

Chapter 2

CODING A DB2 APPLICATION IN JAVA

*Get on the
Fast Track!*



TM

**SYS-ED/
Computer
Education
Techniques, Inc.**

Objectives

You will learn:

- Coding a DB2 application in Java.
- Connection Resource Management in Java.
- Java class libraries.
- Java packages.
- Supported SQL data types in Java.
- Coding JDBC applications and applets.
- SQLj programming.
- DB2 support for SQLj.

1.1 Programming Considerations for Java

DB2 Universal Database implements two standards-based Java programming APIs:

- Java Database Connectivity (JDBC)
- Embedded SQL for Java (SQLj).

Comparison of SQLj to JDBC

The JDBC API allows Java programs to be written that make dynamic SQL calls to databases. SQLj applications use JDBC as a foundation for such tasks as connecting to databases and handling SQL errors. They can also contain embedded static SQL statements in the SQLj source files.

An SQLj source file with the SQLj translator must be translated before compiling the resulting Java source code.

JDBC and SQLj Interoperability

The SQLj language provides direct support for static SQL operations that are known at the time the program is written. If some or all of a particular SQL statement cannot be determined until run time, it is a dynamic operation.

To perform dynamic SQL operations from a SQLj program, use JDBC.

A ConnectionContext object contains a JDBC Connection object, which can be used to create JDBC Statement objects needed for dynamic SQL operations.

Every SQLj ConnectionContext class includes a constructor that takes as an argument a JDBC Connection. This constructor is used to create an SQLj connection context instance that shares its underlying database connection with that of the JDBC connection.

1.2 Advantages of Java Over Other Languages

Programming languages containing embedded SQL are called host languages. Java differs from the traditional host languages C, COBOL, and FORTRAN, in ways that significantly affect how it embeds SQL.

SQLj and JDBC are open standards, allowing for SQLj or JDBC applications to be easily ported from other standards-compliant database systems to DB2 Universal Database.

All Java types representing composite data, and data of varying sizes, have a distinguishing value, null, which can be used to represent the SQL NULL state, giving Java programs an alternative to NULL indicators that are a fixture of other host languages.

Java is designed to support programs that are automatically heterogeneously portable.

Java is designed for binary portability in heterogeneous networks, which promises to enable binary portability for database applications that use static SQL.

1.3 Connection Resource Management in Java

Calling the `close()` method of a connection context instance causes the associated JDBC connection instance and the underlying database connection to be closed.

Connection contexts may share the underlying database connection with other connection contexts and/or JDBC connections, it may not be desirable to close the underlying database connection when a connection context is closed.

A programmer may want to release the resources maintained by the connection context, without actually closing the underlying database connection.

Connection context classes also support a `close()` method, which takes a Boolean argument indicating whether or not to close the underlying database connection.

- the constant `CLOSE_CONNECTION` if the database connection should be closed.

and

- `KEEP_CONNECTION` if it should be retained.

The variant of `close()` that takes no arguments is a shorthand for calling `close(CLOSE_CONNECTION)`.

If a connection context instance is not explicitly closed before it is garbage collected, `close(KEEP_CONNECTION)` is called by the `finalize` method of the connection context. This allows connection-related resources to be reclaimed by the normal garbage collection process while maintaining the underlying database connection for other JDBC and SQLj objects that may be using it.

If no other JDBC or SQLj objects are using the connection, the database connection is closed and reclaimed by the garbage collection process. Both SQLj connection context objects and JDBC connection objects respond to the `close()` method.

When writing an SQLj program, it is sufficient to call the `close()` method on only the connection context object: closing the connection context also closes the JDBC connection associated with it.

However, it is not sufficient to close only the JDBC connection returned by the `getConnection()` method of a connection context: the `close()` method of a JDBC connection does not cause the containing connection context to be closed, and therefore resources maintained by the connection context are not released until it is garbage collected.

