Packages on DB2 UDB

DataDirect Connect® Series ODBC Drivers

Introduction

The concept of "packages" in the DB2 Universal Database (UDB) system is often not well understood. This paper provides both general information about what packages are and how they are created and bound, as well as specific information about both the DataDirect Connect® *for* ODBC (32-bit) and DataDirect Connect64® *for* ODBC (64-bit) drivers. DataDirect Connect *for* ODBC and DataDirect Connect64 *for* ODBC each include a DB2 Wire Protocol ODBC driver that connects to DB2 UDB on Windows, UNIX/Linux, and mainframe servers without the use of DB2 UDB client software. The drivers automatically create and bind packages on DB2 UDB and also allow the user to modify these packages.

In this paper, the term "SQL Request" is used as a generic term referring to all application requests associated with a SQL statement. Internally, DB2 UDB processing actually works in terms of SQL requests rather than SQL statements.

Packages and Sections

When DB2 UDB processes SQL statements, each individual SQL request is mapped to a specific package. For a SQL statement to be processed, it must be associated with a DB2 UDB package. This is the case for application requests such as PREPARE, FETCH, and CLOSE. A more precise statement of how DB2 UDB processes SQL requests is to say that each individual request is mapped to a specific section entry within a specific package within DB2 UDB.

What is a package?

A package is a cluster of information that controls the compilation of any static SQL statement, partially controls the compilation of any dynamic SQL statement, and influences the execution of any SQL request issued within its scope. Package information includes items such as the optimization level used during compilation, whether blocking is used for eligible cursors during execution, and what degree of parallelism is used during execution. All of this information is stored as an entry representing the package in the SYSCAT.PACKAGES catalog table. For static SQL statements, a package also has a section associated with it for each statement. A section is the DB2 UDB executable version of a SQL statement. A package is used by DB2

UDB as a basic authorization control point. Privileges can be granted and revoked on the package as required to permit people to execute or maintain the package. This information is reflected in SYSCAT.PACKAGEAUTH.

Why are packages necessary?

Packages *must exist* for the user to be able *to execute SQL statements* against DB2 UDB.

An application can be written using pure static SQL, a mix of static and dynamic SQL, or pure dynamic SQL. All of these approaches are supported within DB2 UDB through the use of packages.

Static SQL statements are pre-compiled and have a pre-existing access path and therefore do not have to be compiled by the DB2 SQL compiler at run time. They are executed under the authorization of the user who bound the package rather than under that of the run-time user.

Dynamic SQL statements are compiled at run time, under the authorization of the run-time user, and must be used when all or part of the SQL statement is generated during application run time. This is the most common case with ODBC applications. Dynamic SQL statements incur a higher startup cost, but do not have to be recompiled when they are reused. They also use a more optimized access path based on current database statistics.

What is a section entry?

Because an application can have many different SQL statements of both a static and dynamic nature, so can a package. To keep the context of the individual SQL statements clear and to avoid having to provide this information with each request, DB2 UDB subdivides a package into smaller units called section entries. A section entry contains information about the SQL statement itself (if any exists) and about the context in which the SQL statement was found in the application. For example, for a cursor, the section entry contains the name of the cursor, whether it is a WITH HOLD cursor or not, and whether the cursor was defined FOR UPDATE. During execution, a section entry contains information about the current status of any section associated with it. It also performs the same function for the state of any associated application cursors. For dynamic SQL statements, the section entry stored with a package is empty and simply acts as a "bookmark."

There is one unique section entry for each unique PREPARE, DECLARE CURSORS, or static SQL found in an application. There is also one unique entry added when an EXECUTE IMMEDIATE request is found in an application; this entry is shared by all other EXECUTE IMMEDIATE requests within the same application.

What is a section?

A section is the actual executable embodiment of a SQL statement. It contains the logic and data access methods required by DB2 UDB to produce the specified results. A section consists of a series of operators and any associated operands that outline the execution order and optimum operation of the data access. The operators correspond to low-level DB2 UDB functions that access and manipulate the data. The operands represent data elements (for example, rows, tables, and indexes) and control structures. A section is the end result of the compilation of a SQL statement. The SQL Compiler determines the most efficient approach for satisfying the SQL statement and produces a section to implement this plan.

