Herramienta para consultas complejas orientada a usuarios finales

Ing. Pablo A. Vaca – Departamento de Sistemas de Información - Facultad Regional Córdoba - Universidad Tecnológica Nacional. E-mail: vacapablo72@gmail.com
Ing. Maximiliano Abrutsky - Departamento de Sistemas de Información - Facultad Regional Córdoba -Universidad Tecnológica Nacional. E-mail mxtrula@gmail.com

CONTEXTO

El tema que se presenta se encuentra enmarcado por el Proyecto de Investigación PID Promocional de la UTN: Análisis y aplicación de metodologías para la generación de consultas complejas utilizando esquemas OLAP. La presente etapa esta enmarcada dentro de un proyecto que se refiere al estudio de generador y facilitadores de consultas complejas a los usuarios finales. El mismo es el encargado de la extracción de los datos desde las fuentes de para su posterior utilización en el análisis de la información que los mismos contienen.

RESUMEN

La línea de investigación en curso persigue el objetivo de facilitar a los usuarios finales la realización de consultas complejas que respondan a preguntas del negocio. Siendo el principal punto de dificultad la comunicación con la fuente de datos y en este punto en particular la sintaxis a utilizar para extraer la información.

Es así que se plantearon tres alternativas o cursos de avance en la construcción de soluciones, la primera: utilizar el lenguaje nativo de la fuente datos, la segunda: desarrollar un intérprete y compilar que unifique el lenguaje de cara a los usuarios y se encargue de traducir el mismo en el lenguaje especifico de la fuente de datos y por último una alternativa mixta, es decir una parte que sea lenguaje propietario y otra parte que utilice el lenguaje de la fuente de datos.

Con estas tres alternativas planteadas se ha desarrollado una serie de pasos tendientes a decidir por cual de nos inclinaremos, colocando en la balanza las ventajas y desventajas de ellas.

Se presenta en este trabajo los avances alcanzados y las líneas de acción propuestas en el marco del proyecto de investigación en curso.

Palabras clave: *Intérpretes – Código Fuente – Lenguajes de Programación – Datamart – OLAP – Extractores de datos*

1. INTRODUCCION

Para la realización del módulo que se encargue de extraer los datos, los cuales se pueden encontrar en una base de datos relacional, una archivo de texto, un archivo dbf o del tio de x-base, es necesario que todo programa de análisis de información cuente con extractores de información, los cuales no son otra cosa que programas que se comunican a dichas fuentes, y mediante la ejecución de una serie de reglas y sintaxis extraen los datos.

Ahora bien, estas reglas que usan estos programas pueden ser las reglas propias de las fuentes de datos, o sino pueden ser reglas propias que disponga la herramienta en cuestión, y dentro de esta última alternativa disponemos de dos caminos a seguir, uno de los cuales es desarrollar totalmente un lenguaje que se encargue de todos los aspectos de la extracción o en su lugar disponer de un lenguaje que en algunos puntos utilice parte del lenguaje de la fuente de datos.

A continuación se plantean ventajas y desventajas de cada una de estas opciones, las cuales son enumeradas a los fines de justificar luego la elección que tuvo el equipo de investigación sobre cual camino utilizar.

Fuentes consultadas: ver referencia bibliográfica Nro 13.

a. Utilización de lenguajes propietarios

Se entiende por lenguajes propietarios a los que están incorporados como parte de las herramientas que los usan, como por ejemplo el lenguaje PL/SQL que es parte de las bases de datos Oracle y el cual es usado en la elaboración de procedimientos almacenadas o disparadores.

A continuación se enumeran ventajas y desventajas observadas si utilizamos este tipo de lenguajes para la extracción de la información.

Fuentes utilizadas: ver referencias bibliográficas Nro 10 – 11 – 12.

Ventajas.

