Achieving Database Interoperability Across Data Access APIs through SQL Up-leveling

SQL up-leveling provides the capability to write a SQL statement that can be executed across multiple databases, regardless of the databases' SQL implementation. For example, a SQL statement written against Oracle will also be interoperable with DB2.

Each of the major database vendors implements certain functionality in a unique proprietary format. The functionality differences from the ANSI SQL standard fall into the following broad categories:

- Date and time literals representation
- Syntax for outer joins
- Syntax for scalar functions
- Methods for invoking stored procedures
- Dynamic SQL statement format corrections
- Empty string and null conversions
- Long character string insertion support

These proprietary SQL implementations impact the interoperability provided by open data access standards. Adoption of open standards has enabled companies to reduce expenditures by delivering a high degree of interoperability among databases. The inability to provide SQL interoperability has hindered some organizations from realizing the full benefits of adopting these open standards.

DataDirect Technologies® has long been an industry leader in providing companies the highest level of interoperability between databases. One of the key areas of interoperability is SQL up-leveling. DataDirect provides SQL up-leveling across all open data access APIs. This gives developers a common way to implement these types of functionality — using the same code — regardless of the backend DBMS.

SQL Up-leveling Across APIs

SQL up-leveling is not a new concept. DataDirect products have implemented SQL up-leveling from the beginning. In the ODBC data access standard, SQL up-leveling was implemented through *escape clauses*. The escape clauses provided a syntax for certain functions to be performed independently of database vendors' implementation of the SQL standard. When Java emerged as a cross-platform development environment, Sun also implemented these escape clauses in the JDBC API to provide a level of interoperability between databases. The ADO.NET data access standard, however, does not require that ADO.NET data providers implement

proprietary SQL functions across databases in a standard way. This limitation weakens application portability across data sources with ADO.NET.

DataDirect Technologies has overcome this shortcoming in the ADO.NET specification by implementing SQL up-leveling in all DataDirect Connect *for* ADO.NET managed data providers. In addition, DataDirect data providers are written in 100% managed code. This means that developers can take advantage of true database interoperability while taking advantage of the performance and security features provided within the .NET Framework through 100% managed code.

Escape Clauses and SQL Extensions

SQL extensions are just that, extensions to the SQL standard that are unique to open data access standards. SQL extensions provide a consistent method of SQL up-leveling to access SQL features available in many databases, but that have not been incorporated into the SQL standard. These SQL extensions, hereafter referred to as escape syntax, provide a high degree of interoperability for developers who want to use common SQL statements across databases. Each database vendor typically implements these features in a proprietary SQL extension. The escape syntax provides a way of using these features in a common, standard format that is independent of the underlying database.

Using the escape clause, the ODBC and JDBC drivers and DataDirect ADO.NET data providers can easily scan SQL statements for syntax that may require special processing to be understood by the database. The escape clause consists of a pair of curly braces “{ }” surrounding the SQL syntax and a one- or two-character identifier to denote the type of escape clause. The driver or data provider then translates these escape clauses into syntax that is accepted by the database. This provides the greatest level of interoperability for developers who want to maximize the use of SQL statements in applications, regardless of the database implementation. The following sections discuss specific implementations of the escape clause and how to use them.

Date and Time Literals

All major databases support, to some extent, the data types DATE, TIME, and TIMESTAMP or DATETIME. The date and time formats implemented by the database vendors vary greatly, so there is little consistency for developers to leverage. A good example is the DATE format; the format required by Oracle is *mm dd, yyyy*, while for DB2, it is *yyyy-mm-dd*. The various implementations can be problematic for developers using these formats in applications that access multiple databases.

Escape clauses implement the SQL standard definition for the DATE data type, which is *yyyy-mm-dd*. To use a DATE data type across multiple databases, an application developer uses the format {d '*yyyy-mm-dd*'}.

