SQL Open Heterogeneous Data Access

Berthold Reinwald
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
reinwald@almaden.ibm.com

Hamid Pirahesh
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
pirahesh@almaden.ibm.com

Abstract

We describe the open, extensible architecture of SQL for accessing data stored in external data sources not managed by the SQL engine. In this scenario, SQL engines act as middleware servers providing access to external data using SQL DML statements and joining external data with SQL tables in heterogeneous queries. We describe the state-of-the art in object-relational systems and their companion products, and provide an outlook on future directions.

1 Accessing Heterogeneous Data Sources

In today's IT infrastructures, data sources can be categorized into "traditional", desktop, and Web data sources. Traditional data sources are relational or IMS databases, or ISAM/VSAM files with well-established, highly optimized query and data access modules. Desktop data sources are mail systems, documents created by office applications, or spreadsheets. These data sources are primarily updated by the applications which created and which own the data, but they are accessed through a variety of mechanisms. Web data sources are data sources made available through the Web in HTML format and/or as XML data. Examples are product catalogs, investment portfolios, or yellow pages. From a SQL perspective, accessing and combining data from a variety of data sources and integrating the data with well-established SQL data is challenging. A sample application might be a relational database with a product table, and a mail system containing 1000s of messages from customers. A sample query might be an OLAP marketing region analysis for each product in the product table counting the number of complaint messages in the mail database.

2 SQL-Invoked Functions in SQL Standard

The SQL standard includes user-defined functions (UDFs) implemented externally in any host programming language. A UDF may have input parameters, a returns clause, a language and parameter style option for calling and linkage conventions, and various other execution properties. The returns clause may specify a collection type such as a table type with a row type body. In this paper, we refer to external UDFs returning a table type as external table functions. The following example looks up a mailbox, and returns sender, subject, etc. of the mails in a particular inbox.

```
create function inbox (varchar(30))
returns table (sender varchar(30), subject varchar(40), ...)
language C parameter style SQL
external name 'udfsample!inbox';
```

A table function can be used in SQL whenever a table reference or collection derived table is allowed. The following sample query shows the usage of the above table function in a join with a customer table returning the e-mails ordered by revenue of the customer sending the e-mail.

```
select c.name, c.revenue, i.subject
from customer as c, table (inbox('joe')) as i
where c.email = i.sender
order by c.revenue desc;
```

A table function implementation is invoked repeatedly by the SQL engine to return row by row. Various mechanisms are in place to save state information from one invocation to the next, to distinguish between different call requests such as OPEN, FETCH, and CLOSE, and to indicate the end of the table. As anything can be implemented in a programming language, table functions provide a standard, open architecture for presenting any kind of external data in the form of a table.

3 Object-Relational DBMSs

ORDBMSs provide UDF support as well as extensibility APIs. In this section, we describe three ways for providing access to external data sources: (1) built-in data access modules for standard data sources, (2) user-defined table functions implemented by application developers, and (3) user-defined access modules implemented by system integrators.[1]

Informix provides a *gateway manager* (1) that accesses data in multiple external ODBC databases. The external data-source objects are identified in SQL statements by four-part names or by synonyms that resolve to four-part names: ODBC data source, gateway name, object owner and name.

```
select * from oracle_ds@egm_gw:joe.emp;
```

More ad-hoc data sources are accessed in Informix Universal Server through external UDFs that are defined as

[1] Below, (1), (2), and (3) refer to these data-access alternatives.

SIGMOD '98 Seattle, WA, USA

iterator functions (2). Iterator functions return one or more rows of data and therefore require a cursor to execute. The engine passes a function parameter to the DataBlade iterator function implementation containing information about the next iteration operation, saved state, etc.

```
create function TopK (integer, integer)
returns integer not null with (ITERATOR) ...;
```

Informix Universal Server introduces the *virtual table interface* (VTI) (3) to facilitate the development of primary access methods for external data sources. VTI defines a set of database server functions to insert, delete, update, or scan data in an external table (primary access), or access and manipulate an external index (secondary access). VTI passes various descriptors to access method implementations such as scan, qualification, or table statistics descriptors. A table is created using an access method.

```
create function ora_open (pointer) returning int ...;
create primary access_method ora_table_access
        (am_open=ora_open, am_beginscan=ora_scan, ...);
create table emp (...) using ora_table_access;
```

