ACCESSING A HIERARCHICAL DATA STORE THROUGH AN SQL INPUT

Cross-reference to Related Applications

The present application claims benefit of the filing date of U.S. Provisional Application No. 60/104,682 entitled MODELING TOOL SYSTEMS AND METHODS, filed on October 16, 1998, incorporated by reference herein in its entirety.

The present application is related to a co-pending U.S. Patent Application No. _____ (Atty. Docket #22074661-25535) entitled APPARATUS AND METHOD FOR MODELING TOOLS, being concurrently filed on the same day, which is incorporated by reference herein in its entirety.

The present application is related to a co-pending U.S. Patent Application No. _____ (Attorney Docket #22074661-25532) entitled METHOD FOR IMPACT ANALYSIS OF A MODEL, being concurrently filed on the same day, which is incorporated by reference herein in its entirety.

The present application is related to co-pending U.S. Patent Application No. _____ (Attorney Docket #22074661-25531) entitled METHOD FOR DETERMINING DIFFERENCES BETWEEN TWO OR MORE MODELS, being concurrently filed on the same day, which is incorporated by reference herein in its entirety.

The present application is related to co-pending U.S. Patent Application No. _____ (Attorney Docket #22074661-25533) entitled METHOD AND SYSTEM FOR AN EXTENSIBLE MACRO LANGUAGE, being concurrently filed on the same day, which is incorporated by reference herein in its entirety.

Field of the Invention

The present invention relates to a database management system, and in particular to an Application Programming Interface (API) that allows application programs access to various database management systems.

Background Information

Open DataBase Connectivity (ODBC) is an Application Programming Interface (API) that provides a common interface for accessing various structure query language (SQL) based database management systems. Most vendors have added an ODBC interface to their relational databases. One major disadvantage of the conventional

ODBC is that it was designed to access relational database management systems that use standard SQL language in their queries and its use outside the relational databases has met with significant problems because SQL assumes the relational nature of the database.

Therefore it is desirable to provide a system and method for providing an ODBC interface or driver that allow access to non-relational data store such as a hierarchical object/property model.

Summary of the Invention

The present invention provides a driver that allows access of data stored in a hierarchical data store through an SQL input query. Based on the SQL input, the driver transforms a hierarchical data store into relational database tables. Preferably, the tables hold pointers that point to the actual data stored in the hierarchical data store. Then the SQL input is executed using the transformed tables.

Brief Description of the Drawings

Figure 1 is a block diagram showing an ODBC driver interfaced between an SQL report generator and a hierarchical data store according to the present invention.

Figure 2 is a flow chart of the ODBC driver according to the present invention.

Figure 3 illustrates a representation of a hierarchical data store and that of an equivalent relational database.

Detailed Description of the Invention

An hierarchical data store is represented as a graph of objects and properties. Objects may be owned by and conceptually aggregated into other objects. Objects may hold references to other objects via reference properties. Properties are owned by and aggregated into objects. FIG. 3 includes a representation of such an hierarchical data store and that of an equivalent relational database.

The hierarchical data store in FIG. 3 is a model of customers and their order information. The data store includes an object or a group of objects called "Customer" each having a unique internal identification number, "int_id". The "Customer" object has three properties: one scalar property and two vector properties. The one scalar property is of type "Name" which can hold only one value by definition. The first vector property is of type "Phone_number" which may hold many values by definition. The second vector property is of type "Order_ref" which may also hold many values.

Each value of "Order_ref" is a pointer which references another object called "Order". The "Order" object has a property of type "DateOrdered" which stores the date of a particular order that has been placed. The equivalent data store in a relational database can be represented as three separate tables as shown on the right side of FIG. 3. A "Customer" table has two columns named "Id" and "Name", an "Order" table has three columns named "Order_id", "DateOrdered" and "Cust", and a "Phone" table has three columns named "Cust", "Seq" and "Phone_number". The three tables are linked to each other through "Id" of Customer, "Cust" of Phone and "Cust" of Order which have identical values.

