Virtuoso Virtual Database

Executive Summary

Time breeds diversity in the contemporary IT organization. As business goals mature, the application landscape changes with upgrades and wholesale replacements of capital line systems. One central fact has become clear – as this process of maturation evolves, underlying data and systems multiply.

An entire market segment has grown around enterprise middleware. The IT DBMS infrastructure business has struggled mightily to overcome data diversity, yet somehow remains proprietary in its essence.

OpenLink Software's Virtuoso Universal Server is one outstanding example of a platform that successfully generalizes the enterprise data model in all of its diversity. As a Virtual Database, Virtuoso connects to multiple databases and creates a unified data model. As a high-performance object-relational database with Web Services functionality, Virtuoso provides the environment for core applications that extend beyond parochial systems. Spanning the gap between Virtual Database and Core Data storage platform, Virtuoso provides an end-to-end SOA and XML suite of transformative services.

As all IT evolution issues boil down to data integration. We will focus on the Virtuoso VDB general architecture, query processing, optimization, and distributed transaction issues.

Virtuoso Virtual Database (VDB)

Virtuoso's VDB engine provides unified and transparent access to relational data residing on any ODBC/JDBC compliant DBMS. The tables from any number of databases may be linked o a Virtuoso schema, creating a unified data model that may be accessed by the full suite of Virtuoso application functions.

Virtuoso VDB in focus

Information integration is a growing challenge; most organizations have critical information spread across a multi-vendor mix of databases. These distributed systems, by design or organized chaos, are often scattered about multiple physical locations.

Consolidation of diverse data platforms is never easy or practical due to the restrictions of our application requirements. These line applications breed their own form of stasis. And, while stability is the watchword of our IT faith, it certainly limits our agility and opportunity to build out innovative new services and entering, for example, the Web Services arena. Ideally, the far seeing IT captain will seek every opportunity to access and unify data as a single model, preserve stability of current systems, and, eventually, build a new bridge to the brave new world of Web Services.

Virtuoso Universal Server serves the above vision by presenting collections of disparate legacy systems as a seamless whole. Virtuoso also takes step forward as a hosting platform, application logic server, and Web Services platform.

Benefits of the VDB

The best middleware is no middleware at all; to optimally redefine a multi-vendor patchwork of databases as a unified model where the middle tier vanishes is the Holy Grail. If in reality we cannot wave a wand and dissolve this middle-component, let the choice be as robust and transparent as possible.
This unified model should be available to any application through a single point of access, speaking a single SQL dialect. Virtuoso's VDB offers a full spectrum of client API's, including ODBC, JDBC, OLE/DB and .net data adapters.
Sans wand, discrepancies and quirks of the underlying database systems are hidden by Virtuoso's VDB, as the client API and SQL dialect are now normalized to a uniform standard. The VDM also automatically optimizes query execution paths, and provides a single point of security administration and access privileges to the underlying universe of corporate data.
A virtual database should also be a powerful database engine in its own right, with the heart of an enterprise lion, capable of internally executing all DBMS operations such as joins, storage of large interim results sets, maintaining local tables, indices etc. The Virtuoso object relational core engine is capable of hosting capital line applications moved from other databases through the VDB, or via a migration strategy using sophisticated replication methods.

In our new age of service-oriented architectures, Virtuoso brings web services capabilities to bear as a vital point of entry to the virtual database and the world of Open XML standards.

Let's see how Virtuoso makes this happen:

Virtuoso Database Related Components

Virtuoso Universal Server's database is comprised of standard client application interfaces, server side functions, server systems interfaces to external transaction monitors, and OpenLink's famed Universal Data Access technology for remote databases. In most cases, Virtuoso off-loads operations to the attached data source; if a feature is not available on the remote database, the function will execute in the Virtuoso VDB engine.

Clients

- ODBC 3.5 for Windows and Unix
- OLE/DB 2.0 for Windows
- JDBC 3.0 for JDK 1.4, JDBC 2.0 is available for JDK's 1.2 and 1.3
- .Net data provider 1.1

Server

- SQL 92/2K compiler.
- Stored procedure language.
- Local DBMS, including tables and index management, row level locking, local transaction logic, logging, schema management and all other traditional DBMS functions.
- Virtual database engine handling query composition and execution for remote databases.
- Database drivers - Virtuoso uses ODBC drivers supplied by either OpenLink or the remote database manufacturer for connecting to data on remote databases.

Distributed Transactions

- Virtuoso is compatible with MS DTC on Windows for enlisting compatible data sources into distributed transactions.
- Virtuoso can be a resource manager under MS DTC, JTA or Tuxedo (XA). This means that an external transaction monitor manages the commit/rollback /recovery of changes affecting data stored in Virtuoso.
- To this effect, Virtuoso offers a resource manager library for linking with Tuxedo and a XA aware JDBC 3.0 client for use by JTA monitors.
- All Virtuoso clients for Windows, including ODBC, OLE/DB and .net providers support MS DTC interfaces.

