[vertical list of authors]

[cover art/text goes here]

[TBD: check cover margin spec]

[TBD: this page intentionally left blank.]

Contents

Introducing Derby

Welcome to Derby! Derby is a Java- and SQL-based relational database management system (RDBMS). This chapter describes the Derby products.

Deployment options

Derby can be deployed in a number of different ways:

- Embedded in a single-user Java application. Derby can be practically invisible to the user, since it requires no administration and runs in the same Java virtual machine (JVM) as the application.
- Embedded in a multi-user application such as a web server, an application server, or a shared development environment.
- Embedded in a server framework. You can use the Network Server with the IBM DB2 Universal JDBC driver (see (gslib56653.dita#gslib56653)) or a server of your own choice.

System requirements

Derby is a database engine written completely in Java; it will run in any certified Java Virtual Machine (JVM).

Installing and working with Derby

If you are new to Derby and JDBC programming, the following topics will help you begin using the product. If you are already an experienced JDBC programmer, see (gsquck19524.dita#gsquck19524) .

Installing Derby

You must first download the Derby zip or tar archive from the Derby web site in order to follow these instructions.

Extract the file. The contents of the distribution will be extracted into a subdirectory named `apache-derby-10.0` and it will contain two subdirectories:

- The `lib` subdirectory contains the Derby jar files.
- The `javadoc` subdirectory contains the `api` documentation that was generated from source code comments.

Setting up your Java environment

You need to set your PATH environment variable so the JVM and Java applications run correctly. The PATH variable enables your operating system to find the appropriate programs from any directory. If you have more than one JVM installed, the JVM you wish to use must appear before any of the others in the PATH variable.
Note: This is one of the basic steps in running a JVM. It is extremely important for the success of your installation.

- To set the PATH environment variable, add the *bin* subdirectory of the JVM directory to the beginning of the path. For example, if the directory is *C:\JDK1.4*, add *C:\JDK1.4\bin* to the beginning of the path.
- To make sure you set your path correctly, type the following command in a command window:**`java -version`**

If the path is set correctly, you will see a printout telling you the version of your JVM.

Using the tools and startup utilities

Derby tools and utilities include dblook, ij, sysinfo, and the import, export and database class utilities. The */frameworks/embedded/bin* directory contains scripts for running some of the Derby tools and utilities in embedded mode. You can find similar scripts for running tools and utilities for the Network Server in `/frameworks/NetworkServer/bin`. The scripts have self-explanatory names such as *sysinfo.bat* or *ij.ksh*. Like the class path scripts, these scripts end with different extensions depending on your environment.

Windows: *.bat*

UNIX: *ksh*
Note: These scripts serve as examples to help users on all platforms get started with these tools and utilities. However, they might require modification in order to run properly even on Windows or UNIX platforms.

Add the /bin directory to your path to use shortened commands to start the Derby tools.
Note: Users on UNIX platforms need to turn on the execute bit for these files. For example:

```
chmod
+x fileName
```

For reference on these scripts, see (gslib27507.dita#gslib27507) .

Using sysinfo

You can use Derby's `sysinfo` tool to check the versions of Derby products. Once you have the bin directory in your PATH, you can run sysinfo by typing the following command in a command window:

```
sysinfo
```

The sysinfo script sets the appropriate environment variables, including class path, and executes the sysinfo program.

Running ij

You can use the `ij` tool to connect to a Derby database. Once you have the bin directory in your PATH, you can execute the ij command by typing the following:

```
ij
```

The ij script sets up environment variables like class path and executes the ij program.

To create a database with ij, type the following command:

```
ij>
connect 'jdbc:derby:testdb;create=true';
```

This command creates the database called testdb in the current directory and populates the system tables. You can then execute any SQL statements from the ij command line. When you are ready to leave ij, type:

```
ij>
exit;
```

See the *Derby Tools and Utilities Guide* for more information on running ij.

Manually setting environment variables and paths

If you cannot run the scripts for the Derby tools and utilities, you must complete certain steps manually. The following topics show how to manually set your environment and run the tools manually.