FIG. 1 is a block diagram showing an ODBC driver 6 interfaced between an SQL report generator 2 and a hierarchical data store 4. As discussed above, a conventional ODBC is designed to access relational databases using standard SQL language queries. The ODBC driver 6 of the present invention allows standard SQL language queries from an SQL report generator 2 to be used against a standard object/property model of information such as a hierarchical data store 4 according to the steps of FIG. 2. In step 10, the driver 6 receives the SQL input from the generator 2. In step 12, the driver 6 identifies object classes and their properties to be processed based on the received SQL input. In step 14, the hierarchical data store 4 is transformed into relational database tables with various columns such that the hierarchical data store 4 appears to be a relational database to the SQL report generator 2. In a preferred embodiment, step 14 involves the following manipulations such that the hierarchically stored database is transformed into relational database tables:

1. each class of object is transformed into a table. Each table of this type has two pseudocolumns: an "Id" that contains the unique identifier of the instance and is the primary key, and an "Owner" that contains a foreign key reference back to the owning object;

2. each non-reference scalar property of an object becomes a column in the object's class table; and

3. each non-reference vector property is transformed into a table. Each table of this type has two pseudocolumns: a "Sequence" column that contains a sequence value for each element of the vector and is part of the primary key for the table, and an "Owner" column that contains a foreign key back to the owning object and is part of the

primary key; and

4. each reference property, both scalar and vector, may be used to calculate a foreign key relationship from the referenced object to the object owning the reference property.

When the necessary tables and columns are created/transformed, they preferably have pointers that point to the actual data residing in the hierarchical data store, and do not store the data themselves. The query contained in the SQL input is then executed in step 16 on the transformed tables. In step 18, the result of the execution is then passed to the report generator 2 that transmitted the SQL input.

A more detailed explanation of the steps in FIG. 2 will be provided below with reference to a couple of example SQL inputs. As one example, assume that the ODBC driver 6 receives the following SQL input query transmitted from the SQL report generator 2:

Select DateOrdered from Order a, Customer b where a.Cust = b.Id and b.Name = "Doe".

The above SQL input means that the generator would like to receive DateOrdered for all orders that "Doe" has placed. In relational terms, the way the Customer and the Order tables are connected or joined is that the value in the "Cust" column of the Order table should match the value in the Id column of the Customer table. The "Id" is the primary key for the Customer table and "Cust" is a foreign key of the Order table. As can be appreciated by persons of ordinary skill in the art, the hierarchical data store in FIG. 3 is not set up in a form that is readily usable by a conventional ODBC interface. This is one of the problems that is solved by the ODBC driver 6 according to the present invention.

The ODBC driver 6 analyzes the received SQL input and identifies the object classes that need to be processed. The ODBC driver 6 then transforms each object class into a table. In other words, a database table is created for each identified object class because object classes map to tables. In the example above, there are only two objects, "Customer" and "Order". Accordingly, two tables named "Customer" and "Order" are created. As part of the transformation process, the columns for the tables are created partly by analyzing what columns are being referenced in the received SQL language and by analyzing the properties of the object classes since columns map to the

properties. In addition, the internal identification number "int_id" is mapped to a column of the corresponding table. As a result of the transform, the ODBC driver creates a "Customer" table with "Id" and "Name" columns, and an "Order" table with "Order_id", "DateOrdered" and "Cust" columns as shown in FIG. 3. The "int_ id" and "Name" of the Customer object map to "Id" and "Name" columns of the "Customer" table, and the "int_ id" and "DateOrdered" of the Order object map to "Order_id", "DateOrdered" columns of the "Order" table.

At this point, the only column that we need to synthesize is the "Cust" column of the Order table. However, the customer property in the hierarchical data store does not exist in the "Order" object. In other words, there's no property in the Order object pointing back to the "Cust" ("int_id") of the Customer object. In fact, it's the inverse of the relational database because it's the Customer object that points to the Order object. This is a difference in the topologies between a relational database and an object/property model such as the hierarchical data store.