A section is an efficient way to express the logic needed for a SQL statement because it does so directly in terms of specific DB2 UDB internal functions. The contents of a section deal with the bare essentials and the physical realities of the storage mechanisms used for the data. By removing the levels of abstraction provided by SQL, a section can ensure the best performance during the execution of the statement. A SQL statement deals with a specific result set or target set; a set can consist of any number of rows and is treated as a whole by SQL. That is, SQL only recognizes sets of data, not individual rows. Because a section is physically accessing the data as individual rows, it has control at that level. Each step in a section is based on what to do with the current row. The section returns data to the application on a row-by-row basis. Finally, by stripping away the abstraction of the SQL statement, a section allows the SQL statement and its result set to be represented by many different SQL methods getting that result set. Each individual section is a product of the environment in which it is compiled. This flexibility is valuable when supporting static and dynamic SQL created at different times in different contexts.

How is a package created?

There are two basic steps in the creation of a DB2 UDB package: precompilation and binding. Although these two steps are sometimes combined into one from the user's perspective, they are actually two distinctly different steps from DB2 UDB's point of view.

Precompilation occurs when application source files containing SQL statements and host variables are submitted to a precompiler. A precompiler is specific to a programming language. Its job is to parse the application source to find any SQL statements and host variables, remove them, and then replace them in the source file with function calls and variables suitable to the programming language being used.

The resultant calls are to DB2 UDB client functions that communicate with DB2 UDB during the execution of the application. The end result of precompilation is a modified version of the source file and the extracted SQL

statements and variables. The latter can be stored in a file, called a bindfile (.bnd), or directly submitted to DB2 UDB as part of the bind step.

Binding is the step where the SQL information extracted from an application source file is analyzed and stored in the DB2 UDB catalog tables. This information can come directly from the precompilation step or from a bindfile. Information about the package, section entries, and host variables are stored directly in the SYSCAT.PACKAGES catalog table. Static SQL statements are passed through the SQL compiler to have sections generated for them; they are then stored in the SYSCAT.STATEMENTS catalog table. The generated sections for static SQL statements are stored in the SYSIBM.SYSSECTION table. The end result of a successful bind is a DB2 UDB package.

How is a section created?

The process of creating a section from a SQL statement is referred to as compilation. It can also be referred to as optimization or preparing. Compilation of SQL statements within DB2 UDB is performed by the SQL Compiler. The processes explained here apply to static and dynamic SQL statements. The differences between these two processes arise when compilation occurs and the values are used for the compilation environment. Static SQL bases its compilation environment purely on the package information. Dynamic SQL bases its compilation environment on some of the package information but also on the current values of a number of special registers such as CURRENT DEGREE and CURRENT SCHEMA.

The first step in the compilation of a SQL statement is the parsing stage. In this stage, the syntax of the SQL statement is validated and the statement is broken down into its component pieces. Using the information from the catalog tables for any referenced data objects or functions, an internal representation of the statement is constructed in the form of a graph. The representation is referred to as a Query Graph Model or QGM. QGM provides a concise and flexible representation of a statement.

The QGM representation is the basic information structure used by the compiler as it studies and processes the statement. Once the graph representing the basic statement structure has been constructed, it is passed to the next stage of the processing. This stage is referred to as Semantics; it is responsible for supplementing the basic QGM graph with the additional information required for items like referenced views, triggers activated by the statement, and constraints affecting the statement. The QGM graph is extended and modified to account for the additional levels of information brought into play by these entities.

Once the full scope of the statement is understood by the compiler and represented by the QGM graph, the next stage of processing occurs. This stage is referred to as Query Rewrite, or Rewrite. It evaluates the input graph and rewrites the input QGM graph into a version that provides the maximum amount of flexibility to the next stage - the Optimizer. The underlying principle behind the Rewrite component is simply that there are a number of

different ways in SQL of expressing the same result. The form of some of these SQL statements forces certain choices on the compiler, while others do not; thus, it follows that there can be a number of different graphs for representing the same result set and that by transforming a graph from one version to another, more options are made available for potentially better decisions and more efficient access path decisions.

Rewrite is followed by Optimization, the most important stage of compilation. It is within the optimization stage that the QGM graph is analyzed and all possible methods of accessing the data are evaluated and their costs estimated. The Optimizer is a cost-based decision maker that uses complex mathematical models of the varying costs for data access and manipulation to refine and select the most efficient access plan to satisfy the original SQL statement.