Set the DERBY_INSTALL environment variable

During installation, you chose a base directory where the software was installed; we recommended that you name it *Derby_10*. This document refers to that directory as the Derby base directory.

If you do not plan to use the scripts in the `frameworks/embedded/bin` directory and your operating system supports it, create an environment variable called DERBY_INSTALL and set its value to the path of the Derby base directory. For example, if you installed the product in c:\Derby_10, set DERBY_INSTALL to c:\Derby_10:

```
set DERBY_INSTALL=c:\Derby_10
```

Set the class path

The JVM (compiler or interpreter) needs to know the path (operating system instructions about location) of every class file needed by your application. The class path is a list of the class libraries needed by the JVM and other Java applications in order to run your program.

There are two ways to set the class path. You can set the operating system's CLASSPATH environment variable permanently or temporarily. If you set the environment variable temporarily, you must set it each time you open a new command window. Alternatively, you can set the class path with the runtime option, which means specifying the class path at the time you start your Java application and the JVM.

In most development environments, it works best to set the operating system's CLASSPATH environment variable temporarily. Derby provides some scripts to help you set your class path this way; they are found in the`frameworks/embedded/bin` directory or the `frameworks/NetworkServer/bin` directory. Run a script every time you open a new command window.

For example, you can set your class path as follows:

```
set CLASSPATH=%DERBY_INSTALL%\lib\derby.jar;
%DERBY_INSTALL%\lib\derbytools.jar;%CLASSPATH%
```

To manually run the sysinfo utility, type the following in a command window or shell:

```
java org.apache.derby.tools.sysinfo
```

To manually run the ij utility, type the following in a command window or shell:

```
java org.apache.derby.tools.ij
ij> connect 'jdbc:derby:testdb;create=true';
```

When you are finished running the ij utility, type:

```
ij>
exit;
```

For more information on running the ij and sysinfo utilities, see the *Derby Tools and Utilities Guide*.

Start programming with examples

You can use the examples that come with Derby to begin programming.

Simple example

The simple example illustrates basic tasks such as:

- Starting Derby, loading the Derby JDBC driver
- Running in an embedded or a client/server environment (Network Server)
- Establishing a connection
- Turning off auto-commit
- Creating a table
- Inserting and selecting data
- Disconnecting
- Shutting down a Derby system

You can find the simple example in the */demo/programs/simple* directory. Open the `example.html` file. The `example.html` file explains how to run the simple example.

You can run the simple example in both the embedded server and the Network Server environments to familiarize yourself with the different configurations of Derby.

Derby Network Server examples

The Derby Network Server examples are included with the Derby demo programs. The following examples programs demonstrate how to obtain an embedded connection and client connections using the Network Server to connect to the same database:

- *Simple Network Server Sample*

  This example uses two programs to illustrate how a normal client program that starts up in its own JVM can connect to the Network Server that the server program starts. The client program (SimpleNetworkClientSample) and the server program (SimpleNetworkServerSample) each run in their own JVMs. This example shows the Derby jar files that are needed at the client side and server side to use the Network Server. The *SimpleNetworkClientSample* program also shows how to use either the DriverManger or a DataSource to obtain client connections.
- *Network Server Demo*

  This example program (*NsSample*) starts the Network Server and shows how to obtain client and embedded connections using the Network Server to connect to the same database, all in one JVM.

The *Simple Network Server Sample* and *Network Server Demo* programs are located in the `demo/programs/nserverdemo/` directory.

Quick start for experienced JDBC users

This chapter is for experienced JDBC programmers who already know the basics about how to set class path, how to run a Java program, and how to use a JDBC driver.
Tip: You'll find more help on similar topics in the first chapters of the *Derby Developer's Guide* and the *Derby Tools and Utilities Guide*.

Environment quick start

Before you configure your system for running Derby, it is useful to understand something about the different environments in which Derby can run, because these environments affect the class path, driver name, and database connection URL. See the *Derby Developer's Guide* for more information on Derby environments.

Embedded environment

When an application starts up an instance of Derby within its JVM, the application is said to run in an *embedded environment*. In this environment, only a single application can access a database at one time, and no network access occurs. Loading the embedded driver starts Derby.

