PROGRESS DataDirect

ACCESS SCADA AND DATA HISTORIAN SYSTEMS

INTRODUCTION

Open Database Connectivity (ODBC) compliant access to data has become a primary requirement by end users on vendors supplying applications containing data. It is becoming even more crucial as corporate organizations accelerate the deployment of client/server computing. Widely used tools such as word processors, spreadsheets and database application development tools are ODBC enabled and they provide means of building powerful end-user applications quickly through ODBC access.

SCADA systems support archiving and storing of real-time data collected from many different real-time data sources. In addition to the collected data, they maintain configuration data. These systems are normally run on UNIX, VAX/VMS, Alpha/OpenVMS or Windows NT type of platforms. End users want to access this information from the desktop for reporting, application development and analysis. More recently, the ability to make this information available on the web provides a new paradigm for data access and distribution.

In order to make this information easily accessible and usable, one needs a software infrastructure that:

- Supports access to information stored in proprietary databases from standard desktop tools like Microsoft Access, PowerBuilder, Visual Basic, Crystal Reports and many others. Today this means ODBC, OLE DB and JDBC compatible access.
- Limits the types of queries that can be issued.
- Supports many hardware platforms and operating systems.
- Can be developed with minimal man-power investment without having to build in-house expertise in ODBC, JDBC, OLE-DB or SQL.

Once a user has the ability to use ODBC to access your SCADA system, the possibilities are endless.

HIGHLIGHTS:

- Quickly implement a data connectivity driver
- Design the Interface Provider (IP) to perform data access functions
- Support stored procedures
- Implement business rules

ODBC ACCESS TO A SCADA SYSTEM – SAMPLE CASE AND REQUIREMENTS

For this use case, assume we have a SCADA system that stores current values, history of values and configuration information. The configuration of the system is based on the use of unique tag names for each I/O point. Each tag point can be associated to a physical I/O or state point. The current values are stored in memory and are accessible by snapshot API functions. The archived data is stored in compressed files and is accessible through archive API functions.

PROGRESS

We want the following features:

1. **ODBC access** - Provide ODBC access to the current values, archived data and configuration data.
2. **Fast data access** – Based on the tag name and time range, optimized access to the archive data.
3. **Data in tabular format** - Archive data retrieved in column format where the first column is the time stamp and each additional column contains values for different tag names.
4. **Access raw or interpolated data** - Since the SCADA system will compress the data, the user needs to be able to get the data as it was stored or interpolated to provide uniformly spaced data points.
5. **Restricted query types** – Restrict queries to avoid extremely large data set retrieval. This may be done through restrictions on the timeframe for each request.

For this sample case, assume we want to expose three tables: SNAPSHOT, ARCHIVE, and TAGDIR.

The SNAPSHOT table provides access to the current values and is defined to have columns TAG, VALUE, TIME and STATUS. The ARCHIVE table provides access to the historical data and is defined to have columns TAG, VALUE and TIME. The TAGDIR table describes each of the tags in terms of its name, I/O location and other SCADA specific information. Also, assume that the SCADA system exposes functions to allow data to be retrieved by specifying the tag name and the time interval.

HOW TO QUICKLY IMPLEMENT AN ODBC DRIVER FOR YOUR SCADA SYSTEM

The Progress® DataDirect® OpenAccess™ ODBC SDK product can help you create an ODBC driver for your SCADA system within weeks! We provide 90% of the solution in supported headache-free binary code. All you need to do is implement the functions to read and write rows of data from and to your SCADA database and configure the Schema Database with the table definitions—no need to learn ODBC or SQL. Figure 1 shows the proposed architecture using the OpenAccess ODBC SDK product. The only portion you develop is the box labeled "Interface Provider," or IP. This code is responsible for accepting a data access request along with the optimization information from the Database Access Manager (DAM) and using this information to build a set of rows.

For this IP, we will use the supplied Schema Manager for our schema definition. We use SQL queries to set up the table and column definitions as required. These are entered into a text file and then the file is executed using the interactive SQL tools provided with the toolkit.

