| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Connection.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/sql/Clob.html)   [**NEXT CLASS**](http://docs.google.com/java/sql/DatabaseMetaData.html) | [**FRAMES**](http://docs.google.com/index.html?java/sql/Connection.html)    [**NO FRAMES**](http://docs.google.com/Connection.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#2et92p0) | DETAIL: [FIELD](#3dy6vkm) | CONSTR | [METHOD](#26in1rg) |

## **java.sql**

Interface Connection

**All Superinterfaces:** [Wrapper](http://docs.google.com/java/sql/Wrapper.html)

public interface **Connection**extends [Wrapper](http://docs.google.com/java/sql/Wrapper.html)

A connection (session) with a specific database. SQL statements are executed and results are returned within the context of a connection.

A Connection object's database is able to provide information describing its tables, its supported SQL grammar, its stored procedures, the capabilities of this connection, and so on. This information is obtained with the getMetaData method.

**Note:** When configuring a Connection, JDBC applications should use the appropritate Connection method such as setAutoCommit or setTransactionIsolation. Applications should not invoke SQL commands directly to change the connection's configuration when there is a JDBC method available. By default a Connection object is in auto-commit mode, which means that it automatically commits changes after executing each statement. If auto-commit mode has been disabled, the method commit must be called explicitly in order to commit changes; otherwise, database changes will not be saved.

A new Connection object created using the JDBC 2.1 core API has an initially empty type map associated with it. A user may enter a custom mapping for a UDT in this type map. When a UDT is retrieved from a data source with the method ResultSet.getObject, the getObject method will check the connection's type map to see if there is an entry for that UDT. If so, the getObject method will map the UDT to the class indicated. If there is no entry, the UDT will be mapped using the standard mapping.

A user may create a new type map, which is a java.util.Map object, make an entry in it, and pass it to the java.sql methods that can perform custom mapping. In this case, the method will use the given type map instead of the one associated with the connection.

For example, the following code fragment specifies that the SQL type ATHLETES will be mapped to the class Athletes in the Java programming language. The code fragment retrieves the type map for the Connection object con, inserts the entry into it, and then sets the type map with the new entry as the connection's type map.

java.util.Map map = con.getTypeMap();  
 map.put("mySchemaName.ATHLETES", Class.forName("Athletes"));  
 con.setTypeMap(map);

**See Also:**[DriverManager.getConnection(java.lang.String, java.util.Properties)](http://docs.google.com/java/sql/DriverManager.html#getConnection(java.lang.String,%20java.util.Properties)), [Statement](http://docs.google.com/java/sql/Statement.html), [ResultSet](http://docs.google.com/java/sql/ResultSet.html), [DatabaseMetaData](http://docs.google.com/java/sql/DatabaseMetaData.html)

| **Field Summary** | |
| --- | --- |
| static int | [**TRANSACTION\_NONE**](http://docs.google.com/java/sql/Connection.html#TRANSACTION_NONE)            A constant indicating that transactions are not supported. |
| static int | [**TRANSACTION\_READ\_COMMITTED**](http://docs.google.com/java/sql/Connection.html#TRANSACTION_READ_COMMITTED)            A constant indicating that dirty reads are prevented; non-repeatable reads and phantom reads can occur. |
| static int | [**TRANSACTION\_READ\_UNCOMMITTED**](http://docs.google.com/java/sql/Connection.html#TRANSACTION_READ_UNCOMMITTED)            A constant indicating that dirty reads, non-repeatable reads and phantom reads can occur. |
| static int | [**TRANSACTION\_REPEATABLE\_READ**](http://docs.google.com/java/sql/Connection.html#TRANSACTION_REPEATABLE_READ)            A constant indicating that dirty reads and non-repeatable reads are prevented; phantom reads can occur. |
| static int | [**TRANSACTION\_SERIALIZABLE**](http://docs.google.com/java/sql/Connection.html#TRANSACTION_SERIALIZABLE)            A constant indicating that dirty reads, non-repeatable reads and phantom reads are prevented. |

