Chapter 13 Advanced Java Database Programming

1. You can set autoCommit to true using the setAutoCommit method in the Connection class.

2. The general information includes the URL, username, product name, product version, driver name, driver version, available functions, available data types, and so on. The methods for general information usually return a string, an integer, except that the method for retrieving available data types returns a ResultSet. Most methods of this type don't have parameters.

3. Information on database capabilities includes such matters as whether the database supports the GROUP BY operator, the ALTER TABLE command with add column option, and entry-level or full ANSI92 SQL grammar. The methods for finding database capabilities return a boolean value indicating whether the database possesses a certain capability. Most of methods of this type don't have parameters and are named with prefix *supports*. Table 13.3 gives some of.

4. The methods for getting database objects shown in Table 13.4 return lists of information in ResultSets. You can use the normal ResultSet methods, such as getString and getInt, to retrieve data from these ResultSets. If a given form of metadata is not available, these methods should throw a SQLException.

5. To create an instance of DatabaseMetaData, use the getDatabaseMetaData() method from a Connection object.

6. The ResultSetMetaData interface describes information pertaining to the result set. A ResultSetMetaData object can be used to find out about the types and properties of the columns in a ResultSet. The methods in ResultSetMetaData have a single int parameter representing the column except that the getColumnCount method has no parameters. All these methods return int, boolean, or String. To create an instance of ResultSetMetaData, use getResultSetMetaData from an instance of ResultSet.

7. Once a connection to a particular database is established, it can be used to send SQL statements from your program to the database. JDBC provides Statement, PreparedStatement, and CallableStatement interfaces to facilitate sending statements to a database for execution and receiving execution results from the database. The Statement interface is used to execute static SQL

statements that contain no parameters. The PreparedStatement, extending Statement, is used to execute a precompiled SQL statement with or without IN parameters. The CallableStatement, extending PreparedStatement, is used to execute a call to a database-stored procedure. To create instances of Statement, PreparedStatement, and CallableStatement, use the methods createStatement, prepareStatement, and prepareCall.