Modelado de la Metadata para el Desarrollo de Herramientas de Productividad sobre Múltiples Manejadores de Bases de Datos Relacionales

Héctor Andrés Melgar Sasieta, Abraham Eliseo Dávila Ramón

Pontificia Universidad Católica del Perú, Departamento de Ingeniería, Grupo de Investigación y Desarrollo en Ingeniería de Software,
Lima, Perú, Lima 32
{amelgar, edavila}@pucp.edu.pe

Resumen. En la especialidad de Ingeniería Informática de nuestra universidad se han venido desarrollando herramientas que ayuden a mejorar la productividad de las personas que desarrollan software y que trabajan con bases de datos relacionales. En este documento presentamos los modelos que hemos empleado para la construcción de herramientas de productividad que trabajan sobre múltiples sistemas administradores de bases de datos relacionales, una comparación de los modelos y la definición de la arquitectura más adecuada a las necesidades de escalabilidad hacia otros sistemas administradores de bases de datos relacionales.

1 Introducción

Hace algunos años atrás, en el curso de Desarrollo de Programas de la sección de Ingeniería Informática, se propuso el desarrollo de un conjunto de herramientas que trabajen con sistemas administradores de bases de datos relacionales (SABDR). En esa oportunidad se implementaron: navegadores de objetos de bases de datos, replicadores de bases de datos, generadores de datos de prueba, optimizadores de bases de datos, entre otros. Los requerimientos comunes a todos ellos fueron: (i) conectar a diversos SABDR comerciales siendo el mínimo de dos desde una misma sesión de la herramienta, (ii) definir una arquitectura que pueda escalar fácilmente hacia otros SABDRs y (iii) ofrecer algún servicio no existente hasta ese momento en otros productos.

Para que los alumnos puedan desarrollar sus productos, investigaron sobre la metadata de diversos SABDRs y utilizaron el paradigma de orientación a objetos para su modelado y posterior construcción. Los productos desarrollados emplearon los conceptos de herencia y polimorfismo en el diseño de la arquitectura, lo que le dio la flexibilidad y escalabilidad deseada. Las primeras versiones desarrolladas en esa oportunidad presentaban pocas funcionalidades por que solamente disponían de 15 semanas para completar el producto y solamente trabajaron con Microsoft SQL Ser-

ver 7.0 y Oracle 8i, por la disponibilidad de las licencias en la universidad al momento de ejecutarse los proyectos.

Tiempo después de concluir los cursos, algunos alumnos tomaron sus productos y ampliaron los servicios. Los proyectos de construcción de la nueva versión de cada producto fue tomado como Trabajos de Fin de Carrera para la obtención del título profesional de Ingeniero Informático. Los proyectos fueron dirigidos por el Grupo de Investigación y Desarrollo en Ingeniería de Software (GIDIS), a través de una línea de investigación aplicada en el desarrollo de herramientas de productividad en múltiples SABDRs. El grupo ha mejorado los productos anteriores llegando a convertirlos en potenciales productos para el mercado de software y trabajan ahora con Microsoft SQL Server 2000, Oracle 9i por que son los que tienen mejor posicionamiento en el mercado [20] [26] y también con MySQL y PostgreSQL, por su amplio uso en plataformas libres [3] [23] [25] [28] [33].

En GIDIS hemos iniciado un proceso de evaluación de las diversas arquitecturas empleadas en nuestros productos para que sea usado como base para futuros desarrollos. El estudio de la arquitectura se divide en partes, habiéndose optado por estudiar primero la gestión de la metadata de los SABDRs, pues luego de las investigaciones iniciales se entiende ahora que la correcta administración de la metadata mejorará el desempeño de nuestras herramientas.

El objetivo de esta investigación se centra en encontrar un modelo adecuado para el desarrollo de herramientas de productividad que operen sobres múltiples SABDRs, sobre diversas versiones de los SABDRs, que sea escalable a otros SABDRs y que sea sirva de soporte a las herramientas a desarrollar.

Los diversos SABDRs gestionan su metadata de diversas maneras, existiendo cambios inclusive de versión en versión. Si bien es cierto que se puede obtener la metadata de los SABDRs de una manera sencilla, también es cierto que existen diversas maneras de realizar esta labor, haciendo que esta gestión sea una labor complicada. En este sentido el diseño de un modelo con las características mencionadas anteriormente será de mucha ayuda para el desarrollo de aplicaciones sobre esta línea.