| **Method Summary** | |
| --- | --- |
| void | [**clearWarnings**](http://docs.google.com/java/sql/Connection.html#clearWarnings())()            Clears all warnings reported for this Connection object. |
| void | [**close**](http://docs.google.com/java/sql/Connection.html#close())()            Releases this Connection object's database and JDBC resources immediately instead of waiting for them to be automatically released. |
| void | [**commit**](http://docs.google.com/java/sql/Connection.html#commit())()            Makes all changes made since the previous commit/rollback permanent and releases any database locks currently held by this Connection object. |
| [Array](http://docs.google.com/java/sql/Array.html) | [**createArrayOf**](http://docs.google.com/java/sql/Connection.html#createArrayOf(java.lang.String,%20java.lang.Object%5B%5D))([String](http://docs.google.com/java/lang/String.html) typeName, [Object](http://docs.google.com/java/lang/Object.html)[] elements)            Factory method for creating Array objects. |
| [Blob](http://docs.google.com/java/sql/Blob.html) | [**createBlob**](http://docs.google.com/java/sql/Connection.html#createBlob())()            Constructs an object that implements the Blob interface. |
| [Clob](http://docs.google.com/java/sql/Clob.html) | [**createClob**](http://docs.google.com/java/sql/Connection.html#createClob())()            Constructs an object that implements the Clob interface. |
| [NClob](http://docs.google.com/java/sql/NClob.html) | [**createNClob**](http://docs.google.com/java/sql/Connection.html#createNClob())()            Constructs an object that implements the NClob interface. |
| [SQLXML](http://docs.google.com/java/sql/SQLXML.html) | [**createSQLXML**](http://docs.google.com/java/sql/Connection.html#createSQLXML())()            Constructs an object that implements the SQLXML interface. |
| [Statement](http://docs.google.com/java/sql/Statement.html) | [**createStatement**](http://docs.google.com/java/sql/Connection.html#createStatement())()            Creates a Statement object for sending SQL statements to the database. |
| [Statement](http://docs.google.com/java/sql/Statement.html) | [**createStatement**](http://docs.google.com/java/sql/Connection.html#createStatement(int,%20int))(int resultSetType, int resultSetConcurrency)            Creates a Statement object that will generate ResultSet objects with the given type and concurrency. |
| [Statement](http://docs.google.com/java/sql/Statement.html) | [**createStatement**](http://docs.google.com/java/sql/Connection.html#createStatement(int,%20int,%20int))(int resultSetType, int resultSetConcurrency, int resultSetHoldability)            Creates a Statement object that will generate ResultSet objects with the given type, concurrency, and holdability. |
| [Struct](http://docs.google.com/java/sql/Struct.html) | [**createStruct**](http://docs.google.com/java/sql/Connection.html#createStruct(java.lang.String,%20java.lang.Object%5B%5D))([String](http://docs.google.com/java/lang/String.html) typeName, [Object](http://docs.google.com/java/lang/Object.html)[] attributes)            Factory method for creating Struct objects. |
| boolean | [**getAutoCommit**](http://docs.google.com/java/sql/Connection.html#getAutoCommit())()            Retrieves the current auto-commit mode for this Connection object. |
| [String](http://docs.google.com/java/lang/String.html) | [**getCatalog**](http://docs.google.com/java/sql/Connection.html#getCatalog())()            Retrieves this Connection object's current catalog name. |
| [Properties](http://docs.google.com/java/util/Properties.html) | [**getClientInfo**](http://docs.google.com/java/sql/Connection.html#getClientInfo())()            Returns a list containing the name and current value of each client info property supported by the driver. |
| [String](http://docs.google.com/java/lang/String.html) | [**getClientInfo**](http://docs.google.com/java/sql/Connection.html#getClientInfo(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Returns the value of the client info property specified by name. |
| int | [**getHoldability**](http://docs.google.com/java/sql/Connection.html#getHoldability())()            Retrieves the current holdability of ResultSet objects created using this Connection object. |
| [DatabaseMetaData](http://docs.google.com/java/sql/DatabaseMetaData.html) | [**getMetaData**](http://docs.google.com/java/sql/Connection.html#getMetaData())()            Retrieves a DatabaseMetaData object that contains metadata about the database to which this Connection object represents a connection. |
| int | [**getTransactionIsolation**](http://docs.google.com/java/sql/Connection.html#getTransactionIsolation())()            Retrieves this Connection object's current transaction isolation level. |
| [Map](http://docs.google.com/java/util/Map.html)<[String](http://docs.google.com/java/lang/String.html),[Class](http://docs.google.com/java/lang/Class.html)<?>> | [**getTypeMap**](http://docs.google.com/java/sql/Connection.html#getTypeMap())()            Retrieves the Map object associated with this Connection object. |
| [SQLWarning](http://docs.google.com/java/sql/SQLWarning.html) | [**getWarnings**](http://docs.google.com/java/sql/Connection.html#getWarnings())()            Retrieves the first warning reported by calls on this Connection object. |
| boolean | [**isClosed**](http://docs.google.com/java/sql/Connection.html#isClosed())()            Retrieves whether this Connection object has been closed. |
| boolean | [**isReadOnly**](http://docs.google.com/java/sql/Connection.html#isReadOnly())()            Retrieves whether this Connection object is in read-only mode. |
| boolean | [**isValid**](http://docs.google.com/java/sql/Connection.html#isValid(int))(int timeout)            Returns true if the connection has not been closed and is still valid. |
| [String](http://docs.google.com/java/lang/String.html) | [**nativeSQL**](http://docs.google.com/java/sql/Connection.html#nativeSQL(java.lang.String))([String](http://docs.google.com/java/lang/String.html) sql)            Converts the given SQL statement into the system's native SQL grammar. |
| [CallableStatement](http://docs.google.com/java/sql/CallableStatement.html) | [**prepareCall**](http://docs.google.com/java/sql/Connection.html#prepareCall(java.lang.String))([String](http://docs.google.com/java/lang/String.html) sql)            Creates a CallableStatement object for calling database stored procedures. |
| [CallableStatement](http://docs.google.com/java/sql/CallableStatement.html) | [**prepareCall**](http://docs.google.com/java/sql/Connection.html#prepareCall(java.lang.String,%20int,%20int))([String](http://docs.google.com/java/lang/String.html) sql, int resultSetType, int resultSetConcurrency)            Creates a CallableStatement object that will generate ResultSet objects with the given type and concurrency. |
| [CallableStatement](http://docs.google.com/java/sql/CallableStatement.html) | [**prepareCall**](http://docs.google.com/java/sql/Connection.html#prepareCall(java.lang.String,%20int,%20int,%20int))([String](http://docs.google.com/java/lang/String.html) sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)            Creates a CallableStatement object that will generate ResultSet objects with the given type and concurrency. |
| [PreparedStatement](http://docs.google.com/java/sql/PreparedStatement.html) | [**prepareStatement**](http://docs.google.com/java/sql/Connection.html#prepareStatement(java.lang.String))([String](http://docs.google.com/java/lang/String.html) sql)            Creates a PreparedStatement object for sending parameterized SQL statements to the database. |
| [PreparedStatement](http://docs.google.com/java/sql/PreparedStatement.html) | [**prepareStatement**](http://docs.google.com/java/sql/Connection.html#prepareStatement(java.lang.String,%20int))([String](http://docs.google.com/java/lang/String.html) sql, int autoGeneratedKeys)            Creates a default PreparedStatement object that has the capability to retrieve auto-generated keys. |
| [PreparedStatement](http://docs.google.com/java/sql/PreparedStatement.html) | [**prepareStatement**](http://docs.google.com/java/sql/Connection.html#prepareStatement(java.lang.String,%20int%5B%5D))([String](http://docs.google.com/java/lang/String.html) sql, int[] columnIndexes)            Creates a default PreparedStatement object capable of returning the auto-generated keys designated by the given array. |
| [PreparedStatement](http://docs.google.com/java/sql/PreparedStatement.html) | [**prepareStatement**](http://docs.google.com/java/sql/Connection.html#prepareStatement(java.lang.String,%20int,%20int))([String](http://docs.google.com/java/lang/String.html) sql, int resultSetType, int resultSetConcurrency)            Creates a PreparedStatement object that will generate ResultSet objects with the given type and concurrency. |
| [PreparedStatement](http://docs.google.com/java/sql/PreparedStatement.html) | [**prepareStatement**](http://docs.google.com/java/sql/Connection.html#prepareStatement(java.lang.String,%20int,%20int,%20int))([String](http://docs.google.com/java/lang/String.html) sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)            Creates a PreparedStatement object that will generate ResultSet objects with the given type, concurrency, and holdability. |
| [PreparedStatement](http://docs.google.com/java/sql/PreparedStatement.html) | [**prepareStatement**](http://docs.google.com/java/sql/Connection.html#prepareStatement(java.lang.String,%20java.lang.String%5B%5D))([String](http://docs.google.com/java/lang/String.html) sql, [String](http://docs.google.com/java/lang/String.html)[] columnNames)            Creates a default PreparedStatement object capable of returning the auto-generated keys designated by the given array. |
| void | [**releaseSavepoint**](http://docs.google.com/java/sql/Connection.html#releaseSavepoint(java.sql.Savepoint))([Savepoint](http://docs.google.com/java/sql/Savepoint.html) savepoint)            Removes the specified Savepoint and subsequent Savepoint objects from the current transaction. |
| void | [**rollback**](http://docs.google.com/java/sql/Connection.html#rollback())()            Undoes all changes made in the current transaction and releases any database locks currently held by this Connection object. |
| void | [**rollback**](http://docs.google.com/java/sql/Connection.html#rollback(java.sql.Savepoint))([Savepoint](http://docs.google.com/java/sql/Savepoint.html) savepoint)            Undoes all changes made after the given Savepoint object was set. |
| void | [**setAutoCommit**](http://docs.google.com/java/sql/Connection.html#setAutoCommit(boolean))(boolean autoCommit)            Sets this connection's auto-commit mode to the given state. |
| void | [**setCatalog**](http://docs.google.com/java/sql/Connection.html#setCatalog(java.lang.String))([String](http://docs.google.com/java/lang/String.html) catalog)            Sets the given catalog name in order to select a subspace of this Connection object's database in which to work. |
| void | [**setClientInfo**](http://docs.google.com/java/sql/Connection.html#setClientInfo(java.util.Properties))([Properties](http://docs.google.com/java/util/Properties.html) properties)            Sets the value of the connection's client info properties. |
| void | [**setClientInfo**](http://docs.google.com/java/sql/Connection.html#setClientInfo(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) name, [String](http://docs.google.com/java/lang/String.html) value)            Sets the value of the client info property specified by name to the value specified by value. |
| void | [**setHoldability**](http://docs.google.com/java/sql/Connection.html#setHoldability(int))(int holdability)            Changes the default holdability of ResultSet objects created using this Connection object to the given holdability. |
| void | [**setReadOnly**](http://docs.google.com/java/sql/Connection.html#setReadOnly(boolean))(boolean readOnly)            Puts this connection in read-only mode as a hint to the driver to enable database optimizations. |
| [Savepoint](http://docs.google.com/java/sql/Savepoint.html) | [**setSavepoint**](http://docs.google.com/java/sql/Connection.html#setSavepoint())()            Creates an unnamed savepoint in the current transaction and returns the new Savepoint object that represents it. |
| [Savepoint](http://docs.google.com/java/sql/Savepoint.html) | [**setSavepoint**](http://docs.google.com/java/sql/Connection.html#setSavepoint(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Creates a savepoint with the given name in the current transaction and returns the new Savepoint object that represents it. |
| void | [**setTransactionIsolation**](http://docs.google.com/java/sql/Connection.html#setTransactionIsolation(int))(int level)            Attempts to change the transaction isolation level for this Connection object to the one given. |
| void | [**setTypeMap**](http://docs.google.com/java/sql/Connection.html#setTypeMap(java.util.Map))([Map](http://docs.google.com/java/util/Map.html)<[String](http://docs.google.com/java/lang/String.html),[Class](http://docs.google.com/java/lang/Class.html)<?>> map)            Installs the given TypeMap object as the type map for this Connection object. |

