Estudio Comparativo y Análisis de Rendimiento de los Lenguajes de Manipulación de Datos en Bases de Datos Orientadas a Objetos y Bases de Datos Objeto-Relacionales

[1] FALAPPA, Marcelo Alejandro, [1] COBO, María Laura, [1] MARTÍNEZ, Diego César, [2] BENEDETTO Marcelo Gabriel, [2] CARABIO, Ana Lía Ramona, [2] ALVEZ Carlos Eduardo,, [2] FERNÁNDEZ, Miguel Antonio, [2] ETCHART Graciela Raquel, [2] CABRERA, Sergio Alberto

[1] Departamento de Ciencias e Ingeniería de la Computación - Universidad Nacional del Sur
Avenida Alem 1253 - Bahía Blanca (B8000CPB) - Tel.: +54(0291)4595135
{mfalappa, mlc, dcm}@cs.uns.edu.ar

[2] Facultad de Ciencias de la Administración - Universidad Nacional de Entre Ríos
Monseñor Tavella 1424 – Concordia, Entre Ríos (3200) - Tel.: +54(0345)4231433
{marben, anacar, caralv, migfer, graetc, sercab}@fcad.uner.edu.ar

Resumen

El modelo relacional puro tiene una restricción muy fuerte: los tipos de datos deben ser atómicos para satisfacer la primera forma normal, así como las formas más altas. Para tratamiento de objetos complejos existen dos opciones: utilizar lenguajes de manipulación de datos que respeten el paradigma orientado a objetos y manejen objetos persistentes, o bien, utilizar bases de datos objeto-relacionales, las cuales proveen constructores para la manipulación de datos complejos respetando la robustez del modelo relacional.

Por su parte, los lenguajes de programación (LP) han tenido un desarrollo creciente, se han adaptado a las necesidades de diferentes dominios de aplicación, liberando versiones periódicamente y adaptados a estándares. Este crecimiento, generó dentro de los LP, adaptaciones a pautas de diseño e implementación de programas, permitiendo su utilización en el desarrollo de aplicaciones para bases de datos (BD), entre otras. Estas adaptaciones producidas en LP no específicos para BD, poseen la restricción impuesta por el modelo de BD y el modelo del LP, lo que no se produce en lenguajes específicos para BD. Además, los LP poseen características que procuran la producción de software con cualidades como confiabilidad, mantenibilidad y eficiencia, entre otras.

Este proyecto plantea la realización de un estudio comparativo de los lenguajes de manipulación de datos en sistemas de BD, analizando el rendimiento de lenguajes orientados a objetos en función de las cualidades del software que se produce con ellos, y comparándolos con los lenguajes de manipulación de datos en BD objeto-relacionales.

Palabras Clave: Bases de Datos, Lenguajes de Manipulación de Datos, Lenguajes de Programación, Lenguajes Orientados a Objetos, Cualidades del Software, Análisis de Rendimiento.

Contexto

Este proyecto se enmarca dentro del Convenio Específico de Colaboración entre Programas de Investigación y Postgrado, celebrado en el año 2008 entre la Facultad de Ciencias de la Administración de la Universidad Nacional de Entre Ríos y el Departamento de Ciencias e Ingenierías de la Computación de la Universidad Nacional del Sur.

Además, se suscribió un Acuerdo de Colaboración Académico-Científico entre la Facultad de Ciencias de la Administración de la U.N.E.R. y el Instituto de Ciencias e Ingeniería de la Computación (ICIC) del Departamento de Ciencias e Ingeniería de la Computación de la U.N.S. para el desarrollo del presente proyecto de investigación.

Uno de los principales objetivos de este proyecto es la formación de recursos humanos para investigación en la Facultad de Ciencias de la Administración de la UNER, especializados en la línea prioritaria de investigación denominada "*Ingeniería de Software y Lenguajes de Programación*" establecida por Res. 25/11 del C.D. Al ser también ésta una de las principales líneas de investigación del Departamento de Ciencias e Ingeniería de la Computación de la UNS, se justifica la creación de un equipo de investigación inter-universidades que sea contenedor del desarrollo de investigadores de la U.N.E.R. en el área.