Este artículo presenta inicialmente una breve descripción de la metadata de diversos SABDRs que han sido utilizados como referencia, ello permitirá identificar la diversidad de implementaciones de metadata en los SABDRs. Luego se presenta los modelos implementados en los proyectos que hemos desarrollado dentro del grupo a través de una diagrama de clases que representa mejor la arquitectura. Finalmente, se presentan la comparación de los modelos antes descrito, las conclusiones a las que hemos podido llegar en esta investigación aplicada y los trabajos futuros que desarrollaremos al interior del grupo.

2 La Metadata

En esta sección presentaremos una breve definición de la metadata y a continuación presentamos la forma de recuperación de la metadata para los SABDRs estudiados.

2.1 Definición y servicio básico requerido

La metadata es la información acerca de la estructura de la base de datos [12] [13]. Cada SABDR posee diversos mecanismos que posibilitan su acceso.

La necesidad de manejar diversos SABDRs establece restricciones al momento de diseñar la arquitectura de la herramienta. La figura 1 muestra, esquemáticamente, la necesidad de conexión de la herramienta con diversos SABDRs lo que se traduce en el diseño de un modelo que soporte la recuperación de la metadata.

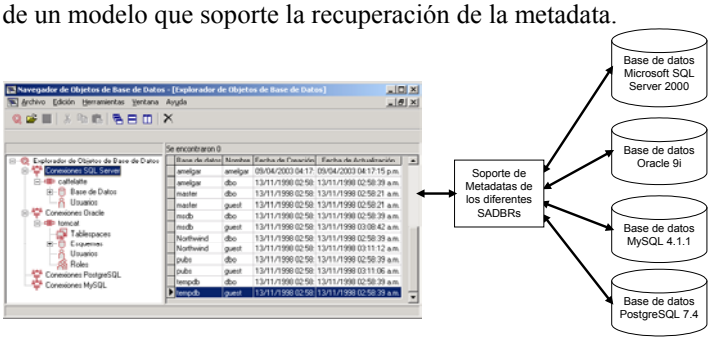

Fig. 1. Esquema de acceso a bases de datos a través de un soporte de metadata. Según el esquema, la aplicación interactúa con el soporte de la metadata, el cual está formado por una seria de componentes especializados que permiten que la aplicación logre independizarse del motor de bases de datos. Las conexiones y la recuperación de datos en la aplicación se realizan de manera transparente, sin importar a que SADBR se está accediendo.

Sabemos que la gran mayoría de productos desarrollados se conectan solamente a motores relacionales debido a que son los más ampliamente utilizados a nivel mundial [31]. También, sabemos que el modelo relacional propuesto por CODD [8] sirve de base a los SABDRs y que emplean el Structure Query Language (SQL) para la creación y administración de las bases de datos; este último es un lenguaje estándar que se divide en dos categorías: lenguaje de definición de datos (DDL) y lenguaje de manipulación de datos (DML) [7] [13].

2.2 Microsoft SQL Server

Existen diversas formas de recuperar la metadata en Microsoft SQL Server. En las versiones 7.0 y 2000, se puede obtener mediante procedimientos almacenados de catálogo (como por ejemplo *sp_tables*, *sp_*stored*_procedures*, *sp_databases*, etc.), mediante funciones de metadata (como por ejemplo *DB_NAME, DB_ID, COL_LENGTH*, etc.), mediante vistas del esquema de información (como por ejem-

plo *TABLES, COLUMNS, VIEWS*, etc.) y mediante el acceso directo a tablas del catálogo de la base de datos (como por ejemplo *SYSOBJECTS, SYSDATABASES, SYSTYPES*, etc.) [21] [22].

2.2 Oracle

Para recuperar la metadata de una base de datos Oracle 8i o 9i se puede utilizar las vistas del diccionario de datos. Por ejemplo la siguiente sentencia *"SELECT DISTINCT * FROM ALL_OBJECTS"*, recupera la lista de objetos que existen en la base de datos. En Oracle 9i existe un paquete denominado *DBMS_METADATA* que permite obtener la metadata de una manera más sencilla que leer el diccionario de datos [27], por ejemplo la siguiente sentencia *"select dbms_metadata.get_ddl('TABLE','DEPT','SCOTT')from dual;"* recupera la sentencia de creación de la tabla *DEPT* del esquema *SCOTT*.