| **Methods inherited from interface java.sql.**[**Wrapper**](http://docs.google.com/java/sql/Wrapper.html) |
| --- |
| [isWrapperFor](http://docs.google.com/java/sql/Wrapper.html#isWrapperFor(java.lang.Class)), [unwrap](http://docs.google.com/java/sql/Wrapper.html#unwrap(java.lang.Class)) |

| **Field Detail** |
| --- |

### TRANSACTION\_NONE

static final int **TRANSACTION\_NONE**

A constant indicating that transactions are not supported.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.sql.Connection.TRANSACTION_NONE)

### TRANSACTION\_READ\_UNCOMMITTED

static final int **TRANSACTION\_READ\_UNCOMMITTED**

A constant indicating that dirty reads, non-repeatable reads and phantom reads can occur. This level allows a row changed by one transaction to be read by another transaction before any changes in that row have been committed (a "dirty read"). If any of the changes are rolled back, the second transaction will have retrieved an invalid row.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.sql.Connection.TRANSACTION_READ_UNCOMMITTED)

### TRANSACTION\_READ\_COMMITTED

static final int **TRANSACTION\_READ\_COMMITTED**

A constant indicating that dirty reads are prevented; non-repeatable reads and phantom reads can occur. This level only prohibits a transaction from reading a row with uncommitted changes in it.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.sql.Connection.TRANSACTION_READ_COMMITTED)

### TRANSACTION\_REPEATABLE\_READ

static final int **TRANSACTION\_REPEATABLE\_READ**

A constant indicating that dirty reads and non-repeatable reads are prevented; phantom reads can occur. This level prohibits a transaction from reading a row with uncommitted changes in it, and it also prohibits the situation where one transaction reads a row, a second transaction alters the row, and the first transaction rereads the row, getting different values the second time (a "non-repeatable read").

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.sql.Connection.TRANSACTION_REPEATABLE_READ)

### TRANSACTION\_SERIALIZABLE

static final int **TRANSACTION\_SERIALIZABLE**

A constant indicating that dirty reads, non-repeatable reads and phantom reads are prevented. This level includes the prohibitions in TRANSACTION\_REPEATABLE\_READ and further prohibits the situation where one transaction reads all rows that satisfy a WHERE condition, a second transaction inserts a row that satisfies that WHERE condition, and the first transaction rereads for the same condition, retrieving the additional "phantom" row in the second read.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.sql.Connection.TRANSACTION_SERIALIZABLE)

| **Method Detail** |
| --- |

### createStatement

[Statement](http://docs.google.com/java/sql/Statement.html) **createStatement**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Creates a Statement object for sending SQL statements to the database. SQL statements without parameters are normally executed using Statement objects. If the same SQL statement is executed many times, it may be more efficient to use a PreparedStatement object.

Result sets created using the returned Statement object will by default be type TYPE\_FORWARD\_ONLY and have a concurrency level of CONCUR\_READ\_ONLY. The holdability of the created result sets can be determined by calling [getHoldability()](http://docs.google.com/java/sql/Connection.html#getHoldability()).

**Returns:**a new default Statement object **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed connection

### prepareStatement

[PreparedStatement](http://docs.google.com/java/sql/PreparedStatement.html) **prepareStatement**([String](http://docs.google.com/java/lang/String.html) sql)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Creates a PreparedStatement object for sending parameterized SQL statements to the database.

A SQL statement with or without IN parameters can be pre-compiled and stored in a PreparedStatement object. This object can then be used to efficiently execute this statement multiple times.

**Note:** This method is optimized for handling parametric SQL statements that benefit from precompilation. If the driver supports precompilation, the method prepareStatement will send the statement to the database for precompilation. Some drivers may not support precompilation. In this case, the statement may not be sent to the database until the PreparedStatement object is executed. This has no direct effect on users; however, it does affect which methods throw certain SQLException objects.

Result sets created using the returned PreparedStatement object will by default be type TYPE\_FORWARD\_ONLY and have a concurrency level of CONCUR\_READ\_ONLY. The holdability of the created result sets can be determined by calling [getHoldability()](http://docs.google.com/java/sql/Connection.html#getHoldability()).

**Parameters:**sql - an SQL statement that may contain one or more '?' IN parameter placeholders **Returns:**a new default PreparedStatement object containing the pre-compiled SQL statement **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed connection

### prepareCall

[CallableStatement](http://docs.google.com/java/sql/CallableStatement.html) **prepareCall**([String](http://docs.google.com/java/lang/String.html) sql)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Creates a CallableStatement object for calling database stored procedures. The CallableStatement object provides methods for setting up its IN and OUT parameters, and methods for executing the call to a stored procedure.

**Note:** This method is optimized for handling stored procedure call statements. Some drivers may send the call statement to the database when the method prepareCall is done; others may wait until the CallableStatement object is executed. This has no direct effect on users; however, it does affect which method throws certain SQLExceptions.

