Heurística y Software Para Aprovechamiento De Metadatos De Base De Datos Relacionales - Prometeo

Marciszack, Marcelo
Maldonado, Calixto
Martinez Spessot, Cesar
Muñoz, Roberto
Navarro, Adrián
Peretti, Juan Pablo
Rogero, Luis
Universidad Tecnológica Nacional, Facultad Regional Córdoba

Resumen
"A partir del desarrollo de una heurística de explotación de meta datos y la construcción de un software capaz de implementar aquella metodología, como aporte a aumentar la productividad en el proceso de construcción de software. El producto responde al nombre de Prometeo y con este nombre fue presentado y aprobado en el programa de Grupos de Reciente Formación de la Agencia Córdoba Ciencia en el año 2006 y aceptado por la Secretaría de Ciencia y Técnica de la UTN, como grupo de investigación en el 2009".

Palabras Clave
Metadatos, **generación de sql**, **Query by example**.

Introducción

En presentaciones en WICC2005 [10] y 35JAIIO [11] se describía el proyecto Prometeo como "un sistema de software que se ejecute en múltiples plataformas de sistemas operativos, en principio, Windows 2000 y Red Hat Linux 7.1 y que sea capaz de generar sentencias SQL basados en los metadatos de los diccionarios de datos de Oracle y PostgreSQL: Consultas simples con SELECT, Consultas multi tablas de las tablas relacionadas con Constraints Referenciales, DML para insertar, borrar y actualizar columnas, DDL para creación de objetos como vistas y tablas accesorias, modificación y borrado de objetos como Vistas, OQL o lenguaje de consulta de Objetos cuando la base de datos estudiada contenga esa opción"

El proyecto, que responde al nombre extenso de "Desarrollo de un método y una herramienta para el aprovechamiento de Metadatos de Base de Datos Relacionales" fue acreditado este año por la SeCyt del UTN, y ha logrado sólidos avances. El nombre extenso es habitualmente reemplazado por el nombre abreviado de Prometeo, que según Wikipedia [20] "es el Titán amigo de los mortales, honrado principalmente por robar el fuego de los dioses en el tallo de una cañaheja, darlo a los humanos para su uso y ser castigado por este motivo." La metáfora del nombre es por que este software ayuda a conocer y aprovechar la información almacenada en los metadatos y con la ayuda de unos criterios o Heurística permite construir consultas, ejecutarlas y guardarlas en tablas propias.

La propuesta inicial fue presentada en un paper que fue aceptado en las 34 JAIIO desarrolladas en Rosario en el mes de Septiembre del 2005, en el simposio ASIS [10]

Elementos del Trabajo y metodología

El hecho que permite obtener provecho de los metadatos es la capacidad de la sentencia SELECT de poder combinar el

resultado de seleccionar el contenido de las columnas de las vistas del Diccionario con literales para poder escribir sentencias SELECT validas. Ejemplo:

Dada la vista del diccionario de datos Oracle [CON05] USER_TABLES, que contiene todas las tablas del usuario conectado, se podría generar una sentencia select de todas las columnas de las tablas existentes con la siguiente sentencia:

*SELECT 'select * from '||table_name||' ;'*
FROM USER_TABLES;

La ejecución de esta consulta generará en la salida estándar de la herramienta usada:

*select * from tabla1;*
*select * from tabla2;*

Una sentencia SELECT por cada registro consultado de la vista del diccionario USER_TABLES.

Agregándole una línea de texto podríamos crear una sentencia SQL para crear una vista que guarde esta consulta como un objeto de la base. Ejemplo:

*SELECT 'create or replace view '||table_name ||'_v as select * from '||table_name||' ;'*
FROM USER_TABLES;

Lograríamos esta sentencia valida SQL, basada en todas las columnas de la tabla 'PEPE' perteneciente al usuario actual:

*create or replace view as PEPE_v as select * from PEPE ;*

Este funcionamiento del lenguaje SQL es la base del generador de sentencias SQL, basadas en los metadatos.

Sobre la Heurística, se ha diseñado con el objetivo del aprovechamiento de los metadatos residentes en motores de bases de datos comerciales como Postgresql y Oracle y en desarrollo, como TecnoDB.

