Comparing OLE DB and ODBC

This document compares the basic features of OLE DB to the basic features of ODBC (Open Database Connectivity). It is intended to help clarify when to use one over the other.

Introduction

OLE DB and ODBC are both specifications created by Microsoft to address universal data access. Each is intended as an industry standard that will make access between one vendor's data store and another vendor's data consumer seamless.

Microsoft's definition of OLE DB is

> a strategic system-level programming interface to data across the organization. OLE DB is an open specification designed to build on the success of ODBC by providing an open standard for accessing all kinds of data.

And the definition of ODBC is

> an industry standard and a component of Microsoft® Windows® Open Services Architecture (WOSA). The ODBC interface makes it possible for applications to access data from a variety of database management systems (DBMSs). ODBC permits maximum interoperability—an application can access data in diverse DBMSs through a single interface. Furthermore, that application will be independent of any DBMS from which it accesses data. Users of the application can add software components called drivers, which create an interface between an application and a specific DBMS.

The two primary differences between the two are

- OLE DB is a component based specification and ODBC is a procedural based specification
- SQL is the core of accessing data using ODBC but just one of the means of data access through OLE DB
- ODBC is constrained to relational data stores; OLE DB supports all forms of data stores (relational, hierarchical, etc)

In general, OLE DB provides a richer and more flexible interface for data access because it is not tightly bound to a command syntax (like SQL in the case of ODBC). As Microsoft points out:

> Whereas ODBC was created to access relational databases, OLE DB is designed for relational and non-relational information sources, including mainframe ISAM/VSAM and hierarchical databases; e-mail and file system stores; text, graphical, and geographical data; custom business objects; and more.

ODBC on the other hand is a more established interface because it has been around longer; there are proven drivers and applications available in the market place. OLE DB is creating a better known presence, but consumer support is probably still considered limited. With the release of Windows 2000 Professional, OLE DB will be installed as part of the operating system. In Microsoft Office2000 all member applications will be OLE DB consumers. When these two major products ship, OLE DB will be much more ensconced in the marketplace.

Thomas Cox does a good job of outlining ODBC's limitations in his SUGI paper "What's up with OLE DB?". In recap they are:

- tightly bound to the SQL language and relational data
- no standard leveling of functionality (i.e. drivers do not support standard levels of functionality)
- each flavor of SQL makes each driver unique undermining the universal intent of the API
- difficult to deploy due since many software pieces must be synchronized (server, driver, OS, etc)

Microsoft created OLE DB to address these issues and better support Internet integration and multi-threading.

Political pros and cons

The debate over ODBC versus OLE DB boils down to two areas: political and technical.

In the political arena, we have the "because Microsoft said so" argument against the "Microsoft always changes their mind" counterpoint. Over the past two to three years, Microsoft has been heavily touting OLE DB as the successor

to ODBC. At PDC 98 they went as far as to state that there would no longer be continuing to develop ODBC (they would fix problems in their drivers, as needed, but not pursue future versions of the API). They plan to invest all new development into OLE DB.
So the politically correct thing to tell a user that asks which technology is to start migrating their applications to OLE DB because it is the industry direction:

- OLE DB is Microsoft's answer to universal data access in all forms (rectangular, relational, hierarchical, multi-dimensional, etc.)
- With the next major release of Microsoft's flagship products (its operating systems, and Office suite) OLE DB will be fully integrated in Microsoft's desktop environment.
- Microsoft's RAD tools (VisualBasic, Visual C++, Visual Interdev) currently support OLE DB.

In following this line of reasoning, we anticipate that our own ODBC drivers will eventually be replaced by OLE DB providers. Users should consider when and how they may be able to migrate their applications from ODBC to OLE DB if this trend is realized.

Technical pros and cons

Given the frequent political controversies around Microsoft's marketing decisions, it is also important (probably more so than the political motivations!) to share the technical pros and cons between ODBC and OLE DB when talking to users:

- If the user wants to access data independently of the SQL language, he or she should migrate to OLE DB.

  As we have pointed out, ODBC is bound to the SQL language. If the user's problem can best be solved with direct table manipulation, OLE DB is a better solution.
- If the user wants parallel interfaces for rectangular and multi-dimensional data, he or she should migrate to OLE DB.

  At present the only open industry standard for multi-dimensional data is based on OLE DB. If the user has a problem that involves the integration of these two forms of data, OLE DB is a better solution. It will reduce the amount of code needed.
- If the user wants to access data in an object server, he or she ***must*** migrate to OLE DB; there is no ODBC outlet.

  There is no ODBC access to an IOM object server. The OLE DB provider that ships with this product is the *only* way a user has to get to SAS data sets in such a server.
- If the user wants to access different data stores (local, SHARE, IOM, etc) from a single application, he or she should migrate to OLE DB.

  If the user plans to access data in an object server and any other data store, he should consider migrating the entire application to OLE DB. Otherwise he will be maintaining separate code paths for each access method. This argument is further supported by the theoretical idea that OLE DB reduces the likelihood of differences between drivers, SQL dialects and DBMSs that has been problematic among ODBC configurations.
- If the user wants to exploit as much of the SAS data model as possible from a thin-client application, he or she should migrate to OLE DB.

  While ODBC is constrained by SQL, OLE DB is not. The OLE DB specification is much richer and more extensible than ODBC. We intend to develop our providers in future releases to encapsulate as much of the SAS data model as possible. To fully exploit this data model in a thin-client environment the user should migrate to OLE DB.
- If the user needs the ability perform concurrent updates, he or she should migrate to OLE DB.

  To update records using the ODBC driver the user must issue SQL UPDATE statements which are basically batch oriented updates. There is no concept of record locking in the ODBC model. OLE DB accomodates several locking models that allow multiple concurrent updates. This feature of OLE DB enables the development of much more data management applications than with ODBC.