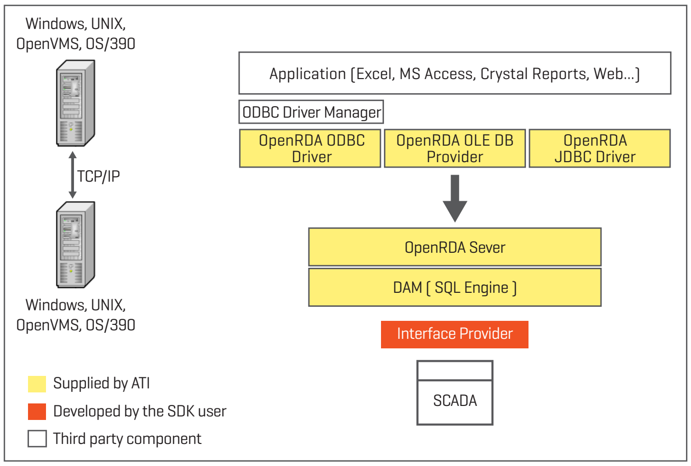

Figure 1: *OpenAccess ODBC SDK architecture*

Next, we design the IP to perform the data access functions. The following steps describe the operation the IP performs in the course of executing a query:

1. **Receive a connection from the client with their user name and password** - use this information as is or with a different user name and password to connect to your SCADA system. The IP receives this request when the client issues an ODBC connect call.
2. **Receive an execute request** - the IP is called to execute a query when the client issues an ODBC SQLExecute call. At this time, the IP can call the functions in the DAM to find out what table is being accessed and what conditions are specified in the query. For example take the query given below:

   ```
   select tag, value from ARCHIVE where tag = 'T1001' and time
   between 'jan 1,1997 and feb 1,1997' (Q1)
   ```

   To execute this query, the IP will first determine the table being accessed and the requested operation. In this case, the operation is SELECT and the table is ARCHIVE. Next, it will ask the DAM to report restrictions on tag and time columns. In this case, the DAM returns (=,'T1001') on TAG and (between, jan 1,1997, feb 1,1997) on TIME. The IP then uses this information to call the SCADA functions to retrieve the data. For each value retrieved, the IP builds a row and passes it to the DAM for further evaluation and for placing in the result set.

   If the DAM returned no restrictions on the TIME columns, the IP can assume defaults (current time) or report an error indicating all required values not specified.

   Let's now assume the query is of the form:

   ```
   select tag, value, time from ARCHIVE where (tag = 'T1001' or
   tag='T2002') and time between 'jan 1,1997 and feb 1,1997' and
   status ='OK' (Q2)
   ```

   Here we have added a tag and one more restriction. Since the condition on STATUS will not be of any use when retrieving data using the SCADA functions, we still do as before and only get the restrictions on TAG and TIME and then use that information to call the SCADA function to get the data. First, we get data where tag is T1001, then we get data where tag is T2002. The rows that do not match the condition status='OK' will be filtered by the DAM.

   This query will generate a result set with rows of data for T1001 and rows of data for T2002. This may not be acceptable when displaying in a spreadsheet format where we want data to be side by side for each tag. Instead of looking like:

TAG	VALUE	TIME
T1001	1.01	jan 1,1997
T1001	1.23	jan 2,1997
T2002	1.50	jan 1,1997
T2002	1.90	jan 2,1997

   we want the data to look like:

TIME	T1001	T2002
jan 1,1997	1.01	1.50
jan 2,1997	1.23	1.90
...		

The only way to achieve this with a standard SQL is to formulate query Q2 as a join on itself as shown below:

```
SELECT T1.TIME, T1.VALUE AS T101 _ VALUE, T2.VALUE AS T102 _
VALUE FROM ARCHIVE T1, ARCHIVE T2 WHERE (T1.TAG = 'T1001' AND
T1.TIME between 'jan 1,1997 and feb 1,1997') and ( T2.TAG =
'T1002' AND T2.TIME between 'jan 1,1997 and feb 1,1997') and
T1.TIME = T2.TIME
```

The DAM allows the IP to be aware of this type of query and to execute them efficiently. Normal join processing of two tables is handled through a cross product that results in N*N rows of data being generated. In the optimized scheme supported by the DAM, the IP will be first asked to build all rows where TAG = T1001 and the time is between jan 1 and feb 1. Then, for each row in this result set, the IP will be called to process queries where TAG='T2002' and TIME as it appears in the row. Before this happens, the IP is notified that it will be called N number of times, where N is the number of result rows of the query with TAG set to T1001. The IP can either build one row each time it is called or build all rows at the first call and return rows as requested. Building all rows at once requires only one call to the SCADA system. Because of the optimized join support in the DAM, joins are handled very efficiently.

3. **Receive a disconnect** - the IP frees any resources allocated on behalf of the client connection.

   As seen by the above steps, the OpenAccess ODBC SDK makes it very simple for the IP to access data in an optimized way. The above steps can be implemented in less than 50 lines of code.

