OLE DB: A Component DBMS Architecture
José A. Blakeley
Microsoft Corporation
One Microsoft Way, Redmond WA, 98052-6399
joseb@microsoft.com

Abstract

This talk describes an effort at Microsoft whose primary goal is to enable applications to have uniform access to data stored in diverse DBMS and non-DBMS information containers. Applications continue to take advantage of the benefits of database technology such as declarative queries, transactional access, and security without having to transfer data from its place of origin to a DBMS. Our approach consists of defining an open, extensible collection of interfaces that factor and encapsulate orthogonal, independently reusable portions of DBMS functionality. These interfaces define the boundaries of DBMS components such as record containers, and query processors that enable uniform, transactional access to data among such components. The proposed interfaces extend Microsoft's OLE Component Object Model (COM) with database functionality, hence these interfaces are collectively referred to as OLE DB. The OLE DB functional areas include data access and updates (rowsets), query processing, catalog information, notifications, transactions, security, and distribution. This talk presents an overview of the OLE DB approach and its areas of componentization.

1. Introduction

Today, a vast amount of critical information necessary to conduct day-to-day business is found outside the traditional, production corporate databases. Instead, this information is found in file systems, indexed-sequential access methods (e.g., Btrieve), personal databases (e.g., Access, Paradox), and productivity tools (e.g., spreadsheets, project management, email). To take advantage of database technology (e.g., declarative queries, concurrency control, recovery, security), today's applications must move the data from their original containing system into a DBMS. This process is expensive and impractical. However, applications want to exploit the advantages of database technology not just when accessing data within a DBMS, but also when accessing data from any other information container.

OLE DB leverages the COM infrastructure, which reduces unnecessary duplication of services and provides a higher degree of interoperability not only among diverse information sources, but also among programming environments and tools already developed for this environment. Indeed, OLE DB is the way to access data in a COM environment.

2. OLE DB Architecture

In OLE DB, a client is any system or application component that consumes an OLE DB interface; this includes OLE DB components themselves. A provider is any component that exposes an OLE DB interface. The following subsections define the current areas of componentization in OLE DB.

2.1 Rowsets

A rowset is the unifying abstraction that enables OLE DB components to consume and produce tabular data. Conceptually, a rowset is a multiset of rows where each row has columns of data. Query processors consume rowsets during query evaluation and produce rowsets as results. Rowsets may have additional properties that determine the ability of the rowset to insert, delete, and update rows as well as the rowset's navigation capabilities such as direct access or scrollability. All access and update operations are performed through rowset interfaces.

2.2 Commands

In OLE DB, data definition and data manipulation statements are referred to as commands. A command object encapsulates the services of a query processor in today's DBMSs. Command objects expose various interfaces representing different pieces of functionality of a query processor such as query formulation, validation, and execution. Queries are a kind of
command that return results in the form of a rowset. Data definition commands, for example to create a table, index, or view do not return rowsets. Commands can be expressed in the form of SQL text, or in the form of a command tree.

2.3 Schema Information

All schema information within a database is presented to the client through rowsets. There is an interface for clients to obtain information about tables, columns, data types, indexes, views, triggers, assertions and constraints, statistics, character sets, collations, domains, as well as some other relationships among them (e.g., constraints and tables, constraints and columns, columns and domains).

2.4 Notifications

OLE DB uses a well known model for notifications widely used in OLE/COM to allow notifications among OLE DB components and clients. There are two notification models: local notifications and data source notifications. Local notifications are used by groups of cooperating clients sharing a rowset under a single transaction. Local notifications enable cooperating clients to be informed about rowset changes originated by their peers. Data source notifications (watches) are designed to enable clients to be notified about changes to the underlying data source performed by other concurrent components running under different transactions.

2.5 Transactions

OLE DB provides interfaces that enable applications to define units of work as transactions. Transactions support the ACID properties. They also permit the specification of various isolation levels and optimistic concurrency control policies to allow more flexible access to data among concurrent clients. There is also a model for nested transactions. In cooperation with OLE Transactions, OLE DB components may participate in coordinated distributed transactions.

2.6 Security

OLE DB provides interfaces that allow authenticated and authorized access to data among components and applications. OLE DB provides a unified view of security mechanisms supported by the operating system and database components.

2.7 Distribution

Distributed OLE DB offers an infrastructure for clients to interact with providers transparently across process and machine boundaries. OLE DB client code is entirely unaware of whether the provider component is executing locally or remotely.

3. Summary

OLE DB allows applications to share and access information stored in diverse information containers including file systems; personal databases such as desktop DBMSs, spreadsheets, email, personal finance databases; ISAMs; traditional DBMSs; and more. The degree of componentization of database function promoted by OLE DB enables a richer, more flexible, and scaleable set of client-server applications.