Sharing Enterprise Data with ODBC

Andy Sarnhart, SAS Institute, Cary, NC

The information in this paper was presented as part of a Formal Online Demonstration at SUGI 19, entitled The SAS® System for Windows, Release 6.10, presented by Carol Williams Rigsbee and John Schroedl, SAS Institute Inc.

ABSTRACT

ODBC provides an interface to many diverse data sources on the back end, and many applications support it. The SAS® System for Microsoft® Windows exploits this capability to provide enterprise solutions for sharing data from multiple sources.

INTRODUCTION

ODBC stands for Open Database Connectivity. It is a specification for a common API (Application Programming Interface) for connecting to different databases in the same fashion. This means that a program can be written to the ODBC specification as a client, and access any database which provides an ODBC driver interface without changing the application. Using drivers which are available today as an example, a program could switch from using local DBase® data to using ORACLE® on a remote server simply by specifying a different data source with no change to underlying program code.

The ODBC specification is provided by Microsoft as part of the ODBC SDK (Software Development Kit) in the Programmer's Reference Manual. It is also available for download in the WINEXT forum of CompuServe® Information Services.

ODBC was developed from a snapshot of the interface being designed by the SQL Access Group (SAG) called the Client Library Interface (CLI). At this point, the SAG CLI has been further defined, and ODBC and SAG CLI are not completely compatible, but a later release of ODBC will be.

ODBC is tightly coupled to SQL (Structured Query Language) as both the syntax for making requests of a database as well the model for presentation of the data - all data must be presented as if it were table oriented. This does not mean that you must learn SQL in order to benefit from ODBC. Many popular programs which access data through ODBC include a front end to present the data in a more visual model and create the SQL queries for you. The SAS System provides this capability in the SQL window under SAS/ASSIST®.

ODBC is part of WOSA (Windows Open Services Architecture), a set of standards which allow programs to implement consistent interfaces in various functional areas. MAPI is a WOSA standard for e-mail; TAPI will support telephony equipment, and a number of other emerging standards will provide easier interoperability between products from various vendors. ODBC is currently available only on the PC under Microsoft® Windows.

ODBC allows databases to be treated by applications in much the same way that a printer driver provides access to varying printers; an ODBC conformant application will not have to be customized for new data bases if they provide an adequate level of ODBC capability. To go further with the printer example, if a program requires high resolution and color, a consistent interface will not make an inadequate printer work, but will allow the application to tell the difference. To take this analogy to ODBC, some drivers will be read only, some will not provide index information or directory type functions, while most tend to be highly functional. The interface allows programs to determine the available functionality and make use of what is provided.

The ODBC API provides fairly rich functionality, if fully implemented. However, there are three levels of compliance: CORE, Level 1 and Level 2 (or low, medium and high level of functionality respectively). The SQL grammar supported also has three levels of compliance: minimum, core or extended (once again - low, medium and high respectively).
The term "data source" has special meaning in ODBC. A data source is a specified combination of a driver and driver specific parameters. Each driver may have multiple data sources. For instance, with a DBase driver, you might configure a data source named "Payroll" which had a path configured to point to a certain directory where payroll files are kept, and you might have another called "Football Statistics" which pointed to another directory.

ODBC drivers may be referred to as single or multi-tiered. A single tier driver is a driver which operates on data which resides on the same machine using direct reads and writes, while a multi-tiered driver does not directly manipulate the database, but either calls across the network or process space, or uses a library to access the database.

In operation, there are a number of different software components which are required to implement ODBC:

**ODBC Administrator or ODBC Setup**

The ODBC Administrator shipped with the initial ODBC development disk, and is a program which was added to the Program Manager as an icon. It is being phased out and being replaced with the ODBC Setup program, which is added to the Control Panel, taking its place. ODBC Setup is used to install drivers and create/configure data sources. The ODBC Setup program is provided by Microsoft and is redistributed with ODBC drivers from other vendors.

**ODBC Manager**

The DLL (Dynamic Link Library) that a client application is linked to. It loads the appropriate driver requested by the client for a connection and does some system level management, but primarily passes all calls through to the ODBC driver. The ODBC manager is provided by Microsoft and is redistributed with ODBC drivers from other vendors.

**ODBC Driver DLL**

This is the DLL which actually provides the interface to a specific database. It is provided by the database vendor or a 3rd party.

**ODBC Setup DLL**

A DLL which is used to create a data source. It is called by the ODBC Setup program. It usually has one or more dialog boxes to input the information necessary to define a data source for a given driver. For example, it might ask for the server name and database name in the case of a SQL Server setup (or node name, connection type and LIBNAME if it were SAS). It is provided by the database vendor or a 3rd party.