This last stage is the code generation, or Codegen, phase where the actual section is produced. The Codegen stage translates the "theoretical" access plan selected by the Optimizer into a "practical" access plan as embodied by the section. This section is then returned to the requester for execution.

The DB2 Wire Protocol Driver and Packages

The DB2 Wire Protocol driver does not work properly unless packages exist on every server to which you intend to connect. The driver can, however, create and bind these packages.

IMPORTANT: You must have the appropriate privileges for the driver to create and bind packages with your user ID. These privileges are BINDADD for binding packages and GRANT for executing the packages. These are typically the permissions of a Database Administrator (DBA). If you do not have these privileges, someone that has a user ID with DBA privileges needs to create packages by connecting with the driver.

When connecting for the first time, the driver determines whether or not packages exist on the server. If packages do not exist, the driver creates them automatically using driver data source default values.

NOTE: The initial driver connection to a particular server may take a few minutes because of the number and size of the packages that must be created on the server. Subsequent connections do not incur this delay.

By default, the packages contain 200 dynamic sections and are created in the collection named NULLID. You can override the default number of dynamic sections through the Dynamic Sections option on the Modify Bindings tab of the driver Setup dialog box or by setting the DynamicSections connection string attribute. Similarly, you can override the collection in which the packages are created through the Package Collection option on the Modify Bindings tab of the driver Setup dialog box or by setting the PackageCollection attribute.

If you change default values in a data source before connecting with the driver for the first time, the new defaults are used when creating the packages. If you want to change these values after the packages have been created, you can re-create the packages with the new values by one of the following methods:

Windows

On Windows, the driver allows you to create or modify packages from the Modify Bindings tab.

UNIX and Linux

On UNIX and Linux, the driver allows you to create or modify packages through a special bind utility. Depending on the platform of the DB2 server, the minimum attribute values that must be set in the data source to bind packages are:

Linux/UNIX/Windows DB2 Servers

IpAddress, Database, TcpPort

z/OS and iSeries DB2 Servers

IpAddress, Location, TcpPort

There are also other attribute values that affect binding. See the appropriate product User's Guide and Reference for details about configuring data sources for the DB2 Wire Protocol driver on UNIX and Linux. These books are available at:

http://www.datadirect.com/download/docs/dochome/index.ssp

The bind utility is located in the /bin directory of the product installation directory. After specifying the appropriate connection string attribute values in the data source, create or modify packages by entering the command:

`bind`*`xx dsn`*

where *`xx`* is the driver level number in the driver file name and *`dsn`* is the ODBC data source name in the system information file. You are prompted for a user ID and password if they are not stored in the data source. If packages are created and bound successfully, a message indicating success appears. If there are problems connecting or creating the packages, an appropriate error message appears.

Summary

Packages provide flexibility for processing both static and dynamic SQL on DB2 UDB. There are two basic steps in the creation of a DB2 UDB package: precompilation and binding.

To keep the context of individual SQL statements clear, a package is subdivided into section entries. A section entry contains information about the SQL statement itself and about the context in which the SQL statement was found in the application. A section itself is the actual executable embodiment of a SQL statement. It contains the logic and data access methods required by DB2 UDB to produce the specified results.

The DB2 Wire Protocol driver can create and bind packages, and does so automatically when it first connects to DB2 UDB. You can change the defaults used for this initial package creation or you can modify the package bindings later. This is done through the Modify Bindings tab of the driver Setup dialog box on Windows or through the bind utility on UNIX and Linux.

Reference:

Inside DB2 Universal Database: The Life and Times of an SQL Statement, Paul Bird, The IDUG Solutions Journal, Fall 1999 - Volume 6, Number 3.

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

DataDirect Technologies is focused on standards-based data connectivity, enabling software developers to quickly develop and deploy business applications across all major databases and platforms. DataDirect Technologies offers the most comprehensive, proven line of data connectivity components available anywhere. Developers worldwide at more than 250 leading independent software vendors and thousands of corporate IT departments rely 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. Developers also depend on DataDirect to radically simplify complex data integration projects using XML products based on the emerging XQuery and XQJ standards. DataDirect Technologies is an operating company of Progress Software Corporation (Nasdaq: PRGS), a US$300+ million global software industry leader. Headquartered in Bedford, Mass., DataDirect Technologies can be reached on the Web at http://www.datadirect.com or by phone at +1-800-876-3101.