EasyQL - White Paper

CS W4115: Programming Languages and Translators
Professor Stephen A. Edwards
Computer Science Department
Fall 2006 Columbia University

EasyQL Members:

Kangkook jee (kj2181)
Kishan Iyer (ki2147)
Smridh Thapar (st2385)
Sahil Peerbhoy (sap2126)

Table of Contents

Introduction

The EasyQL language is a database manipulation language where the primary aim is to allow users to handle and manipulate data from several database instances simultaneously. The language will simplify for the programmer, tasks such as connecting to a database and performing operations on data residing on different database instances.

Why EasyQL?

We believe that while traditional database-related tasks such as querying a table, inserting/deleting records and so on still make up the majority of the operations on data, there is still a need to be able to handle various types of data within a single environment. This is what EasyQL seeks to address.

What are we going to implement?

We are trying to implement a simple interpretive environment from which a user can connect to multiple database instances at a time and can easily switch from one connection to another. Manipulation and transfer of data between tables, tablespaces and instances will be enabled.

While SQL is meant to operate on data from a table, between tables and at most between tablespaces, EasyQL focuses on a broader scope of data objects such as ones from different connections, different instances and various metadata that define the attributes of a database.

A programmer will also be allowed to input SQL queries if he/she wishes to do so.

You may encounter the situation that you need to access retrieve data from legacy database that resides on decade's old mainframe as a form of hierarchical db or ISAM file that you never heard of or learn how to play with it. You may want to move and store your personal database (might be in MS ACCESS or anything else) into bigger DBMS (such as Oracle or MYSQL) to share it from the web.

The solution for these situations is what EasyQL will provide eventually i.e. interchange and migrate data from multiple database connections under the same interface.

Features

To better explain the features of our language, consider the following 2 tables which reside in different databases on the servers of different companies.

Table schema for EmplA table from HR_tablespace of CompanyA db instance

Id	Number (7)
Name	Varchar2 (25)
Salary	Number (11,2)
Title	Varchar2 (25)

Table schema for EmpB table from HR_tablespace of CompanyB db instance

Id	Number (7)
Name	Varchar2 (25)
Salary	Number (11,2)
Title	Varchar2 (25)
Dependent	Number (2)

- **Supports multiple database connections**

EasyQL allows the programmer to connect to multiple database instances from the command line. An example showing connections being established to 3 different databases is illustrated below.

```
%connect companyA@10.10.10.10:1521  as dbuser/dbpasswd
connected (some connection message..)

companyA> connect companyB@localhost:2000 as guest/guestpasswd
connected (some connection message..)
```

- **Operations such as comparing metadata**

The language enables one to compare the metadata of tables residing on different databases. If there is a match, one can perform operations such as appending rows from a table in one database instance to another. A possible append operation which our language will include is shown below.

```
companyB>print COMP(empB, companyA.HR_tablespace.empA);
2      #return value means that empA table can have records from empB without any loss of data)

companyB>if (COMP(empB , companyA . HR_tablespace.empA)
                tempC.APPEND(companyA.HR_tablespace.emp);
```

- **Procedural and program oriented**

  Enhancements with respect to most present query languages include the availability of procedural and control flow in EasyQL. For instance, in order to attempt a connection to a particular database, one could make use of the following statements. A connection will be attempted every second for 10 seconds.

```
i=0;
while(i<10)
{
   sleep(1);
   connect username/password@ip_address;
   i++;
}
```

- **Embedded SQL statements**

  To provide as much flexibility as possible to the programmer, EasyQL will allow one to input SQL queries directly into the program. This feature will be useful for one who feels that the user's queries can be better expressed in SQL.

```
SQL("Create table Automobile (licence_no CHAR(10), make CHAR(10), PRIMARY KEY(licence_no))")
```

  An equivalent of the SQL Select statement would be:

```
Employee.Condition(salary<50000).Show(name,ssn);
```

  This will be translated to the SQL query:

```
SELECT name, ssn
FROM Employee
WHERE salary<50000;
```

Who is going to use EasyQL?

We feel that a regular database administrator whose working environment is multi-vendor oriented will need to manipulate various data from different databases at the same time will find the language rather useful.
This will also be a very useful tool for those who are working in projects which involve data migration, data integration, data mining etc.

Summary

The EasyQL Language provides some flexibility for a programmer who needs to operate on data residing in different databases. The language, as the name suggests, is easy to use and should find some utility among several users.