Result sets created using the returned CallableStatement object will by default be type TYPE\_FORWARD\_ONLY and have a concurrency level of CONCUR\_READ\_ONLY. The holdability of the created result sets can be determined by calling [getHoldability()](http://docs.google.com/java/sql/Connection.html#getHoldability()).

**Parameters:**sql - an SQL statement that may contain one or more '?' parameter placeholders. Typically this statement is specified using JDBC call escape syntax. **Returns:**a new default CallableStatement object containing the pre-compiled SQL statement **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed connection

### nativeSQL

[String](http://docs.google.com/java/lang/String.html) **nativeSQL**([String](http://docs.google.com/java/lang/String.html) sql)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Converts the given SQL statement into the system's native SQL grammar. A driver may convert the JDBC SQL grammar into its system's native SQL grammar prior to sending it. This method returns the native form of the statement that the driver would have sent.

**Parameters:**sql - an SQL statement that may contain one or more '?' parameter placeholders **Returns:**the native form of this statement **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed connection

### setAutoCommit

void **setAutoCommit**(boolean autoCommit)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets this connection's auto-commit mode to the given state. If a connection is in auto-commit mode, then all its SQL statements will be executed and committed as individual transactions. Otherwise, its SQL statements are grouped into transactions that are terminated by a call to either the method commit or the method rollback. By default, new connections are in auto-commit mode.

The commit occurs when the statement completes. The time when the statement completes depends on the type of SQL Statement:

* For DML statements, such as Insert, Update or Delete, and DDL statements, the statement is complete as soon as it has finished executing.
* For Select statements, the statement is complete when the associated result set is closed.
* For CallableStatement objects or for statements that return multiple results, the statement is complete when all of the associated result sets have been closed, and all update counts and output parameters have been retrieved.

**NOTE:** If this method is called during a transaction and the auto-commit mode is changed, the transaction is committed. If setAutoCommit is called and the auto-commit mode is not changed, the call is a no-op.

**Parameters:**autoCommit - true to enable auto-commit mode; false to disable it **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, setAutoCommit(true) is called while participating in a distributed transaction, or this method is called on a closed connection**See Also:**[getAutoCommit()](http://docs.google.com/java/sql/Connection.html#getAutoCommit())

### getAutoCommit

boolean **getAutoCommit**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the current auto-commit mode for this Connection object.

**Returns:**the current state of this Connection object's auto-commit mode **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed connection**See Also:**[setAutoCommit(boolean)](http://docs.google.com/java/sql/Connection.html#setAutoCommit(boolean))

### commit

void **commit**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Makes all changes made since the previous commit/rollback permanent and releases any database locks currently held by this Connection object. This method should be used only when auto-commit mode has been disabled.

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called while participating in a distributed transaction, if this method is called on a closed conection or this Connection object is in auto-commit mode**See Also:**[setAutoCommit(boolean)](http://docs.google.com/java/sql/Connection.html#setAutoCommit(boolean))

### rollback

void **rollback**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Undoes all changes made in the current transaction and releases any database locks currently held by this Connection object. This method should be used only when auto-commit mode has been disabled.

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called while participating in a distributed transaction, this method is called on a closed connection or this Connection object is in auto-commit mode**See Also:**[setAutoCommit(boolean)](http://docs.google.com/java/sql/Connection.html#setAutoCommit(boolean))

### close

void **close**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Releases this Connection object's database and JDBC resources immediately instead of waiting for them to be automatically released.

Calling the method close on a Connection object that is already closed is a no-op.

It is **strongly recommended** that an application explicitly commits or rolls back an active transaction prior to calling the close method. If the close method is called and there is an active transaction, the results are implementation-defined.

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - SQLException if a database access error occurs

### isClosed

boolean **isClosed**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves whether this Connection object has been closed. A connection is closed if the method close has been called on it or if certain fatal errors have occurred. This method is guaranteed to return true only when it is called after the method Connection.close has been called.

This method generally cannot be called to determine whether a connection to a database is valid or invalid. A typical client can determine that a connection is invalid by catching any exceptions that might be thrown when an operation is attempted.

**Returns:**true if this Connection object is closed; false if it is still open **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs

### getMetaData

[DatabaseMetaData](http://docs.google.com/java/sql/DatabaseMetaData.html) **getMetaData**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves a DatabaseMetaData object that contains metadata about the database to which this Connection object represents a connection. The metadata includes information about the database's tables, its supported SQL grammar, its stored procedures, the capabilities of this connection, and so on.

**Returns:**a DatabaseMetaData object for this Connection object **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed connection

### setReadOnly

void **setReadOnly**(boolean readOnly)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Puts this connection in read-only mode as a hint to the driver to enable database optimizations.

**Note:** This method cannot be called during a transaction.

**Parameters:**readOnly - true enables read-only mode; false disables it **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed connection or this method is called during a transaction

### isReadOnly

boolean **isReadOnly**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves whether this Connection object is in read-only mode.

**Returns:**true if this Connection object is read-only; false otherwise **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - SQLException if a database access error occurs or this method is called on a closed connection

### setCatalog

void **setCatalog**([String](http://docs.google.com/java/lang/String.html) catalog)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the given catalog name in order to select a subspace of this Connection object's database in which to work.

If the driver does not support catalogs, it will silently ignore this request.

**Parameters:**catalog - the name of a catalog (subspace in this Connection object's database) in which to work **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed connection**See Also:**[getCatalog()](http://docs.google.com/java/sql/Connection.html#getCatalog())

### getCatalog

[String](http://docs.google.com/java/lang/String.html) **getCatalog**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves this Connection object's current catalog name.

**Returns:**the current catalog name or null if there is none **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed connection**See Also:**[setCatalog(java.lang.String)](http://docs.google.com/java/sql/Connection.html#setCatalog(java.lang.String))

### setTransactionIsolation

void **setTransactionIsolation**(int level)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Attempts to change the transaction isolation level for this Connection object to the one given. The constants defined in the interface Connection are the possible transaction isolation levels.

**Note:** If this method is called during a transaction, the result is implementation-defined.

**Parameters:**level - one of the following Connection constants: Connection.TRANSACTION\_READ\_UNCOMMITTED, Connection.TRANSACTION\_READ\_COMMITTED, Connection.TRANSACTION\_REPEATABLE\_READ, or Connection.TRANSACTION\_SERIALIZABLE. (Note that Connection.TRANSACTION\_NONE cannot be used because it specifies that transactions are not supported.) **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed connection or the given parameter is not one of the Connection constants**See Also:**[DatabaseMetaData.supportsTransactionIsolationLevel(int)](http://docs.google.com/java/sql/DatabaseMetaData.html#supportsTransactionIsolationLevel(int)), [getTransactionIsolation()](http://docs.google.com/java/sql/Connection.html#getTransactionIsolation())

### getTransactionIsolation

int **getTransactionIsolation**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves this Connection object's current transaction isolation level.

