

New Features and Changes in PowerHouse 4GL 8.4D1

New Features and Changes Specific to Relational Database Support

DataDirect ODBC Support (UNIX)

PowerHouse 4GL 8.43D1 supports the DataDirect 5.0 ODBC connection to Microsoft SQL Server. The DATABASE statement in PDL now allows TYPE ODBC. For details on PowerHouse support of ODBC, see Chapter 4 of the PowerHouse 4GL Version 8.4 New Features document.

SQL 92 Compatibility

PowerHouse 4GL 8.4xD introduced strict SQL 92 compatibility. In many cases this removed ambiguity and differences between databases. However, in some cases, it meant that code from previous releases caused parse errors. To remove the requirement for strict SQL 92 compatibility in version 8.4xD1, use the STRICT_SQL92 environment variable and set the value to NO. The default value is YES.

Quoted Stored Procedure Calls

As part of the strict SQL 92 compatibility introduced in PowerHouse 4GL 8.4xD, quoted stored procedure calls, where the quoted procedure call syntax was passed directly to the database, caused parse errors. For example:

```
> DECLARE mycursor CURSOR FOR CALL "myproc('param')"
```

is no longer accepted. In order to allow the double quotes and pass what is between the double quotes directly to the database in version 8.4xD1, specify the **quotedproccall** program parameter at compile time.

Resetting Bind Variables in SQL Statements

A bind variable is a placeholder in SQL generated at compile time where a value will be substituted at execution time. For example, if a request value for a Find is needed in generated SQL, a bind variable acts as the placeholder in the WHERE clause. Each bind variable has a unique identifier made up of a number and the field name. In previous versions, the number was incremented from statement to statement even though the field was the same. This meant that generated SQL was different even though the SQL statements themselves were the same. Because the generated SQL was different, it could not be reused by the database.

In 8.4xD1, the **resetbindvar** program parameter specifies that the bind variables are to be reset for each SQL statement. This allows the generated SQL to be identical for identical SQL syntax. The bind variables will be a letter and a number. The letter is S for Select operations, U for update operations, I for insert operations, and D for delete operations. The number is incremented from 1.

The program parameter is available in QDESIGN, QUICK, QUIZ, and QTP. The default is **resetbindvar**. To override the default, use **noresetbindvar**. There is also a resource file statement RESETBINDVAR|NORESETBINDVAR.

Concurrency Model for DB2 (UNIX, Windows)

The Concurrency transaction model has been changed to use a single transaction for all activities and cursor retention is supported. The section titled Concurrency Model for DB2 in Chapter 3 of the PowerHouse 4GL Version 8.4 New Features Document should be changed to read as follows:

The Concurrency model for DB2 defines both the Query and Update transactions for all screen phases, however, the Query transaction is not used. The Update transaction is used for all screen phases. All DB2 activities are associated with a single Update transaction.

Finding Existing Data

To find or select existing records, the Update transaction retrieves the data.

Changing Existing Data

When you use a screen to retrieve and change existing data, all three screen phases are used. The Update transaction starts when retrieving the existing data, continues to be used as needed to perform field processing, such as lookups, and performs any database updates.

Entering New Data

When you use a screen to enter new data, only the Process and Update phases are involved. The Update transaction is started, if required, during the Process phase, for example to satisfy lookups during field processing, or to send the new data to the database during the Update phase.

When updating in QUICK, the rows to be updated are locked, re-fetched, and checksummed to ensure that they have not been changed before being updated.

Cursor Retention

For the Concurrency Model DB2 retains the read position after the Update transaction has been committed. Cursor retention is supported.

Updating and Deleting Rows Containing BLOBs

This section is no longer needed.

DO BLOB BINARY|TEXT Option (OpenVMS, Windows)

If a blob is copied to or from a file using the wrong mechanism, it can be corrupted. The BINARY|TEXT option allows you to specify the contents of the blob and how it should be copied. The default on OpenVMS is TEXT. On OpenVMS, blobs have a 2048 byte fixed-length record format. The default on Windows is BINARY.