Introducción

Las bases de datos son masivamente utilizadas en las aplicaciones de hoy en día. Detrás de la mayoría de los sistemas informáticos, existe una base de datos así como un sistema de manejo de la misma, que permite el acceso a los datos, brindando seguridad al usuario, recuperación ante fallos, posibilidad de acceso concurrente, y mecanismos de control de concurrencia que garantizan la atomicidad de las transacciones, y la correcta modificación de los datos. Dentro de los diferentes modelos, sin lugar a dudas, el modelo relacional es el modelo que más éxito ha tenido y que más ha perdurado en el tiempo. Probablemente la razón de esto sea debido a la solidez formal del modelo en sí, como así también a la eficiencia de los sistemas de manejo de bases de datos que lo utilizan. Sin embargo, el modelo relacional puro tiene una fuerte limitación derivada de las restricciones impuestas por las formas normales: los campos de las relaciones (tablas) deben ser atómicos. Actualmente, existen dos formas de extender dichos sistemas: mediante bases de datos objeto-relacionales, o mediante la interacción con lenguajes de programación orientados a objetos que permitan la manipulación de objetos persistentes.

En la actualidad, los lenguajes de programación poseen interfaces de desarrollo de aplicaciones y acceso a sistemas de base de datos. La mayoría de las aplicaciones de hoy, poseen interfaces amigables para el usuario que permiten interactuar con la base de datos. Existen varias técnicas para incluir estas interacciones en un lenguaje de programación: una consiste en escribir programas de aplicación en lenguajes de alto nivel, que integran instrucciones que permiten soportar la funcionalidad de una base de datos; o bien utilizar una técnica más dinámica pero más compleja, como las API's (API: *Application Programming Interface*), que son librerías de funciones y procedimientos que pueden ser utilizados por otro software como una capa de abstracción; en el contexto de bases de datos. Estas dos metodologías son las más comunes, pero presentan problemas derivados de las diferencias entre el modelo de la base de datos y el modelo del lenguaje de programación. Este problema disminuye con el uso de una tercera técnica, que es la utilización de un lenguaje de programación específico para construcción de aplicaciones que tienen mucha interacción con la base de datos, brindando así, compatibilidad entre ambos modelos.

Las bases de datos y los lenguajes de programación han tenido un desarrollo creciente y, en la mayoría de los casos, se han ido adaptando a las necesidades propias de los diferentes dominios de aplicación. Los lenguajes de programación, en particular, son una herramienta de vital importancia para el desarrollo de muchas aplicaciones de software. Éstos poseen características particulares, tales como *simplicidad*, *legibilidad*, *facilidad de escritura*, facilidades para la *auto-documentación*, herramientas para un adecuado diseño de *interfaces*, que procuran la producción de software con cualidades tales como *confiabilidad*, *mantenibilidad* y *eficiencia*, entre otras.

Desde que el software se utiliza para la resolución de tareas complejas y/o críticas, la confiabilidad ha cobrado mayor importancia. El desarrollo de sistemas debe contemplar que los mismos sean tolerantes a fallas, es decir que continúen brindando soporte al usuario aún en presencia de eventos no frecuentes o indeseables, tales como anomalías de hardware o software.

En referencia a la mantenibilidad, no es económicamente factible descartar el software existente y desarrollar aplicaciones de reemplazo desde el principio, dado que el costo de desarrollo de software se ha incrementado y, por lo tanto, las aplicaciones existentes deben modificarse para satisfacer los nuevos requerimientos. En particular, la mayoría de los sistemas de software que utilizan bases de datos, y en especial, las que usan el modelo relacional, han utilizado, reusado y adaptado las bases de datos iniciales más allá de la evolución del sistema.