Este método o heurística guía a la generación de las sentencias útiles dentro de todas las combinaciones posibles y contiene 21 puntos o criterios a seguir y los resultados a obtener, mas adelante se mostrrán como ejemplo algunos de estos critarios.

Aplicando esta heurística, se espera poder generar sentencias SQL con algun grado de optimización [3] que permitan ahorrar trabajo y errores comunes a los desarrolladores de aplicaciones en estas bases de datos. Estamos buscando mejorar y completar el método que viene desarrollandosé desde su inicio y aumentar las funcionalidades de su interfaz, para que pueda definir qué consultas se deben generar para que asista al desarrollador a construir reportes, generar conjuntos de filas de pruebas de integridad y desarrollar una herramienta que la aplique par lograr que con el solo hecho de ejecutarla, obtener un subconjunto de sentencias útiles sin necesidad de conocer profundamente el modelo de datos objetivo. Una de las formas de mejora en el proyecto es diseñar el método para generar Java Beans, es decir, extender la accion a construir codigo basado en JAVA, generando codigo reusable [21] [19]

Prometeo tiene como objetivo también aportar al proyecto TecnoDB sirviendo como aporte al diseño del diccionario de datos, aún no finalizado en el desarrollo del motor de Bases de datos relacional.

Las sentencias que genera Prometeo son inicialmente SELECT, y CREATE VIEW. Se completará con las sentencias INSERT y CREATE TABLE, pudiendo aumentar esta capacidad a otras sentencias mas. Las sentencias INSERT van ser generadas para insertar conjuntos de datos de prueba de las tablas elegidas, respetando las restricciones de integridad referencial detectadas en el modelo[8] [9] [6].

En imágenes una explicación de cómo este software, genera, muestra, ejecuta, muestra los resultados y grababa en sus propias tablas, las sentencias SQL obtenidas con las indicaciones del usuario. En la figura 1 se muestra un esquema del funcionamiento.

Fig. 1 Pasos en la generación de Sentencias

En la figura 2 el modelo de datos inicial, modificado durante 2008 al requerir almacenar mas información.

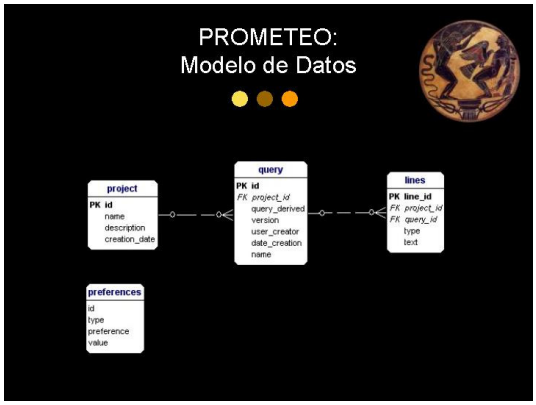

Fig. 2

La interfaz se conecta al diccionario de datos de la base de datos objetivo y guarda en la base de datos repositorio (que puede ser la misma) la información acerca de los objetos analizados en tablas propias. Permite ejecutar las consultas generadas o almacenadas y mostrar el resultado en la pantalla, para luego guardarla en el repositorio. Los lenguajes a utilizar en la construcción son PHP y JAVA para la interfaz y procedimientos y SQL para la comunicación con las bases de datos objetivo. Se realizó la comunicación con PostgreSQL y se esta trabajando para realizar la conexión con MySQL.

También se avanzará en que la interfaz pueda recopilar información de documentación con los contenido de Comentarios sobre tablas y columnas posibles de ser guardados en las en las vistas del diccionario de las bases de datos objetivo, por ejemplo, en Oracle [4] la vista ALL_TAB_COMMENTS y la ALL_COL_COMMENTS permitiran mejorar la documentacion de los desarrollos que utilicen a Prometeo como herramienta.

Se están definiendo los casos de usos correspondientes a cada punto enunciado para poder diseñar la aplicación que lo resuelva y que sirva como base al proceso de testing [5] [14] [13].

