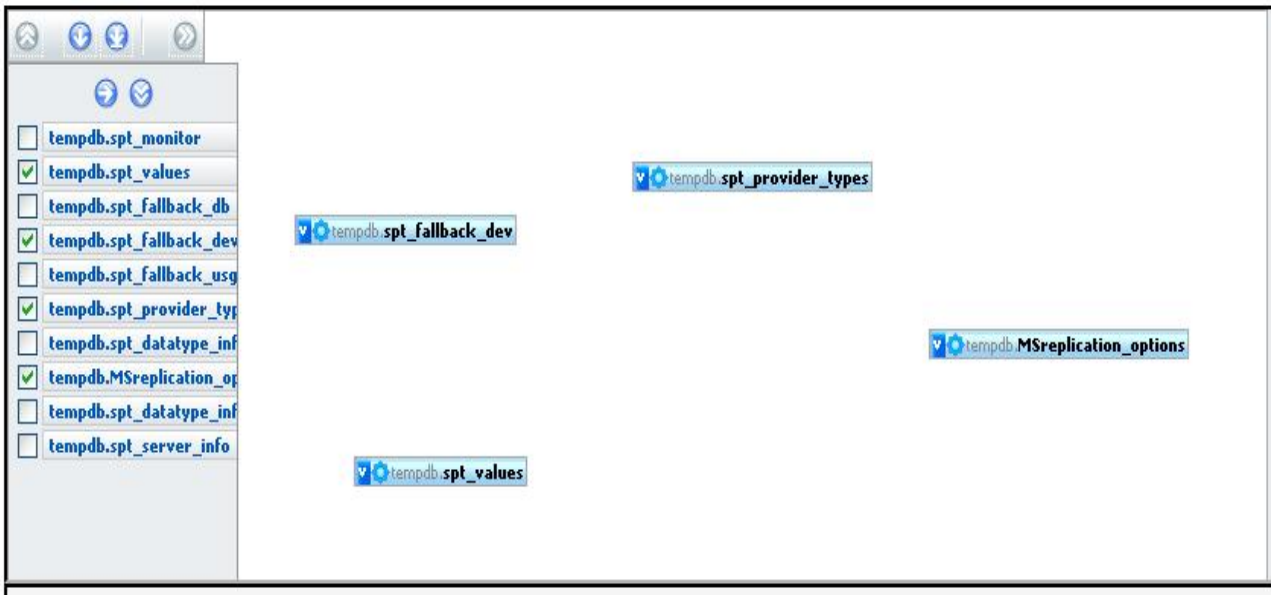


Figura 8. Vista de las tablas. Fuente: Query Wizard.

Para ver los campos contenidos en cada tabla así como sus tipos, será necesario ver el contenido de la misma, una vez que haya sido seleccionada y este en la vista diseño esto se podrá lograr de forma sencilla con solo presionar el botón  , que se encuentra en la parte superior izquierda de la tabla. Si se desea que todas se muestren en el mismo momento se debe accionar el botón  , presente en el pequeño menú en la parte superior de la vista. La otra opción del menú superior  , es solamente para desplazar el mismo hacia la derecha o hacia la izquierda, según la posición donde se encuentre. Las tablas y su contenido se ven de la siguiente forma (Figura 9).

