BASIS SQL Gets Even Better

As databases evolve over the years, it often becomes necessary to add SQL functionality. While SQL is a relatively standard language for querying databases, various DBMS vendors often add syntax to their SQL not supported by all databases. BBj® now boasts new SQL support for several SQL features supported by other databases, as well as significant improvements to the way it handles SQL views.

Execute SQL/MySQL Script

The Enterprise Manager now allows an administrator to execute an SQL script on any BBj database. Script files should be plain text and contain one or more SQL statements, terminated by a semicolon and a new line. Statements can include CREATE and DROP statements for TABLE, VIEW, PROCEDURE, INDEX, and TRIGGER as well as CALL, INSERT, UPDATE, DELETE, ALTER, GRANT, and REVOKE statements. To execute a script, click the button shown in **Figure 1**, located on the database Information tab.

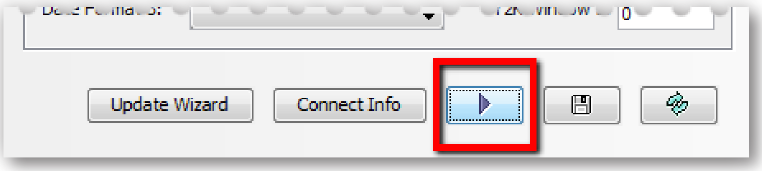
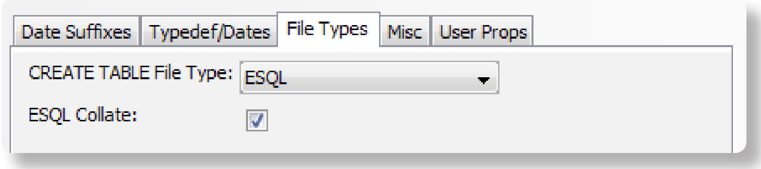


Figure 1. Execute an SQL script file

This powerful feature also includes support of most MySQL syntax for creating, modifying, and populating tables so administrators can easily import MySQL database exports into a BBj database. When importing from a MySQL script, set the "CREATE TABLE File Type" to ESQL as shown in **Figure 2**.



CREATE TABLE File Type: ESQL
ESQL Collate: ✓

Figure 2. CREATE TABLE File Type Setting

ENUM Type

BBj 12 now supports the use of the ENUM type (Enumerated Type) when creating a table. Note that this feature only works with ESQL tables as it relies on ESQL files' constraint capabilities. An ENUM type column's values are restricted to a set of explicit values specified at the time of table creation. For example, the following statement creates a table with an ENUM type:

```
CREATE TABLE my_table (id INTEGER PRIMARY KEY, gender ENUM ('M', 'F')) ESQL
```

This next statement generates an error because 'Q' does not meet the criteria for the enumeration:

```
INSERT INTO my_table VALUES (1, 'Q')
```

This statement will succeed:

```
INSERT INTO my_table VALUES (1, 'M')
```

The new ENUM type guarantees data integrity as only valid information may be written to the database.

REPLACE Statement

The REPLACE statement works just like INSERT with the exception that if the value specified for the primary key in the REPLACE statement already exists, it simply replaces the existing record with the new record specified in the REPLACE. For example, if ID is a

By Jeff Ash
Software Engineer

primary key, the first statement would insert a new record because it is new, while the second would change the NAME to `John`.

```
REPLACE my_table (id, name) VALUES (10, 'Jeff')
REPLACE my_table (id, name) VALUES (10, 'John')
```

Using REPLACE is a great timesaver for developers, as previously this would require one of the following multi-step processes to accomplish the same thing.