2.3 MySQL 4.1.1

Existen algunas funciones e instrucciones que permiten recuperar la metadata de un manejador de bases de datos MySQL [24]. A continuación mostramos algunos ejemplos:

- Obtención del nombre de la base de datos: *"SELECT DATABASE();"*
- Obtención del listado de tablas de la base de datos: *"SHOW TABLES;"*
- Obtención de la sentencia de creación una tabla: *"SHOW CREATE TABLE RH;"*

2.4 PostgreSQL 7.4

Cada base de datos PostgreSQL posee un esquema denominado pg_catalog, este esquema contiene todas las tablas del sistema. Estas tablas del sistema almacenan la metadata del SABD (como por ejemplo *pg_database, pg_trigger, pg_type,* etc.). Además existe el esquema *information_schema* que contiene vistas que permiten acceder a la metadata dejando de lado el esquema *pg_catalog* (como por ejemplo las vistas *TABLES, VIEWS, TRIGGERS,* etc.). A continuación mostramos algunos ejemplos de recuperación de la metadata en este SABD:

- Para obtener el nombre de la BD: *"select current_database();"*
- Para mostrar el listado de las tablas: *"select * from pg_tablas;"*
- Para mostrar el listado de usuarios: *"select * from pg_shadow;"*

3 Acceso a Múltiples Orígenes de Datos

La capacidad de soporte a múltiples SABDR ha sido tema de estudio a lo largo de varios años. Diversas organizaciones, entre ellas empresas importantes como Microsoft, IBM, Sun, Oracle, Informix, entre otras, han investigado este aspecto. Como resultado de estos estudios se tienen: el estándar SQL, el estándar ODBC, los Microsoft ActiveX Data Objects, el *midleware* BDE, la interfaz DatabaseMetadata, entre otros. En este punto se hace un breve resumen de estos trabajos.

3.1 El estándar SQL 99

El estándar SQL 99, es la última versión de los estándares desarrollados para el lenguaje SQL. Este estándar habilita la portabilidad de aplicaciones SQL y es el resultado del esfuerzo de muchas organizaciones como Computer Associates, IBM, Informix, Oracle, Sybase, Microsoft, entre otros [17]. En este estándar cada catálogo contiene un esquema de información denominado `INFORMATION_SCHEMA`, que incluye la descripción de los objetos contenidos en la base de datos [1]. Entre los objetos que se proponen en este estándar se tienen las vistas: `COLUMNS`, `TABLES`, `TRIGGERS`, `VIEWS`, entre otras [2]. Los SABDRs intentan implementar por lo menos parte del estándar SQL, por ejemplo MySQL soporta la cláusula `LIMIT` para la sentencias `SELECT` y `DELETE` [14] [32], Microsoft SQL Server 7.0 [22] y MySQL [9] soportan los esquemas de información.

3.2 Open DataBase Connectivity

ODBC (Open DataBase Connectivity) es un estándar desarrollado originalmente por Microsoft, que permite que las aplicaciones se conecten a diferentes orígenes de datos a través de una única interfaz [11] [29]. Este estándar posee algunas funciones que permiten obtener información acerca de la metadata, como por ejemplo: `SQLDescribeCols`, que permite conocer el nombre, tipo de dato, presición, escala y nulidad de las columnas de una tabla [29] [30]. La ventaja de usar este estándar es lograr una independencia de los SABDRs [30].

3.3 Microsoft ActiveX Data Objects (ADO)

Es la estrategia de Microsoft para proveer acceso a múltiples orígenes de datos relacionales y no relacionales, esto lo logra a través de OLE DB (Object Link Embedding for DataBase). OLE DB es una colección de interfaces COM (Component Object Model) que encapsulan los servicios de los administradores de orígenes de datos, y ha sido desarrollado siguiendo la misma filosofía de ODBC en el sentido de ser una especificación abierta. La diferencia esta radica en que mientras ODBC fue creado para dar soporte de acceso a bases de datos relacionales, OLE DB ha sido diseñado para orígenes de datos relacionales y no relacionales [4] [6] [16].

3.4 Interfaz DataBaseMetadata

En Java existe una interfaz denominada DatabaseMetaData, que permite obtener información acerca de la metadata de un SABDR, cada proveedor de SABDRs debe implementar esta interfaz de manera específica de acuerdo a la forma en que almacena su metadata [10].