- No se requiere de una traducción antes de ejecutar la extracción en la base de datos, esto redunda en un mejor rendimiento en la extracción de la información.
- Es posible encontrar expertos en cada uno de los lenguajes, lo cual disminuye los tiempos de aprendizaje de la herramienta.
- Trabajar con los tipos de datos nativos, se disminuyen la probabilidad de introducir errores en la traducción de la información.
- Se podrían ejecutar procedimientos almacenados en la base de datos, los cuales ayudarían a transformar la información.
- No existe límite en cuanto a las fuentes de información a acceder, siempre y cuando tengan un modo de conexión (ej. ODBC u OleDb).

Desventajas.

- Si se quiere acceder a variadas fuentes de datos será necesario conocer todos los lenguajes de cada una de ellas. Esto incluye todas las funciones, por ejemplo se dispondrán de distintas funciones que realicen la misma tarea como ser el formateo de fechas o números.
- Para proyectos que involucren varias fuentes de datos se deberá contar con un súper especialista o con muchas personas que conozcan cada una parte.
- Queda en manos de cada fuente de datos la validación de la sintaxis y tendríamos que manejar innumerables códigos de error para facilitarle al usuario la interpretación del mismo.
- Restricción a las fuentes de información para las cuales hayamos desarrollado los traductores correspondientes.
- Si un usuario tuviese que aprender todos los lenguajes antes de usar la herramienta el costo de dicho aprendizaje sería muy alto.

b. <u>Desarrollar un lenguaje propio</u>

Otra alternativa con la que cuentan algunas herramientas en disponer de un lenguaje propio, es decir que cuentan con su propia sintaxis y reglas con las cuales se encargan de extraer los datos desde las fuentes de información, si bien no existen herramientas que apliquen este punto en forma completa es decir que no utilicen para nada lenguaje propio de la fuente de datos, existen lenguajes de programación que para la extracción de datos encapsulan todo lo relacionado con el origen y permiten crear programas que sirvan para varias bases de datos relacionales por ejemplo. Un ejemplo de esto son los Framework de persistencia utilizados en el modelo MVC de desarrollo tales como hibernate.

A continuación se enumeran ventajas y desventajas de desarrollar un módulo co estas características.

Fuentes consultadas: ver referencia bibliográfica Nro 14.

Ventajas.

- Es necesario aprender un solo lenguaje, lo cual disminuye los tiempos de capacitación para los usuarios.
- El conjunto de funciones es único evitando problemas con el uso equivocado y también con las posibles incompatibilidades con los tipos de datos.
- Uniformidad en la escritura del código necesario para extraer la información, facilitando la lectura y comprensión del mismo.

Desventajas.

- Es necesario desarrollar los traductores para la parte del código de extracción que interactue con las fuentes de datos, dado que será imposible sino lograr comunicarse con las mismas.
- Se agrega una nueva etapa de traducción lo cual atenta contra el rendimiento total de la etapa de extracción de datos.
- La complejidad del nuevo lenguaje puede ser muy elevada si no se delimita adecuadamente el alcance del mismo. Esto podría generar una carga de trabajo excesiva, siendo que solo forma una etapa del proyecto total.
- Restricción a las fuentes de información para las cuales hayamos desarrollado las interfaces necesarias, aunque esto tiene un costo menor que desarrollar un traductor.

c. <u>Desarrollar un lenguaje híbrido</u>

La mayoría de las herramientas utilizan una variante de las dos expuestas anteriormente, es decir cuentan con una parte propietaria y otra parte en la cual utilizan el lenguaje propio de la base de datos o fuente que contiene la información, con lo cual se disminuyen los costos de desarrollo del módulo ya que se reutilizan las funcionalidades del lenguaje ya existentes..

A continuación se muestran ventajas y desventajas de esta alternativa.

Fuentes consultadas: ver referencias Nro 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 15.

Ventajas.

- Es posible unificar las funciones que manipulen

los datos extraídos, como ser funciones de formateo de fechas y números, manejos de cadenas de caracteres, de conversión de datos.
- No es necesario desarrollar traductores que se encarguen por ejemplo de transformar una consulta en el lenguaje específico que utiliza cada fuente de datos.
- Al tener una parte propietaria, no es necesario conocer innumerables funciones para trabajar con los datos lo cual permitiría un rápido aprendizaje de la herramienta.
- Se simplifica el desarrollo de la parte propietaria, lo cual permite incorporar aspectos comunes a un menor costo.
- Al tener una parte en lenguaje de la fuente de datos es posible utilizar procedimientos almacenados.