1. Execute a SELECT statement to determine if the record already exists
   - If the record exists, execute an SQL UPDATE statement
   - If the record does not exist, execute an SQL INSERT statement
2. Attempt to insert the record by executing an SQL INSERT statement, trapping for an error
   - If the SQL INSERT failed because the record already exists in the table, execute an SQL UPDATE statement

DATEDIFF Scalar Function

The DATEDIFF scalar function returns the number of days between two specified dates. Parameters can be literal expressions or the name of a date type column; it subtracts the second parameter from the first parameter. For example, the following returns a value of 7:

```
SELECT DATEDIFF('2012-08-08', '2012-08-01')
```

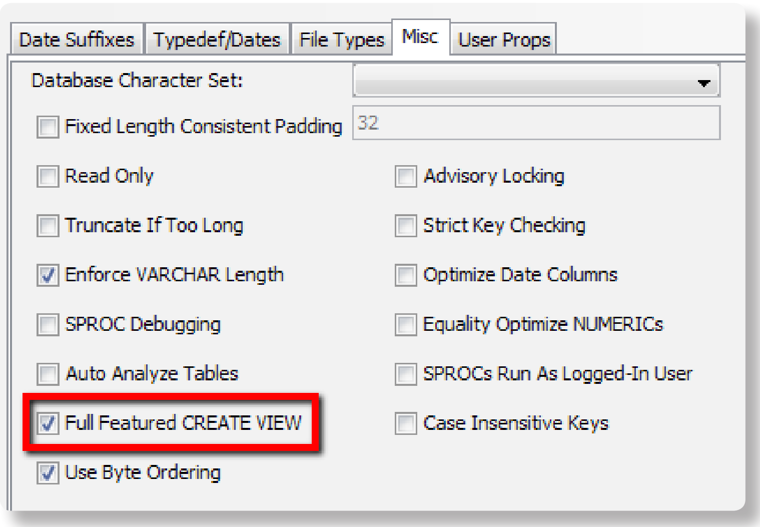
Reversing the order of the parameters returns -7:

```
SELECT DATEDIFF('2012-08-01', '2012-08-08')
```

This simple-to-use scalar function makes it very easy to perform this common date operation without the necessity for writing any application code. Further, using a scalar function to perform calculations makes it possible to use the SQL statement in reports without the need for custom report scripting.

Full Featured Views

Full featured views are a tremendous improvement over the way BASIS SQL traditionally supported views. Prior to full featured views, SQL views were limited to a SELECT column list, list of tables, and a limited WHERE clause. ORDER BY, LEFT JOIN, GROUP BY, etc. were unsupported. However, full featured views support all valid SELECT queries regardless of size or complexity and were first available in BBj 12. By default, adding a database to BBj configures that database to create full featured views. **Figure 3** shows the setting for enabling full featured views.



Database Character Set:

[ ] Fixed Length Consistent Padding 32
[ ] Read Only
[ ] Advisory Locking
[ ] Truncate If Too Long
[ ] Strict Key Checking
[✓] Enforce VARCHAR Length
[ ] Optimize Date Columns
[ ] SPROC Debugging
[ ] Equality Optimize NUMERICs
[ ] Auto Analyze Tables
[ ] SPROCs Run As Logged-In User
[✓] Full Featured CREATE VIEW
[ ] Case Insensitive Keys
[✓] Use Byte Ordering

Figure 3. Enabling Full Featured Views in Enterprise Manager

The following example of a complex CREATE VIEW statement now works in BBj with the addition of full featured views, it demonstrates the power and flexibility of this new capability.

```
CREATE VIEW my_great_view AS
SELECT t.name,
      count(t.order_num) AS NUM_ORDERS
FROM   (SELECT trim(c.last_name) + ', '
               + c.first_name AS NAME,
               o.order_num
        FROM   customer c
          LEFT JOIN order_header o
            ON c.cust_num = o.cust_num) t
GROUP BY t.name
ORDER BY t.name desc
```

Summary

All currently maintained database management systems evolve and grow over time to include new features, new syntax, and improvements to existing functionality. Because BASIS is always listening to customers for ideas about improving our products, count on more improvements and new features to the BASIS toolset. The addition of ENUM, REPLACE, DATEDIFF and full featured views gives BBj developers several more tools to use when developing database applications. Please keep the ideas coming! ■