3.5 Borland Database Engine

El DBE es un midleware que permite el acceso a distintos orígenes de datos a través de aplicaciones desarrolladas en entornos Borland. Este estándar implementa algunas funciones que obtienen la metadata de las bases de datos como por ejemplo `DbiOpenDatabaseList` que permite obtener una lista de bases de datos disponibles para la conexión realizada [5].

4 Modelos Implementados

Se han desarrollado principalmente 2 tipos de modelos, uno que modela los objetos de SABDRs directamente y otro que modela la metadata de los SABDRs en donde hemos desarrollado tres variantes. A continuación se describen los modelos implementados.

4.1 Modelado a través de los objetos de los SABDRs

En este modelo se ha realizado un mapeo directo de los distintos objetos de los SABDR en el diagrama de clases tal como lo presenta la figura 2. La arquitectura soporta múltiples SABDRs y cada clase representa un objeto de la base de datos. El modelado es rápido y tiende a ser más detallado en lo que respecta a cada tipo de objetos

A este modelo lo hemos denominado Mod-Obj y fue utilizado para la implementación de una herramienta de optimización del rendimiento de SABDRs [15].

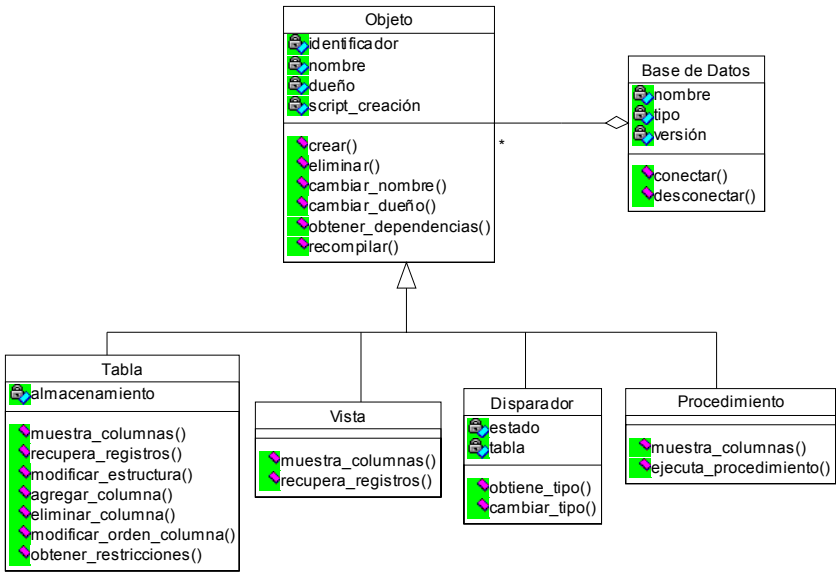

Fig. 2. Mod-obj, modelado a través de los objetos de los SABDRs. La figura presenta un modelo en donde se mapean los tipos de objetos de bases de datos con los que se desea interactuar en la aplicación. En este modelo existe una relación directamente proporcional de la cantidad de clases que posee el modelo y la cantidad de tipos de objetos que se deseen manejar en la aplicación, mientras más tipos de objetos se manejen, más clases contendrá el modelo. La implementación de los métodos es compleja ya que estos soportan múltiples SABDRs. Este modelo fue implementado en un optimizador de bases de datos relacionales.

4.2 Modelado a través de la metadata de un SABDR sin soporte a versiones

El modelado se hace sobre la metadata como una entidad abstracta tal como es presentado en la figura 3. Las diversas implementaciones de SABDR representan clases derivadas de la clase metadata.

A este modelo lo hemos denominado Mod-Met-SinVer y fue utilizado para la implementación de un navegador de objetos con soporte para múltiples bases de datos relacionales[19].

