

Application Note for OpenAccess™

Subject: Exposing Data from ERP Systems through ODBC, OLE DB or JDBC

Date: March 2003

Markets: ERP/CRM Software Vendors

Need

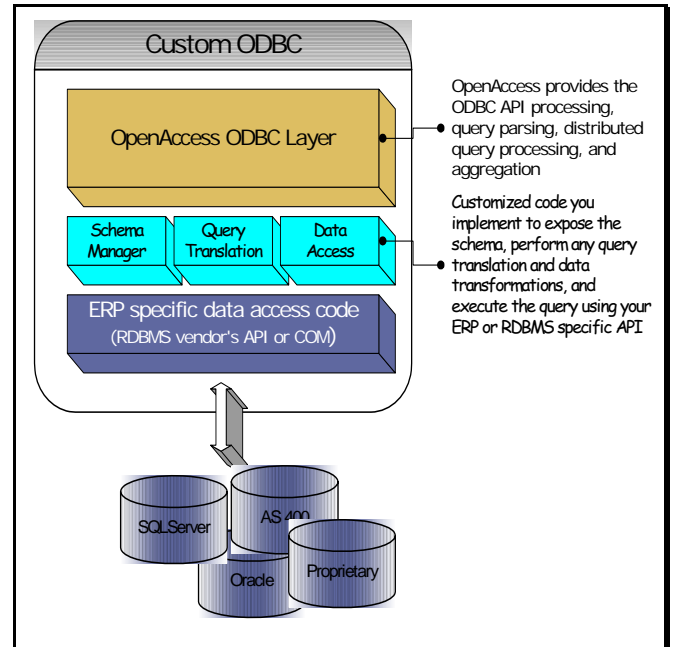
Are you happy with your existing reporting solution for your ERP system? Do you need to easily create reports using tools like Crystal Reports, Brio, Impromptu, MS Access, and others? Do you need to quickly analyze data using tools like Microsoft Access, Excel, Brio, and others?

Do you need to isolate your reports and applications from changes in the underlying ERP system due to version upgrades, customization and/or data relocation?

Do you want to use a business view (meaningful table and column names) of your data during report creation?

ERP systems store their data in relational databases like Microsoft SQL Server, Oracle, and others. This data is stored in complex schema that consists of hundreds of tables and thousands of columns. Each site in turn can customize their ERP system through changes in how the data is stored, addition of fields, and other customization. This makes it difficult to supply general purpose reporting tools. Writing applications that go directly to the database schema has limitations:

- Users need to understand how the data is stored in the database – relations.
- The table/column names can be very cryptic – F1001.
- The data may be stored in physically different databases of the same or different kinds – requires distributed query processing.
- Data conversion and look-ups are not possible through the use of a single query limit imposed by most reporting and analysis tools.
- Security and other business logic cannot be enforced.



OpenAccess Framework For Custom ODBC Driver

Features

An ideal solution would provide the following features:

1. **ODBC Compliant** - to provide read access to your application data from off the shelf commercial tools like Microsoft Access, Crystal Reports, Brio, Excel, ADO, ADO.NET and hundreds of other applications. Options to support OLE DB and JDBC as required.
2. **Flexibility** – work with site-specific implementation of the ERP system. This means the tables and columns displayed to the user are based on the current meta-data at the site.
3. **User Friendly** – ERP systems store data in hundreds of tables and thousands of columns whose names are cryptic. Using more meaningful names for the tables and the column makes it easier for the user to work with your custom implementation of the ERP system at the customer's site. This means the tables and

columns displayed to the user are based on the current meta-data.

4. **Data Location Independence** – support the distribution of data and meta-data in two or more databases. Reports and applications built to the logical layer exposed by the custom ODBC driver remain unchanged even when the physical layout of the data changes.
5. **Database Independence** – provide support for the databases your ERP system supports – Oracle, SQL Server, AS/400, etc. and any data encoding used for storing dates and numeric values.
6. **Performance** – take full advantage of your data source's SQL engine and client/server protocol.
7. **Platform Independence** – allow the distributed query processing to occur on any server platform – including Solaris, AIX, Linux, HP-UX, NT, and others.
8. **Quick Time To Market** – quickly get a functional ODBC out to customers based on proven technology.

Proposed Solution

OpenAccess DQP SDK provides the framework and pre-built components to quickly allow one or more data source (s) to be exposed as a single logical data source that behaves like a SQL compliant RDBMS database with standardized APIs that include ODBC, OLE DB and JDBC.