For example, a developer writing a SQL statement that selects employees hired on October 11 or October 15, 2003, uses the following SQL syntax:

```
SELECT EmpName, HireDate
FROM Emp
WHERE HireDate = {d '2003-10-11'}
OR HireDate = {d '2003-10-15'}
```

DataDirect's Oracle JDBC driver translates this syntax to the following:

```
SELECT EmpName, HireDate
FROM Emp
WHERE HireDate = 'Oct 11, 2003'
OR HireDate = 'Oct 15, 2003'
```

DataDirect's SQL Server JDBC driver translates the syntax to:

```
SELECT EmpName, HireDate
FROM Emp
WHERE HireDate = '10-11-2003'
OR HireDate = '10-15-2003'
```

The escape clauses for TIME and TIMESTAMP use the SQL standard format for the syntax of the literals, using the `t` and `ts` character designations, respectively. Thus, the TIME escape clause is implemented as {t '*hh:mm:ss*'}, and the TIMESTAMP escape clause is implemented as {ts '*yyyy-mm-dd hh:mm:ss*'}. DataDirect products implement these escape syntaxes for DATE and TIME literals in the same way across all data access APIs. This means that developers can use a single SQL statement across different vendors' databases, as well as across JDBC, ODBC, and ADO.NET.

Outer Joins

Database vendors also differ in their implementation of the outer join functionality. Outer joins provide a convenient way to match records from two tables by forming pairs of related rows from the two different tables. The difference between outer joins and inner joins is that, if no pairs of data match, the inner join does not return a row, whereas the outer join returns a null value for a column in the row with unmatched data.

Outer joins are implemented in escape clauses via the following syntax:

```
{oj table1 LEFT|RIGHT|FULL OUTER JOIN table2 ON search
condition}
```

For a developer to write a query that shows all sales representatives in the Raleigh sales office, the outer join escape syntax is:

```
SELECT emp.EmpName, office.City
FROM {oj  emp LEFT OUTER JOIN office ON emp.EmpId
=office.EmpId}
WHERE emp.City = 'Raleigh'
```

DataDirect's Oracle ADO.NET data provider translates this statement to:

```
SELECT emp.EmpName, office.City
FROM cust, ord
WHERE emp.EmpId =office.EmpID  (+) AND emp.City = 'Raleigh'
```

DataDirect's SQL Server ADO.NET data provider translates the syntax to:

```
SELECT emp.EmpName, office.City
FROM emp, office
WHERE emp.EmpId *= office.EmpId AND office.city = 'Raleigh'
```

As developers plan to leverage SQL statements across applications that access multiple databases from different vendors, the need for SQL up-leveling in often complex outer join statements becomes extremely important.

Most ADO.NET data providers today do not support SQL up-leveling, a capability that is not built into the ADO.NET specification. In contrast, DataDirect has extended its successful implementation of SQL up-leveling capabilities to include its DataDirect Connect *for* ADO.NET data providers so developers can maximize SQL statement portability.

Scalar Functions

Database vendors implement scalar functions in various proprietary formats. The escape syntax provides the ability to perform scalar functions in a consistent manner across databases. Scalar functions operate against a single value, in contrast to aggregate functions which operate on a given set of values. The four main types of scalar functions are numeric, string, time and date, and system. The DataDirect product lines implement these scalar functions in a consistent manner across all APIs, allowing developers to leverage the maximum amount of SQL syntax reuse, regardless of database.

Numeric Functions

Numeric functions provide the ability to determine many mathematic values, such as sine, cosine, square root, absolute value and logarithms. Each of the many supported functions is called using the escape syntax, regardless of the database implementation. Data access components, however, can only provide access to functions that are available on the database backend. The following example shows how to call the square root function:

```
SELECT {fn SQRT(c_squared)} FROM table1 WHERE shape='triangle'
AND type='isosceles'
```

String Functions

String functions enable a developer to dynamically manipulate strings within the SQL result set, for example, setting all characters in a string to upper case, or returning a certain portion of a string from beginning or end. Some of the string functions available to developers are UCASE, LCASE, RTRIM, LTRIM, LENGTH, and SUBSTRING. The following statement returns UCASE values for a row that is mixed case in the database.

```
SELECT {fn UCASE(LastName)} FROM emp WHERE office = 'Raleigh'
```

Time and Date Functions

The time and date functions are very useful when developers need to manipulate date and time values, obtain days of week, or get the current time to use in a SQL statement. The following example uses the DAYNAME function to retrieve the date and day of the week the order was placed for the customer Progress.

```
SELECT date,{fn DAYNAME(date)} FROM orders WHERE
customer='Progress'
```

System Functions

System functions include the ability to retrieve a database name and user name for the current connection, and the ability to check for a NULL value and substitute with another value if the expression is a NULL. The following example uses the USER function in a Select statement to retrieve all the vacation days available for the current user:

```
SELECT VacationDays FROM emp WHERE EmpName={fn USER()}
```

Invoking Stored Procedures

Developers who want to maximize performance in applications typically use stored procedures whenever possible. Database vendors implement stored procedure syntax in proprietary formats, using different markers for the parameters accepted as arguments in the stored procedure. DataDirect provides an escape syntax that is consistent across all data access APIs. The escape syntax used for stored procedure invocation is `{call` *`procedure_name`* `(?,?)}` where the ? is the parameter marker if the stored procedure uses parameters. The ? parameter marker is used across all commercially available databases and APIs, ensuring that stored procedure invocation code does not need to be changed to access different databases. For example, a stored procedure call to return the property tax rate for a particular county, which accepts the parameter of a county name, uses the following syntax:

```
{CALL county_prop_tax (?)}
```

where the bound parameter is the county name submitted by the user.