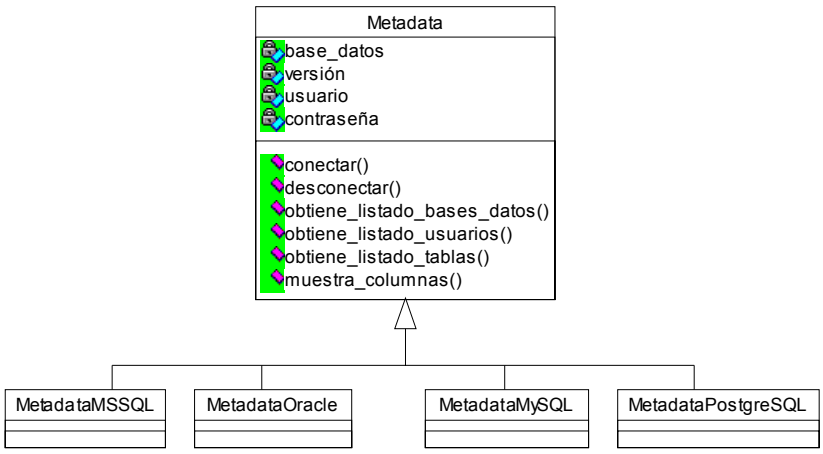

Fig. 3. Mod-Met-SinVer, modelado a través de la metadata de un SABDR sin soporte a versiones. La figura presenta un modelo en donde existe una clase abstracta de la metadata, cada manejador extenderá el comportamiento de esta clase abstracta. En este modelo existe una relación directa entre el número de manejadores que se desea soportar en la aplicación y la cantidad de clases existentes en el modelo, si se quiere soportar más SABDRs, se debe agregar al modelo una clase por cada SABDR adicionado, esta clase deberá heredar de la clase abstracta. La implementación de los métodos no es muy compleja, pero se debe tener en cuenta las diferencias entre versiones del mismo motor. Este modelo fue implementado en un navegador de objetos bases de datos.

4.3 Modelado a través de la metadata de un SABDR con soporte a versiones

Este modelo es muy parecido al anterior (Mod-Met-SinVer), conceptualmente. La figura 4 presenta la arquitectura que incorpora una capa más de clases abstractas (intermedia) para poder manejar las diferentes versiones que puedan existir de los SABDRs para una herramienta trabaje con múltiples SABDRs. Las clases abstractas del segundo nivel (derivadas para cada producto) permitirán manejar los métodos comunes a que hay entre versión y versión de un mismo SABDRs y las clases específicas a cada versión manejarán solamente los métodos propios de cada versión.

A este modelo lo hemos denominado Mod-Met-ConVer y está siendo utilizado actualmente para la implementación de un navegador de objetos con soporte para múltiples bases de datos relacionales todavía en desarrollo.

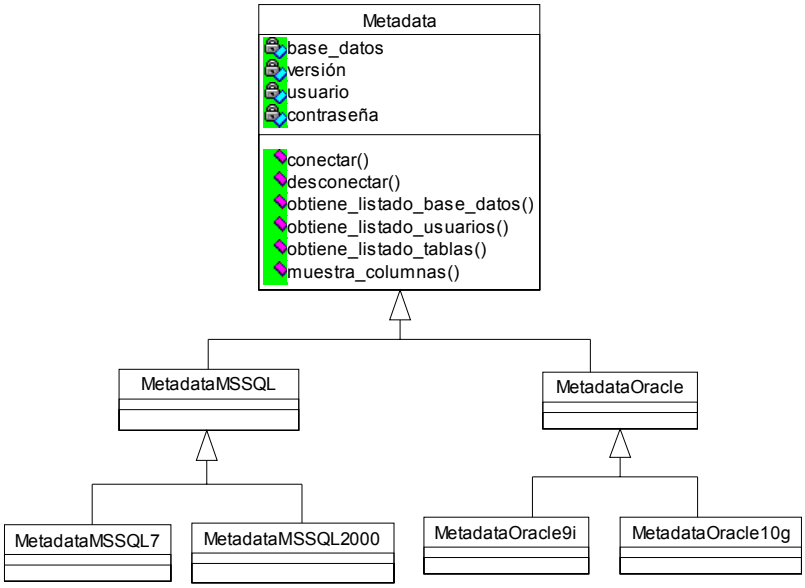

Fig. 4. Mod-Met-ConVer, modelado a través de la metadata de un SABDR con soporte a versiones. La figura presenta un modelo similar al de la figura anterior, pero con un nivel adicional de abstracción para la metadata de cada SABDR. Cada versión extenderá la implementación de los métodos de la clase padre si es que lo requiere. La implementación de los métodos no es compleja y el soporte de nuevas versiones de un SABDR existente se logra en un tiempo bastante rápido debido a que entre versión y versión los cambios de los mecanismos de recuperación de la metadata suelen ser los mismos o cambian muy poco. Este modelo está siendo usado en un navegador de objetos bases de datos.

