Top of Form

#### What is DUAL table?

The DUAL is special one row, one column table present by default in all Oracle databases. The owner of DUAL is SYS (SYS owns the data dictionary, therefore DUAL is part of the data dictionary.) but DUAL can be accessed by every user. The table has a single VARCHAR2(1) column called DUMMY that has a value of 'X'. MySQL allows DUAL to be specified as a table in queries that do not need data from any tables. In SQL Server DUAL table does not exist, but you could create one.

The DUAL table was created by Charles Weiss of Oracle corporation to provide a table for joining in internal views.

See the following commands :

**The following command displays the structure of DUAL table :**

DESC DUAL;

Output:

Name Null? Type

--------------------------- ------

DUMMY VARCHAR2(1)

**The following command displays the content of the DUAL table :**

SELECT \* FROM DUAL;

Output:

DUMMY

----------

X

**The following command displays the number of rows of DUAL table :**

SELECT COUNT(\*) FROM DUAL;

Output:

COUNT(\*)

----------

1

**The following command displays the string value from the DUAL table :**

SELECT 'ABCDEF12345' FROM DUAL;

Output:

'ABCDEF1234

-----------

ABCDEF12345

**The following command displays the numeric value from the DUAL table :**

SELECT 123792.52 FROM DUAL;

Output:

123792.52

----------

123792.52

**The following command tries to delete all rows from the DUAL table :**

DELETE FROM DUAL;

Output:

DELETE FROM DUAL

\*

ERROR at line 1:

ORA-01031: insufficient privileges

**The following command tries to remove all rows from the DUAL table :**

TRUNCATE TABLE DUAL;

Note : The DELETE command is used to remove rows from a table. After performing a DELETE operation you need to COMMIT or ROLLBACK the transaction to make the change permanent or to undo it. TRUNCATE removes all rows from a table. The operation cannot be rolled back.

Output:

TRUNCATE TABLE DUAL

\*

ERROR at line 1:

ORA-00942: table or view does not exist

**The following command select two rows from dual :**

SELECT dummy FROM DUAL

UNION ALL

SELECT dummy FROM DUAL;

Output

DUMMY

----------

X

X

**Example - 1**

You can also check the system date from the DUAL table using the following statement :

SELECT sysdate FROM DUAL ;

Output:

SYSDATE

---------

11-DEC-10

**Example - 2**

You can also check the arithmetic calculation from the DUAL table using the following statement :

SELECT 15+10-5\*5/5 FROM DUAL;

Output:

15+10-5\*5/5

-----------

20

**Example - 3**

Following code display the numbers 1..10 from DUAL :

SELECT level

FROM DUAL

CONNECT BY level <=10;

Output:

LEVEL

----------

1

2

3

4

5

6

7

8

9

10

**Example - 4**

In the following code, DUAL involves the use of decode with NULL.

SELECT decode(null,null,1,0)

FROM DUAL;

Output:

DECODE(NULL,NULL,1,0)

---------------------

1

#### DUAL table : Oracle vs MySQL

We have already learned that DUAL is a special one row one column table. For Oracle, it is useful because Oracle doesn't allow statements like :

SELECT 15+10-5\*5/5;

Output:

SELECT 15+10-5\*5/5

\*

ERROR at line 1:

ORA-00923: FROM keyword not found where expected

But the following command will execute (see the output of the previous example) :

SELECT 15+10-5\*5/5 FROM DUAL;

In case of MySQL the following command will execute :

SELECT 15+10-5\*5/5;

Output:

mysql no dual result

Bottom of Form

Top of Form

#### ****SQL | SEQUENCES****

Sequence is a set of integers 1, 2, 3, … that are generated and supported by some database systems to produce unique values on demand.

* A sequence is a user defined schema bound object that generates a sequence of numeric values.
* Sequences are frequently used in many databases because many applications require each row in a table to contain a unique value and sequences provides an easy way to generate them.
* The sequence of numeric values is generated in an a**scending or descending order** at defined intervals and can be configured to restart when exceeds max\_value.

**Syntax:**

CREATE SEQUENCE sequence\_name

START WITH initial\_value

INCREMENT BY increment\_value

MINVALUE minimum value

MAXVALUE maximum value

CYCLE|NOCYCLE ;

**sequence\_name:** Name of the sequence.

**initial\_value:** starting value from where the sequence starts.

Initial\_value should be greater than or equal

to minimum value and less than equal to maximum value.

**increment\_value:** Value by which sequence will increment itself.

Increment\_value can be positive or negative.