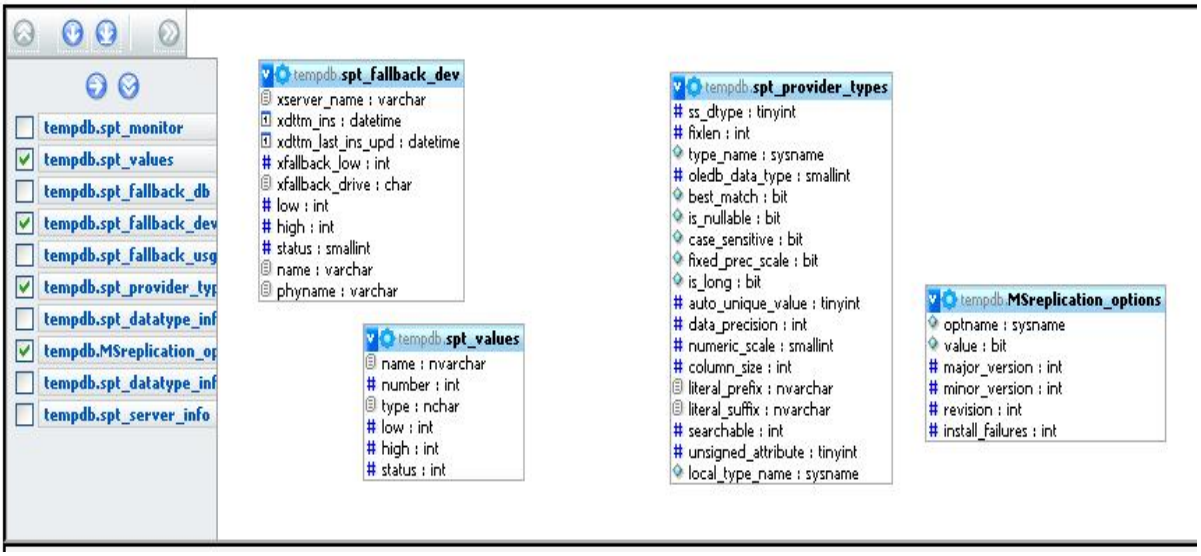


Figura 9. Contenido de las tablas. Fuente: Query Wizard.

Si se desea seleccionar mas de una se hace el mismo procedimiento hasta que se quiera.

Si alguna de las tablas que se vayan seleccionando tiene relación con una que este ya en la Vista de Diseño automáticamente se mostrara la relación existente entre ambas (Figura 10).

Figura 10. Relaciones entre tablas. Fuente: Query Wizard.

Es necesario aclarar que si se cambia de selección tanto en el gestor como en la base de datos esta ventana carga automáticamente los nuevos datos seleccionados por lo que se debe tener cuidado a la hora de hacer cualquier cambio, pues se puede perder información de forma indeseada.

Con estas facilidades el usuario obtendrá para sí una mayor información, aspecto que redundará finalmente en una construcción más correcta de la consulta lo cual es objetivo final de este asistente. Es muy útil poder conocer la información sobre todo de que tipos de datos

estamos encuestando en el momento de escribir algún criterio sobre los mismos, si no se conociesen se puede cometer un errores que posiblemente no darían al traste con la ejecución de la consulta pero si llevarían a obtener resultados diferentes a los necesitados.

3.3 Constructor de consultas.

La opción mas importante de este módulo asistente es dotar al que lo utilice de la facilidad de construir una consulta, tanto simple como compuesta y que permita al usuario relacionar diferentes bases de datos que se encuentren en un mismo gestor por eso podemos afirmar que la parte de más importante es donde finalmente el usuario realiza consultas.

Este se trato de diseñar de una forma que tuviera puntos de contactos con su similar de MSAcces de forma tal que quien anteriormente hubiese utilizado este software pudiese ver en el constructor de consultas algunas similitudes con el mismo y le hiciese más fácil su manejo (Figura 11).