This method of invoking the stored procedure ensures that developers who want to use stored procedures to maximize performance can do so in a consistent format across databases and data access APIs.

Dynamic SQL Statement Formatting Corrections

Certain databases require a specific format for SQL statements that extend to more than one line.

A good example of dynamic format correction is with “newline” characters in SQL statements. Databases such as DB2 do not recognize newline characters, so statements that use them generate database syntax errors. The DataDirect products for DB2 dynamically replace these newline characters with white space so that DB2 can process the SQL statement without syntax errors.

The functionality is implemented by the driver or data provider and requires no action from the developer. This level of interoperability between databases with regard to the syntax of SQL statements is implemented across all APIs and is dependent on the underlying database syntactical requirements.

Empty String and NULL Conversions

Developers writing SQL statements to the ANSI standard use two single quoted identifiers in a SQL statement to denote an empty string. However, databases such as Oracle use two single quoted identifiers to denote a null value. Developers who use data access components provided by database vendors are forced to keep track of these differences in their code, either by coding separate SQL statements or by programmatically altering the identifiers based on the underlying database’s implementation. DataDirect provides this conversion dynamically to the developer.

If a developer uses the following syntax in an application connected to Oracle:

```
INSERT INTO emp (FirstName, LastName, VacationDays) VALUES
('Tom', 'Smith', '' )
```

The DataDirect Connect *for* ODBC driver for Oracle alters the syntax to:

```
INSERT INTO emp (FirstName, LastName, VacationDays) VALUES
('Tom', 'Smith', "")
```

Null values are unable to use operators and are forced to use ISNULL and ISNOTNULL conditions. This ensures that the correct values are inserted into the database and that all operator logic with regards to strings still applies.

Long Character String Insertion Support

Some databases limit the length of a character string that can be inserted,. For example, Sybase throws an exception if a user attempts to insert strings that contain more than 2000 characters. To get around this issue, developers must check the length of the string before inserting it into the database. Alternatively, developers can catch the thrown exception, and bind the string and insert it as a parameter. Developers using DataDirect products have the latter approach performed for them dynamically. If the database throws an

exception while attempting to insert a long string, the driver or data provider catches the exception and inserts the string as a parameter dynamically.

Conclusion

Many organizations have adopted open data access standards, such as ODBC, JDBC and ADO.NET, to achieve application portability and improved developer productivity. However, the SQL standard is not always portable between different vendors' databases.

DataDirect delivers the necessary interoperability through a consistent implementation of SQL up-leveling. Developers can write SQL statements that are highly portable across the leading databases and data access APIs. Developer productivity is improved and deployment time and application maintenance is reduced, increasing an organization's return on investment.

We welcome your feedback! Please send any comments concerning documentation, including suggestions for other topics that you would like to see, to:

docgroup@datadirect.com

FOR MORE INFORMATION

800-876-3101

Worldwide Sales

Belgium (French).....0800 12 045
Belgium (Dutch).....0800 12 046
France.....0800 911 454
Germany 0800 181 78 76
Japan 0120.20.9613
Netherlands.....0800 022 0524
United Kingdom.....0800 169 19 07
United States.....800 876 3101

022009

DataDirect TECHNOLOGIES

DataDirect Technologies is the software industry's only comprehensive provider of software for connecting the world's most critical business applications to data and services, running on any platform, using proven and emerging standards. Developers worldwide depend on DataDirect® products to connect their applications to an unparalleled range of data sources using standards-based interfaces such as ODBC, JDBC™ and ADO.NET, XQuery and SOAP. More than 300 leading independent software vendors and thousands of enterprises rely on DataDirect Technologies to simplify and streamline data connectivity for distributed systems and to reduce the complexity of mainframe integration. DataDirect Technologies is an operating company of Progress Software Corporation (Nasdaq: PRGS). For more information, visit www.datadirect.com/.