**Returns:**the current transaction isolation level, which will be one of the following constants: Connection.TRANSACTION\_READ\_UNCOMMITTED, Connection.TRANSACTION\_READ\_COMMITTED, Connection.TRANSACTION\_REPEATABLE\_READ, Connection.TRANSACTION\_SERIALIZABLE, or Connection.TRANSACTION\_NONE. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed connection**See Also:**[setTransactionIsolation(int)](http://docs.google.com/java/sql/Connection.html#setTransactionIsolation(int))

### getWarnings

[SQLWarning](http://docs.google.com/java/sql/SQLWarning.html) **getWarnings**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the first warning reported by calls on this Connection object. If there is more than one warning, subsequent warnings will be chained to the first one and can be retrieved by calling the method SQLWarning.getNextWarning on the warning that was retrieved previously.

This method may not be called on a closed connection; doing so will cause an SQLException to be thrown.

**Note:** Subsequent warnings will be chained to this SQLWarning.

**Returns:**the first SQLWarning object or null if there are none **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed connection**See Also:**[SQLWarning](http://docs.google.com/java/sql/SQLWarning.html)

### clearWarnings

void **clearWarnings**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Clears all warnings reported for this Connection object. After a call to this method, the method getWarnings returns null until a new warning is reported for this Connection object.

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - SQLException if a database access error occurs or this method is called on a closed connection

### createStatement

[Statement](http://docs.google.com/java/sql/Statement.html) **createStatement**(int resultSetType,  
 int resultSetConcurrency)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Creates a Statement object that will generate ResultSet objects with the given type and concurrency. This method is the same as the createStatement method above, but it allows the default result set type and concurrency to be overridden. The holdability of the created result sets can be determined by calling [getHoldability()](http://docs.google.com/java/sql/Connection.html#getHoldability()).

**Parameters:**resultSetType - a result set type; one of ResultSet.TYPE\_FORWARD\_ONLY, ResultSet.TYPE\_SCROLL\_INSENSITIVE, or ResultSet.TYPE\_SCROLL\_SENSITIVEresultSetConcurrency - a concurrency type; one of ResultSet.CONCUR\_READ\_ONLY or ResultSet.CONCUR\_UPDATABLE **Returns:**a new Statement object that will generate ResultSet objects with the given type and concurrency **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed connection or the given parameters are not ResultSet constants indicating type and concurrency [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method or this method is not supported for the specified result set type and result set concurrency.**Since:** 1.2

### prepareStatement

[PreparedStatement](http://docs.google.com/java/sql/PreparedStatement.html) **prepareStatement**([String](http://docs.google.com/java/lang/String.html) sql,  
 int resultSetType,  
 int resultSetConcurrency)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Creates a PreparedStatement object that will generate ResultSet objects with the given type and concurrency. This method is the same as the prepareStatement method above, but it allows the default result set type and concurrency to be overridden. The holdability of the created result sets can be determined by calling [getHoldability()](http://docs.google.com/java/sql/Connection.html#getHoldability()).

**Parameters:**sql - a String object that is the SQL statement to be sent to the database; may contain one or more '?' IN parametersresultSetType - a result set type; one of ResultSet.TYPE\_FORWARD\_ONLY, ResultSet.TYPE\_SCROLL\_INSENSITIVE, or ResultSet.TYPE\_SCROLL\_SENSITIVEresultSetConcurrency - a concurrency type; one of ResultSet.CONCUR\_READ\_ONLY or ResultSet.CONCUR\_UPDATABLE **Returns:**a new PreparedStatement object containing the pre-compiled SQL statement that will produce ResultSet objects with the given type and concurrency **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed connection or the given parameters are not ResultSet constants indicating type and concurrency [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method or this method is not supported for the specified result set type and result set concurrency.**Since:** 1.2

### prepareCall

[CallableStatement](http://docs.google.com/java/sql/CallableStatement.html) **prepareCall**([String](http://docs.google.com/java/lang/String.html) sql,  
 int resultSetType,  
 int resultSetConcurrency)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Creates a CallableStatement object that will generate ResultSet objects with the given type and concurrency. This method is the same as the prepareCall method above, but it allows the default result set type and concurrency to be overridden. The holdability of the created result sets can be determined by calling [getHoldability()](http://docs.google.com/java/sql/Connection.html#getHoldability()).

**Parameters:**sql - a String object that is the SQL statement to be sent to the database; may contain on or more '?' parametersresultSetType - a result set type; one of ResultSet.TYPE\_FORWARD\_ONLY, ResultSet.TYPE\_SCROLL\_INSENSITIVE, or ResultSet.TYPE\_SCROLL\_SENSITIVEresultSetConcurrency - a concurrency type; one of ResultSet.CONCUR\_READ\_ONLY or ResultSet.CONCUR\_UPDATABLE **Returns:**a new CallableStatement object containing the pre-compiled SQL statement that will produce ResultSet objects with the given type and concurrency **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed connection or the given parameters are not ResultSet constants indicating type and concurrency [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method or this method is not supported for the specified result set type and result set concurrency.**Since:** 1.2

### getTypeMap

[Map](http://docs.google.com/java/util/Map.html)<[String](http://docs.google.com/java/lang/String.html),[Class](http://docs.google.com/java/lang/Class.html)<?>> **getTypeMap**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the Map object associated with this Connection object. Unless the application has added an entry, the type map returned will be empty.

**Returns:**the java.util.Map object associated with this Connection object **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed connection [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.2 **See Also:**[setTypeMap(java.util.Map>)](http://docs.google.com/java/sql/Connection.html#setTypeMap(java.util.Map))

### setTypeMap

void **setTypeMap**([Map](http://docs.google.com/java/util/Map.html)<[String](http://docs.google.com/java/lang/String.html),[Class](http://docs.google.com/java/lang/Class.html)<?>> map)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Installs the given TypeMap object as the type map for this Connection object. The type map will be used for the custom mapping of SQL structured types and distinct types.

**Parameters:**map - the java.util.Map object to install as the replacement for this Connection object's default type map **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed connection or the given parameter is not a java.util.Map object [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.2 **See Also:**[getTypeMap()](http://docs.google.com/java/sql/Connection.html#getTypeMap())

### setHoldability

void **setHoldability**(int holdability)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Changes the default holdability of ResultSet objects created using this Connection object to the given holdability. The default holdability of ResultSet objects can be be determined by invoking [DatabaseMetaData.getResultSetHoldability()](http://docs.google.com/java/sql/DatabaseMetaData.html#getResultSetHoldability()).

**Parameters:**holdability - a ResultSet holdability constant; one of ResultSet.HOLD\_CURSORS\_OVER\_COMMIT or ResultSet.CLOSE\_CURSORS\_AT\_COMMIT **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access occurs, this method is called on a closed connection, or the given parameter is not a ResultSet constant indicating holdability [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the given holdability is not supported**Since:** 1.4 **See Also:**[getHoldability()](http://docs.google.com/java/sql/Connection.html#getHoldability()), [DatabaseMetaData.getResultSetHoldability()](http://docs.google.com/java/sql/DatabaseMetaData.html#getResultSetHoldability()), [ResultSet](http://docs.google.com/java/sql/ResultSet.html)

### getHoldability

int **getHoldability**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the current holdability of ResultSet objects created using this Connection object.