**minimum\_value:** Minimum value of the sequence.

**maximum\_value:** Maximum value of the sequence.

**cycle:** When sequence reaches its set\_limit

it starts from beginning.

**nocycle:** An exception will be thrown

if sequence exceeds its max\_value.

**Example**

Following is the sequence query creating sequence in ascending order.

* **Example 1:**
* CREATE SEQUENCE sequence\_1
* start with 1
* increment by 1
* minvalue 0
* maxvalue 100
* cycle;
* Above query will create a sequence named sequence\_1.Sequence will start from 1 and will be incremented by 1 having maximum value 100. Sequence will repeat itself from start value after exceeding 100.
* **Example 2:**
* Following is the sequence query creating sequence in descending order.
* CREATE SEQUENCE sequence\_2
* start with 100
* increment by -1
* minvalue 1
* maxvalue 100
* cycle;
* Above query will create a sequence named sequence\_2.Sequence will start from 100 and should be less than or equal to maximum value and will be incremented by -1 having minimum value 1.
* **Example to use sequence :** create a table named students with columns as id and name.
* CREATE TABLE students
* (
* ID number(10),
* NAME char(20)
* );
* Now insert values into table
* INSERT into students VALUES(sequence\_1.nextval,'Ramesh');
* INSERT into students VALUES(sequence\_1.nextval,'Suresh');
* where sequence\_1.nextval will insert id’s in id column in a sequence as defined in sequence\_1.
* **Output:**
* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* | ID | NAME |
* ------------------------
* | 1 | Ramesh |
* | 2 | Suresh |
* ----------------------

Bottom of Form

Top of Form

#### Using basic data types

The Microsoft JDBC Driver for SQL Server uses the JDBC basic data types to convert the SQL Server data types to a format that can be understood by the Java programming language, and vice versa. The JDBC driver provides support for the JDBC 4.0 API, which includes the SQLXML data type, and National (Unicode) data types, such as NCHAR, NVARCHAR, LONGNVARCHAR, and NCLOB.

#### Data type mappings

The following table lists the default mappings between the basic SQL Server, JDBC, and Java programming language data types:

1 To use java.sql.Time with the time SQL Server type, you must set the sendTimeAsDatetime connection property to false.

2 You can programmatically access values of datetimeoffset with [DateTimeOffset Class](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/datetimeoffset-class?view=sql-server-ver15).

3 Note that java.sql.Timestamp values can no longer be used to compare values from a datetime column starting from SQL Server 2016. This limitation is due to a server-side change that converts datetime to datetime2 differently, resulting in non-equitable values. The workaround to this issue is to either change datetime columns to datetime2(3), use String instead of java.sql.Timestamp, or change database compatibility level to 120 or below.

#### Retrieving data as a string

If you have to retrieve data from a data source that maps to any of the JDBC basic data types for viewing as a string, or if strongly typed data is not required, you can use the [getString](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/getstring-method-sqlserverresultset?view=sql-server-ver15) method of the [SQLServerResultSet](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/sqlserverresultset-class?view=sql-server-ver15) class, as in the following:

try(Statement stmt = con.createStatement();) {

ResultSet rs = stmt.executeQuery("SELECT lname, job\_id FROM employee WHERE (lname = 'Brown')");

rs.next();

short empJobID = rs.getShort("job\_id");

}

#### Retrieving data by data type

If you have to retrieve data from a data source, and you know the type of data that is being retrieved, use one of the get<Type> methods of the SQLServerResultSet class, also known as the getter methods. You can use either a column name or a column index with the get<Type> methods, as in the following:

try(Statement stmt = con.createStatement();) {

ResultSet rs = stmt.executeQuery("SELECT lname, job\_id FROM employee WHERE (lname = 'Brown')");

rs.next();

short empJobID = rs.getShort("job\_id");

}

#### Updating data by data type

If you have to update the value of a field in a data source, use one of the update<Type> methods of the SQLServerResultSet class. In the following example, the [updateInt](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/updateint-method-sqlserverresultset?view=sql-server-ver15) method is used in conjunction with the [updateRow](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/updaterow-method-sqlserverresultset?view=sql-server-ver15) method to update the data in the data source:

try (Statement stmt = con.createStatement(ResultSet.TYPE\_SCROLL\_SENSITIVE, ResultSet.CONCUR\_UPDATABLE);) {

ResultSet rs = stmt.executeQuery("SELECT lname, job\_id FROM employee WHERE (lname = 'Brown')");

rs.next();

int empJobID = rs.getInt(2);

empJobID++;

rs.first();

rs.updateInt(2, empJobID);

rs.updateRow();

}

**Note:** The JDBC driver cannot update a SQL Server column with a column name that is more than 127 characters long. If an update to a column whose name is more than 127 characters is attempted, an exception is thrown.