SQL Capabilities

Virtuoso supports SQL 92 with multiple SQL 2K features.

- Full SQL 92 data types, including wide characters, text and binary large objects, decimal floating point, datetimes with timezone etc.
- SQL 2K user defined types, including inheritance, methods, polymorphism, and persistence of user defined type instances as column values.
- Large selection of aggregates plus possibility for user defined aggregates.
- Basic OLAP extensions, grouping sets, rollup and cube.
- All SQL structures such as derived tables, sub queries, unions and other set operations, all join types etc.

All of these features are available to remote databases attached via Virtuoso's VDB.

Distributed Query Considerations

Virtuoso uses cost based SQL optimization that evaluates join orders and join types, additionally factoring in communication overhead associated with remote data sources.

Let us consider the choices involved in compiling the following query, against the well-known Northwind database:

Figure – SQL Query against Northwind Database

```
select ProductName, p.CategoryID, CategoryName
from Products, V2.Categories c
where
p.CategoryID = p.CategoryID;
```

Now let us say that Products is a local table and V2.Categories is a remote table.

The attached source SQL compiler may consider the following possibilities:

- Loop over categories on the remote and look up the product. ***Suboptimal performance penalty due to the lack of an index on a product's category.***
- Loop over Products and seek the category by doing a lookup on the remote. ***Suboptimal due to round-trip session to remote in order to get the category name for each product***
- Read categories and make a hash table from id to category name. Then loop over products and look up the name from the hash.
  Much better Optimization: ***because there is only one message to the remote for filling the join's temporary hash resulting in a table scan of products locally with very fast lookup for returning category name***.

To effect these optimizations, Virtuoso's SQL compiler counts the rows in each table, available indices, primary keys, and the count of distinct values for each column (and possibly a histogram indicating the distribution of values for a column). Very sophisticated, indeed.

Alternatively, if the query is modified to retrieve one product by adding an extra condition for ProductID = 111, then the compiler will change the join into a loop join, where the outer loop is on the product and the inner on category. In this case of retrieving one category, it is best to retrieve category by id, rather than create a hash of categories. Repetitive retrieve operations benefit substantially from the hash join.

Virtuoso measures server query latency and data transfer rate whenever a remote table is attached to a Virtuoso schema. As a result of these empirical tests, alternative SQL execution plans are composed for performance optimization. A remote table can be dropped and subsequently reattached to a Virtuoso schema, thus bringing all meta information (including access times, index availability, columns etc.) up to date. Stored procedures or queries (depending on the table) will be automatically recompiled for any potential performance enhancement.

If both tables are located within same remote database, a query would be passed through directly. In the event of a 'pass through' query, Virtuoso will evaluate the join order and endeavor to optimize the query using the most advantageous join order. However, final decisions on join order and join type are the province of the remote database.

Remote and local tables are treated identically when gathering SQL optimization statistics.

VDB Adaptation to Remote Databases

SQL dialect variations are a vexing integration challenge in a multi-vendor environment. To ameliorate this curse, Virtuoso exploits ODBC metadata functions in order to determine which operations may be relayed to a particular database. Virtuoso uses highly optimized, data source specific methods for the market leading remote databases.

Having the flexibility to exploit specific 'power features' is crucial when accessing legacy systems that support functions beyond the SQL standard, such as outer joins, sub-queries, and OLAP extensions. Without this 'fine grained' adaptability to specific database features, Virtuoso would be forced to execute a sub-optimal query.

Cursors Functions

Virtuoso provides scrollable cursor functionality for bookmarks and cursor state information. Virtuoso will translate scrollable cursor event operations into discrete forward only operations on the remote database, effectively hiding differences in scrollable cursor support between attached databases.

Legacy Business Logic

Most database systems allow business logic to be expressed as triggers and stored procedures. Virtuoso takes the middle-tier a step further by allowing the attachment and marshaling of stored procedure from VDB attached remote databases. These procedures (and the program logic they encapsulate) are now eligible to be called as a local Virtuoso stored procedures. The power of 'rounding up' SQL stored procedures is made manifest by exposing these procedures as web services, or as know in the vernacular of SOA, executable end-points.[1]

Virtuoso allows any stored procedure to return a result set as a sub-table within a SQL query. Procedure result sets may therefore be joined with tables and other result sets to be sorted, filtered, or otherwise processed. This is ideal for adding non-relational data sources into the virtual data model. One prime example would be a stored procedure for reporting.

Procedure tables can be expressed in any hosted language, thus allowing complex logic to be added to the relational world.