**Returns:**the holdability, one of ResultSet.HOLD\_CURSORS\_OVER\_COMMIT or ResultSet.CLOSE\_CURSORS\_AT\_COMMIT **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed connection**Since:** 1.4 **See Also:**[setHoldability(int)](http://docs.google.com/java/sql/Connection.html#setHoldability(int)), [DatabaseMetaData.getResultSetHoldability()](http://docs.google.com/java/sql/DatabaseMetaData.html#getResultSetHoldability()), [ResultSet](http://docs.google.com/java/sql/ResultSet.html)

### setSavepoint

[Savepoint](http://docs.google.com/java/sql/Savepoint.html) **setSavepoint**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Creates an unnamed savepoint in the current transaction and returns the new Savepoint object that represents it.

if setSavepoint is invoked outside of an active transaction, a transaction will be started at this newly created savepoint.

**Returns:**the new Savepoint object **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called while participating in a distributed transaction, this method is called on a closed connection or this Connection object is currently in auto-commit mode [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**[Savepoint](http://docs.google.com/java/sql/Savepoint.html)

### setSavepoint

[Savepoint](http://docs.google.com/java/sql/Savepoint.html) **setSavepoint**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Creates a savepoint with the given name in the current transaction and returns the new Savepoint object that represents it.

if setSavepoint is invoked outside of an active transaction, a transaction will be started at this newly created savepoint.

**Parameters:**name - a String containing the name of the savepoint **Returns:**the new Savepoint object **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called while participating in a distributed transaction, this method is called on a closed connection or this Connection object is currently in auto-commit mode [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**[Savepoint](http://docs.google.com/java/sql/Savepoint.html)

### rollback

void **rollback**([Savepoint](http://docs.google.com/java/sql/Savepoint.html) savepoint)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Undoes all changes made after the given Savepoint object was set.

This method should be used only when auto-commit has been disabled.

**Parameters:**savepoint - the Savepoint object to roll back to **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called while participating in a distributed transaction, this method is called on a closed connection, the Savepoint object is no longer valid, or this Connection object is currently in auto-commit mode [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**[Savepoint](http://docs.google.com/java/sql/Savepoint.html), [rollback()](http://docs.google.com/java/sql/Connection.html#rollback())

### releaseSavepoint

void **releaseSavepoint**([Savepoint](http://docs.google.com/java/sql/Savepoint.html) savepoint)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Removes the specified Savepoint and subsequent Savepoint objects from the current transaction. Any reference to the savepoint after it have been removed will cause an SQLException to be thrown.

**Parameters:**savepoint - the Savepoint object to be removed **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed connection or the given Savepoint object is not a valid savepoint in the current transaction [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4

### createStatement

[Statement](http://docs.google.com/java/sql/Statement.html) **createStatement**(int resultSetType,  
 int resultSetConcurrency,  
 int resultSetHoldability)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Creates a Statement object that will generate ResultSet objects with the given type, concurrency, and holdability. This method is the same as the createStatement method above, but it allows the default result set type, concurrency, and holdability to be overridden.

**Parameters:**resultSetType - one of the following ResultSet constants: ResultSet.TYPE\_FORWARD\_ONLY, ResultSet.TYPE\_SCROLL\_INSENSITIVE, or ResultSet.TYPE\_SCROLL\_SENSITIVEresultSetConcurrency - one of the following ResultSet constants: ResultSet.CONCUR\_READ\_ONLY or ResultSet.CONCUR\_UPDATABLEresultSetHoldability - one of the following ResultSet constants: ResultSet.HOLD\_CURSORS\_OVER\_COMMIT or ResultSet.CLOSE\_CURSORS\_AT\_COMMIT **Returns:**a new Statement object that will generate ResultSet objects with the given type, concurrency, and holdability **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed connection or the given parameters are not ResultSet constants indicating type, concurrency, and holdability [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method or this method is not supported for the specified result set type, result set holdability and result set concurrency.**Since:** 1.4 **See Also:**[ResultSet](http://docs.google.com/java/sql/ResultSet.html)

### prepareStatement

[PreparedStatement](http://docs.google.com/java/sql/PreparedStatement.html) **prepareStatement**([String](http://docs.google.com/java/lang/String.html) sql,  
 int resultSetType,  
 int resultSetConcurrency,  
 int resultSetHoldability)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Creates a PreparedStatement object that will generate ResultSet objects with the given type, concurrency, and holdability.

This method is the same as the prepareStatement method above, but it allows the default result set type, concurrency, and holdability to be overridden.

**Parameters:**sql - a String object that is the SQL statement to be sent to the database; may contain one or more '?' IN parametersresultSetType - one of the following ResultSet constants: ResultSet.TYPE\_FORWARD\_ONLY, ResultSet.TYPE\_SCROLL\_INSENSITIVE, or ResultSet.TYPE\_SCROLL\_SENSITIVEresultSetConcurrency - one of the following ResultSet constants: ResultSet.CONCUR\_READ\_ONLY or ResultSet.CONCUR\_UPDATABLEresultSetHoldability - one of the following ResultSet constants: ResultSet.HOLD\_CURSORS\_OVER\_COMMIT or ResultSet.CLOSE\_CURSORS\_AT\_COMMIT **Returns:**a new PreparedStatement object, containing the pre-compiled SQL statement, that will generate ResultSet objects with the given type, concurrency, and holdability **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed connection or the given parameters are not ResultSet constants indicating type, concurrency, and holdability [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method or this method is not supported for the specified result set type, result set holdability and result set concurrency.**Since:** 1.4 **See Also:**[ResultSet](http://docs.google.com/java/sql/ResultSet.html)

### prepareCall

[CallableStatement](http://docs.google.com/java/sql/CallableStatement.html) **prepareCall**([String](http://docs.google.com/java/lang/String.html) sql,  
 int resultSetType,  
 int resultSetConcurrency,  
 int resultSetHoldability)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Creates a CallableStatement object that will generate ResultSet objects with the given type and concurrency. This method is the same as the prepareCall method above, but it allows the default result set type, result set concurrency type and holdability to be overridden.

**Parameters:**sql - a String object that is the SQL statement to be sent to the database; may contain on or more '?' parametersresultSetType - one of the following ResultSet constants: ResultSet.TYPE\_FORWARD\_ONLY, ResultSet.TYPE\_SCROLL\_INSENSITIVE, or ResultSet.TYPE\_SCROLL\_SENSITIVEresultSetConcurrency - one of the following ResultSet constants: ResultSet.CONCUR\_READ\_ONLY or ResultSet.CONCUR\_UPDATABLEresultSetHoldability - one of the following ResultSet constants: ResultSet.HOLD\_CURSORS\_OVER\_COMMIT or ResultSet.CLOSE\_CURSORS\_AT\_COMMIT **Returns:**a new CallableStatement object, containing the pre-compiled SQL statement, that will generate ResultSet objects with the given type, concurrency, and holdability **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed connection or the given parameters are not ResultSet constants indicating type, concurrency, and holdability [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method or this method is not supported for the specified result set type, result set holdability and result set concurrency.**Since:** 1.4 **See Also:**[ResultSet](http://docs.google.com/java/sql/ResultSet.html)

### prepareStatement

[PreparedStatement](http://docs.google.com/java/sql/PreparedStatement.html) **prepareStatement**([String](http://docs.google.com/java/lang/String.html) sql,  
 int autoGeneratedKeys)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Creates a default PreparedStatement object that has the capability to retrieve auto-generated keys. The given constant tells the driver whether it should make auto-generated keys available for retrieval. This parameter is ignored if the SQL statement is not an INSERT statement, or an SQL statement able to return auto-generated keys (the list of such statements is vendor-specific).

