

Feature Note for OpenAccess SDK

Subject: Implement a custom ODBC or OLE DB access to a SQL compatible database
Date: 3/4/99
Proposed Release : 3.5 + Custom enhancements

Need

You have a database where the information is stored in a commercial SQL database (Oracle, Sybase, SQL Server, etc.) in some packed format that you chose because it made sense for your application. Now people want to be able to use standard desktop tools that use SQL queries to access data. This requires your data to be exposed in an unpacked way such that a off the shelf tool like Crystal Reports or Microsoft Access can be used to query the database and use the data directly without having to perform unpacking operations.

Your requirements are:

- **Standardized connectivity** - Provide connectivity through an ODBC Driver or a OLE DB provider
- **Expose modified schema** - the user does not see all the tables/columns in the target database or may want to see different names.
- **Modify query** - you may need to modify the query before it is executed.
- **Enforce security** – control who can do what with what data
- **Execute query using underlying RDBMS** - you want to be able to pass the query down to the target database for execution.
- **Modify data** – the data may be packed or in a non-standard format that needs to be mapped to a more standard format. An example is numbers stored in a packed format or a date stored in a Julian format.

Sample Case

We have a database that resides in IBM DB/2. It contains data in tables that are shared by many users. We want each user to only be able to see the rows that belong to them. This is done by having each table containing an additional column called CUSTOMER_ID that is hidden from the user. For example, assume we want multiple users to be able to maintain their order information in a table called ORDERS. This table is to be exposed to the end user as:

```
order_no      char(20)
part_no       char(64)
master        text(2000)
```

However it is defined in DB/2 as:

```
customer_id   int
order_no      char(20)
partn_no      char(64)
master        text(2000)
```

The user should be able to issue the following query:

```
> select * from ORDERS
```

and get back the fields order_no, part_no and master for all the rows belonging to that customer. This can be achieved by modifying the user's query with a CUSTOMER_ID restriction as follows:

```
>select order_no, part_no, master from ORDERS where CUSTOMER_ID=10001
```

We want the following features:

1. to provide a ODBC driver to access this information.
2. to expose a logical schema to the end user such that certain columns are hidden.
3. to be able to modify the query with customer restrictions before executing it

Proposed Solution

The OpenAccess product can help you create a customizable infrastructure for managing the distribution of information from your storage to the end user's desktop. The basic idea is to use the OpenAccess SDK product to build a custom ODBC driver or OLE DB provider for your database(s). Because you are using a software development kit, you have full control on how to process the query. The OpenAccess Database Access Manager (DAM) is a SQL processor that is used to provide a SQL interface to non-SQL databases or to parse SQL queries and obtain information that can be used to build a modified SQL query.

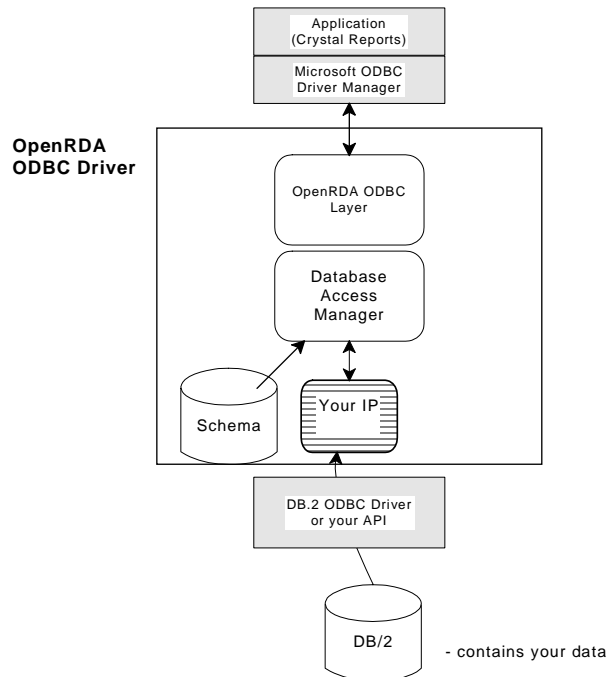


Figure 1: OpenAccess Solution

For this example we will use OpenAccess SDK in the SQL pass thru mode:

1. Expose schema that defines tables and columns in the format usable by desktop tools
2. Parse SQL queries submitted by the end user to obtain the required information in order for you to build a query to retrieve data from the DB/2 database. Once the query is parsed, your

code (Your IP) will call DAM function to retrieve the select list and the where expression and use this information to retrieve data from the DB/2 database.