The `isClosed()` method of a connection context returns true if any variant of the `close()` method has been called on the connection context instance. If `isClosed()` returns true, calling `close()` has no effect, and calling any other method is undefined.

1.4 Source and Output Files for Java

The extensions for the source files are:

java	Java source files, which require no precompiling. These files can be compiled with the javac Java compiler included with the Java development environment.
Sqlj	SQLj source files, which require translation with the sqlj translator. The translator creates: one or more .class bytecode files and one .ser profile file per connection context

The corresponding output files have the following extensions:

.class	JDBC and SQLj bytecode compiled files.
.ser	SQLj serialized profile files. Packages are created in the database for each profile file with the db2prof utility.

1.5 Java Class Libraries

DB2 Universal Database provides class libraries for JDBC and SQLj support.

These libraries must be provided in the CLASSPATH or included with the applets.

db2jcc.jar	Provides the JDBC Type 4 driver.
db2java.zip	Provides the JDBC driver and JDBC and SQLj support classes, including stored procedure and UDF support.
sqlj.zip	Provides the SQLj translator class files.
runtime.zip	Provides Java run-time support for SQLj applications and applets.

1.6 Java Packages

In order to use the class libraries included with DB2 in applications, the appropriate import package statements are required to be included at the top of your source files.

These packages can be used in Java applications:

java.sql.*	JDBC API is included in the JDK. This package must be imported into every JDBC and SQLj program.
sqlj.runtime.*	SQLj support is included with every DB2 client. This package must be imported in every SQLj program.
sqlj.runtime.ref.*	SQLj support included with every DB2 client. This package must be imported into every SQLj program.

1.7 Supported SQL Data Types in Java

This Java equivalent of each SQL data type, based on the JDBC specification for data type mappings is:

SQL Column Type	Java Data	Type SQL Column	Type Description
SMALLINT	short	16-bit, signed integer	(500 or 501)
INTEGER	int	32-bit, signed integer	(496 or 497)
BIGINT	long	64-bit, signed integer	(492 or 493)
REAL	float	Single precision floating point	(480 or 481)
DOUBLE	double	Double precision floating point	(480 or 481)
DECIMAL(p,s)	java.math.BigDecimal	Packed decimal	(484 or 485)
CHAR(n)	java.lang.String	Fixed-length character string of length n where n is from 1 to 254	(452 or 453)
CHAR(n) FOR BIT DATA	byte[]	Fixed-length character string of length n where n is from 1 to 254	
VARCHAR(n)	java.lang.String	Variable-length character string	(448 or 449)
VARCHAR(n) FOR BIT DATA	byte[]	Variable-length character string	
LONG VARCHAR	java.lang.String	Long variable-length character string	(456 or 457)
LONG VARCHAR FOR BIT DATA	byte[]	Long variable-length character string	
BLOB(n)	java.sql.Blob	Large object variable-length binary string	(404 or 405)
CLOB(n)	java.sql.Clob	Large object variable-length character string	(408 or 409)
DBCLOB(n)	java.sql.Clob	Large object variable-length double-byte character string	(412 or 413)
DATE	java.sql.Date	10-byte character string	(384 or 385)
TIME	java.sql.Time	8-byte character string	(388 or 389)
TIMESTAMP	java.sql.Timestamp	26-byte character string	(392 or 393)

1.8 Coding JDBC Applications and Applets

When coding a JDBC application or applet, the following tasks will need to be implemented:

1. Import the appropriate Java packages and classes (java.sql.*).
2. Load the appropriate JDBC driver:
 - For type 2 JDBC, COM.ibm.db2.jdbc.app.DB2Driver for applications.
 - For type 3 JDBC, COM.ibm.db2.jdbc.net.DB2Driver for applets.
 - The type 3 driver is deprecated in Version 8.
 - For type 4 JDBC, com.ibm.db2.jcc.DB2Driver for both applications and applets.
3. Connect to the database, specifying the location with a URL as defined in the JDBC specification and using the db2 subprotocol.
4. Pass SQL statements to the database.
5. Receive the results.
6. Close the connection.

