

Feature Note for OpenAccess™ SDK

Subject: Wrapping a SQL Component with ODBC, OLE DB, JDBC API

Date: 3/27/97, Revised Jan 7,2000

Available in : 3.5+

Author: OpenAccess Software, Inc.

Need

You may have an existing middleware component that is capable of executing SQL queries against various databases using your own mechanism. You now have a requirement to allow applications to use ODBC driver, OLE DB provider, or a JDBC driver to access this data.

- support access to information stored in relational and/or proprietary databases from standard desktop tools like Microsoft Access, PowerBuilder, Visual Basic, Crystal Reports, PERL, ASP, ADO and many other tools.
- use existing middleware for SQL processing
- limit the types of queries that can be issued
- expose schema for only the tables/columns the connected user is allowed to see
- expose a global schema of a distributed database that is physically stored in different databases

Sample Case

For this application note, assume we have data stored in one or more database but this data can only be accessed using a CORBA compliant business object that can take SQL queries and execute them against one or more instances of Oracle databases.

We want the following features:

1. to provide a single ODBC driver to access this information – the ODBC driver may be used on the same system or in client/server configuration
2. to expose only certain tables to the client applications.
3. to have full control over the execution of the SQL query – the Business Object is fully capable of doing this.
4. to have full control over data mapping as the data is transferred from the business object to the client application.
5. Support on both UNIX and NT platforms.
6. Support for desktop and client/server configurations.

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 application. The basic idea is to use the OpenAccess SDK product to build an ODBC interface to your database(s). Because you are using a software development kit, you have full control on how to process the query. And you can start from our existing library of interfaces for Oracle, Sybase, Informix, Ingres, DB2 and others. This allows you to have customized access to these databases in weeks.

For this example we will use OpenAccess in the SQL database access mode where the SQL queries are executed not by the OpenAccess SQL engine but by the code you provide. The code you write is referred to as the SQL IP. This allows you to make use of your existing software that is capable of executing SQL statements. We choose this mode because the component we are using to access the data in this example is already capable of executing SQL queries. As a side note, optionally the OpenAccess SQL engine can be used in this configuration to validate the SQL queries. Basically we parse the query and provide the parse tree to you such that you can control what queries you want to allow.

Figure 1 shows OpenAccess in client/server architecture in which you build a server process that includes our OpenRDA Server code and your code that will communicate with your Business Object. Figure 2 shows OpenAccess in desktop configuration where you build a DLL or a shared library that includes the OpenAccess ODBC or OLE DB code and your code that will communicate with your Business Object. The code in the green box labeled Your SQL IP Code is responsible for implementing the functions listed in Table 1. These functions are called by the OpenAccess code to handle ODBC/OLE DB API calls made by an application. Please refer to *the OpenAccess SQL IP Programmer's Guide* for more details.

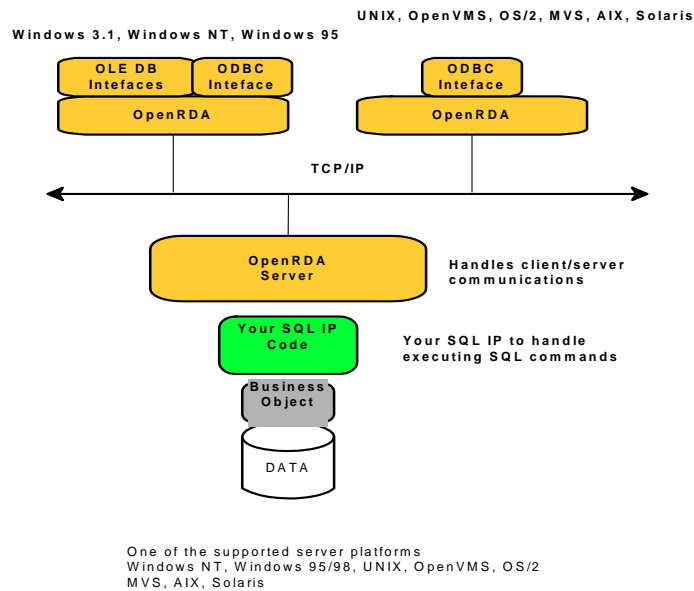


Figure 1: Client/server Architecture

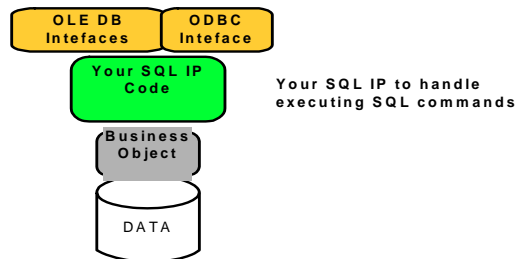


Figure 2: Desktop Configuration

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

1. Receive connection from the client with their user name and password - use this information as is or with different user name and password to connect to your Business Object.
2. Receive the query from the application
3. Your SQL IP Code is called to execute the query after which it is called again to retrieve the data. Your IP retrieves each row of data from the Business Object. The IP handles data manipulation queries and schema information queries.