4.4 Modelado a través de la metadata de un SABDR basado en el manejo de la metadata en Java

En este modelo la arquitectura se ha basado en la propuesta de Java a través de su interfaz DataBaseMetadata, muchos trabajos se han desarrollado utilizando este modelo. En la figura 5 se presenta un modelo simplificado usando una interfaz como la desarrollada por Sun.

A este modelo lo hemos denominado Mod-MetaJava y se han utilizado en dos proyectos del grupo [18][34].

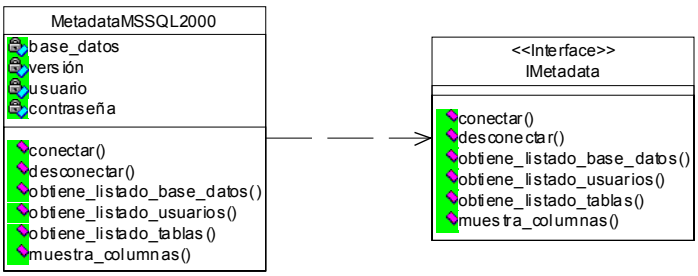

Fig. 5. Mod-MetaJava, Modelado a través de la metadata de un SABDR basado en el manejo de la metadata en Java. En la figura se presenta un modelo en donde se ha modelado la metadata como una interfase, la aplicación trabaja con esta interfase y la implementación es realizada en clases para cada versión de cada manejador. Este mecanismo esta basado en un desarrollo de Sun y la ventaja es que existen drivers que implementan esta interfase. La desventaja es que estas implementaciones, en muchos casos, están ocultas para el desarrollador lo cual no es deseable.

5 Comparación de los modelos presentados

Los modelos que hemos utilizado tienen ventajas y desventajas a ser consideradas para la selección de una arquitectura que cumpla con el requisito de escalabilidad y flexibilidad para una arquitectura que trabajará con múltiples RDBMS desde la misma sesión de trabajo. En la siguiente tabla se presenta una comparación de algunas características de los modelos empleados.

En el Mod-Obj, se ve que existe una dependencia entre el número de clases a manejar y la cantidad de tipos de objetos a controlar, siendo el mantenimiento de este modelo un poco engorroso, pero no depende del número de SABDRs a soportar como si ocurre con los demás.

Todos los modelos comparados soportan diferentes versiones de un mismo SABDR. Sin embargo, existen dos enfoques para la implementación de los métodos para cada versión: (i) hacer una diferenciación por clase o (ii) hacer una diferenciación en los métodos. En el segundo caso, el mantenimiento del método se volvería cada vez más complejo en la medida que aumenten las versiones de los manejadores a soportar.

Por lo general el soporte de metadata no cambia drásticamente entre versión y versión, existiendo compatibilidad con las versiones anteriores. Es muy probable que un método diseñado para soportar la recuperación de una lista de tablas en una base de datos Microsoft SQL Server 7.0 sea igual al método diseñado para soportar la recuperación de una lista de tablas en una base de datos Microsoft SQL Server 2000.

La ventaja de usar la interfaz DataBaseMetadata de Java es que la implementación de dicha interfaz ya ha sido realizada por los drivers de cada SABDR, pero esto trae consigo depender del puente JDBC y del lenguaje Java.

Tabla 1. Comparación de los modelos presentados.

Aspecto \ Modelo	**Mod-Obj**	**Mod-Met-SinVer**	**Mod-Met-ConVer**	**Mod MetaJava**
La cantidad de clases a implementar, depende del número de tipos de objetos de bases de datos a soportar.	SI	NO	NO	NO
La cantidad de clases a implementar, depende del número de SABDRs a soportar.	NO	SI	SI	SI
La cantidad de métodos a implementar por clase es pequeña.	SI	NO	SI	SI
Soporte de distintas versiones de SABDRs.	SI	SI	SI	SI
El soporte de las versiones de los SABDRs se realiza en las clases.	NO	NO	SI	SI
El soporte de las versiones de los SABDRs se realiza en los métodos.	SI	SI	NO	NO
Existe escalabilidad hacia otros SABDRs.	SI	SI	SI	SI
Independencia de las forma de conexión a los SABDRs.	SI	SI	SI	NO
Independencia de la herramienta de programación.	SI	SI	SI	NO

6 Conclusiones y Trabajos Futuros

La arquitectura que modela la metadata y que da soporte a las versiones de diversos SABDRs (Mod-Met-ConVer**)** y la basada en Java son arquitecturas que cumplen con la escalabilidad y flexibilidad mejor que los otros modelos empleados. Sin embargo, nuestros proyectos iniciales no fueron implementados en Java, por lo que si introducimos la independencia del lenguaje como un aspecto relevante, entonces la mejor alternativa es la primera.