**Note:** This method is optimized for handling parametric SQL statements that benefit from precompilation. If the driver supports precompilation, the method prepareStatement will send the statement to the database for precompilation. Some drivers may not support precompilation. In this case, the statement may not be sent to the database until the PreparedStatement object is executed. This has no direct effect on users; however, it does affect which methods throw certain SQLExceptions.

Result sets created using the returned PreparedStatement object will by default be type TYPE\_FORWARD\_ONLY and have a concurrency level of CONCUR\_READ\_ONLY. The holdability of the created result sets can be determined by calling [getHoldability()](http://docs.google.com/java/sql/Connection.html#getHoldability()).

**Parameters:**sql - an SQL statement that may contain one or more '?' IN parameter placeholdersautoGeneratedKeys - a flag indicating whether auto-generated keys should be returned; one of Statement.RETURN\_GENERATED\_KEYS or Statement.NO\_GENERATED\_KEYS **Returns:**a new PreparedStatement object, containing the pre-compiled SQL statement, that will have the capability of returning auto-generated keys **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed connection or the given parameter is not a Statement constant indicating whether auto-generated keys should be returned [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method with a constant of Statement.RETURN\_GENERATED\_KEYS**Since:** 1.4

### prepareStatement

[PreparedStatement](http://docs.google.com/java/sql/PreparedStatement.html) **prepareStatement**([String](http://docs.google.com/java/lang/String.html) sql,  
 int[] columnIndexes)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Creates a default PreparedStatement object capable of returning the auto-generated keys designated by the given array. This array contains the indexes of the columns in the target table that contain the auto-generated keys that should be made available. The driver will ignore the array if the SQL statement is not an INSERT statement, or an SQL statement able to return auto-generated keys (the list of such statements is vendor-specific).

An SQL statement with or without IN parameters can be pre-compiled and stored in a PreparedStatement object. This object can then be used to efficiently execute this statement multiple times.

**Note:** This method is optimized for handling parametric SQL statements that benefit from precompilation. If the driver supports precompilation, the method prepareStatement will send the statement to the database for precompilation. Some drivers may not support precompilation. In this case, the statement may not be sent to the database until the PreparedStatement object is executed. This has no direct effect on users; however, it does affect which methods throw certain SQLExceptions.

Result sets created using the returned PreparedStatement object will by default be type TYPE\_FORWARD\_ONLY and have a concurrency level of CONCUR\_READ\_ONLY. The holdability of the created result sets can be determined by calling [getHoldability()](http://docs.google.com/java/sql/Connection.html#getHoldability()).

**Parameters:**sql - an SQL statement that may contain one or more '?' IN parameter placeholderscolumnIndexes - an array of column indexes indicating the columns that should be returned from the inserted row or rows **Returns:**a new PreparedStatement object, containing the pre-compiled statement, that is capable of returning the auto-generated keys designated by the given array of column indexes **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed connection [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4

### prepareStatement

[PreparedStatement](http://docs.google.com/java/sql/PreparedStatement.html) **prepareStatement**([String](http://docs.google.com/java/lang/String.html) sql,  
 [String](http://docs.google.com/java/lang/String.html)[] columnNames)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Creates a default PreparedStatement object capable of returning the auto-generated keys designated by the given array. This array contains the names of the columns in the target table that contain the auto-generated keys that should be returned. The driver will ignore the array if the SQL statement is not an INSERT statement, or an SQL statement able to return auto-generated keys (the list of such statements is vendor-specific).

An SQL statement with or without IN parameters can be pre-compiled and stored in a PreparedStatement object. This object can then be used to efficiently execute this statement multiple times.

**Note:** This method is optimized for handling parametric SQL statements that benefit from precompilation. If the driver supports precompilation, the method prepareStatement will send the statement to the database for precompilation. Some drivers may not support precompilation. In this case, the statement may not be sent to the database until the PreparedStatement object is executed. This has no direct effect on users; however, it does affect which methods throw certain SQLExceptions.

Result sets created using the returned PreparedStatement object will by default be type TYPE\_FORWARD\_ONLY and have a concurrency level of CONCUR\_READ\_ONLY. The holdability of the created result sets can be determined by calling [getHoldability()](http://docs.google.com/java/sql/Connection.html#getHoldability()).

**Parameters:**sql - an SQL statement that may contain one or more '?' IN parameter placeholderscolumnNames - an array of column names indicating the columns that should be returned from the inserted row or rows **Returns:**a new PreparedStatement object, containing the pre-compiled statement, that is capable of returning the auto-generated keys designated by the given array of column names **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed connection [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4

### createClob

[Clob](http://docs.google.com/java/sql/Clob.html) **createClob**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Constructs an object that implements the Clob interface. The object returned initially contains no data. The setAsciiStream, setCharacterStream and setString methods of the Clob interface may be used to add data to the Clob.

**Returns:**An object that implements the Clob interface **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an object that implements the Clob interface can not be constructed, this method is called on a closed connection or a database access error occurs. [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this data type**Since:** 1.6

### createBlob

[Blob](http://docs.google.com/java/sql/Blob.html) **createBlob**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Constructs an object that implements the Blob interface. The object returned initially contains no data. The setBinaryStream and setBytes methods of the Blob interface may be used to add data to the Blob.

**Returns:**An object that implements the Blob interface **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an object that implements the Blob interface can not be constructed, this method is called on a closed connection or a database access error occurs. [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this data type**Since:** 1.6

### createNClob

[NClob](http://docs.google.com/java/sql/NClob.html) **createNClob**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Constructs an object that implements the NClob interface. The object returned initially contains no data. The setAsciiStream, setCharacterStream and setString methods of the NClob interface may be used to add data to the NClob.

**Returns:**An object that implements the NClob interface **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an object that implements the NClob interface can not be constructed, this method is called on a closed connection or a database access error occurs. [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this data type**Since:** 1.6

### createSQLXML

[SQLXML](http://docs.google.com/java/sql/SQLXML.html) **createSQLXML**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Constructs an object that implements the SQLXML interface. The object returned initially contains no data. The createXmlStreamWriter object and setString method of the SQLXML interface may be used to add data to the SQLXML object.

**Returns:**An object that implements the SQLXML interface **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if an object that implements the SQLXML interface can not be constructed, this method is called on a closed connection or a database access error occurs. [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this data type**Since:** 1.6

### isValid

boolean **isValid**(int timeout)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Returns true if the connection has not been closed and is still valid. The driver shall submit a query on the connection or use some other mechanism that positively verifies the connection is still valid when this method is called.