Table 1: SQL IP API – Function your code needs to implement

Operation	Description
init	Called at startup to initialize the IP. The IP performs all the required startup processing.
exit	Called when the OpenRDA Server (in client/server) or the client DLL (in local case) is shutting down. The IP should free all resources and shutdown any open connections.
alloc_connect	Called when a client wants to establish a connection with a datasource served by the IP. Setup a handle to be used for this connection.
connect	Called when a client wants to establish a connection with a datasource served by the IP. Authentication information such as the user name and password are passed in. The IP attempts to connect to the database at this time.
disconnect	Close the connection. The IP should close any files or other connections established on behalf of this connection.
free_connect	Free the connection handle. The IP should close any files or other connections established on behalf of this connection.
start_transaction	Called to initiate a new transaction. The IP can use this entry point to perform transaction management for each connection.
commit	Commit the changes
rollback	Called with operation code COMMIT or ROLLBACK.
alloc_stmt	Allocate a statement handle to be used for executing
free_stmt	Free a statement handle that was allocated by the ALLOC_STMT
execute_immediate	Executes the given non-select SQL statement and return the number of rows affected by the statement.
prepare	Parse the given statement and generate plan for executing the statement. The IP should collect information about any parameters in the statement and also description about result columns (if stmt is of type SELECT)
execute	Execute the prepared non-select stmt and return the number of rows affected by the stmt. It is recommended that the IP create a new memory tree to be used for statement execution and free the tree after completion. Note that the IP should be able to handle execution of the same stmt multiple times.
declare_cursor	Associate the cursor name with the prepared select stmt. If the IP has no use for a cursor name, it can have an empty function for this.
open_cursor	Open the cursor i.e., execute the select stmt. Save the result table and place the cursor before the first row in the table. Return the number of rows selected. It is recommended the IP create a new memory tree for execution which it should free when close_cursor() is called.
close_cursor	Close the cursor and clear the result table. Free the Exec memory tree if the IP had a separate memory tree for statement execution.
is_cursor_open	Check if the cursor is open on the current statement handle
fetch_row	Advance the cursor to the next row of the result table. On reaching End of the Table, return SQLDRV_EOS
get_numparams	Returns the number of parameter markers in the prepared stmt. If the stmt does not have any parameters, returns 0.

init_param	Initialize the parameter with the specified value..
get_numcols	Return the number of columns in the result set. If the statement is a DML (INSERT, UPDATE, DELETE), it will return 0. If statement is a query (SELECT) it will return number of columns
get_colspec	Returns the description about the column.
get_colval	Return the column value in the same format as the type of the column. For example, if the column is defined as XO_TYPE_CHAR, then the data is a character string.
error	Return the error during the last operation on either the connection or statement handle. Once the error is returned, delete the error. This function is called until it returns SQLDRV_EOS to indicate no more errors exist.

Your Development Effort

1. Start with our SQL IP template or one we have developed and enhance with your rules for accepting a query and accounting for the usage. We have production quality SQL IP s that have been developed for Oracle, ODBC, Sybase and others.
2. Link with our OpenRDA Server libraries or desktop libraries to create the server executable or the shared library.

Expected time of completion: 1 man week

Benefits of using OpenAccess

- **Binary support on many platforms** – the code you get is supported in binary format on most popular operating systems.
- **Compatibility** – the ODBC drivers developed using OpenAccess are compatible with the Microsoft ODBC driver manager version 2.5+ on Windows and with Intersolv and iODBC driver managers versions 2.5+ on UNIX.
- **Thread safe** – the ODBC drivers developed using OpenAccess are thread safe and can be configured to be fully multi-threaded or to be thread safe in which only one thread is allowed to perform an operation at a time.
- **Based on proven technology** – the OpenAccess components you use to build an ODBC driver for your SQL component are the same pieces used by hundreds of our OEM to implement driver for non-SQL databases and are the same pieces used by thousands of end users to access Microsoft SQL Server and other SQL databases on NT from UNIX.
- **Write code once and deploy in desktop or in client/server configuration** – the SQL IP code you write stays the same whether you will be running in desktop configuration or in client/server configuration – just build with different libraries.
- **Support** – our developers provide the support so that you can quickly and accurately get your questions answered.
- **Flexible licensing** – we will work with you to make it cost effective for you to use our technology.