Client/server environment

When multiple applications connect to Derby over the network, they are running in a client/server environment. Derby runs embedded in a server framework that allows multiple network connections. (The framework itself starts up an instance of Derby and is running in an embedded environment. However, the client applications are not in the embedded environment. See the *Derby Server and Administration Guide* for more information on how to run Derby on a server.)

It is also possible to embed Derby in any Java server framework.

Libraries and class path

See (gslib19524.dita#gslib19524) for complete reference.

Driver

The following drivers are available depending on the environment you choose for Derby:

- *org.apache.derby.jdbc.EmbeddedDriver*

  For embedded environments, when Derby runs in the same JVM as the application.

- *com.ibm.db2.jcc.DB2Driver*

  For the Network Server environment.

Database connection URL

For the Derby-provided driver listed above, here is the format for the database connection URL for connecting to an existing database:

```
jdbc:derby:databaseName;URLAttributes
```

The italicized items stand for something the user fills in:

- *databaseName*

The name of the database you want to connect to

- *URLAttributes*

  One or more of the supported attributes of the database connection URL, such as ;*locale=ll_CC* or ;*create=true*. For more information, see the *Derby Developer's Guide*.

The Derby documents

Derby comes with a complete set of documentation describing Derby concepts and tasks, and includes reference information.

The Derby library

- *Derby Developer's Guide*

  Describes the functionality and features of Derby common to all deployments, such as Derby's JDBC and SQL specifics, deploying Derby applications, security, and other advanced features.
- *Derby Reference Manual*

  A reference for the core SQL language, Derby's Java-enhanced dialect of SQL. Also provides reference information for Derby's JDBC and JTA implementations, keywords, system tables, properties, and *SQLExceptions*.
- *Derby Tuning Guide*

  Explains how to configure and tune Derby through properties and provides reference information on properties. It also offers performance tips, an in-depth discussion of performance, and information about the Derby optimizer.
- *Derby Tools and Utilities Guide*

  A guide for working with the Derby tools such as `ij`, and more advanced utilities such as import/export and the database class loader.
- *Derby Server and Administration Guide*

  Part One of this guide discusses configuration of servers, how to program client applications, and database administration.

  In addition, some systems might require administrative tasks such as backing up databases. These tasks are independent of any server framework but are unique to multi-user or large systems.

  Part Two of this guide discusses administrative issues such as backups and debugging deadlocks.
- Derby API javadoc

  Automatically generated for all public Derby classes. (No javadoc is provided for the JDBC API, which is part of the Java™ 2 Platform, Standard Edition) For more information about the classes in the API, see the *Derby Reference Manual*.

Documentation conventions

Terminology, syntax, and typographical conventions of the Derby documentation.

Terminology

The Derby documentation uses some specialized vocabulary. Here are some definitions that will help you understand Derby:

environment
How your application interacts with Derby. Sometimes referred to as a framework. The two environments are *embedded environment (#docs99397/docs35643)* and *client/server environment (#docs99397/docs13780)* .

embedded environment
When an application starts up an instance of Derby within its JVM, the application is said to run in an embedded environment. In this environment, only a single application can connect to a database at one time, and no network access occurs.

client/server environment
When multiple applications connect to Derby over the network, they are said to run in a client/server environment. Derby runs embedded in a server or connectivity framework that allows multiple network connections. (The framework itself starts up an instance of Derby and, strictly speaking, *it* is running in an embedded environment; the client applications, however, are not.)

Syntax

SQL syntax is presented in modified BNF notation. The meta-symbols of BNF are:

Symbol	Meaning
|	“or.” Choose one of the items
`[ ]`	Enclose optional items.
`*`	Flags items that you can repeat 0 or more times. Has a special meaning in some SQL statements.
`{ }`	Groups items so that they can be marked with one of the other symbols, i.e. [ ], |, or *.
`( ) . ,`	Other punctuation that is part of the syntax.

