Microsoft Universal Data Access Platform

José A. Blakeley, Michael J. Pizzo
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
+1 425 936 5477

{joseb,mikep}@microsoft.com

1. ABSTRACT

Microsoft Universal Data Access defines a platform for developing multi-tier enterprise applications that require efficient access to diverse relational or non-relational data sources across intranets or the Internet. Universal Data Access consists of a collection of software components that interact with each other using system-level interfaces defined by OLE DB and providing an application-level data access model called ActiveX Data Objects (ADO). This talk provides an overview of the platform.

1.1 Keywords

Data access and manipulation, OLE DB, ADO, ODBC, Database extensibility, Component databases, OLAP, Spatial

2. INTRODUCTION

Modern enterprise applications require integration and analysis of information stored in a mix of data sources including traditional database management systems, indexed-sequential files, the file system, desktop databases, spreadsheets, project management tools, electronic mail, directory services, multimedia data stores, and spatial data stores among others. To meet this challenge, several database companies are providing solutions based on a traditional database-centric approach, which we refer to as the *universal database*. In this approach, the database vendor extends the database engine and programming interface to support new data types, including text, spatial, video, and audio. The vendor requires customers to move *all* data needed by the application, which can be distributed in diverse sources across the enterprise, into one database system. This process can be expensive, time consuming, wasteful, and difficult to maintain. *Universal Data Access* is an effective alternative that allows applications to efficiently access data where it resides without replication, transformation, or conversion. Open interfaces allow connectivity among all data sources. Independent services provide for distributed queries, caching, update, data remoting, resource pooling, distributed transactions, and content indexing among sources. The Universal Data Access approach can leverage the universal database approach for data that makes sense to store in a common database, but allows data to interoperate with data outside the database. Universal Data Access encompasses system and application level APIs along with software components and services that implement these interfaces. These components may play one of three roles:

Data providers are components that contain and expose data such as SQL databases, indexed-sequential files, and spreadsheets. Providers expose their information uniformly using a common abstraction, called the *rowset* [1].

Services are components that process and transform data, or provide a common service. For example a distributed query processor filters, sorts, and combines data from multiple OLE DB sources. A distributed transaction coordinator is a service that coordinates transactions among multiple data sources.

Consumers are components that use data. Examples include services such as a query processor; high-level data access models such as ADO, business applications written in languages like Visual Basic, C++, or Java as well as development tools.

3. DATA PROVIDERS

A growing number of providers deliver access to relational data stores as well as documents stored in the file system via a content search engine, multidimensional data (OLAP) [2], email, and spatial data [3].

Relational. To date, there are OLE DB providers for most relational databases including Oracle, Microsoft SQL Server, IBM DB2, Sybase, Informix, and CA-Ingres. Access to all ODBC sources is available through an OLE DB provider for ODBC. In addition, there are providers for legacy sources such as VSAM, AS-400 files, and IMS/DB.

OLAP. OLE DB for OLAP defines how typical OLE DB consumers interact with multidimensional data sources, and defines common extensions for exposing OLAP-specific functionality native to such data sources [2]. It extends OLE DB by defining the concept of a *cube*, which is a set of related dimensions and measures; a new object, the *dataset*, which represents an instance of a cube; and interfaces to access and navigate the dataset. Additional schema rowsets represent metadata objects such as cubes, measures, dimensions, hierarchies, levels, properties (for each dimension level), and members. A query language called multidimensional expressions (MDX); an extension of SQL enables the computation of datasets.

File System. Microsoft Index Server is a content search engine over documents stored in the Windows NT file system. The engine contains filters that enable content and property indexing of files in formats such as ASCII, HTML, Word, Excel, PowerPoint, NNTP news documents, and Adobe PDF. Since Index Server exposes its services as an OLE DB provider, it is possible to develop data-intensive applications over data stored in the file system. The following is an example of a query that finds all Word documents in the directory 'c:\My Documents' containing the words "Data" and "Access" within 20 words of each other, created after June 1st, 1997 ordered by their size in bytes.

```
SELECT FileName, DocAuthor, Size
FROM SCOPE('c:\My Documents')
WHERE
CONTAINS('"Data" NEAR(word,20) "Access"')>0
AND create >= '1997/06/01'
AND FileName LIKE '%[.]doc' ORDER BY Size Desc
```

Spatial. The OpenGIS Consortium has developed a set of standard extensions to OLE DB to enable efficient access to geo-spatial sources [3]. The extensions consist of new schema rowsets to identify tables that correspond to a *geographic feature*; enumerate the columns that contain *geometry* including their type such as line, point, area, polylines, and their spatial reference system; and enumerate and characterize the *spatial reference systems* supported by the provider.