Let's walk through the sequence of steps we need to perform in order to execute a validated SQL statement:

1. The application connects to the OpenRDA ODBC driver and executes a query: "select * from ORDERS"
2. The DAM takes this query and parses in and validates that the table name and the columns in the query are valid.
3. The EXECUTE function in Your IP is called. In your code, you access our data structures that contain the parse SQL query to build up a modified query to execute against your database. The DAM breaks down the SQL statement into all its basic elements for you such that you can easily generate the required query for your backend. In our example you would build "select order_no, part_no, master from ORDERS" and then add the " where CUSTOMER_ID=10001" to that tables restrictions. You can modify JOINED or NESTED queries.
4. You execute the modified SQL query using the ODBC driver for DB/2 or other API you may have for executing the SQL query against it. You then retrieve the results and provide them to the DAM for sending back to the client application.

Figure below shows a configuration for implementing a custom OLE DB provider for consuming existing ADO compatible data sources. Again the DAM is used to parse the query and your code handles the execution of the query against the target database.

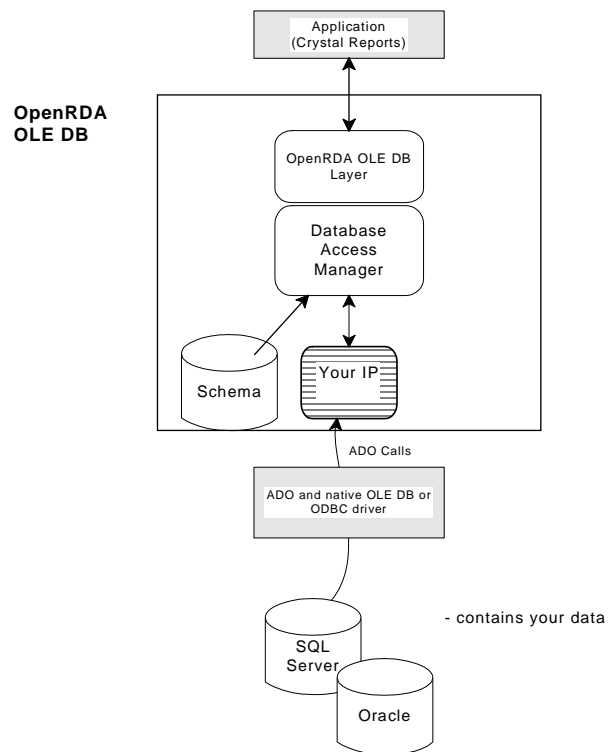


Figure 2: OLE DB Solution

Some Notes

- When building a row for the logical table as exposed to the end user, the data can come from multiple tables in the underlying database. This means you don't need to expose one table for every table in your DB/2 database. During schema definition you should decide what information the user needs and design a schema to allow them to easily get to this information.
- Details of pass thru mode can be found in the OpenAccess SDK Programmer's Guide. Example code can be downloaded from our support section at <http://www.odbcSDK.com>. The example code shows use of ODBC from the IP.

Schema Management

One of the key requirements is to expose the tables and columns to the end user to hide columns and tables in the target database. In our proposed solution, the schema will be managed by the DAM. If the schema information is static then it can be stored in our schema database. If it is dynamic then you can provide code as part of your IP to provide this information at run-time. In this case the DAM will call a schema function you register for tables, columns, index, and other information.

Your Development Effort

1. Design and implement the schema you want to expose. Here you need to decide what information you want to expose to the end user and how.
2. Start with our template IP and add your code to execute SQL queries using ADO or ODBC. We have sample code on how to use ADO or ODBC. If you already have a high-level API written to access your data then you can make use of it instead of making direct ADO/ ODBC calls. You can get access to a single table working within a day or so.
3. Link with our OpenRDA ODBC libraries to create the OpenRDA ODBC DLL.

Expected time of completion: 3-4 man weeks.

The example we provide is capable of accessing any ADO or ODBC compliant database. It is set up to map all tables with specific names into tables that are exposed through the OpenAccess ODBC/OLE DB layer.

Conclusion

OpenAccess SDK provides the required features to easily create a custom ODBC driver, OLE DB provider, or a JDBC driver for a SQL compliant data source in which schema, connection management, and/or data needs to be modified.