**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



DATABASE MANAGEMENT SYSTEM

Lab Assignment #6

**Submitted by:**

Lokendra Puri  
013BSCCSIT023

**Submitted to:**

Er. Sanjay Kr. Yadav

Lecturer  
Department of Computer Science

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

1. Join

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.

1. Theta join

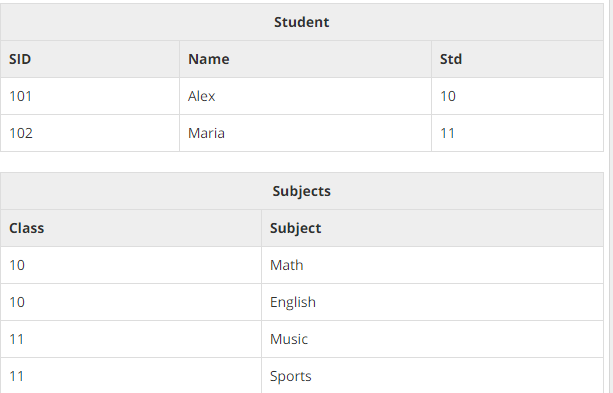
Theta join combines tuples from different relations provided they satisfy the theta condition. The join condition is denoted by the symbol **θ**.

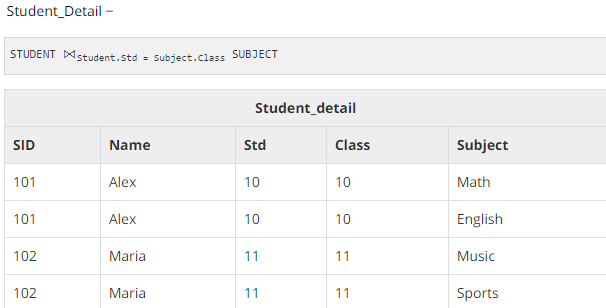
**Notation**

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.

Theta join can use all kinds of comparison operators.

****

****

1. 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.

* right join

All the tuples from the Right relation, S, are included in the resulting relation. If there are tuples in S without any matching tuple in R, then the R-attributes of resulting relation are made NULL.

Syntax:

SELECT column\_name(s)  
FROM table1  
RIGHT JOIN table2  
ON table1.column\_name=table2.column\_name;

* left join

All the tuples from the Left relation, R, are included in the resulting relation. If there are tuples in R without any matching tuple in the Right relation S, then the S-attributes of the resulting relation are made NULL.

Syntax:

SELECT column\_name(s)  
FROM table1  
LEFT JOIN table2  
ON table1.column\_name=table2.column\_name;

* inner join

The INNER JOIN creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate. The query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row.

Syntax:

The basic syntax of **INNER JOIN** is as follows:

SELECT table1.column1, table2.column2...

FROM table1

INNER JOIN table2

ON table1.common\_field = table2.common\_field;

Consider the following two tables, (a) CUSTOMERS table is as follows:

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

| ID | NAME | AGE | ADDRESS | SALARY |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

(b) Another table is ORDERS as follows:

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

| OID | DATE | ID | AMOUNT |

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

| 102 | 2009-10-08 00:00:00 | 3 | 3000 |

| 100 | 2009-10-08 00:00:00 | 3 | 1500 |

| 101 | 2009-11-20 00:00:00 | 2 | 1560 |

| 103 | 2008-05-20 00:00:00 | 4 | 2060 |

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

Now, let us join these two tables using INNER JOIN as follows:

SQL> SELECT ID, NAME, AMOUNT, DATE

FROM CUSTOMERS

INNER JOIN ORDERS

ON CUSTOMERS.ID = ORDERS.CUSTOMER\_ID;

This would produce the following result:

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

| ID | NAME | AMOUNT | DATE |

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

| 3 | kaushik | 3000 | 2009-10-08 00:00:00 |

| 3 | kaushik | 1500 | 2009-10-08 00:00:00 |

| 2 | Khilan | 1560 | 2009-11-20 00:00:00 |

| 4 | Chaitali | 2060 | 2008-05-20 00:00:00 |

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

1. Rename operation

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter **rho** *ρ*.

**Notation** − *ρ* x (E)

Where the result of expression **E** is saved with name of **x**.

1. Assignment operation
2. Division operation
3. 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 |

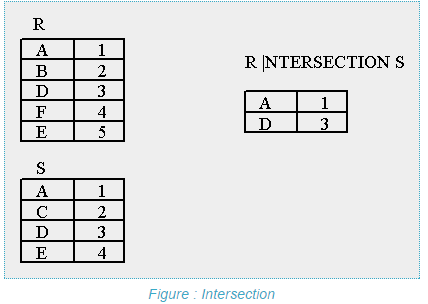
| a |

| p |

1. Additional operation

1. Set intersection operation

Intersect operation is used to combine two SELECT statements, but it only returns the records which are common from both SELECT statements. In case of Intersect the number of columns and data type must be same. MySQL does not support INTERSECT operator.



1. Natural join operation

If the two relations being joined have exactly one attribute (domain) name in common, then we assume that the single attribute in common is the one being compared to see if a new tuple will be inserted in the result.

Assuming that we've augmented the domain names in our lab database so that we use MachineName, PrinterName, ServerName, and UserName in place of the generic domain "Name", then

**join(Workstations,Printers)**

is a natural join, on the shared attribute name Room. The result is a relation of all workstation/printer attribute pairs that are in the same room.

it is a binary operation and a combination of certain selections and a Cartesian product into one operation.

* It is denoted as |X| .
* It is associative.

It forms a Cartesian product of its two arguments.  
Then performs a selection forcing equality on those attributes those appear in both the relations.  
And finally removes duplicates attributes.  
  
r(R): r is a relation with attributes R.  
s(S): s is a relation with attributes S.  
  
If R **∩**S = Ф i.e. they have no attributes in common then **r |X| s = r X s**Example:-  
Table "r":-

Code:

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

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

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

| a | b | c |

| d | e | f |

| g | h | i |

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

Table "s" :-

Code:

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

| **B** | **D** |

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

| b | g |

| p | r |

| e | t |

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

Therefore, r |X| s :-

Code:

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

| **A**  | **B** | **C** | **D**  |

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

| a | b | c | g |

| d | e | f | t |

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

i.e;  
if r (A, B, C), s (B, D) then

Code:

**r|X|s** = **Π r.A, r.B, r.C, s.D (σ r.B = s.B (r X s))**