**ODBC Client application**

This is an application program which is using the ODBC API to access data. SAS using the ODBC pass through engine is a client application.
SAS/ACCESS® to ODBC

The SAS/ACCESS to ODBC is implemented as a SQL pass through engine under PROC SQL. It is not a full SAS/ACCESS engine, and does not use view descriptors. It can be used to perform queries against defined ODBC data sources to create SAS data sets or views, which appear in SAS as if they were data sets, but cannot be updated. A view also requires that the data source is available when SAS accesses the view. In PROC SQL, the two are done either by CREATE TABLE or CREATE VIEW, and you can think of creating a table as data migration (into SAS) and creating a view as data integration.

The Query window in SAS/ASSIST® has a new access type for ODBC, and allows users to easily navigate to desired tables and columns through the use of a friendly graphical interface. Complex queries can be generated without knowledge of SQL and saved as views, imported as tables, queried immediately into the output window or used to generate reports.

SQLDriverConnect and SQLTables. SQLDriverConnect allows "wildcard" connections where users select the data source from a list, and the driver prompts the user for any information not in the data source definition needed to connect (logically or physically) to the data base. SQLTables returns a list of available tables. The requirements seem to be in keeping or less restrictive than the industry as a whole, as we have not yet encountered a single ODBC driver which does not provide all the functionality we need. If functions beyond the required APIs are available in the driver, some which return result sets can be accessed through special queries.

In order to present data from diverse origins without loss of precision, ODBC provides a very rich set of data types. The types provided by ODBC drivers are converted to the most appropriate SAS type and FORMAts are automatically assigned as appropriate (DATE, DATETIME, TIME, DOLLAR). All type conversion necessary is performed by the pass through; we do not require (perhaps trust is a better word) drivers to do conversions for us. For instance, if an integral type is to be returned, we request it as an integral type then convert it to a double (SAS NUM format internal representation), even though the specification says we could just request it as a double.

SAS/ODBC DRIVER®

The SAS/ODBC DRIVER is a new product in version 6.10 for Windows. The driver is multi tiered and Level 1 conformant. It does not directly read and write SAS data sets. It communicates with a SAS server running SHARE or the BASE server (a new companion product providing access to SAS on the local machine) and uses Remote Library Services to access the data sets.

A SQL pass through is not really read only, but for most users and applications it will seem to be. In order to write to tables, the appropriate textual command for the INSERT or UPDATE would have to be fabricated and then could be submitted to execute. There are a few SCL programs which have been written to do this with other pass through engines, so a blanket categorization as a read only interface would not be correct, but it is correct to say that writing is not trivial and not supported with most procedures.

SAS/ACCESS to ODBC only requires CORE level APIs to be supported in a driver for pass through to work in PROC SQL. For the Query window to work, there must also be two Level 1 functions supported:
SAS' data hierarchy of Library and Data Set becomes Owner and Table in ODBC terminology. We do not provide Qualifier information (this is generally provided by desktop drivers to give a file name of the actual file like c:\db\test.dbf or something).

The SAS/ODBC DRIVER is installed just like most "standard" Windows products. A single diskette contains SETUP.EXE and will install all necessary components including the ODBC manager and administrator if they do not already exist. If the user already has the ODBC Manager, the driver can be added from the disk without running SETUP. There is a standard information file structure on the disk that the manager can use to unpack and install the disk contents.

SAS data sources are configured using the ODBC manager which calls into the SAS driver setup API. To configure a SAS data source, a server must first be configured (also within same setup) and selected. A server definition defines an access method, such as TCP, all the information required by the particular method in order to connect and an indication of whether it is a SHARE or BASE server. Multiple data source definitions can be made for any given server definition. At the data source level, libraries can be assigned (these are basically LIBNAME statements which are to be performed on connect) and numerous SQL options (described as relevant below) may be selected.

We claim minimum SQL grammar and when asked (via an info API call) about JOIN support we answer that we cannot, but this is actually due to inconsistency of grammar, not lack of functionality. If a user/programmer consults the PROC SQL documentation, then he/she will find additional SQL commands available beyond the minimum specification, but with syntax differing from the ODBC "standard". All the minimum SQL grammar we support does adhere to this standard.