Por otra parte, la eficiencia ha sido siempre una cualidad deseable de cualquier software. Este requisito afecta tanto al lenguaje de programación, como a la elección del algoritmo a utilizar. Si bien el costo del hardware continúa descendiendo y, al mismo tiempo, su performance continúa en aumento, la necesidad de una ejecución eficiente persiste porque se utilizan computadoras con aplicaciones cada vez más exigentes.

La calidad del software que se desarrolla está directamente relacionada con la calidad que el lenguaje de programación brinda a través de sus características y atributos asociados. Pero también, la calidad

del software es el resultado de ciertos atributos asociados que no tienen relación directa con lo que el software hace, sino con la organización del programa fuente, la documentación y su comportamiento en ejecución.

Las cualidades deseables del software se pueden dividir en cualidades externas e internas. Las primeras son visibles para los usuarios de las aplicaciones, mientras que las internas son aquellas que afectan a los desarrolladores y se relacionan en gran medida con la estructura del software. En general, los usuarios sólo se preocupan por las cualidades externas, aunque son las cualidades internas las que ayudan a los desarrolladores a lograr dichas cualidades externas. En muchos casos, estas cualidades están estrechamente relacionadas y la distinción entre lo interno y externo no es tan fácil de evidenciar.

Entre las cualidades internas y externas del software más representativas, se encuentran:

Correctitud: un programa es funcionalmente correcto si se comporta de acuerdo a la especificación de las funciones que debe proporcionar.

Confiabilidad: se puede definir la confiabilidad en términos de comportamiento estadístico, como la probabilidad de que el software funcionará como se espera, en un intervalo de tiempo específico.

Robustez: un programa es robusto si se comporta de forma razonable, incluso en circunstancias que no se hayan contemplado en la especificación de requerimientos.

Rendimiento: en ingeniería de software, a menudo, se equipara rendimiento con *eficiencia*. Un sistema de software es eficiente si utiliza los recursos informáticos de manera económica.

Facilidad de uso: un software es fácil de usar, si a los usuarios les resulta fácil operarlo. Esta definición denota la subjetividad de la facilidad de uso. La interface de usuario es un componente importante de dicha cualidad.

Verificabilidad: un software es verificable si sus propiedades pueden ser chequeadas con facilidad. El diseño modular, buenas prácticas de codificación y el uso de un lenguaje adecuado contribuyen a la verificabilidad. Suele ser una cualidad interna, aunque a veces se convierte también en una cualidad externa.

Mantenibilidad: este término se utiliza comúnmente para hacer referencia a las modificaciones que se realizan en un software después de su lanzamiento inicial.

Reusabilidad: es un concepto similar a la capacidad de evolución. En la evolución del software, se modifica un producto para construir una nueva versión del mismo; en la reutilización del software, se producen algunos cambios para construir otro software.

Portabilidad: el software es portable si puede funcionar en diferentes ambientes.

Comprensibilidad: es una cualidad interna del producto que ayuda en el logro de otras cualidades, como la capacidad de evolución y la posibilidad de verificación.

Interoperabilidad: se refiere a la capacidad de un software de coexistir y cooperar con otros sistemas.

Productividad: es una cualidad del proceso de producción de software, que mide la eficiencia del proceso y corresponde a la cualidad de rendimiento aplicada al proceso.

Oportunidad: es una cualidad relacionada con el proceso que se refiere a la capacidad de ofrecer un producto a tiempo. Históricamente, la oportunidad que ha faltado en los procesos de producción de software llevó a la crisis del mismo, lo cual a su vez condujo al nacimiento de la *Ingeniería de Software*. En la actualidad, muchos de los procesos no culminan en un producto oportuno.

Visibilidad: un proceso de desarrollo de software es visible si todos sus pasos y estado actual son claros y están documentados.