Desventajas.

- Depende de la fuente de datos. Lo cual implica cierto grado de conocimiento de la misma.
- Es necesario tener mucho cuidado en la etapa de diseño, ya que es necesario definir cuidadosamente las interfaces entre la parte propietaria y el lenguaje de datos que son recuperados de la fuente de información

INFORMACIÓN COMPLEMENTARIA.

Fuentes consultadas: ver referencias Nro 1 a 9.

Desarrollo de un lenguaje propio o un híbrido:

El desarrollo de un lenguaje propio o un lenguaje híbrido merece una breve explicación ya que es una parte del proyecto que permite aportar varios conocimientos en distintas áreas, por otra parte el desarrollo de un lenguaje propio o híbrido persigue aportar para el logro del objetivo principal del proyecto, que es facilitarle al usuario la labor de la realización de consultas complejas.
La definición de un lenguaje propio conlleva la necesidad de la creación de un intérprete del mismo, el cual conceptualmente tiene la finalidad de analizar y ejecutar el código escrito en el lenguaje propietario, instrucción por instrucción a medida que sea necesario. En este sentido deberemos contemplar dos alternativas, usar aplicaciones para la creación del intérprete o desarrollar totalmente un intérprete.

Alternativas.

Utilización de aplicaciones para la creación de analizadores léxicos y sintácticos

- Lex & Yacc (lenguaje c)
- TP Lex & Yacc (lenguaje Pascal)
- Flex & Bison (versión de código abierto, con licencia GPL de Lex & Yacc)
- JLex & CUP (lenguaje Java)

Desarrollo íntegro del intérprete

- Utilizando la API de Reflection (Java, .Net, etc).
- En algún lenguaje que manipule tecnologías XML, utilizando las como base (XSLT, XPath, XQuery, Schema, etc).
- Otros lenguajes / Tecnologías.

2. LINEAS DE INVESTIGACION y DESARROLLO

El proyecto de investigación se encuentra dentro de lo que se llama investigación aplicada, más específicamente investigación en Inteligencia de negocios.
Esta enfocada en investigar y evaluar software existente, comparar sus características y encontrar puntos en los cuales son plausibles de mejoras.
También se orienta a generar conocimientos que puedan ser transferidos a la comunidad académica en general.

3. RESULTADOS OBTENIDOS/ESPERADOS

En base a la investigación realizada se observa que conceptualmente el funcionamiento de las aplicaciones antes mencionadas es el mismo, variando solamente el lenguaje de programación del código generado como salida. Es por tanto que se ha tomado la decisión de optar por Lex & Yacc por ser el de mayor popularidad y ser la más evolucionada.
Los patrones son codificados para utilizados como entrada en Lex, quien con la entrada de código fuente genera un analizador léxico en c. El analizador léxico busca coincidencias en el código fuente de entrada respecto a los patrones definidos, convirtiendo dichas cadenas en tokens quienes son una representación numérica de las cadenas de caracteres simplificando el procesamiento.
La gramática es la entrada para Yacc, quien también genera en código c un analizador sintáctico o parseador. Éste aplica las reglas gramaticales que le permiten utilizando los tokens provenientes del analizador léxico crear el árbol de sintaxis, en este se impone una estructura jerárquica de los tokens.
Finalmente el último paso es la generación de código el cual procesa al árbol de sintaxis. Algunos compiladores generan código máquina.
Las reglas de traducción en Lex representan la escritura de todos los patrones, o sea, a tal expresión regular, tal acción. En cambio para Yacc, cada regla consta de una producción de la gramática y la acción semántica asociada. Todo el conjunto de dichas reglas representan a la gramática libre de contexto.