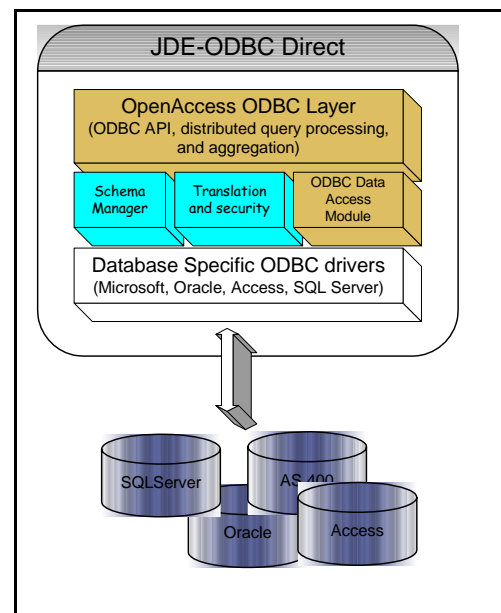
The OpenAccess components provide the ODBC, JDBC, or OLE DB APIs, SQL parsing, distributed query processing, aggregation, and a client/server protocol (if required). These components interact with the Interface Provider code that is implemented for a specific ERP system. The Interface Provider code implements the schema management, any required query translation, security, and the execution of the query against the data source.

End users query the data sources as if they were a single database using SQL through ODBC, OLE DB or JDBC. Queries are parsed and executed against each of the data sources in real-time and then the results are combined into a single result set and sent back to the client.

The ODBC driver developed using the OpenAccess DQP SDK product can enforce all the business rules, data conversions, security, and data configurations supported by your ERP system.

The ODBC driver can be developed to make use of an existing API supported by the ERP system or to directly go to the database(s) where the data is stored. In either case the user is exposed a logical view of the data that shows meaningful names for the table names and columns.

With OpenAccess DQP SDK, the amount of custom code required is minimal. OpenAccess handles all the ODBC related issues and handles all distributed joins where required. The custom code, which we refer to as the Interface Provider (IP), consists of the implementation of a data access module, a schema management module, and a translation layer. The schema manager module is responsible for using your meta-data to expose a schema. OpenAccess is shipped with a data access module for working with Oracle, SQL Server, MS Access, and AS/400 through the vendor's ODBC connectivity.



Architecture for JDE-ODBC Direct Implementation

Case Study – ODBC Driver for JD Edwards ERP System

The OneWorld product from JD Edwards is an example of an ERP system that has many of the characteristics addressed at the beginning of this application note. Unity Enterprise Solutions has used OpenAccess DQP SDK to implement an ODBC driver that can support the use of the Object Configuration Manager tables (meta-data) for exposing the schema and for accessing the data. This product is being marketed as jdeDirect ODBC driver. Please visit <http://www.jde-direct.com> for more information about this product.

The architecture chosen for this application was a local implementation in which the OpenAccess ODBC component runs on the desktop and uses the ODBC drivers for Oracle, SQL Server, AS/400, and MS Access to perform the actual data access. The Schema Manager module was designed to read the OCM and use it to expose the tables and columns for the selected user and environment. The Schema Manager module was custom implemented to understand the JDE OneWorld data dictionary and to provide this information to the OpenAccess layer. The Data Access module used was the ODBC module that comes standard with OpenAccess. The translation layer was implemented to support row-based security and user defined codes. In the figure above the orange colored components are provided with OpenAccess DQP SDK. The RDBMS vendors supply the white colored components. The aqua colored box represents the code custom written to adapt OpenAccess to JDE schema management.

Your Development Effort

1. Design and code the schema manager (15 days)
2. Enhance the data access module as required (10 days)
3. Do your QA (5 days)
4. Package up for distribution (2 days)

Expected time of completion: **32 man days**
Expected time for working prototype: **5 days**

Conclusion

This application note provides overview and details of using the OpenAccess SDK to implement a custom ODBC driver for your ERP system. OpenAccess also supports the development of OLE DB, JDBC, and .NET drivers. All three standards are based on consuming the IP that is developed to adapt your data source to OpenAccess. So if you use OpenAccess to implement ODBC, it's just a matter of linking with the OLE DB flavor of OpenAccess to get OLE DB and linking with JDBC flavor of OpenAccess to get JDBC – no code changes.

©2005 OpenAccess Software, Inc All Rights Reserved. OpenRDA is a registered trademark and OpenAccess is a trademark of OpenAccess Software, Inc. All other marks are of their respective owners. Although OpenAccess Software believes the information contained in this document to be accurate, OpenAccess Software cannot accept responsibility for omissions or errors contained within this document.