In this case, it is recognized that the "Cust" column is a foreign key in the relational database. The foreign key concept is similar to a reference property in the hierarchical data store such as the "Order_ref" property of the "Customer" object. Recognizing that the query requires a foreign key backwards, the "Order_ref" property of the "Customer" object is read, the inverse relationship is synthesized, and the "Cust" column is then synthesized or created using the "int_id" of the Customer object.

With the necessary tables and columns fully built, the tables preferably store pointers that point to the actual data residing in the hierarchical data store, and do not store the data themselves. Alternatively, actual data may be stored in the tables.

In a second example of an SQL language input, assume that the ODBC driver 6 receives the following:

Select phone_number from Customer a, Phone b where a.Id = b.Cust and a.Name = "Sears".

In this example, a different transform occurs because it references a vector property "phone_number". As discussed above, each non-reference vector property is transformed into a table. The property itself maps to one column named "phone_number". Two more columns are synthesized. One column named "Cust" is the id ("int_id") of the object ("Customer") that owned the "phone_number" and a

second column named "seq" is an arbitrary increasing number so that different phone numbers for the same object can be differentiated.

As in the first example, once the necessary tables and columns are fully built, the tables preferably store pointers that point to the actual data residing in the hierarchical data store. The SQL input is then executed using the transformed tables. The data matching the selection criteria of the SQL input are selected from the hierarchical data store pointed to by the pointers stored in the transformed tables and passed to the report generator 2. Again, actual data may be stored in the tables.

From the foregoing, it will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the invention.

What is claimed is:

1. A method of accessing data stored in a hierarchical data store by using an SQL input, comprising the steps of:

receiving an SQL input;

transforming a hierarchical data store into relational database tables based on the received SQL input; and

executing the SQL input using the transformed tables.

2. A method as in claim 1 including analyzing the received SQL input to select at least one of the objects in the hierarchical data store to transform into the relational database.

3. A method as in claim 2 wherein analyzing the received SQL input includes selecting at least one of the objects in the hierarchical data store.

4. A method of accessing particular data from a hierarchical data store, comprising:

receiving an SQL input;

analyzing the received SQL input to determine object classes in the hierarchical data store to be queried;

analyzing the received SQL input to determine columns referenced by the received SQL input;

creating at least one table for an object class that is to be queried, the table having at least one column to represent a property in the object class;

synthesizing at least one column for one foreign key in the at least one table; and

executing the SQL input against the tables created.

5. A method as in claim 4 including transforming a scalar property in an object in the hierarchical data store into the table, which table has at least one column for the scalar property and at least one column for a foreign key associated with the object.

6. A method as in claim 4 including transforming a vector property in an object in the hierarchical data store into the table, which table has at least one column for at least one value in the vector and at least one column for a foreign key associated with the object.

7. A method as in claim 4 including storing in the table at least one instance of actual data.

8. A method as in claim 4 including storing in the table at least one pointer to data.

9. A method as in claim 4 including:

transforming a scalar property in an object in the hierarchical data store into the table, which table has at least one column for the scalar property and at least one column for a foreign key associated with the object; and

transforming a vector property in an object in the hierarchical data store into the table, which table has at least one column for at least one value in the vector and at least one column for a foreign key associated with the object.

10. A method as in claim 9 wherein at least one table includes a column for each of the properties that the SQL input references.

11. A method of accessing particular data from a hierarchical data store, comprising:

receiving an external SQL statement;

analyzing the SQL statement to determine objects in the hierarchical data store to be queried to provide the data that the SQL statement requires;

transforming each determined object into a table that contains at least a column that contains an unique identifier to act as a primary key for the table and a column that contains a foreign key reference back to the owning object;

transforming each non-reference scalar property of an object into a column in the table that is the transformation table for that object;

transforming each non-reference vector property of an object into a table, which table has a sequence column to be at least part of the primary key for that table, a column that contains a foreign key reference back to the owning object;

calculating from each reference property of an object a foreign key relationship from the referenced object to the object owning the reference property;

populating the tables with one of the following: actual data, references to actual data, or a combination of actual data and references to actual data; and

executing the SQL input against the tables.

12. A method as in claim 11 wherein at least some, but not all, of the properties in the hierarchical data store are transformed into the relational database.