De lo expuesto anteriormente, se desprende que en la actualidad la mayoría de las aplicaciones informáticas utilizan bases de datos, existen lenguajes de programación con facilidades para manipular las mismas, se reconocen las cualidades deseables del software, así como también las características que poseen los lenguajes de programación para contribuir al desarrollo de software con dichas cualidades. Uno de los objetivos principales de este proyecto es el estudio y análisis de los diferentes lenguajes de manipulación de datos, y sus efectos en la calidad del software generado por ellos.

Líneas de investigación y desarrollo

Existen lenguajes de programación que ofrecen soporte para acceder y manipular datos

en bases de datos. El presente proyecto de investigación limitará el estudio de lenguajes de programación orientados a objetos, a aquellos que permitan la posibilidad de manipular objetos persistentes, así como lenguajes de manipulación de datos en bases de datos que, además, provean facilidades para el manejo de interfaces gráficas.

Cada uno de estos lenguajes, resulta más o menos adecuado que otro para producir software con ciertas cualidades, en función de las características que ellos presentan. Para poder determinar claramente si el software que producen posee las cualidades de interés para el presente trabajo, es necesario conocer sus características, su funcionalidad, su estructura; y efectuar mediciones, con la finalidad de comparar los resultados.

En base a los lenguajes seleccionados, las cualidades del software y las características elegidas para el estudio, se podrán conocer aquellos lenguajes que experimenten un mejor comportamiento en aplicaciones orientadas a sistemas informáticos con bases de datos, para luego poder compararlos con sistemas que utilicen bases de datos objeto-relacionales.

Resultados y objetivos

El objetivo de este trabajo es realizar un estudio comparativo y análisis de rendimiento de lenguajes orientados a objetos, así como lenguajes de manipulación de datos en bases de datos objeto-relacionales. En cada caso, se buscará medir la calidad del software producido con ellos.

Objetivos Específicos:

-Seleccionar lenguajes de manipulación de datos en bases de datos objeto-relacionales y lenguajes orientados a objetos que manipulen objetos persistentes.

-Realizar un estudio comparativo en base a los lenguajes seleccionados y a las características de los mismos, en función de las cualidades deseables del software que producen.

-Comparar globalmente sistemas desarrollados en lenguajes orientados a objetos que manipulen objetos persistentes con sistemas que utilicen bases de datos objeto-relacionales.

-Analizar los resultados del estudio comparativo efectuado y establecer un diagnóstico del rendimiento de los lenguajes de manipulación de datos estudiados.

Formación de recursos humanos

Se brindará a los integrantes del proyecto, formación en lo que se refiere a técnicas avanzadas de orientación a objetos, desarrollo en lenguajes puros e híbridos dentro del paradigma, así como también en bases de datos orientadas a objetos y objeto-relacionales.

Los integrantes del proyecto, se desempeñan en cátedras relacionadas directamente con el tema central de la investigación, por lo que este trabajo tendrá impacto directo e inmediato en la docencia.

Se procederá a dirigir becarios de investigación, así como también tesinas finales de grado, dirigidos por el director del proyecto de investigación y/o por los docentes-investigadores del mismo. Para estos casos, también se prevé la presentación a convocatorias de becas ante organismos provinciales y nacionales.

Los integrantes participarán de reuniones científicas y técnicas que permitan actualizar los conocimientos en el tema de interés. También se trabajará en la presentación de trabajos en congresos nacionales e internacionales relacionados con el área del proyecto. Estos trabajos servirán para divulgar los conocimientos obtenidos durante el trabajo de investigación.

Uno de los principales objetivos del proyecto es que el personal docente de la UNER dedicado al mismo avance y/o concluya con sus estudios de posgrado, así como también se incorporen becarios realizando investigaciones en temas afines a la temática del proyecto.