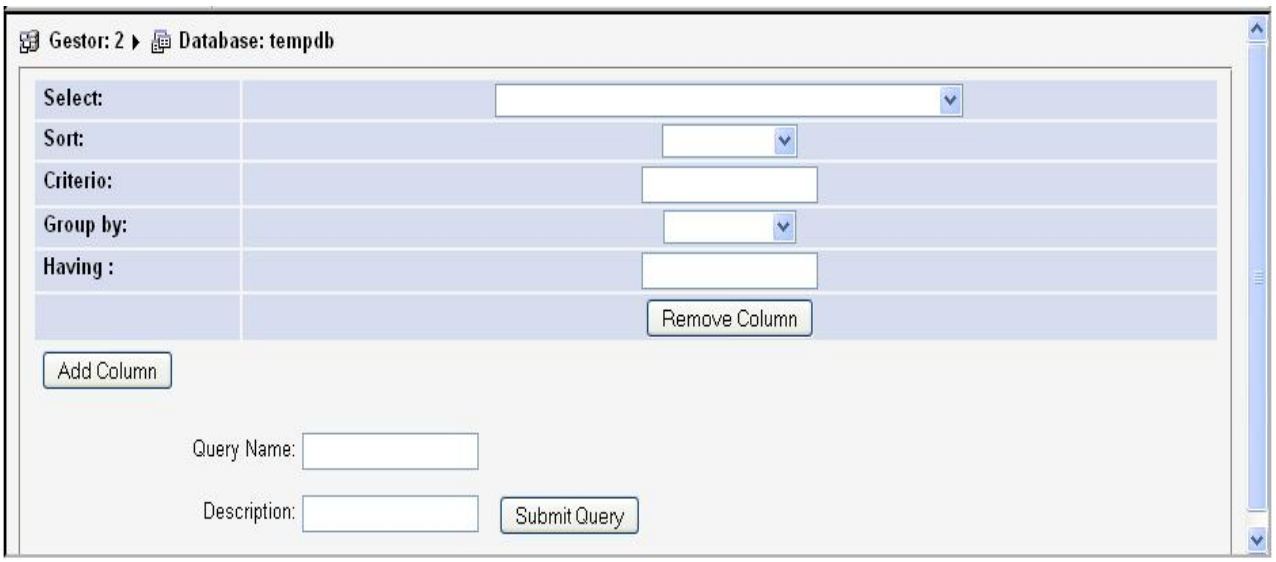


Figura 11. Constructor de consultas. Fuente: Query Wizard.

El campo de selección (SELECT) cargar automáticamente los campos de todas las tablas de la base de datos que este seleccionada y dan la oportunidad al usuario de elegir estos mientras no se cambie la misma, entonces serán elegibles la nueva agrupación de campos que estén dentro de la base de datos a la cual se cambió. Los elementos aparecen ordenados de la forma: base_dato.tabla.campo (Figura 12).

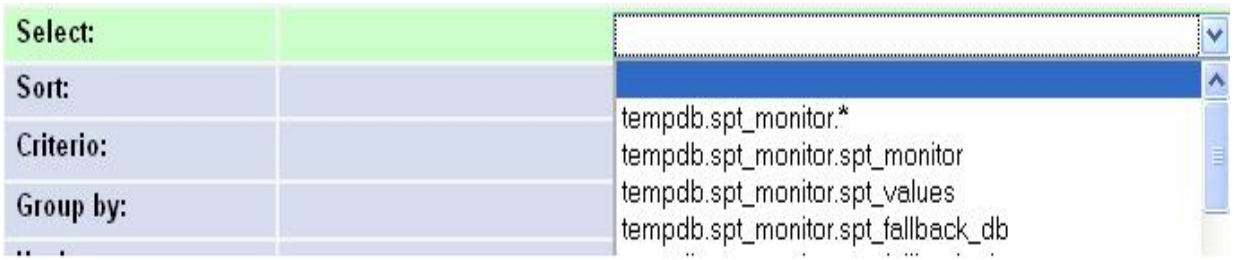


Figura 12. Cláusula Select. Fuente: Query Wizard.

Cuando se lleva esta información a la estructura definida se hace de la misma forma en que aparecen al usuario, separando cada uno por coma (,).

Para ordenar se puede disponer de las dos opciones clásicas, ordenamiento ascendente y descendente (Figura 13).