ODBC provides a couple of mechanisms for "safe" updating. One method is using cursors, which point at rows in a result set returned by a query, to do positioned updates. We do not support cursors. The other method is to identify unique indices. We support the APIs for index reporting, and will provide adequate information on indexes created in SAS to allow safe updates. There is a cursor support library included in ODBC 2.0 which uses the index capability to provide cursor capability to applications even if cursors are not supported by the library. Unique indexes are still required, of course. It is important to note that some popular PC products will not use an index on a floating point column for fear of conversion causing a lack of distinction. This can be alleviated, if the floating point column in question (a SAS NUM field) is truly integer in nature. This is discussed in the paragraph on types.

There is another issue with respect to SAS and safe updates. PROC SQL has an option called UNDO_POLICY which controls whether or not the ability to undo updates made in a multi row update if one row fails is guaranteed. UNDO_POLICY is set to REQUIRED, OPTIONAL or NONE. In SQL, it is acceptable to update multiple rows (UPDATE cars where color="red" set price=price+500) and if the update fails, there is an expectation that no changes were made (or that any made before the error were undone). It makes sense that UNDO_POLICY=REQUIRED would be in effect. Unfortunately, in practice, this will often fail. The reason for this is that the PC program may have (and usually does) multiple result sets open on the table, such as one that is rows of data and one that is just the index values. It may then use another statement to try the update. Because the other statements have the data set open on the SAS side, the updates will fail since SAS cannot get exclusive access to the data. Even though all the statements are open to the same program, the prospect of a multi threaded executable in NT makes any assumption unsafe. So the UNDO_POLICY is configurable to be either OPTIONAL (undo mode on if possible, but don't fail if not possible) or REQUIRED with a data source option.

SAS really has two base data types: CHAR and NUM (double). However, FORMAT information on NUM fields can be used to provide better mapping to the proper ODBC data type. ODBC has a rich set, including a few integral types and floating types, as well as fixed and variable length text and binary
types, and dates, times and timestamps. Using the
FORMAT on a SAS data field, it is possible to infer
a corresponding SQL type for SAS NUM data other
than just SQL_DOUBLE. DATE, TIME and
DATETIME fields are reported as SQL_DATE,
SQL_TIME and SQL_TIMESTAMP respectively,
and the conversions are automatically performed.
The driver provides one extension because of our
awareness of SAS data characteristics: According to
the ODBC specification, requesting conversions
between numeric and time/date types should result
in an error; but we assume that the request is made
because of an understanding of the underlying SAS
type and allow it. There are a couple of options for
other type conversion. If the appropriate option was
selected in the data source definition, NUM fields
with a FORMAT of (N,O) where N <= 11 are
reported as SQL_INTEGER (long). Another option
allows CHAR fields of over 80 characters to be
reported as VAR_CHAR instead of CHAR, so that
PC data bases may be able to store long text fields
which are not always filled more efficiently. Trailing
blanks are stripped from these fields, and yet
another option controls whether trailing blanks are
stripped from all CHAR fields.

All the types which we can support conversion to are
also reported as creatable types, and CREATE
TABLE statements are parsed to convert the
extended types to their "NUM format=" counterparts
for SAS.

On a call to SQLColAttributes to check for financial
data, a check is performed for the DOLLAR formats.
There is not a money type in ODBC, but it is an
attribute which can be queried on numeric columns.

Errors are reported both locally and at the server (if
it occurs there). Many errors are reported simply as
invalid syntax from SAS, but the driver distinguishes
the most common errors (table not found, column
not found) and reports the appropriate specific code
for them as some applications have specific
recoveries for these situations.

In addition to the ODBC specification itself, the
SAS/ODBC DRIVER was designed to conform to a
white paper provided by Microsoft on the "Jet
Engine", their internal interface to external
databases, as a guide for implementation of ODBC
functions which must be supported to provide full
application interoperability. This white paper is
available from Microsoft on CompuServe, and is
recommended as an additional reference (in
addition to the ODBC specification) for custom
development for the SAS/ODBC DRIVER.

CONCLUSION

While ODBC is not the native interface to any
database, it provides a strong common interface to
many diverse sources of data and may be an
integral part of an enterprise wide solution.

SAS Institute is committed to the ODBC standard as
both a client and server, allowing the SAS System
to remain an integral part of your enterprise
solutions both now and in the future.

SAS, SAS/ASSIST, SAS/ODBC DRIVER and
SAS/ASSIST are registered trademarks or
trademarks of SAS Institute Inc. in the USA and
other countries.
© indicates USA registration.

Other brand or product names are registered
trademarks or their respective companies.

REFERENCES

Black, N. and Hecht, S. "White Paper - Jet
Database Engine ODBC Connectivity", Microsoft
Corporation. Available electronically in the WINEXT
forum on Compuserve® Information Services.