See the Web Services and Run Time Hosting white papers for more information.

Administration

Virtuoso is compact and easy to install and administer. In a virtual database setting, administration consists of the following:

Installation

- Defining remote data sources and attaching tables from them.
- Defining local user accounts, roles and granting permissions to them.

After these steps, which can be performed through a web interface, Virtuoso is operational as a virtual database.

For more advanced use, consider the following options:

- Attaching stored procedures from remote databases and publishing these as web services.
- Developing business logic in Virtuoso's stored procedure language or any of the hosted languages, including Java and .net languages.

- Creating a local database on Virtuoso using regular SQL.
- Replicating and synchronizing data between Virtuoso and other databases, including MS SQL Server, Oracle, DB2, Sybase and Informix. For the aforementioned databases, a 2-way incremental replication mechanism is supported. Any ODBC data sources can be imported/exported without synchronization to and from Virtuoso.
- Hosting web interface logic in ASP. Net, PHP, PERL, Python, JSP or Virtuoso's own VSPX dynamic web page language.
- Publishing data as XML, using mapping schemas to translate relational data to XML.
- Developing XML oriented applications using Virtuoso's XSLT, XQuery and XML indexing capabilities.

A web interface allows access to Virtuosos complete repertoire of web services and replication methods. Dynamic SQL clients may be used for custom development.

See the relevant white papers for more information.

Local Database Engine

Virtuoso has a high performance relational database engine with the following key features:

- Clustered B tree index, data follows the primary key inside the index tree.
- Row level locking with dynamic escalation to page locking.
- All four isolation levels, from dirty read to serializable.
- Full text index for text and XML columns. The index supports score, proximity, boolean connectives and a special mode capturing XML hierarchies in the data. Additionally, non-text data can be stored in the text index for high performance retrieval along with text data.
- On-line incremental backup. No activity needs to be interrupted for backing up. The server keeps track of pages changed since last backup for incremental backup.
- Read ahead for indexes and large objects.
- Multithreaded I/O with background flush of dirty buffers and read ahead activity scheduled on a thread per device basis for best throughput. Striping consecutive pages on different devices is recommended.
- 32 terabytes address space per database.

The VDB employs local engine for ordering by group or hash join temporary spaces.

Distributed Transactions

Virtuoso can operate with external transaction monitors including MS DTC, JTA and Tuxedo.

Native Virtuoso transactions for local and remote database reads/updates are guaranteed full ACID integrity.

Using a transaction monitor with Virtuoso will allow a two-phase commit messages from the transaction monitor. Transactions at the prepare phase are logged in order to enable commit cycle failure recovery.

Clients may enlist a Virtuoso connection into a global transaction using MS DTC or XA transaction monitors,. With MS DTC, a remote database operation undertaken on behalf of Virtuoso is enlisted into the global transaction. A Virtuoso stored procedure may also initiate distributed transactions under MS DTC.

A Virtuoso transaction enlisted into an XA transaction may also enlist other data sources if the middleware supports this.

Virtual Database Security

Virtuoso supports identical role base security for local and remote objects. Roles may inherit from other roles, user accounts may also belong to multiple roles. Accounts and/or roles are legitimate grantees for table, stored procedure or user defined type permissions.

Virtuoso's row level security features provide enhanced security for attached legacy systems accessed via the Virtuoso VDB. User access to restricted data may be limited by a table specific policy. This mechanism frees security administrators and developers from having to define, maintain and reference views for special user classes. See the ***Row Level Security*** white paper for more.

Administrator privileges are required for attaching tables from remote databases into a Virtuoso schema. Administrators must provide login credentials in order to effect remote database attachment. Virtuoso will invoke the administrator's credentials when connecting to a remote database via a user request, while hiding access to the login details. A user's access to remote tables and columns are subject to SQL grants.

It is also possible to implement API functions to provide a different login on a remote database. Most of the time, this is unnecessary, however audit trail logging on remote databases may need users to be logged as distinct accounts.

Conclusion

Virtuoso offers a complete virtual database solution, adaptable to a wide range of environments. Virtuoso preserves existing investments in business logic, stored procedures, and design. As a transparent distributed query engine, Virtuoso normalizes data location and overcomes SQL Dialect limitations of attached systems. Disparate infrastructure becomes directly accessible via a unified API, covering the major standards of ODBC, OLE/DB, JDBC and .net.

Virtuoso encourages incremental deployment, with minimal or no re-engineering of existing processes, and straightforward installation.

Virtuoso also opens a migration path to the world of web services, becoming a gateway between time-tested functions in your existing infrastructure, and forward-looking XML-based services oriented architectures.

Learn More

- **Virtuoso Documentation**
- **Virtuoso Tutorials**

[1] See article Virtuoso Web Services infra.