**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**Database Management System**

**Lab Assignment #6**

**Submitted by:**

Binod Marikhu  
013BSCCSIT013

**Submitted to:**

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

**Date of Submission:** September 10, 2015

**JOIN**

Join is a combination of a Cartesian product followed by a selection process. Join combine records from two or more database tables into one row source, one set of rows with the same columns. The columns can be formed from joined tables or using expressions and built-in user-define functions. A join operation pairs two tuples from different relations provided they satisfy the theta condition.

1. **Theta (θ) Join**

The join condition is denoted by the symbol θ. Theta join combines tuples from different relations provided they satisfy the theta condition. In theta join, the condition is applied only on input relation(s) and then only those selected rows are used in the cross product to be merged and included in the output. i.e. in normal cross product all the rows of one relation are mapped/merged with all the rows of second relation, but only selected rows of a relation are made cross product with second relation. It is denoted as RxS where R and S are two relations. Select operation is carried on one relation and then selected rows are cross product with all the rows of second relation.

**NATURAL JOIN**

The NATURAL JOIN matches every row in one table against every row in another based on common values found in columns that share a common name and data type. A Natural Join acts like a nested loop where you match a key from the outer loop against a key from the inner loop. It discards all non-matching values from the outer and inner loops.

**Rules**

1. A natural join can’t have an explicit join statement in the SQL statement.
2. A natural join reads the database catalog to find columns that share a name and implicitly constructs a join statement.
3. A natural join returns the intersection of two sets where all the values in columns that have matching names are compared and found to equal one another.
4. **Right Join**

Return all rows from right table, and the matched rows from the left table.

1. **Left Join**

Returns all rows from left table, and the matched rows from the right table.

1. **Inner Join**

Returns all rows when there is at least one match in both tables.

Example:

**SELECT** \*

**FROM** table\_name1

**INNER** **JOIN** table\_name2 **ON** table\_name1.attribute\_name = table\_name2.attribute\_name;

**RENAME OPERATION**

The REMANE operatopm returns an existing relation under a new name. This is a mechanism used for renaming both relations and attributes, this renaming is highly useful when one has to compare a set of tuples of the same relation with other tuples of that relation. It has the following general format :  
  
old\_name as new-name  
  
The as clauses can appear in both the select and from clauses. For example if we want the attribute name loan \_no to be replaced with the name loan\_id, we can write the query as  
  
Select distinct cust\_name, borrower.loan\_no as loan\_no  
From borrower, loan  
Where borrower.loan\_no = loan.loan.no AND  
Branch\_name = KR CIRCLE

**ASSIGNMENT OPERATION**

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

This 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 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;

[=](https://dev.mysql.com/doc/refman/5.1/en/assignment-operators.html#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 any WHERE 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 SET clause 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

[=](https://dev.mysql.com/doc/refman/5.1/en/assignment-operators.html#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 any WHERE 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 SET clause 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

**DIVISION OPERATION**

**ADDITIONAL OPERATIONS**

1. **Set-Intersection Operation**
2. **Natural Join Operation**

A NATURAL JOIN is a JOIN operation that creates an implicit join clause for you based on the common columns in the two tables being joined. Common columns are columns that have the same name in both tables.

A NATURAL JOIN can be an INNER join, a LEFT OUTER join, or a RIGHT OUTER join. The default is INNER join.

If the SELECT statement in which the NATURAL JOIN operation appears has an asterisk (\*) in the select list, the asterisk will be expanded to the following list of columns (in this order):

• All the common columns

• Every column in the first (left) table that is not a common column

• Every column in the second (right) table that is not a common column

An asterisk qualified by a table name (for example, COUNTRIES.\*) will be expanded to every column of that table that is not a common column.

If a common column is referenced without being qualified by a table name, the column reference points to the column in the first (left) table if the join is an INNER JOIN or a LEFT OUTER JOIN. If it is a RIGHT OUTER JOIN, unqualified references to a common column point to the column in the second (right) table.

Syntax

TableExpression NATURAL [ { LEFT | RIGHT } [ OUTER ] | INNER ] JOIN { TableViewOrFunctionExpression | ( TableExpression ) }