Resultados

Los avances en la Heurística [17] para el uso de los Metadatos en los diccionarios de las Bases de Datos Oracle, PostgreSql, MySql. Se ha tomado como referencia las facilidades que ofrecen herramientas incluidas en los motores comerciales nombrados y herramientas de terceras partes para realizar comparaciones de posibilidades. [18]
Diseño del modelo de datos de Metadatos para TecnoDB.
Diseño de Vistas para las distintas versiones de RDBMS usadas para lograr independencia en las distintas instalaciones.
Creación de Java Beans con la información de los Metadatos para construir automáticamente métodos de acceso, actualización a los datos de las tablas analizadas.

Hasta el 2007 se programó un sistema web, desarrollado en PHP 5 y que obtuviera información de los metadatos de Postgresql [10].

Durante el 2008, se trabajo sobre MySQL lográndose lo mismo que hasta el 2007 y además se pudo construir sentencias SELECT multitablas, es decir relacionaba tablas con constraints de clave foránea para escribir los Joins sobre las mismas tablas. Esto estaba previsto en la Heurística original pero no se había concretado aún [17] [1].

Durante el trabajo con MySql se observó que el modelo de Datos objetivo de los metadatos de MySQL son manejados por el motor a través de la base de datos INFORMATION_SCHEMA, la misma está caracterizada por no contener tablas, sino que la información es manejada a través de vistas. En este Diagrama de Entidad-Relación se pueden observar las vistas más importantes (sobre todo las utilizadas por Prometeo) en forma de tablas y relaciones entre las mismas. [15]

Se modificó esta pantalla con la idea de brindar la posibilidad al usuario de seleccionar el tipo de consulta a realizar. Se vio que sería importante lograr la pantalla de inicio debería mostrar una leyenda de bienvenida al usuario y darle la posibilidad de seleccionar una tarea entre varias, como por ejemplo, generar consultas, seleccionar consultas guardadas de acuerdo a un proyecto particular y ejecutar dicha consulta y/o ver todas las consultas guardadas junto a los proyectos a las cuales pertenecen. (Fig. 3)

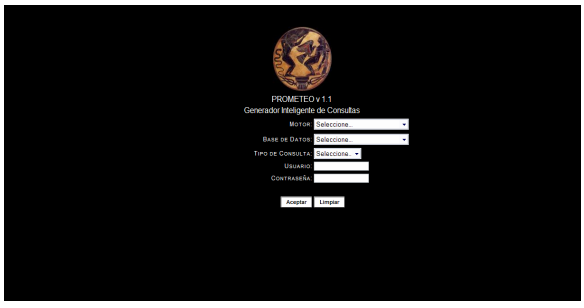

Fig 3 Pantalla de Inicio de la V1.1

La siguiente pantalla (Fig. 4) ofrece dos criterios para lograr los emparejamientos de columnas para las consultas multitablas, pero solo soporta uno solo de los dos criterios (Tablas referenciadas por esta tabla). Es decir, la funcionalidad está implementada para ese criterio solamente, por lo que debería desarrollarse la funcionalidad para el criterio restante.

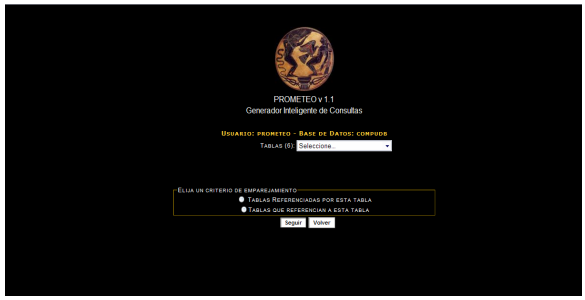

Fig. 4 Pantalla para generar consultas multitablas.

En la siguiente imagen (Fig 5) se muestra página con la consulta generada a partir de los datos seleccionados anteriormente. La funcionalidad de los botones "Guardar Consulta" y "Ejecutar Consulta" esta en desarrollo.

Fig 5. Resultado de la Generación de consultas multitabla.

Son objetivos del trabajo durante este año:

-Extender la aplicación a otros motores comerciales como SQL Server y DB2 [7].
-Avanzar con nuevos productos del análisis de los metadatos como generar automáticamente Java Beans que contengan los métodos get y set de una tabla. [12]
-Migrar la programación de la aplicación de PHP a Java. [16]
-Generar en las tablas estudiadas set de datos de prueba de acuerdo a los criterios

que se podran registrar como preferencias en la herramienta.