A partir de las implementaciones existentes, podemos hacer evaluaciones comparativas de tiempos de implementación de extensiones de las herramientas hacia otros SABDR. También consideramos conveniente evaluar facilidad de mantenimiento que provee que cada arquitectura.

Hemos revisado cuatro modelos, pero no son los únicos y cada uno puede ser útil de acuerdo a las necesidades específicas de cada herramienta a construirse.

Si bien es cierto que en los componentes (*drivers*) JDBC de Java implementan la interfaz DataBaseMetadata, usar este modelo supondría depender siempre de la tecnología Java, lo cual crea una dependencia no deseable en algunos casos, además la implementación de los métodos resulta una caja negra para los desarrolladores, y estos no se podrían modificar ni optimizar.

Hablamos en esta publicación de soporte para múltiples bases de datos relacionales, pues como [31] dice son las más usadas a nivel mundial. Pero existen en el mercado ya algunos sistemas administradores de bases de datos orientadas a objetos a donde se podría extender este estudio.

Agradecimientos

Los autores desean agradecer de manera especial a los ingenieros José Pow Sang y Carla Basurto, del Grupo de Investigación y Desarrollo de Ingeniería de Software quienes realizaron valiosos aportes en el desarrollo del presente artículo.

Este trabajo fue parcialmente financiado por la Dirección Académica de Investigación de la Pontificia Universidad Católica del Perú, a quienes estamos eternamente agradecidos.

Todos los miembros del equipo de desarrollo desean agradecer a la sección de Ingeniería Informática que en todo momento brindó su apoyo cediendo computadoras personales y servidores al equipo de desarrollo.

Referencias

1 ANSI/ISO/IEC International Standard (IS). Database Language SQL – Part 1: SQL Framework. (Septiembre 1999).

2 ANSI/ISO/IEC International Standard (IS). Database Language SQL – Part 2: SQL Foundation. (Septiembre 1999).

3 Axmark David. MySQL The Commercial OpenSource Database. Egov Open Source Conference.(Marzo2003).ftp.iasi.roedu.net/mirrors/ftp.mysql.com/Presentations/presentation-fosdem2003-20030208.ppt, disponible el 14 de Mayo del 2004.

4 Blakeley Jose A. And Pizzo Michael J. Microsoft Universal Data Access Platform. SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data. (June 2-4, 1998, Seattle, Washington, USA).

5 Borland International. 32-Bit Borland Database Engine 3.0 for Windows 95 and Windows NT: Proven power for high-performance database applications. Borland International, Inc. (Mayo 1996). http://www.tietovayla.fi/BORLAND/BDE/engine3x.html, disponible el 14 de Mayo del 2004.

6 Burger Kevin R. Using ActiveX Data Objects to Publish an Excel Grade Book on the World Wide Web. The Consortium for Computing in Small Colleges. (Marzo 2001).