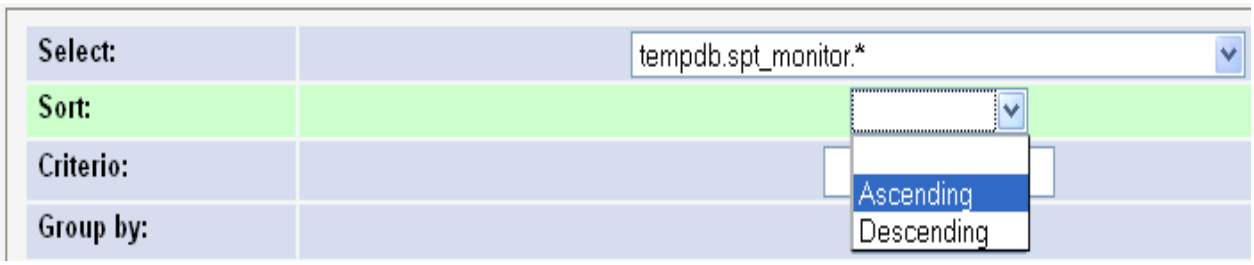


Figura 13. Opciones para el ordenamiento. Fuente: Query Wizard.

Puede ser elegible de cualquier forma por el usuario, sin importar si lo hace intercaladamente, al llevar esta selección al XML que contiene la consulta se especifica cual campo es el que se ordena de una forma u otra agregando a la selección hecha el campo entre paréntesis para que el manager comprenda cual opción es la vinculada a cada elemento.

La opción Group By es para agrupar el campo del SELECT por quien se seleccione en esta fila, es una lista desplegable que contiene una lista con los campos de la tabla del elemento seleccionado en la primera opción (Figura 14).

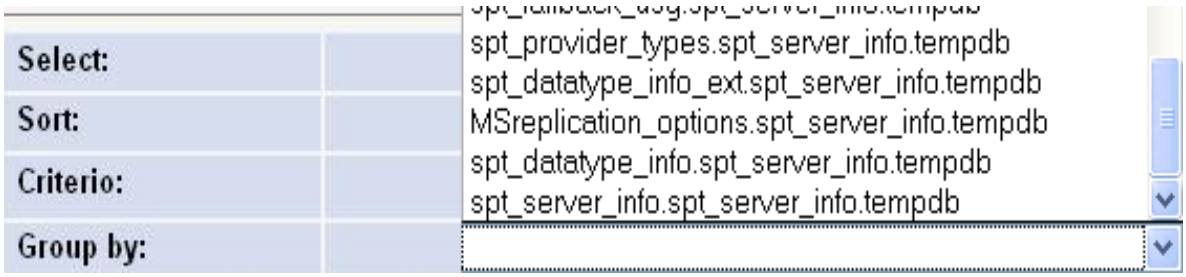


Figura 14. Opción Group By. Fuente: Query Wizard.

Los campos Criterio y Having son similares (el Having no es más que el criterio del Group By), tomando en cuenta que ambos son condiciones que el usuario debe escribir en ellos, se debe ser cuidadoso en el momento de llenarlos pues son campos de texto en los cuales se aceptan cualquier caracter (Figura 15), por lo que un error puede dar al traste con la consulta final.

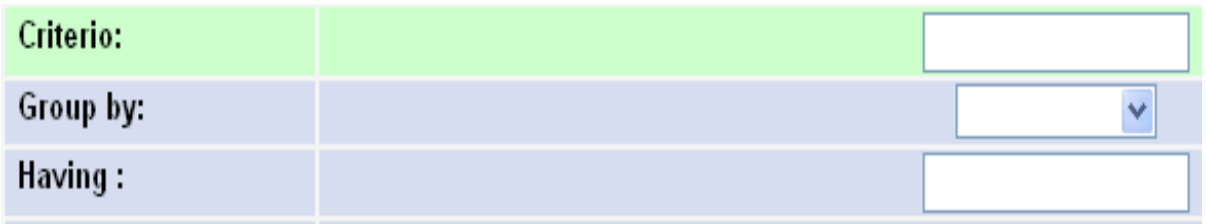
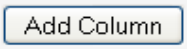


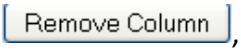
Figura 15. Campos: Criterios y Having. Fuente: Query Wizard.