IBM DB2 supports optimized data access to a variety of well-established data sources such as DB2 MVS, Oracle, Informix, Sybase, etc. through *server mappings* and *nicknames* (1) referring to data stored by external servers.

```
create server mapping from mysrv to node "dbwkst"
        database "test_db" type sybase;
create nickname dept for mysrv.admin.dept;
select * from dept;
```

DB2 Universal Database supports external *table functions* (2) as they are defined in the SQL standard. Additional properties are included such as fenced/unfenced execution, serial/parallel execution in UNI, SMP, and MPP, additional call requests for optimizing multiple table scans and piping data, optimization information such as interested columns in current select clause, etc. Table functions are implemented in C, Java, or Visual Basic through OLE automation. For example, a table function implemented in Visual Basic returns a Microsoft Excel spreadsheet as a table.

```
create function sales ()
returns table (product varchar(10), region char(1), ...)
language ole external name 'db2smpl.sales!list'
```

A DB2 *extender* (3) is a set of UDTs and UDFs. A text extender for instance can index and search text documents stored in files referenced by file handles or URLs, and UDFs implement predicate evaluations. Advanced users can implement their own extenders.

```
select * from mytexttable
where db2tx.contains (db2books,
      '"authorization" in same paragraph as "table"') = 1
```

Oracle supports the definition of *database links* and *synonyms* (1), which allow locating remote data stores and accessing its data. Database links (transparent gateways) encapsulate the communication with external data sources. Oracle 8's distributed database functionality provides optimization and decomposition of heterogeneous queries.

```
create database link db2 using 'mvsdb2';
create synonym emp for emp@db2;
select avg (sal) from emp;
```

For ad-hoc data source access, Oracle 8 allows *table functions* (2) as any arbitrary C, Java or PL/SQL function to return a value of TABLE type, which can be used anywhere a table can be referenced (e.g. in the FROM clause of a query).

```
create type sales_t as object (product varchar(10), ...);
create type salestab_t as table of sales_t;
create function sales () return salestab_t ...;
select * from table(cast(sales() as salestab_t));
```

Additionally, Oracle 8's extensible architecture supports *data cartridges* (3), which can be plugged into the database using extensibility interfaces. Access to different heterogeneous, foreign data sources can then be encapsulated into data cartridges and used the same way as database links. The extensibility interfaces also include APIs for user-defined indexes, cost/selectivities, etc.

Sybase Adaptive Server Anywhere, ASA (1), provides external data access via Java access classes, utilizing all-Java access mechanisms such as JDBC drivers.

```
create server srv class DB2Driver using "jdbc:db2:sample"
create proxy emp location "srv.sample.k55.emp"
select * from emp
```

Adaptive Server/Enterprise (ASE) supports built-in access methods for accessing external data through gateways referred to as DirectConnects. Sybase provides a series of DirectConnect access providers, and ASE's *Component Integration Layer* (3) allows for advanced users to implement and register their own access methods.

```
exec sp_addserver db2, direct_connect
create proxy emp remote table "db2.sample.k55.emp"
```

Microsoft SQL Server 7.0, also known as "Sphinx", supports as part of Microsoft's Universal Data Access a built-in, parameterized table called "OpenRowSet" to run distributed, heterogeneous joins across OLE DB providers.

4 Summary and Future Directions

ORDBMSs support a wide range of options for accessing external data sources. Built-in data access modules provide in-the-box data access to standard data sources such as ODBC. User-defined table functions provide easy-to-use standard means for application developers to loosely connect to any data source. User-defined access modules compatible with the proprietary extensibility architectures of ORDBMSs allow for advanced users to tightly integrate SQL engines with external data sources. Any of these options benefit from better "plumbing" for accessing all kinds of data through OLE DB, JDBC, or XML.

Today, only the user-defined table functions are part of the SQL standard. In the future, (1) SQL extensions for distributed querying capabilities such as registering external data sources are required as synonyms, nicknames, or proxies are proprietary vendor extensions. (2) Standard data access APIs for a wider range of data sources beyond relational data are necessary, especially in the face of the object-relational extensions to SQL and hierarchical, structured and semi-structured data such as XML data. (3) "Cooperative" cost-based query optimization can either be performed through external costing functions, external subplan submissions, or querying physical schema information from external sources, which requires standards for data source interoperability not yet addressed by SQL. (4) Infrastructure for language-independent projection and predicate pushdown is required. This can partly be subsumed in standards for user-defined access modules in the grand scheme.

Acknowledgments

The authors would like to thank Informix, IBM, Oracle, and Sybase for providing input material for this presentation.