También se ha previsto para la próxima etapa del proyecto la inclusión de elementos de diseño, tales como estilos web para lograr una interfaz amigable con el usuario y desarrollar un asistente o Wizard para tener un punto de inicio y donde el usuario pueda seleccionar la/s tarea/s que interesan realizar y que se vayan desarrollando en la secuencia correcta.

Cabe señalar que del desarrollo anterior, solo se hizo una adaptación de lo existente para ser utilizado con MySQL, no se desarrolló de nuevo la funcionalidad para consultas simples, y que deben completarse ciertos aspectos que son heredados del desarrollo anterior.

En cuanto a la formación de Recursos Humanos, los participantes son tres alumnos de la UTN-FRC que están cursando los últimos años de la carrera de Ingeniería y su trabajo ha sido considerado dentro del ámbito de la materia Práctica Profesional Supervisada o PPS, durante el presente año.

Durante el año 2008 estuvieron trabajando como Becarios de la Secretaria de Alumnos y Extensión al desarrollar la tarea descripta.
Se hará transferencia de los aprendizajes adquiridos en algoritmos de mejora de performance, almacenamiento y procesamiento de consulta de datos a las cátedras de la Universidad Tecnológica Nacional, especialmente a Sistemas de Soporte de Decisión (5to año de la carrera de Ingeniería en Sistemas de Información) donde se tratan temas avanzados de base de datos.

El trabajo esta relacionado con el grupo de desarrollo de TecnoDB de la UTN, con el objetivo de que Prometeo sea una herramienta de TecnoDB para mejorar sus prestaciones al contar con una interfaz amigable para realizar consultas SQL.

Discusión

El objetivo de la Heurística es lograr desarrollar los criterios para que un método, basado en ellos, permita generar automáticamente la mayor cantidad de consultas, basado solamente en la información de metadatos, sin ninguna información o acerca del contenido de estas tablas, no sabemos a priori de qué se tratan las tablas, ni qué contenido tienen, sólo su estructura registrada en el diccionario.

A priori y sin intervención del analista, diseñador o desarrollador y con la información de los metadatos sola, no se puede indicar qué columnas se consultarán juntas ni en qué orden son necesarias, por lo que se plantea que el método de generación debería proveer todas las posibilidades de columnas y de su orden.

El problema es que necesitaremos como máximo una cantidad de vistas equivalentes a todas las variaciones sin repetición del número de columnas de una tabla tomadas por el mismo número de elementos. Que es un número calculado según la formula de variaciones sin repetición [22] a saber:

Si la tabla tiene n columnas y quiero mostrar de a una, de a dos, de a tres columnas y así con las n columnas, sin repetirlas, y además en todos los órdenes posibles, la fórmula es la sumatoria de las variaciones sin repetición de los n arreglos posibles que es el valor de k.
Esto hace que tendremos, para una tabla con

1 columna, 1 consulta a generar, V= 1!/(1-1)! ; V=1,
2 columnas, 4 consultas a generar, V= 2!/(2-2)!+2!/(2-1)! ; V=4
3 columnas, 15 consultas a generar, V= 3!/(3-3)!+3!/(3-2)!+3!/(3-1)! ; V=15

4 columnas, 64 consultas a generar, V= 4!/(4-4)+4!/(4-3)!+4!/(4-2)!+4!/(4-1)! ; V=64
5 columnas, 325 consultas a generar, V= 5!/(5-5)+ 5!/(5-4)+5!/(5-3)!+5!/(5-2)!+5!/(5-1)! ; V=325

Una tabla con 10 columnas y de la que se desea crear una vista por cada forma de mostrar las columnas, de a una, de a dos (…) y de a 10 en los distintos ordenes posibles necesitaremos crear 9.859.061 de consultas

Las tablas dentro de un modelo relacional tienen un límite según la implementación, en el orden de doscientos cincuenta y cinco a mil veinticuatro columnas por tablas, lo que nos indica la imposibilidad de lograr todas las variaciones necesarias.