#### Updating data by parameterized query

If you have to update data in a data source by using a parameterized query, you can set the data type of the parameters by using one of the set<Type> methods of the [SQLServerPreparedStatement](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/sqlserverpreparedstatement-class?view=sql-server-ver15) class, also known as the setter methods. In the following example, the [prepareStatement](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/preparestatement-method-sqlserverconnection?view=sql-server-ver15) method is used to pre-compile the parameterized query, and then the [setString](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/setstring-method-sqlserverpreparedstatement?view=sql-server-ver15) method is used to set the string value of the parameter before the [executeUpdate](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/executeupdate-method?view=sql-server-ver15) method is called.

try(PreparedStatement pstmt = con.prepareStatement("UPDATE employee SET fname = ? WHERE (lname = 'Brown')");) {

String name = "Bob";

pstmt.setString(1, name);

int rowCount = pstmt.executeUpdate();

}

For more information about parameterized queries, see [Using an SQL statement with parameters](https://docs.microsoft.com/en-us/sql/connect/jdbc/using-an-sql-statement-with-parameters?view=sql-server-ver15).

#### Passing parameters to a stored procedure

If you have to pass typed parameters into a stored procedure, you can set the parameters by index or name by using one of the set<Type> methods of the [SQLServerCallableStatement](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/sqlservercallablestatement-class?view=sql-server-ver15) class. In the following example, the [prepareCall](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/preparecall-method-sqlserverconnection?view=sql-server-ver15) method is used to set up the call to the stored procedure, and then the [setString](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/setstring-method-sqlservercallablestatement?view=sql-server-ver15) method is used to set the parameter for the call before the [executeQuery](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/executequery-method-sqlserverstatement?view=sql-server-ver15) method is called.

try(CallableStatement cstmt = con.prepareCall("{call employee\_jobid(?)}");) {

String lname = "Brown";

cstmt.setString(1, lname);

ResultSet rs = cstmt.executeQuery();

}

**Note:** In this example, a result set is returned with the results of running the stored procedure.

For more information about using the JDBC driver with stored procedures and input parameters, see [Using a stored procedure with input parameters](https://docs.microsoft.com/en-us/sql/connect/jdbc/using-a-stored-procedure-with-input-parameters?view=sql-server-ver15).

#### Retrieving parameters from a stored procedure

If you have to retrieve parameters back from a stored procedure, you must first register an out parameter by name or index by using the [registerOutParameter](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/registeroutparameter-method-sqlservercallablestatement?view=sql-server-ver15) method of the SQLServerCallableStatement class, and then assign the returned out parameter to an appropriate variable after you run the call to the stored procedure. In the following example, the prepareCall method is used to set up the call to the stored procedure, the registerOutParameter method is used to set up the out parameter, and then the [setString](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/setstring-method-sqlservercallablestatement?view=sql-server-ver15) method is used to set the parameter for the call before executeQuery method is called. The value that is returned by the out parameter of the stored procedure is retrieved by using the [getShort](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/getshort-method-sqlservercallablestatement?view=sql-server-ver15) method.

try(CallableStatement cstmt = con.prepareCall("{call employee\_jobid (?, ?)}");) {

cstmt.registerOutParameter(2, java.sql.Types.SMALLINT);

String lname = "Brown";

cstmt.setString(1, lname);

ResultSet rs = cstmt.executeQuery();

short empJobID = cstmt.getShort(2);

}

**Note:** In addition to the returned out parameter, a result set might also be returned with the results of running the stored procedure.

Bottom of Form

Top of Form

**Reference Links:**

https://www.geeksforgeeks.org/sql-sequences/

https://www.tutorialspoint.com/sql/sql-using-sequences.htm

https://www.studytonight.com/dbms/sql-sequences.php

https://docs.microsoft.com/en-us/sql/t-sql/statements/create-sequence-transact-sql

https://docs.oracle.com/cd/B28359\_01/server.111/b28286/statements\_6015.htm

https://www.w3resource.com/sql/sql-dual-table.php

https://www.geeksforgeeks.org/dual-table-in-sql/

**Video Links:**

https://youtu.be/zADj0k0waFY

https://youtu.be/SO\_wA88hpPs

Bottom of Form