7 Casanova Marco A. y Bernstein Philip A.. A Formal System for Reasoning about Programs Accessing a Relational Database. ACM Transactions on Programming Languages and Systems. Vol. 2, No. 3, (Julio de 1980), pp 386-414.
8 Codd E. F.. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM. Vol. 13, No. 6, (Junio de 1970) , pp. 377-387.
9 Gulutzan Peter. MySQL and the SQL Standards: A Detailed Analysis and Progress Report. MySQL AB. MySQL Users Conference. (Abril del 10 - 12 del 2003). San José, California - USA. http://www.mysql.com/events/uc2003/, disponible el 14 de Mayo del 2004.
10 JavaTM 2 Platform Std. Ed. v1.5.0: Interface DatabaseMetaData. http://java.sun.com/j2se/1.3/docs/api/java/sql/DatabaseMetaData.html, disponible el 14 de Mayo del 2004.
11 Jebb Richard. Using the ODBC Function Library. (Mayo 2001). http://www.pcfltd.co.uk/howtos/odbc_lib.pdf, disponible el 14 de Mayo del 2004.
12 Jeffery Keith G. Metadata The Future of Information Systems. http://www.wmo.ch/web/www/WDM/ET-IDM/Doc-2-3.html, disponible el 14 de Mayo del 2004.
13 Korth Henry F.and Silberschatz Abraham. Fundamentos de Bases de Datos. Mc Graw-Hill. 1993. Madrid.
14 Lammel Ralf. SQL and MySQL. Werkcollege InleidingGegevensverwerking. (Febrero 2004).
15 Laura S. Yasin y Echegaray A. Jessica. Administrador de Bases de Datos Enfocado a la Optimización para Múltiples Plataformas de Bases de Datos. Tesis para optar por el título de Ingeniero Informático. Pontificia Universidad Católica del Perú. 2001.
16 Lazar David. Microsoft Strategy for Universal Data Access. MSDN Library. (Mayo 1998). http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnuda/html/msdn_udastrat.asp, disponible el 14 de Mayo del 2004.
17 Mattos Nelson M. SQL 99, SQL/MM, and SQLJ: An Overview of the SQL Standards. IBM Database Common Technology. (1999).
18 Meléndez Ll. Karin. Sistema Generador de Datos Basados en Reglas. Tesis para optar por el título de Ingeniero Informático. Pontificia Universidad Católica del Perú. 2003.
19 Melgar S. Andrés. Navegador de Objetos de Múltiples Bases de Datos Relacionales. Tesis para optar por el título de Ingeniero Informático. Pontificia Universidad Católica del Perú. 2002.
20 Microsoft Corporation. SQL Server Customers Get Worldwide Recognition for Enterprise-Class Scalability. http://www.microsoft.com/sql/evaluation/compare/wintercorp.asp, disponible el 14 de Mayo del 2004.
21 Microsoft SQL Server 2000 – Books Online. Microsoft Corporation. 1999.
22 Microsoft SQL Server 7.0 – Books Online. Microsoft Corporation. 1998.
23 Miller Mike. A Contact Database using MySQL and PHP. 1998. http://www.webtechniques.com/archives/1998/01/note/, disponible el 14 de Mayo del 2004.
24 MySQL AB. MySQL Reference Manual. MySQL AB. (2003). http://dev.mysql.com/doc/, disponible el 14 de Mayo del 2004.
25 MySQL Web Page. http://www.mysql.com/, disponible el 14 de Mayo del 2004.
26 Oracle Corporation. The World's Largest Commercial Database Runs Oracle http://www.oracle.com/solutions/performance_scalability/index.html?winter2003.html, disponible el 14 de Mayo del 2004.
27 Oracle Corporation. Oracle Notes. http://www.oraclenotes.com/dba/list_package_detail.cfm?PKG_ID=603
28 PostgreSQL Web Page. http://www.postgresql.com/, disponible el 14 de Mayo del 2004.
29 Postgress Software Corporation. Postgress ODBC Driver Guide. (Mayo 2001). http://www.progress.com/progress/products/documentation/docs/sql92/odr/odr.pdf, disponible el 14 de Mayo del 2004.

30 University of Waterloo. CLI and ODBC. http://db.uwaterloo.ca/~david/cs338/lect-ODBC.pdf, disponible el 14 de Mayo del 2004.
31 Vela Belén y Marcos Esperanza. Bases de Datos Orientadas a Objetos. Universidad Rey Juan Carlos de Madrid, Escuela Superior de Ciencias Experimentales y Tecnología. Área de Lenguajes y Sistemas Informáticos. 2000.
32 Versavel Dirk. The Open Source Database for Mission-Critical, Heavy-Load Applications. MySQL White Paper. (Enero 2003). http://www.encima.be/docs/mysql.pdf, disponible el 14 de Mayo del 2004.
33 Wolff David. Mysql, Postgresql, And Php: Open Source Technologies For A Database Management. The Journal of Computing in Small Colleges. Volume 17 , Issue 2 (Diciembre 2001), Pages: 91 - 92
34 Zapata D. Claudia. Sistema de Apoyo a la Gestión de Replicación de Datos de Diferentes Manejadores de Bases de Datos Relacionales. Tesis para optar por el título de Ingeniero Informático. Pontificia Universidad Católica del Perú. 2003.