Esto en principio nos da una dimensión máxima del problema a resolver. La propuesta es avanzar en la definición de pautas para elegir cuáles son las consultas que sería necesario generar para que nos ayude a que el número posible a resolver sea importante, ya que cuantas más posibilidades se cubran, menos trabajo de desarrollo de consultas será necesario con un menor margen de error que hacerlo manualmente.

Considerar inicialmente el uso los metadatos de las vistas del Diccionario de Datos y con su información, se podrán fijar, sin pretender agotarlas en esta presentación el tema, basados en el sentido común y la experiencia en desarrollo de sistemas, las siguientes pautas:

1)Crear las vistas con el nombre de la tabla concatenado al de la o las columnas intervinientes con el prefijo ‘get’ y el sufijo ‘v’
Se desea generar un patrón de nombres similar a los métodos get_atributo() de la definición de una Clase de datos en JAVA, Ejemplo:

get_TABLA_COLUMNA_v

2) Se creará una consulta para cada columna de la tabla
Esta pauta es una abreviación de la enunciada en el punto Estudio del problema y busca ejemplificar como se podría resolver las siguientes pautas.
Ejemplo:

```
SELECT 'create or replace view get_'||table_name||'_'||column_name||
                ' as select '||column_name||'
        from '||table_name ||';'
FROM user_tab_cols
ORDER BY table_name
```

Si seguimos con la tabla PEPE, esta tiene tres columnas y las queremos mostrar con esta vista de a una, obtendríamos:

```
create or replace view get_PEPE_ID as select ID from PEPE
create or replace view get_PEPE_DESCRIPCION as select DESCRIPCION from PEPE
create or replace view get_PEPE_MONTO as select MONTO from PEPE
```

creamos las vistas y ahora sabiendo la forma de nombrado de la generación, podríamos hacer la siguiente consulta:

```
select * from get_pepe_id;
```

en vez del tradicional:

```
select id from pepe;
```

No hemos acortado la longitud del texto, es más, la aumentamos, pero se generó sin escribirla y además aplicamos un patrón previamente definido, seleccionable de una lista de valores. En la segunda tuvimos que conocer el nombre de la tabla y la columna y digitarla sin errores. El método de descubrimiento de las consultas previamente generadas, es más sencillo que hacerlo manual ya que necesitamos conocer

de antemano que existe esa columna en esa tabla.

3) Se generará sólo una vista para cada tipo de datos.
Esta definición permitiría acotar el número de consultas necesario a tres para cada familia de tipos de datos, Numéricos (NUMBER, INT, y otros variaciones), de Carácter (VARCHAR, CHAR y otras variaciones) y de Fecha (DATE, TIMESTAMP y otras variaciones) y una adicional que contenga las columnas de los tipos de datos no incluidos en la enumeración anterior. Esto limitaría a que sobre toda tabla es necesario generar una consulta para todas las columnas, y cuatro consultas más para los distintos tipos de datos posibles. El orden de las columnas no será tomado en cuenta y es el que la consulta original devuelva, obteniéndose así una sola variación entre todas las posibles.

4) El orden de las columnas lo determinara el usuario que dirige la generación
La pauta anterior limita a 5 consultas por tabla y ofrece un orden fijo para generar una sola variación entre todas las posibles.
Se puede considerar permitir que el usuario defina diferentes órdenes de acuerdo a los requerimientos del negocio y que esta preferencia se guardará en un modelo de datos especial para que el proceso lo considere y así generar las consultas o vistas con las columnas en el orden preferido por el usuario.

El proceso que usara este método en PROMETEO puede tener un fin de generar código a pedido o un proceso en lote que genere todos los objetos de una sola corrida. Para esto se puede definir que el usuario seleccione las columnas que desee y el orden final en el que se mostraran las columnas.

5) Crear una consulta por cada tipo de dato y de estas, una por cada tamaño diferente
Este criterio aumentará a un número no determinado a priori de consultas, pero con la ventaja de aumentar la probabilidad de reunir columnas que tengan un significado similar, agrupando importes, claves numéricas que normalmente son números enteros.