OpenVMS Examples

This statement copies the contents of a blob named file_blob in text format to a file named pmhcp.txt.

```
> DO BLOB file_blob command " " FILE "pmhcp.txt"
```

This statement copies the contents of a blob named file_blob in binary format to a file named prmhcp.pdf:

```
> DO BLOB file_blob BINARY command " " FILE "prmhcp.pdf"
```

Windows Examples

This statement copies the contents of a blob named file_blob in binary format to a file named prmhcp.pdf:

```
> DO BLOB file_blob command " " FILE "prmhcp.pdf"
```

This statement copies the contents of a blob named file_blob in text format to a file named pmhcp.txt.

```
> DO BLOB file_blob TEXT command " " FILE "pmhcp.txt"
```

SET FILE OPEN READ in QTP Creates Read-Only Transaction

Specifying SET FILE OPEN READ in QTP in versions prior to 8.4xD1 caused a read-write transaction to be created. In 8.4xD1, a read-only transaction is created.

Checksums no Longer Include Calculated Columns

When PowerHouse reads a row, a checksum is calculated based on the contents of the row. When the row is updated, it is reread and another checksum calculated for comparison to the original. This ensures that another user has not changed the data. If calculated columns are included in the checksum, the values may have been changed by the

database, resulting in a checksum mismatch. This can easily occur if the user does an Update Stay. Removing calculated columns from the checksum calculation eliminates these false errors.

Generating Database Creation SQL

QSHOW provides the capability to generate SQL CREATE syntax. The syntax is:

```
SET LANGUAGE SQL
GENERATE DATABASE file
```

If the named file is a relational database, QSHOW will generate the SQL from the database schema. If the named file is not a relational database, QSHOW will generate the SQL based on the dictionary definition. The generated SQL is created in a file named QSHOSQL (MPE/iX), QSHOSQL.TXT (OpenVMS), and qshogen.sql (UNIX, Windows).

This allows you to generate skeleton SQL for indexed files. To convert multiple indexed files into a single relational database, generate the SQL for each file one by one. After each file, copy the results from each file into another text file and make the appropriate changes. QSHOW does not provide the ability to change the name of the output file.

New Features and Changes Specific to PowerHouse for Windows

Large File Support (Windows)

Direct and sequential files, non-indexed subfiles, and portable subfiles can now exceed the two gigabyte limit in total number of bytes. The file must be recreated to be able to grow beyond two gigabytes. An existing file cannot grow beyond the two gigabyte limit. The file system must be able to handle large files and must be configured to allow them.

PowerHouse 4GL can still only process up to 2,147,483,647 records.

SET JOB Available in QUIZ and QTP

SET JOB functionality is now available in QUIZ and QTP in PowerHouse 4GL 8.41D1. Previously QUIZ accepted the syntax but no job was submitted. QTP would not accept the syntax.

The job is submitted as a separate spawned process just before the product exits. By default the spawned process window is hidden. To show the window, use the **setjobshow** program parameter. The default is **nosetjobshow**. There is also a resource file statement SETJOBSHOW|NOSETJOBSHOW.

SET REPORT DESTINATION and PH_PRINTER

You can specify the destination printer as a string in

```
SET REPORT DEVICE PRINTER string
```

Now, you can also use the PH_PRINTER environment variable and the DESTINATION option in

```
SET REPORT DESTINATION printer
```

Additional New Features and Changes

REPORT and RUN Statements

The REPORT and RUN statements and the equivalent RUN REPORT and RUN RUN verbs are available in QDESIGN. These statements and verbs execute QUIZ or QTP respectively and take the compiled report or run as a parameter. The syntax is:

```
REPORT|RUN filespec
and
```

Time Rounding in the DATEEXTRACT Function

Due to the use of floating point when extracting portions of a datetime, in Series 8 versions prior to 8.4xD1, using the DATEEXTRACT might result in incorrect values. It was possible to get hours, minutes, and seconds values greater than 60. The rounding errors have been corrected.

When DATEEXTRACT was used to extract the TIME, 9 digits were returned representing hours, minutes, seconds, tenths of a second, hundredths of a second, and thousands of a second. In fact, the thousands of a second was a meaningless number because it was greater than the precision of the floating point value. Therefore, the thousandths of a second will no longer be returned by DATEEXTRACT. Rather, a zero will always be returned so that the TIME will continue to return 9 digits.