Para identificar posteriormente las consultas hechas y tener una breve descripción del objetivo que llevó al usuario a realizar la misma se cuentan con los campos Querry Name y Description. En el primero se introducirá un nombre para la consulta formulada y en el segundo una breve descripción acerca de de la misma, estos datos son de suma importancia en el momento de salvar las consultas en la base de datos que las almacenará para su posterior búsqueda (Figura 16).

Figura 16. Nombre y descripción de la consulta. Fuente: Query Wizard.

Para poder enviar la consulta que se hace es necesario escribir en ambos campos si no se encontraría el usuario con que esta opción estaría deshabilitada.

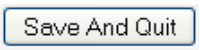
Cuando accedemos a el constructor de consultas este tiene solamente una columna para seleccionar elementos en ella, cuando el usuario quiera relacionar más de uno debe añadirle tantas columnas como elementos desee en su consulta. Para ello se cuenta con el botón 

, al accionarlo añade una columna a la vista. También es posible que se desee eliminar alguna de las columnas añadidas anteriormente por lo cual debajo de cada una se encuentra el botón 

, el mismo elimina la columna en la cual está si es activado, aún si esta ya contiene alguna selección, por lo que se debe ser muy cuidadoso a la hora de borrar no sea se pierda alguna información que realmente no se quiera que sea así.

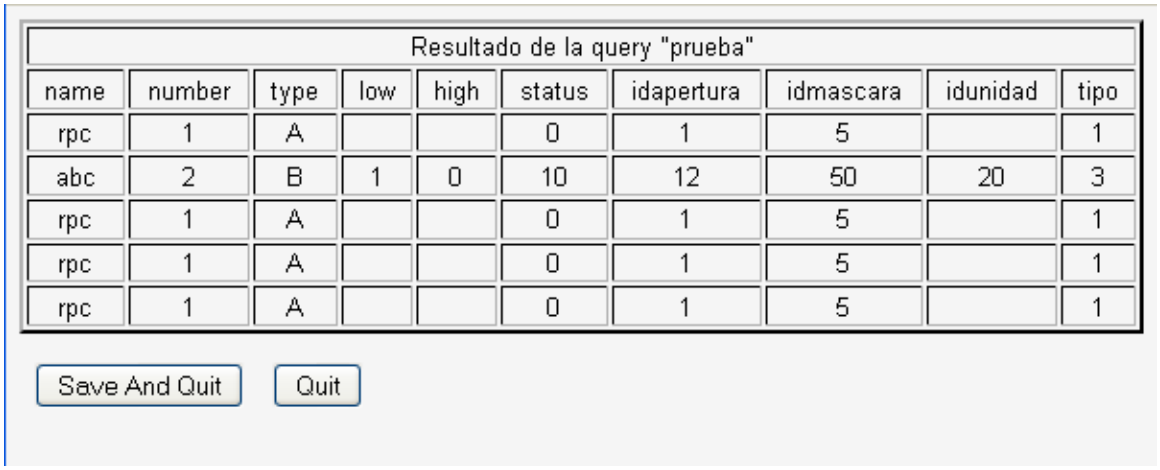
Por ultimo tenemos la opción de enviar la consulta a la cual se llega después de seguir todos los pasos descritos anteriormente, es importante que el usuario verifique bien toda la información que desea tener en su consulta antes de enviar la misma ya que pudiese un error no permitir que esta sea ejecutada por el manager. Al presionar el botón 

, este "envía" la consulta hacia el Manager en el XML previamente definido del que este obtiene e interpreta la información recogida finalmente en la consulta hecha.

Cuando el gestor de consultas ejecuta la misma y obtiene la solución, envía esta nuevamente al asistente en un XML, este permite al usuario ver en una nueva ventana en forma de tabla el resultado que se obtuvo (Figura 17). Además se cuenta con la opción de Salvar Consulta 

, que envía un mensaje al gestor de consultas para que guarde la estructura de la misma no su resultado, en una base de datos creada al efecto, o simplemente puede presionar el botón 

, que cerrará la ventana y llevará el usuario nuevamente hacia el asistente para que comience de nuevo el proceso si así lo desea.