Como resultado de esta etapa de la investigación estamos en condiciones de adoptar una postura sobre la alternativa a utilizar, a la cual hemos escogido por brindar facilidades a los usuarios finales y al equipo de desarrollo de la herramienta, dicha alternativa es desarrollar un híbrido utilizando una aplicación que ayude en el desarrollo de dicho intérprete.
La alternativa elegida permite al equipo de desarrollo adquirir conocimientos y habilidades en distintas áreas de la informática relacionadas con la creación de intérpretes, las cuales pueden ser utilizadas para generar transmisión del conocimiento dentro y fuera de la universidad.

Los objetivos en curso son los siguientes:

- Desarrollar una herramienta que extraiga y contenga la información para que los usuarios finales puedan realizar consultas complejas sobre dicha información sin requerir conocimientos técnicos específicos sobre consultas.
- Desarrollar modelos de extracción de datos.
- Desarrollar un modelo teórico de un generador de consultas complejas.

4. FORMACION DE RECURSOS HUMANOS

Se formarán recursos humanos en los siguientes aspectos:

* Dirección de proyectos, ya que la participación como co-director del proyecto por parte del Ing Pablo A. Vaca lo formará en los aspectos de gestión y dirección de proyectos de investigación.

* Recursos humanos formados en las áreas de Inteligencia de negocios, al contar con la participación de otros egresados y alumnos, estos adquirirán conocimientos avanzados de DataWarehouse, y de otros aspectos de la especialidad Inteligencia de Negocios, aspectos muy requeridos en las organización actuales y futuras.

* Recursos humanos formados en áreas de desarrollo de algoritmos de extracción y compresión de datos. Estas personas obtendrán como resultado de su participación acabados conocimientos de técnicas de extracción de datos, técnicas de cmpresión de datos y técnicas de almacenamiento y manejo en memoria de los datos.

* Recursos humanos formados en la metodología de documentación y modelado UML. La cual es ampliamente usada en todas las organizaciones que desarrollan sistemas informatizados.

5. BIBLIOGRAFIA

1. lex & yacc, Second Edition By John Levine, Tony Mason, Doug Brown - ISBN 10: 1-56592-000-7 | ISBN 13: 9781565920002
2. Compilers: Principles, Techniques, and Tools. by Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Monica S. Lam - ISBN-13: 9780321486813
3. Writing Compilers and Interpreters, by Ronald Mark - ISBN-10: 0471113530 # ISBN-13: 978-0471113539
4. Compiladores - Teoria y Construccion, by by F. Sanchis Llorca - ISBN-10: 8428314691 # ISBN-13: 978-8428314695.
5. Web Oficial: http://dinosaur.compilertools.net/
6. Publicaciones en ePapers: http://epaperpress.com
7. Johnson, Stephen C. [1975]. Yacc: Yet Another Compiler Compiler. Computing Science Technical Report No. 32, Bell Laboratories, Murray hill, New Jersey.
8. Lesk, M. E. and E. Schmidt [1975]. Lex – A Lexical Analyzer Generator. Computing Science Technical Report No. 39, Bell Laboratories, Murray Hill, New Jersey. A COMPACT GUIDE TO LEX & YACC. - by Tom Niemann
9. Wikipedia: Artículos introductorios de lingüística computacional. Listado: http://en.wikipedia.org/wiki/Category:Computational_linguistics.
10. Introducción a los Sistemas de Bases de Datos - C.J. Date. ISBN: 968-444-419-2 - Editorial Pearson.
11. Lantimes SQL 2 - Autor Groff
12. Oracle9i ISBN: 84-481-3653-5 Editorial: Mc Graw Hil.
13. Data WareHousing. Autor Ing. Alejandro Sueldo. Página de Internet.http://alejandrosueldo.com.ar/joomla/index.php?option=com_content&task=view&id=3&Itemid=2 - Consultado en el mes de Abril de 2007.
14. http://www.hibernate.org/. Documentación contenida en el sitio en forma on-line y en formato pdf.
15. http://www.qliktech.com. Documentación del producto QlikView. Herramienta de BI basada en conceptos de Lenguaje de consultas por asociación basado en memoria.