4. SERVICE ARCHITECTURE

Universal Data Access provides an architecture that allows individual, specialized components to implement discrete sets of database functionality around less capable stores. These sets of database functionality are defined by OLE DB interfaces. Composition of services is achieved transparently to consumers and providers. This architecture represents a new approach to extending database functionality dynamically, at run time, by aggregating OLE DB services through COM Aggregation. This architecture enables the creation of service components that can be reused to enrich the capabilities of any OLE DB source. The rest of the section gives an example of five specific services that can be combined using the service component architecture.

4.1 Cursor Service

The Cursor Service consumes data from a rowset provider, caches the data in a local store, and exposes the cursor data plus rich data-manipulation functionality such as local filtering, sorting and scrolling through rowset interfaces. The input to this service can come from a native provider, a service component, or a business application producing results as a rowset.

4.2 Update and Synchronization Service

For applications using local data, the Synchronization Service component complements the Cursor and Remote Data Services by providing the ability to send updates back to the provider and to refresh the data in local cursors with current data from the provider. Both the update and refresh operations can be applied to one or more rows simultaneously and can be coupled together in a single user command. Both operations can ensure consistency of multi-table cursors if the structure of the join relation between the participating tables is known.

4.3 Shape Service

The Shape Service provides a hierarchical view of data as well as the reshaping and hierarchical navigation capabilities that user interfaces and reporting tools require. This service creates new hierarchical rowsets or aggregates existing rowsets into hierarchies. It implements three types of hierarchies, based on relationship, parameterized queries, and grouping. The shape service allows local hierarchies to be reshaped to offer applications a different perspective on the data.

4.4 Remote Data Service

The Remote Data Service (RDS) component provides client applications efficient access to OLE DB data sources over "connectionless" environments such as the Internet. RDS provides efficient marshaling of rowsets over HTTP and DCOM protocols. RDS works with the Cursor Service to provide client-side caching of rowsets, and the Synchronization Service to provide asynchronous update and refresh of data.

4.5 Distributed Query Service

Microsoft SQL Server contributes to the Universal Data Access platform by supporting distributed heterogeneous queries and transactional updates against a variety of relational and non-relational OLE DB data sources. An OLE DB data source is registered in SQL Server as a *linked server*. Once a linked server is defined, its data can be accessed using the four-part name <linked-server>.<catalog>.<schema>.<object>. The following example establishes a linked server to an Oracle server using the Microsoft OLE DB Provider for Oracle:

```
exec sp_addlinkedserver OraSvr,
     'Oracle 7.3', 'MSDAORA', 'OracleServer'
```

A query against this linked server is expressed as:

```
SELECT * FROM OraSvr.CORP.ADMIN.SALES
```

In addition, SQL Server supports built-in, parameterized table-valued functions called *OpenRowset* and *OpenQuery* which allow sending uninterpreted queries to a provider or linked server, respectively, in the dialect supported by the provider. The following query combines information stored in an Oracle and Index Server linked servers. It lists all documents and their author containing the words *Data* and *Access* ordered by the author's department and name.

```
SELECT e.dept, f.DocAuthor, f.FileName
FROM OraSvr.Corp.Admin.Employee e,
     OpenQuery( EmpFiles,
  'select DocAuthor, FileName
  from scope(''c:\EmpDocs'')
  where
  contains(''"Data" near() "Access"'')>0') f
WHERE e.name = f.DocAutor
ORDER BY e.dept, f.DocAuthor
```

SQL Server uses Microsoft Distributed Transaction Coordinator and the OLE DB transaction interfaces of the provider to ensure atomicity of transactions spanning multiple data sources.

5. ACTIVEX DATA OBJECTS (ADO)

While OLE DB is a powerful API for manipulating data, most application developers do not need the fine-grained control that OLE DB offers. Most developers are not interested in managing memory resources, manually aggregating components, and other low-level operations. ActiveX Data Objects (ADO) is an application-level data access object model that allows easy access to any OLE DB data source from languages such as Visual Basic, C++, Java, VBScript, and JavaScript.

6. REFERENCES

[1] J.A. Blakeley. *Data Access for the Masses through OLE DB,* Proc. ACM Sigmod 1996 Conf., pp. 161-172.

[2] Microsoft. *OLE DB for OLAP,* Version 1.0 specification, http://www.microsoft.com/data/oledb/olap, Feb. 6, 1998.

[3] OpenGIS. *OpenGIS Features forOLE-COM: Implementation Specification.* Document 97-023, July 28, 1997.