**St. Xavier’s College**

**Maitighar, Kathmandu**



**DATABASE MANAGEMENT SYSTEM**

**ASSIGNMENT #6**

**Submitted By**

Shaurabh Chapagain

013BSCCSIT037

**Submitted To**

|  |  |
| --- | --- |
| Er. Sanjay Kumar Yadav  Lecturer, St. Xavier’s College |  |

Department of Computer Science

Date of Submission: 10th September, 2015

**Join**

The JOIN operations, which are among the possible [Table Expression](https://docs.oracle.com/javadb/10.8.3.0/ref/rreftableexpression.html#rreftableexpression)s in a [FROM clause](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqlj21583.html#rrefsqlj21583), perform joins between two tables. Join is a combination of a Cartesian product followed by a selection process. A Join operation pairs two tuples from different relations, if and only if a given join condition is satisfied.

**Theta Join**

Theta join combines tuples from different relations provided they satisfy the theta condition. The join condition is denoted by the symbol θ.

Notation

R1 ⋈θ R2

R1 and R2 are relations having attributes (A1, A2, ..,An) and (B1, B2,.. ,Bn) such that the attributes don’t have anything in common, that is R1 ∩ R2 = Φ.

## Natural Join (⋈)

Natural join does not use any comparison operator. It does not concatenate the way a Cartesian product does. We can perform a Natural Join only if there is at least one common attribute that exists between two relations. In addition, the attributes must have the same name and domain. Natural join acts on those matching attributes where the values of attributes in both the relations are same.

SELECT \*

FROM customers

NATURAL JOIN orders;

# RIGHT OUTER JOIN operation

A RIGHT OUTER JOIN is one of the [JOIN operations](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqlj29840.html#rrefsqlj29840) that allow you to specify a JOIN clause. It preserves the unmatched rows from the second (right) table, joining them with a NULL in the shape of the first (left) table. A LEFT OUTER JOIN B is equivalent to B RIGHT OUTER JOIN A, with the columns in a different order.

Syntax

[*TableExpression*](https://docs.oracle.com/javadb/10.8.3.0/ref/rreftableexpression.html#rreftableexpression) **RIGHT [ OUTER ] JOIN** [*TableExpression*](https://docs.oracle.com/javadb/10.8.3.0/ref/rreftableexpression.html#rreftableexpression)

**{**

**ON** booleanExpression **|**

[***USING*** *clause*](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljusing.html#rrefsqljusing)

**}**

The scope of expressions in the ON clause includes the current tables and any tables in query blocks outer to the current SELECT. The ON clause can reference tables not being joined and does not have to reference either of the tables being joined (though typically it does).

# LEFT OUTER JOIN operation

A LEFT OUTER JOIN is one of the [JOIN operations](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqlj29840.html#rrefsqlj29840) that allow you to specify a join clause. It preserves the unmatched rows from the first (left) table, joining them with a NULL row in the shape of the second (right) table.

## Syntax

[*TableExpression*](https://docs.oracle.com/javadb/10.8.3.0/ref/rreftableexpression.html#rreftableexpression) **LEFT [ OUTER ] JOIN** [*TableExpression*](https://docs.oracle.com/javadb/10.8.3.0/ref/rreftableexpression.html#rreftableexpression)

**{**

**ON** booleanExpression **|**

[*USING clause*](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljusing.html#rrefsqljusing)

**}**

The scope of expressions in either the ON clause includes the current tables and any tables in query blocks outer to the current SELECT. The ON clause can reference tables not being joined and does not have to reference either of the tables being joined.

# INNER JOIN operation

An INNER JOIN is a [JOIN operation](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqlj29840.html#rrefsqlj29840) that allows you to specify an explicit join clause.

## Syntax

[*TableExpression*](https://docs.oracle.com/javadb/10.8.3.0/ref/rreftableexpression.html#rreftableexpression) **[ INNER ] JOIN** [*TableExpression*](https://docs.oracle.com/javadb/10.8.3.0/ref/rreftableexpression.html#rreftableexpression)

**{**

**ON** booleanExpression **|**

[*USING clause*](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljusing.html#rrefsqljusing)

**}**

You can specify the join clause by specifying ON with a boolean expression.

The scope of expressions in the ON clause includes the current tables and any tables in outer query blocks to the current SELECT. In the following example, the ON clause refers to the current tables:

**SELECT \***

**FROM SAMP.EMPLOYEE INNER JOIN SAMP.STAFF**

**ON EMPLOYEE.SALARY < STAFF.SALARY**

The ON clause can reference tables not being joined and does not have to reference either of the tables being joined (though typically it does).

**Rename operation:**

Following statement renames one or more tables.

RENAME TABLE *tbl\_name* TO *new\_tbl\_name*

[, *tbl\_name2* TO *new\_tbl\_name2*] .

The rename operation is done atomically, which means that no other session can access any of the tables while the rename is running. For example, if you have an existing table old\_table, you can create another tablenew\_table that has the same structure but is empty, and then replace the existing table with the empty one as follows (assuming that backup\_table does not already exist):

CREATE TABLE new\_table (...);

RENAME TABLE old\_table TO backup\_table, new\_table TO old\_table;

If the statement renames more than one table, renaming operations are done from left to right. If two table names to be swapped, following operation is done (assuming that tmp\_table does not already exist):

RENAME TABLE old\_table TO tmp\_table,

new\_table TO old\_table,

tmp\_table TO new\_table;

As long as two databases are on the same file system, you can use [RENAME TABLE](https://dev.mysql.com/doc/refman/5.0/en/rename-table.html) to move a table from one database to another:

RENAME TABLE *current\_db.tbl\_name* TO *other\_db.tbl\_name;*

**Assignment Operation:**

| Name | Description |
| --- | --- |
| [=](https://dev.mysql.com/doc/refman/5.1/en/assignment-operators.html#operator_assign-equal) | Assign a value (as part of a [SET](https://dev.mysql.com/doc/refman/5.1/en/set-statement.html) statement, or as part of the SET clause in an [UPDATE](https://dev.mysql.com/doc/refman/5.1/en/update.html) statement) |
| [:=](https://dev.mysql.com/doc/refman/5.1/en/assignment-operators.html#operator_assign-value) | Assign a value |

Table 12.5 Assignment Operators

* **[:=](https://dev.mysql.com/doc/refman/5.1/en/assignment-operators.html" \l "operator_assign-value)**

Assignment operator causes the user variable on the left hand side of the operator to take on the value to its right. The value on the right hand side may be a literal value, another variable storing a value, or any legal expression that yields a scalar value, including the result of a query (provided that this value is a scalar value). You can perform multiple assignments in the same [SET](https://dev.mysql.com/doc/refman/5.1/en/set-statement.html) statement. You can perform multiple assignments in the same statement-

Unlike [=](https://dev.mysql.com/doc/refman/5.1/en/assignment-operators.html#operator_assign-equal), the [:=](https://dev.mysql.com/doc/refman/5.1/en/assignment-operators.html#operator_assign-value) operator is never interpreted as a comparison operator. This means you can use [:=](https://dev.mysql.com/doc/refman/5.1/en/assignment-operators.html#operator_assign-value) in any valid SQL statement (not just in [SET](https://dev.mysql.com/doc/refman/5.1/en/set-statement.html) statements) to assign a value to a variable.

mysql> SELECT @var1, @var2;

-> NULL, NULL

mysql> SELECT @var1 := 1, @var2;

-> 1, NULL

mysql> SELECT @var1, @var2;

-> 1, NULL

mysql> SELECT @var1, @var2 := @var1;

-> 1, 1

mysql> SELECT @var1, @var2;

-> 1, 1

mysql> SELECT @var1:=COUNT(\*) FROM t1;

-> 4

mysql> SELECT @var1;

-> 4

Value assignments can be made using [:=](https://dev.mysql.com/doc/refman/5.1/en/assignment-operators.html#operator_assign-value) in other statements besides [SELECT](https://dev.mysql.com/doc/refman/5.1/en/select.html), such as [UPDATE](https://dev.mysql.com/doc/refman/5.1/en/update.html), as shown here:

mysql> SELECT @var1;

-> 4

mysql> SELECT \* FROM t1;

-> 1, 3, 5, 7

mysql> UPDATE t1 SET c1 = 2 WHERE c1 = @var1:= 1;

Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> SELECT @var1;

-> 1

mysql> SELECT \* FROM t1;

-> 2, 3, 5, 7

While it is also possible both to set and to read the value of the same variable in a single SQL statement using the[:=](https://dev.mysql.com/doc/refman/5.1/en/assignment-operators.html#operator_assign-value) operator, this is not recommended. [Section 9.4, “User-Defined Variables”](https://dev.mysql.com/doc/refman/5.1/en/user-variables.html), explains why you should avoid doing this.

* **[=](https://dev.mysql.com/doc/refman/5.1/en/assignment-operators.html" \l "operator_assign-equal)**

This operator is used to perform value assignments in two cases, described in the next two paragraphs.

Within a [SET](https://dev.mysql.com/doc/refman/5.1/en/set-statement.html) statement, = is treated as an assignment operator that causes the user variable on the left hand side of the operator to take on the value to its right. (In other words, when used in a [SET](https://dev.mysql.com/doc/refman/5.1/en/set-statement.html) statement, = is treated identically to [:=](https://dev.mysql.com/doc/refman/5.1/en/assignment-operators.html#operator_assign-value).) The value on the right hand side may be a literal value, another variable storing a value, or any legal expression that yields a scalar value, including the result of a query (provided that this value is a scalar value). You can perform multiple assignments in the same [SET](https://dev.mysql.com/doc/refman/5.1/en/set-statement.html) statement.

In the SET clause of an [UPDATE](https://dev.mysql.com/doc/refman/5.1/en/update.html) statement, = also acts as an assignment operator; in this case, however, it causes the column named on the left hand side of the operator to assume the value given to the right, provided anyWHERE conditions that are part of the [UPDATE](https://dev.mysql.com/doc/refman/5.1/en/update.html) are met. You can make multiple assignments in the same SETclause of an [UPDATE](https://dev.mysql.com/doc/refman/5.1/en/update.html) statement.

In any other context, = is treated as a [comparison operator](https://dev.mysql.com/doc/refman/5.1/en/comparison-operators.html#operator_equal).

mysql> SELECT @var1, @var2;

-> NULL, NULL

mysql> SELECT @var1 := 1, @var2;

-> 1, NULL

mysql> SELECT @var1, @var2;

-> 1, NULL

mysql> SELECT @var1, @var2 := @var1;

-> 1, 1

mysql> SELECT @var1, @var2;

-> 1, 1

**The division operation:**

* It is denoted as ÷.

Let r(R) and s(S) be relations.  
  
**r ÷ s: -**  The result consists of the restrictions of tuples in r to the attribute names unique to R, i.e. in the Header of r but not in the Header of s, for which it holds that all their combinations with tuples in s are present in r.

Example:  
  
Relation or table "r":-

Code:

| **A** | **B** |

+--------+-------+

| a | 1 |

| b | 2 |

| a | 2 |

| p | 3 |

| p | 4 |

+--------+-------+

Relation or table "s":-

Code:

+------+

| B |

+------+

| 2 |

| 3 |

+------+

Therefore, r ÷ s

Code:

+------+

| A |

+------+

| b |

| a |

| p |

+------+