Resultado de la query "prueba"

name	number	type	low	high	status	idapertura	idmascara	idunidad	tipo
rpc	1	A			0	1	5		1
abc	2	B	1	0	10	12	50	20	3
rpc	1	A			0	1	5		1
rpc	1	A			0	1	5		1
rpc	1	A			0	1	5		1

Figura 17. Visualización de los resultados de la consulta. Fuente: Query Wizard.

CONCLUSIONES

A partir de un estudio, sobre las principales diferencias de la implementación del lenguaje de consulta SQL en los SGBDR más populares, se ha implementado un módulo que da un primer paso hacia el objetivo de hacer transparentes al usuario estas diferencias, en un ambiente corporativo heterogéneo.

1. El tratamiento de la heterogeneidad en los dialectos de SQL con respecto del estándar, representa una oportunidad para mejorar en un ambiente heterogéneo, la capacidad de la organización de capitalizar sus recursos informativos.
2. Los mecanismos diseñados probaron la factibilidad de tratar el problema estudiado, a través de capas de abstracción de bases de datos (Databases Abstraction Layer).
3. La solución propuesta elimina la necesidad de que el usuario domine las diferencias entre los diferentes dialectos, adiciona la capacidad de realizar consultas, a través de una interfaz Web amigable, a múltiples SGBDR todo lo cual constituye una respuesta afirmativa a las preguntas de investigación planteadas.
4. Las capacidades actuales de la solución propuestas, una vez que extendidas, garantizan la habilidad de la solución de recuperar información a través de múltiples gestores, lo que demuestra la flexibilidad y adaptación de los mecanismos diseñados.

RECOMENDACIONES

Con el objetivo de darle continuidad a este proyecto y hacer de el una valiosa herramienta se plantean las siguientes recomendaciones.

1. Brindar la posibilidad de hacer consultas más complejas, dotando para ello al sistema de la capacidad de generar y resolver todas las posibles cláusulas de una sentencia SQL, acorde a un estándar.

2. Estudiar e implementar los métodos necesarios de forma que el sistema sea capaz de resolver consultas que involucren información de distintos gestores.

3. Explotar las potencialidades de la base de datos diseñada para el sistema, con el fin de asistir al usuario en el proceso de creación de consultas siendo proactivo ante sus necesidades de información.

4. Probar la efectividad del sistema sobre múltiples plataformas.

Bibliografía

1 Aalst, W. v. d. & Hee, K. v. (2002): Workflow management: models, methods, and systems. Hee, W. v. d. A. a. K. v. London, The MIT Press Cambridge, Massachusetts London, England

2 Anderson, J. D. (2003): Organization of knowledge. International Encyclopedia of Information and Library Science. 2nd. London. 471-490*pp*

3 Cali, A., Calvanese, D., Giacomo, G. D. & Lenzerini, M. (2000): Accessing Data Integration Systems through Conceptual Schemas. http://www.dis.uniroma1.it/~lastname *Última visita:* 15.11.2004

4 Cuellar Almeida, A. & Sáez Mosquera, I. (2004): Diseño de la arquitectura de una capa de abstracción de información para sistemas de información de soporte a la toma de decisiones logística. Departamento de Ingeniería Industrial. Universidad Central "Marta Abreu" de Las Villas. Santa Clara. Cuba. 61*p*

5 DBDS, D. D. S. (2001): Metadata solutions and their return of investment. http://www.dbdsolutions.com. *Última visita:* 22.10.2006. 7*p*

6 decision-making-confidence.com (2006): Decision Making Model. http://www.decision-making-confidence.com/decision-making-models.html. *Última visita:* 29.05.2007