OPENACCESS FEATURES SUITED FOR SCADA SYSTEMS

This section highlights some of the features in OpenAccess ODBC SDK that make it ideal for accessing current value, configuration and historical data maintained in SCADA systems.

Specifically, the OpenAccess ODBC SDK provides:

- Access to full details of the query
- IP defined expression processing
- Optimized join processing
- Schema management
- Ability to support stored procedure
- Ability to implement business rules

ACCESS TO FULL DETAILS OF THE QUERY

The DAM allows the IP to find out as little or as much as it wants about a query. This helps the IP use information contained in the query to limit the number of rows it reads. Information about the query can be obtained on individual columns or on the entire WHERE clause. You can retrieve the entire WHERE clause as a set of expressions on the referenced columns. This information can be used to reformulate the query into your own syntax if your database supports a query language. The default information provided to the IP is sufficient to implement optimized access to SCADA systems. You can also retrieve the full expression information to implement powerful optimization schemes.

In certain implementations, our customers wanted to utilize specialized hardware to increase the rate of data processing. They do this by retrieving the full WHERE expression and then use the features of the hardware to optimize its execution.

IP DEFINED EXPRESSION PROCESSING

Many systems have their own formats for specifying the time data type. For example, you may support a format of the type: time between now and now-2hr. This allows the end user to specify time in a very flexible way. The DAM easily supports this by letting you handle this expression. You decide if the rows you return have time data between now and now-2hr. In this case the DAM will not attempt to evaluate this expression.

OPTIMIZED JOIN PROCESSING

Joins are very expensive unless handled properly. The DAM handles all joins. The IP is only responsible for accessing data from a single table at a time. For join processing, the IP is called to process a SELECT for each table. The simplest way to implement a join is for the IP to execute the query on one of the tables, execute the query on the next table, and then perform a Cartesian product between the two tables. A Cartesian product of two result sets of size M and N is M*N. This can be a large number as M and N grow and this happens very often when a single piece of information is spread amongst many tables. The DAM optimizes this by first building a result set for the first table and then going through each row in this set and passing the required information to the next query as restrictions. This way, the off diagonal elements are not even created. The IP is called to process SELECT on the second table as many times are there are rows in the first result set. The IP does not have to build the M*N set in which most of the data gets thrown away.

SCHEMA MANAGEMENT

The OpenAccess ODBC SDK allows you to provide the schema information (data dictionary) for your database tables by either populating the Schema Database provided as part of the DAM or by implementing schema function in the IP. Using the built in schema manager makes it very easy for you to configure your tables. You simply write SQL statements to insert the table and column definitions into the schema manager database. If you choose, you can implement functions in the IP to use your existing data dictionary.

STORED PROCEDURES

OpenAccess ODBC SDK allows client applications to invoke stored procedures within the IP. This allows the IP to implement many functions on the server as part of the IP code. A stored procedure is implemented as part of the IP code or within your database system and exposed by a name and the required arguments.

BUSINESS RULES

OpenAccess ODBC SDK allows the IP to easily implement business logic. Since an IP is exposing a view, it has complete control over how SELECTS, INSERTS, UPDATES and DELETES are processed on each of the tables. The IP can take advantage of this to validate the queries and to guarantee that business rules are not broken. You can easily check the validity of the input data and control what kinds of queries are allowed.

YOUR DEVELOPMENT EFFORT

1. Define the schema for the data to be exposed (2 man days)
2. Implement the IP code to access the tables defined in item 1 (5 man days)
3. Test your IP with ODBC applications (2 man days)
4. Perform optimization as needed to IP (3 man days)
5. Package-up with run-time libraries for distribution (2 man days)

Total: **14 man days**

PROGRESS SOFTWARE

Progress Software Corporation (NASDAQ: PRGS) is a global software company that simplifies the development, deployment and management of business applications on-premise or in the cloud, on any platform or device, to any data source, with enhanced performance, minimal IT complexity and low total cost of ownership.

WORLDWIDE HEADQUARTERS

Progress Software Corporation, 14 Oak Park, Bedford, MA 01730 USA Tel: +1 781 280-4000 Fax: +1 781 280-4095 On the Web at: www.progress.com

Find us on facebook.com/progresssw twitter.com/progresssw youtube.com/progresssw

For regional international office locations and contact information, please go to www.progress.com/worldwide

Progress, DataDirect, DataDirect Connect, OpenAccess and SequeLink are trademarks or registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and other countries. Any other marks contained herein may be trademarks of their respective owners. Specifications subject to change without notice.

© 2008, 2014 Progress Software Corporation. All rights reserved.

Rev. 9/14

PROGRESS