El Director del proyecto, Marcelo Alejandro FALAPPA, es Doctor en Ciencias de la Computación egresado de la Universidad Nacional del Sur (UNS) en el año 1999. Actualmente es Profesor Asociado con Dedicación Exclusiva en el Departamento de Ciencias e Ingeniería de la Computación de la UNS a cargo de la materia curricular *Bases de Datos*, es Investigador Adjunto del CONICET especializado en Actualización de Bases de Datos Deductivas y Teoría de Cambio, y tiene Categoría II en el Programa de Incentivos.

Uno de los codirectores del proyecto, Marcelo Gabriel BENEDETTO, es Magíster en Sistemas de Información, se encuentra realizando los cursos del Doctorado en Informática de la Universidad de Murcia

(España) y cursando la Maestría en Desarrollo Local en la Universidad Nacional de General San Martín; posee categoría IV en el Programa de Incentivos.

La codirectora, Licenciada en Sistemas de Información Ana Lía Ramona CARABIO, posee categoría V en el Programa de Incentivos y se encuentra realizando la Maestría en Redes de la U.N.LP., restando la presentación de la Tesis; el integrante Miguel Antonio FERNÁNDEZ es categoría IV en el Programa de Incentivos y se encuentra actualmente cursando la Especialización en Gestión de la Innovación y la Vinculación Tecnológica; el integrante Licenciado en Sistemas Sergio Alberto CABRERA se encuentra realizando la Maestría en Sistemas de Información de la UNER, restando sólo la presentación de la Tesis; la integrante María Laura COBO es Doctora en Ciencias de la Computación responsable de la cátedra *Lenguajes Formales y Autómatas*; el integrante Diego César MARTÍNEZ es Doctor en Ciencias de la Computación, responsable de las cátedras *Tecnologías de Programación* e *Ingeniería de Aplicaciones Web*, categoría III en el Programa de Incentivos; el integrante Carlos Eduardo ALVEZ es Doctor en Ingeniería - Mención Sistemas de Información, egresado de la Universidad Tecnológica Nacional - Facultad Regional Santa Fe en el año 2012, actualmente responsable de las cátedras *Lógica para las ciencias informáticas* y *Base de Datos*, y tiene Categoría II en el Programa de Incentivos. Finalmente, la Licenciada en Sistemas Graciela Raquel ETCHART se encuentra realizando los cursos del Doctorado en Ciencias de la Computación de la UNS y posee categoría V en el Programa de Incentivos.

Referencias

GHEZZI, Carlo; JAZAYERI, Mehdi; MANDRIOLI, Dino. *Fundamentals of Software Engineering*. Second Edition. Prentice Hall Inc., 2002.

GHEZZI, Carlo; JAZAYERI, Mehdi. *Programming Language Concepts.* Third Edition. United States of America, John Wiley & Sons Inc., 1997.

ELMASRI, Ramez; NAVATHE, Shamkant B. *Fundamentals of Database Systems*. United States of America, Addison Wesley, 2006.

KUHN, Thomas S. *The Structure of Scientific Revolutions*. Third Edition. Chicago, University of Chicago Press, 1993.

LOUDEN, Kenneth C. *Lenguajes de Programación: Principios y Práctica*. Segunda Edición. México, Thomson Internacional, 2004.

MEYER, Bertrand. *Object-Oriented Software Construction*. Second Edition. 17th Printing 2011. United States of America, Prentice-Hall, 1997.

PRATT, Terrence W.; ZELKOWITZ, Marvin V. *Programming Languages. Design and Implementation*. Fourth Edition. United States of America, Prentice-Hall, 2001.

PRESMAN, Roger S. *Ingeniería del Software. Un enfoque práctico*. Sexta Edición. McGraw-Hill, 2005.

SEBESTA, Robert. *Concepts of Programming Languages*. Ninth Edition. United States of America, Addison Wesley, 2009.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. *Database System Concepts*. Sixth Edition. Mc. Graw Hill, 2010.

SOMMERVILLE, Ian. *Software Engineering*. 9th Edition. Pearson Educación, 2010.