An example of how SQL syntax is presented:

```
CREATE [ UNIQUE ] INDEX IndexName     ON TableName ( SimpleColumnName [ ,
SimpleColumnName ]
* )
```

Command-line syntax for running Java programs and utilities (as well as examples) always begins with the word *java*:

```
java org.apache.derby.tools.ij
```

In addition, this documentation uses the the IBM Software Development Kit style for setting JVM arguments and properties. If you use another Java Virtual Machine, the way you set JVM arguments and properties might be different.

Typography

This documentation uses some typographical conventions to highlight elements of the SQL language, operating system commands, and the Java programming language.

Typeface	Usage	Examples
Italic	New terms	defined by *keys*
	File and directory names	*C:\derby*
	Dictionary objects	The *Employees* table
	In syntax, items that you do not type exactly as they appear, but replace with the appropriate name	**`CREATE TABLE`** ***`tableName`***
Bold, fixed-width	SQL examples	**`SELECT city.getName() FROM Cities`**
	Java application examples	**`Connection conn = DriverManager.getConnection ("jdbc:derby:Sample")`**
Bold, fixed-width	Things you type in a command prompt	**`java org.apache.derby.tools.ij`**
Roman, fixed-width	Comments within examples	`--This line ignored`
All caps	SQL keywords (commands)	You can use a CREATE TABLE statement

Derby Libraries and Scripts: Complete Reference

This appendix describes Derby libraries and scripts.

Libraries Provided by Derby

Library Name and Path	Use
Engine Libraries. You always need this library for embedded environments. For client/server environments, you only need this library on the server.	
derby.jar	For embedded databases
Tools Libraries For embedded environments, you need a library in the class path to use a tool. For a client/server environment, you need a library on the client only.	
derbytools.jar	Required for running all the Derby tools (such as ij, dblook, and import/export).
The Network Server Libraries	
derbynet.jar	Required to start the Derby Network Server.

Libraries not provided by Derby

The Derby Network Server requires a JDBC client that talks DRDA. The IBM DB2 JDBC Universal driver is available for download from IBM developerWorks (http://www.ibm.com/developerworks/cloudscape) and includes the two jar files listed in the table below:

Library name	Use
db2jcc.jar	Required to use the DB2 JDBC Universal Driver. You need to put this jar file in your client class path to run your application.
db2jcc_license_c.jar	Required to use the DB2 JDBC Universal Driver. You need to put this jar file in the client class path to run your application.

In a Java Development Kit Version 1.3 environment, some special Derby features require that you install additional libraries and place them in your class path (Environments using Java Development Kit, Version 1.4 includes all of these libraries):

- LDAP (see the *Derby Developer's Guide*)
- JTA (see the *Derby Reference Manual*)
- JDBC 2.0 Extensions (see the *Derby Reference Manual*)

Scripts provided by Derby

Derby provides scripts in the */bin* directory. Each script comes in two flavors, one ending in *.bat* and one ending in *.ksh*. Here is a complete listing:

- *frameworks/embedded/bin/ij*

Starts ij.

- *frameworks/embedded/bin/setEmbeddedCP*

  Puts all the Derby libraries for an embedded environment in the class path.
- *frameworks/embedded/bin/sysinfo*

  Runs sysinfo.
- *frameworks/embedded/bin/dblook*

  Runs dblook.
- *frameworks/NetworkServer/bin/dblook*

  Runs dblook in a Network Server client context.
- *frameworks/NetworkServer/bin/ij*

  Runs ij in a Network Server client context.
- *frameworks/NetworkServer/bin/NetworkServerControl*

  Runs NetworkServerControl.
- *frameworks/NetworkServer/bin/setNetworkClientCP*

  Puts the libraries needed to connect to Derby Network Server into the class path.
- *frameworks/NetworkServer/bin/setNetworkServerCP*

  Puts the libraries needed for Derby Network Server into the class path.
- *frameworks/NetworkServer/bin/startNetworkServer*

  Starts the Network Server on the local machine.
- *frameworks/NetworkServer/bin/stopNetworkServer*

  Stops the Network Server on the local machine.
- *frameworks/NetworkServer/bin/sysinfo*

  Gets the system information from a running Derby Network Server.