After coding a program, it can be compiled the same way as any other Java program.

It will not be necessary to perform any special precompile or bind steps.

1.9 Example: JDBC Program

Every JDBC program must perform the following steps:

1. Import the JDBC package.
Every JDBC and SQLj program must import the JDBC package.
2. Declare a Connection object.
The Connection object establishes and manages the database connection.
3. Set the database URL variable.
The DB2 application driver accepts URLs that take the form of:

```
jdbc:db2:database-name
```

4. Connect to the database.
The DriverManager.getConnection() method is most often used with the following parameters:

```
getConnection(String url)
```

This parameter establishes a connection to the database specified by a URL, and uses the default user ID and password.

getConnection(String url, String userid, String password) will establish a connection to the database specified by a URL, and uses the user ID and password that are specified by userid and password respectively.

All JDBC sample programs use the Db class defined in the Util.java program to perform the connection.

INSERT Statement

```
Statement stmt = con.createStatement();
stmt.executeUpdate( "INSERT INTO staff(id, name, dept, job, salary) " +
" VALUES (380, 'Pearce', 38, 'Clerk', 13217.50), "+
" (390, 'Hachey', 38, 'Mgr', 21270.00), " +
" (400, 'Wagland', 38, 'Clerk', 14575.00) ");
stmt.close();
```

UPDATE Statement

```
Statement stmt = con.createStatement();
stmt.executeUpdate(
"UPDATE staff " +
" SET salary = salary + 1000 " +
" WHERE id >= 310 AND dept = 84");
stmt.close();
```

DELETE Statement

```
Statement stmt = con.createStatement();
stmt.executeUpdate("DELETE FROM staff " +
" WHERE id >= 310 AND salary > 20000");
stmt.close();
```

1.10 SQLj Programming

DB2 SQLj support is based on the SQLj ANSI standard.

These SQL constructs can be used in SQLj programs:

SQL Construct	Example(s)
Queries	SELECT statements and expressions
SQL Data Change Statements (DML)	INSERT, UPDATE, DELETE
Data Statements	FETCH, SELECT..INTO
Transaction Control	COMMIT, ROLLBACK, etc.
Data Definition Language	CREATE, DROP, ALTER
Calls to stored procedures	CALL MYPROC(:x, :y, :z)
Invocations of functions	VALUES(MYFUN(:x))

1.11 DB2 Support for SQLj

DB2 SQLj support is provided by the DB2 Application Development Client.

IN addition to the JDBC support provided by the DB2 client, DB2 SQLj support provides for the creation, building, and running of:

- Embedded SQL for Java applications.
- Applets.
- Stored procedures.
- User-defined functions (UDFs).

These contain static SQL and use embedded SQL statements that are bound to a DB2 database.

The SQLj support provided by the DB2 Application Development Client includes:

Support	Purpose
SQLj translator, sqlj	Replaces embedded SQL statements in the SQLj program with Java source statements and generates a serialized profile containing information about the SQL operations found in the SQLj program. The SQLj translator uses the sqllib/java/sqlj.zip file.
SQLj run-time classes	These are available in sqllib/java/runtime.zip.
DB2 SQLj Profile Customizer,	The db2profc, precompiles the SQL statements stored in the generated profile and generates a package in the DB2 database.
DB2 SQLj Profile Printer	The db2profp prints the contents of a DB2 customized profile in plain text.
SQLj Profile Auditor Installer	The profdb installs or uninstalls debugging class-auditors into an existing set of binary profiles. Once installed, all RTStatement and RResultSet calls made during application run time are logged to a file (or standard output), which can then be inspected to verify expected behavior and trace errors. Only those calls made to the underlying RTStatement and RResultSet call interface at run time are audited.
The SQLj Profile Conversion Tool	The profconv, converts a serialized profile instance to class bytecode format. Some browsers do not yet have support for loading a serialized object from a resource file associated with the applet. This utility needs to be run in order to perform the conversion.