7 Duchesi, P. & Chengalur-Smith, I. (1998): Client/Server benefit, problems, best practices. Communication of the ACM. Vol 41. 5. New York. 87-94 *pp*

8 Fernández, L. S. & García, N. F. (2005): La Web semántica: fundamentos y breve estado del arte. Novática. Vol 178. ATI. España. 6-11 *pp*

9 Foss, K. & Rodgers, W. (2006): The use of information in organizations: a cognitive perspective on Line Managers' involment in cross-unit activities. Copenhagen Business School. Center for Strategic Management & Globalization. http://www.google.com.cu/search?q=%22decisional+guidance%22&hl=es&client=firefox-a&rls=org.mozilla:en-US:official&hs=xYE&start=20&sa=N. *Última visita:* 18.01.2008. 30*p*

10 Fuß, C., Gatzemeier, F., Kirchhof, M. & Meyer, O. (2004) Inferring Structure Information from Typography. Lecture Note**,** 44-55*pp*.

11 George, G. (2000) A taxonomy of business process modelling and information systems modelling techniques. Brunel University (UK). Department of Information Systems and Computing**,** 1-34*pp*.

12 George S. Nezlek, H. K. J. a. D. L. N. (1999): An integrated approach to enterprise computing architectures. Communication of the ACM. Vol 42. 12. New York. 82-90 *pp*

13 Hauch, R. M. (2004): The Role of Model-Driven Enterprise Information Integration in Service Oriented Architectures. *Última visita:* 11.03.06

14 Hey, J. (2004): The Data, Information, Knowledge, Wisdom Chain: The Metaphorical link. Numberg, G. & Duguid, P.

15 Hirschheim, R. & Lacity, M. (2000): The myths and realities of information technology insourcing. Communication of the ACM. Vol 43. 2. New York. 99-108 *pp*

16 Hirsh, H., Basu, C. & Davison, B. D. (2000): Learning to personalize. Communication of the ACM. Vol 43. 8. New York. 102-106 *pp*

17 JTC, I. I. (2006): Part 1: Framework (SQL/Framework). *Information Technology- Databases languages - SQL.* New York, ANSI Customer Service Department.

18 Kerschberg, L. & Jeong, H. (2005): Just-in-Time Knowledge Management. eceb.gmu.edu/pubs/JIT_KM_Kerschberg_Jeong.pdf. *Última visita:* 23.11.2004