6) Crear una consulta para cada columna con la condición where y la clave primaria
De la definición de la Clave primaria podemos crear una consulta que es muy valiosa que es el acceso directo a un registro por el valor de esta clave. La clave primaria puede ser simple o compuesta y en este caso deberá contemplarse a todas las columnas que la conforman, excepto la misma clave primaria por redundante.

El mecanismo pensado es a través de un SELECT ejemplo:

```
SELECT 'create or replace view get_'||
            table_name
||'_'||tc.column_name||'_v_Pk'||  '  as  select
'||tc.column_name||
            ' from '||tc.table_name ||
            ' where '||(select column_name
                        from
user_cons_columns cc, user_constraints c
                        where
cc.constraint_name = c.constraint_name
                        and  cc.table_name  =
tc.table_name
                        and  constraint_type  =
'P')||' = valor;'
FROM user_tab_cols tc
WHERE tc.table_name = 'PEPE'
AND tc.column_name not in
    (select column_name
     from  user_cons_columns  cc,
user_constraints c
     where  cc.constraint_name  =
c.constraint_name
     and cc.table_name = tc.table_name
     and constraint_type = 'P')
ORDER BY tc.table_name;
```

Lo que producirá, si se ha registrado en el diccionario que la columna ID de PEPE es su clave primaria, el siguiente grupo de

sentencias de creación de una vista por columna :

create or replace view get_PEPE_DESCRIPCION_v_Pk as select DESCRIPCION from PEPE where ID = valor;

create or replace view get_PEPE_MONTO_v_Pk as select MONTO from PEPE where ID = valor;

7) Obtener una consulta que cuente los registros de cada tabla
Se la puede denominar get_count_TABLA_v y contendrá la expresión con la función de grupo Count y el comodín (*)

8)Obtener una consulta para todas las columnas definidas como numéricas afectadas por cada funcion de grupo
Generar una vista generada que mostrará a todas las columnas numéricas afectadas a una función de grupo existente, una consulta por función. Las funciones de grupo son SUM, MAX, MIN, AVG, STDDEV, VARIANCE. Se la denominaria get_tabla_columna_funcion_v .

9)Obtener una consulta para todas las columnas definidas como alfanuméricas afectadas por cada funcion de grupo aplicable
Generar una vista generada que mostrará a todas las columnas numéricas afectadas a una función de grupo existente, una consulta por función. Las funciones de grupo aplicables a las columnas alfanuméricas son MAX y MIN. Se la denominaría get_tabla_columna_funcion_v .

10) Obtener una consulta para todas las columnas definidas como fechas afectadas por cada función de grupo aplicable
Generar una vista generada que mostrará a todas las columnas numéricas afectadas a una función de grupo existente, una consulta por función. Las funciones de grupo aplicables a las columnas de fechas son MAX y MIN. Se la denominaría get_tabla_columna_funcion_v .

El método completo, descripto hasta estos diez primeros puntos, contiene hasta el momento 21 puntos [17].

Se observó que los resultados alcanzados nos reorientan hacia mayores desafíos al encontrar nuevas posibilidades de mas aprovechamiento de los Metadatos a alcanzar en las siguientes etapas.

Conclusión

La idea original de la heurística de definir 21 formas de aprovechar los Metadatos, dio paso a la construcción de herramientas que cumplieron los primeros puntos y se observó que como se había previsto, los resultados fueron de diversa utilidad.
Al definir grupos de columnas de una tabla con coherencia formal, como los agrupamientos por tipo de dato, es decir, el generar una sentencia select que traiga todos las columnas numéricas (con o sin funciones de grupo), las alfabéticas (con o sin formateo), se llegó a consultas que no tenían la utilidad de producto final, como para un reporte requerido por el usuario, pero era una base para lograr cumplir un requerimiento. Esto nos mostró que podía usarse como base y abreviar la curva de aprendizaje de un modelo de datos que puede ser muy complejo y por ende demorar la incorporación de nuevos recursos humanos al grupo de desarrollo. Este resultado, si bien no tuvo el mismo valor que cuando Prometeo, generaba las consultas multitabla de todas las combinaciones posible, nos dio pistas de usos ulteriores a las primeras consultas de agrupación por tipo de datos.
En general el avanzar en construir las consultas y ver los resultados, derivó en el descubrimiento de que deben buscarse mas cantidad de resultados derivados directamente y de derivados indirectos para