The query submitted by the driver to validate the connection shall be executed in the context of the current transaction.

**Parameters:**timeout - - The time in seconds to wait for the database operation used to validate the connection to complete. If the timeout period expires before the operation completes, this method returns false. A value of 0 indicates a timeout is not applied to the database operation.

**Returns:**true if the connection is valid, false otherwise **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the value supplied for timeout is less then 0**Since:** 1.6

**See Also:**[DatabaseMetaData.getClientInfoProperties()](http://docs.google.com/java/sql/DatabaseMetaData.html#getClientInfoProperties())

### setClientInfo

void **setClientInfo**([String](http://docs.google.com/java/lang/String.html) name,  
 [String](http://docs.google.com/java/lang/String.html) value)  
 throws [SQLClientInfoException](http://docs.google.com/java/sql/SQLClientInfoException.html)

Sets the value of the client info property specified by name to the value specified by value.

Applications may use the DatabaseMetaData.getClientInfoProperties method to determine the client info properties supported by the driver and the maximum length that may be specified for each property.

The driver stores the value specified in a suitable location in the database. For example in a special register, session parameter, or system table column. For efficiency the driver may defer setting the value in the database until the next time a statement is executed or prepared. Other than storing the client information in the appropriate place in the database, these methods shall not alter the behavior of the connection in anyway. The values supplied to these methods are used for accounting, diagnostics and debugging purposes only.

The driver shall generate a warning if the client info name specified is not recognized by the driver.

If the value specified to this method is greater than the maximum length for the property the driver may either truncate the value and generate a warning or generate a SQLClientInfoException. If the driver generates a SQLClientInfoException, the value specified was not set on the connection.

The following are standard client info properties. Drivers are not required to support these properties however if the driver supports a client info property that can be described by one of the standard properties, the standard property name should be used.

* ApplicationName - The name of the application currently utilizing the connection
* ClientUser - The name of the user that the application using the connection is performing work for. This may not be the same as the user name that was used in establishing the connection.
* ClientHostname - The hostname of the computer the application using the connection is running on.

**Parameters:**name - The name of the client info property to setvalue - The value to set the client info property to. If the value is null, the current value of the specified property is cleared.

**Throws:** [SQLClientInfoException](http://docs.google.com/java/sql/SQLClientInfoException.html) - if the database server returns an error while setting the client info value on the database server or this method is called on a closed connection

**Since:** 1.6

### setClientInfo

void **setClientInfo**([Properties](http://docs.google.com/java/util/Properties.html) properties)  
 throws [SQLClientInfoException](http://docs.google.com/java/sql/SQLClientInfoException.html)

Sets the value of the connection's client info properties. The Properties object contains the names and values of the client info properties to be set. The set of client info properties contained in the properties list replaces the current set of client info properties on the connection. If a property that is currently set on the connection is not present in the properties list, that property is cleared. Specifying an empty properties list will clear all of the properties on the connection. See setClientInfo (String, String) for more information.

If an error occurs in setting any of the client info properties, a SQLClientInfoException is thrown. The SQLClientInfoException contains information indicating which client info properties were not set. The state of the client information is unknown because some databases do not allow multiple client info properties to be set atomically. For those databases, one or more properties may have been set before the error occurred.

**Parameters:**properties - the list of client info properties to set

**Throws:** [SQLClientInfoException](http://docs.google.com/java/sql/SQLClientInfoException.html) - if the database server returns an error while setting the clientInfo values on the database server or this method is called on a closed connection

**Since:** 1.6

**See Also:**[setClientInfo(String, String)](http://docs.google.com/java/sql/Connection.html#setClientInfo(java.lang.String,%20java.lang.String))

### getClientInfo

[String](http://docs.google.com/java/lang/String.html) **getClientInfo**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Returns the value of the client info property specified by name. This method may return null if the specified client info property has not been set and does not have a default value. This method will also return null if the specified client info property name is not supported by the driver.

Applications may use the DatabaseMetaData.getClientInfoProperties method to determine the client info properties supported by the driver.

**Parameters:**name - The name of the client info property to retrieve

**Returns:**The value of the client info property specified

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the database server returns an error when fetching the client info value from the database or this method is called on a closed connection

**Since:** 1.6

**See Also:**[DatabaseMetaData.getClientInfoProperties()](http://docs.google.com/java/sql/DatabaseMetaData.html#getClientInfoProperties())

### getClientInfo

[Properties](http://docs.google.com/java/util/Properties.html) **getClientInfo**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Returns a list containing the name and current value of each client info property supported by the driver. The value of a client info property may be null if the property has not been set and does not have a default value.

**Returns:**A Properties object that contains the name and current value of each of the client info properties supported by the driver.

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if the database server returns an error when fetching the client info values from the database or this method is called on a closed connection

**Since:** 1.6

### createArrayOf

[Array](http://docs.google.com/java/sql/Array.html) **createArrayOf**([String](http://docs.google.com/java/lang/String.html) typeName,  
 [Object](http://docs.google.com/java/lang/Object.html)[] elements)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Factory method for creating Array objects.

**Note:** When createArrayOf is used to create an array object that maps to a primitive data type, then it is implementation-defined whether the Array object is an array of that primitive data type or an array of Object.

**Note:** The JDBC driver is responsible for mapping the elements Object array to the default JDBC SQL type defined in java.sql.Types for the given class of Object. The default mapping is specified in Appendix B of the JDBC specification. If the resulting JDBC type is not the appropriate type for the given typeName then it is implementation defined whether an SQLException is thrown or the driver supports the resulting conversion.

**Parameters:**typeName - the SQL name of the type the elements of the array map to. The typeName is a database-specific name which may be the name of a built-in type, a user-defined type or a standard SQL type supported by this database. This is the value returned by Array.getBaseTypeNameelements - the elements that populate the returned object **Returns:**an Array object whose elements map to the specified SQL type **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database error occurs, the JDBC type is not appropriate for the typeName and the conversion is not supported, the typeName is null or this method is called on a closed connection [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this data type**Since:** 1.6

### createStruct

[Struct](http://docs.google.com/java/sql/Struct.html) **createStruct**([String](http://docs.google.com/java/lang/String.html) typeName,  
 [Object](http://docs.google.com/java/lang/Object.html)[] attributes)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Factory method for creating Struct objects.

**Parameters:**typeName - the SQL type name of the SQL structured type that this Struct object maps to. The typeName is the name of a user-defined type that has been defined for this database. It is the value returned by Struct.getSQLTypeName.attributes - the attributes that populate the returned object **Returns:**a Struct object that maps to the given SQL type and is populated with the given attributes **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database error occurs, the typeName is null or this method is called on a closed connection [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this data type**Since:** 1.6

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Connection.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/sql/Clob.html)   [**NEXT CLASS**](http://docs.google.com/java/sql/DatabaseMetaData.html) | [**FRAMES**](http://docs.google.com/index.html?java/sql/Connection.html)    [**NO FRAMES**](http://docs.google.com/Connection.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#2et92p0) | DETAIL: [FIELD](#3dy6vkm) | CONSTR | [METHOD](#26in1rg) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).