19 King, J. L. & Lyytinen, K. (2006): Information system: the state of the field. John Wiley & Sons, Ltd. USA. 355 *pp*

20 Kobryn, C. (1999): UML 2001: A standarization odyssey. Communication of the ACM. Vol 42. 10. New York. 29-37 *pp*

21 Kokkoras, F., Bassiliades, N. & Vlahavas, I. (2004) Modelling Information Extraction Wrappers with Conceptual Graphs. Proc.(2nd Volume) 3rd Panhellenic Conference on Artificial Intelligence (SETN'04) 5-8 May 2004.
22 Leymann, F. & Roller, D. (2002) Using flows in information integration. IBM System Journal*,* 41**,** 10*pp*.
23 Lin, C. (2003): Object-Oriented Database Systems: A Survey. www.cs.ucsc.edu/~lcx/courses/cmps277/cmps277-project.pdf. *Última visita:* 11.11.2005
24 Ling, T. W., Lee, M. L. & Dobbie, G. (2005): Semistructured database design. Australia. 300 *pp*
25 McCarthy, W. E. (2003) The REA modeling approach to teaching accounting information systems. Issues in Accounting Education*,* 18**,** 427-441*pp*.
26 McClure, S. (1997): Object Database vs. Object-Relational Databases ce.sharif.edu/.../84-85/1/ce384/resources/root/Object%20Database%20vs_%20Object-Relational%20Databases.pdf. *Última visita:* 01.01.2005
27 McGuinness, D. L. & Silva, P. P. d. (2003): Infrastructure for Web Explanations. LNCS 2870,. Berlin Heidelberg 2003. 1156-1167*pp*
28 McNamara, D. S. & O'Reilly, T. (2004): Learning: Knowledge Representation, Organization, and Acquisition *Knowlegde and information.*
29 Partridge, C. (2002): The role of ontology in integrating semantically heterogeneous Databases. Technical report. Padova, Italy. 26*pp*
30 Patel-Schneider, P. & Siméon, J. (2002): The Yin/Yang Web: A Unified Model for XML Syntax and RDF Semantics. http://portal.acm.org/. *Última visita:* 21.11.2006
31 Pawlowski, T., Barr, P. & Ring, S. (2001): Simulation of enterprise and Network architecture. www.mitre.org/news/events/tech04/briefings/729.pdf. *Última visita:* 04.03.2006
32 Perales, J. C., Catena, A. & Maldonado, A. (2001): Aprendizaje de relaciones de contingencia y causalidad: Hacia un análisis integral del aprendizaje causal desde una perspectiva computacional. Departamento de Psicologia Experimental. 1-62*pp*
33 Pérez-Acosta, A. M. (2002): El modelo Rescorla - Wagner a los veinte. http://www.psicologiacientifica.com/bv/psicologia-139-1-el-modelo-rescorla-wagner-a-los-veinte.html. *Última visita:* 06.02.2007
34 Piccinelli, G., Salle, M. & Zirpins, C. (2001): Service-Oriented Modelling for e-Business Applications Components http://ieeexplore.ieee.org/iel5/7566/20615/00953379.pdf. *Última visita:* 23.05.2001. 7*p*
35 Rada, R. & Craparo, J. (2000): Sharing standards: Standarizing software. Communication of ACM. Vol 43. 12. New York. 21-26 *pp*
36 Rodríguez Palmero, M. L. (2004): La teoría del aprendizaje significativo. Conference on Concept Mapping. Pamplona, Spain. 1-10*pp*
37 Rouse, W. B. (2004): Enterprise (as) system. http://www.ti.gatech.edu/docs/RouseEnterprisesAsSystemsSEVol8No22005.pdf. *Última visita:* 24.11.2004
38 Ruijgrok, C. J., Tavasszy, L. A. & Thissen, M. J. P. M. (2003): Emerging global logistics networks: Implications for transport system and polices. TNO Inro. Delft. 1-19*pp*
39 Sáez Mosquera, I. (2008): Procedimientos y arquitectura de apoyo para la asistencia decisional en procesos estratégicos de gestión logística. Departamento de Ingeniería Industrial. Universidad Central "Marta Abreu" de Las Villas. Santa Clara. 150*p*
40 Sáez Mosquera, I., Marx Gómez, J. & Hernández Pérez, G. (2003): Considerations to model a real world in business integrations. MT´2003. La Habana. Cuba. 22-25.04.2003. 354-361*pp*
41 Soh, C., Siew Kien, S. & Tay-Yap, J. (2000): Cultural fit and misfit: is ERP a universal solutions. Communication of the ACM. Vol 43. 4. New York. 47-51 *pp*
42 Universidad de Costa Rica (2004): El camino hacia Avalon, o de las conclusiones finales. www.ts.ucr.ac.cr/~historia/tcu/tinoco/docs/Conc.pdf. *Última visita:* 04.04.2004

43 Vdovjak, R. & Houben, G.-J. (2002): RDF Based Architecture for Semantic Integration of Heterogeneous Information Sources. wwwis.win.tue.nl/~houben/respub/wiiw01.pdf. *Última visita:* 12.12.2004

44 Wehr, H. (2003): Integrating Heterogeneous Data Sources into Federated Information Systems - Work in Progress. http://www.old.netobjectdays.org/pdf/02/papers/ws-gcse/Wehr-Final.pdf. *Última visita:* 03.12.2004. 1-12*p*

45 Zimanyi, P. E. & Belakhdar, D. O. (2004): Web information wrapping and security management. Universite Libre de Bruxelles. *Última visita:* 10.10.2004.