lograr mayores aplicaciones de la idea, ampliando las prestaciones de la heurística.
Se esta desarrollando la forma de expresar esta heurística en forma que pueda ser patentable.
Otra de las conclusiones de esta etapa del trabajo es que debe mejorarse la herramienta, realizando estudios de Usabilidad a la interfaz para mejorar la adopción de la misma.
Se que a mayor cantidad de Motores de Bases de datos estudiados, mayor valor tendrá el resultado.
Por fin, se puede afirmar que el entusiasmo demostrado por los últimos participantes en la tarea es un impulso muy importante hacia nuevos logros, en la investigación y desarrollo de software que responda a las necesidades de los desarrolladores de aplicaciones.

Agradecimientos (Times New Roman, 10, negrita)
A la Agencia Córdoba Ciencia, y al Ministerio de Ciencia y Técnica de la Provincia de Córdoba.
A la Secretaria de Ciencia y Técnica de la Fac. Regional Córdoba de la UTN.
A todos estos organismos por el apoyo a nuestro trabajo.

Referencias

[1] www.Altova.com/DatabaseSpy visitado Mayo2008

[2] Burgos M. (1999) "Tutorial de SQL", http://members.tripod.com/~MoisesRBB/sql.html Fecha de consulta: mayo 2005

[3] Cisterna M.(2002) "Métodos de Optimización de Consultas para el Lenguaje SQL", http://macine.epublish.cl/tesis/index.html Fecha de consulta: agosto 2005

[4] Documentación oficial Oracle Database, Concepts 10g Release 2- June 2005

[5] Alistair Cockburn "Writing Effective Use Cases, Addison-Wesley

[6] Date C. J. - Darwen H. (1997) "A guide to the SQL Standard", Addison Wesley, Reading – MA, Fourth Edition

[7] DB2 edicion express C http://www-01.ibm.com/software/data/db2imstools/db2tools/db2wqt/ visitado mayo 2008

[8] http://www.sqlmanager.net/products/mssql/datagenerator

[9] http://www.sqlmanager.net/?gclid=CNeW07SouZUCFQyenAod9gQCQA visitado junio 2008

[10] Gastañaga, Iris et al TecnoDB: Desarrollo de una metodología de aprovechamiento de Metadatos de los diccionarios de Datos de Bases de Datos Relacionales para lograr un generador de sentencias SQL JAIIO34 ASIS 2005 Argentine Symposium on Information Systems Rosario, Argentina - September 29-30, 2005

[11] Gastañaga, Iris et al TecnoDB una Base de Datos Relacional y Prometeo un metodo de aprovechamiento de Metadatos y Generador de Consultas WICC 2006

[12] http://java-source.net/open-source/sql-clients/isql-viewer visitado Julio de 2008

[13] Jacobson I. - Booch G. - Rumbaugh J. (2000) "El Proceso unificado de Desarrollo de Software", Addison Wesley, Madrid – España

[14] Craig Larman "Applying UML and Patterns" , Addison- Wesley

[15] http://www.mysql.com/products/tools/query-browser/ visitado Mayo 2008

[16] http://www.quest.com/ visitado en Julio 2008

[17] Martinez Spessot et al, PROMETEO Un método de explotación de Metadatos. JIDIS 07

[18] http://www.toadsoft.com/

[19] http://java.sun.com/javase/technologies/desktop/javabeans/index.jsp

[20] http://es.wikipedia.org/wiki/Prometeo visitado Abril 2009

[21] Java Beans http://es.wikipedia.org/wiki/JavaBean - Fecha de consulta: desde diciembre 2008

[22] Variaciones
http://es.wikipedia.org/wiki/Variación Consultado 06/07/2005

Marcelo Marciszac y Calixto Maldonado
Departamento de sistemas de información
Facultad Regional Córdoba
Universidad Tecnológica Naciona
Maestro Lopez S/N Ciudad Universitaria
mmarciszack@sistemas.frc.utn.edu.ar
calixto@bbs.frc.utn.edu.ar

Datos de Contacto: