**ST.XAVIER’S COLLEGE**

MAITIGHAR, KATHMANDU



Database Management System

Assignment #6

Submitted By:

Aabhash Dhakal

013BSCCSIT001

2nd year/ 4th semester

Submitted to:

|  |  |
| --- | --- |
| Er. Sanjay Kumar Yadav  Lecturer  Department of Computer Science |  |

# 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.

Product(PName, Price)

====================

Laptop, 1500

Car, 20000

Airplane, 3000000

Component(PName, CName, Cost)

=============================

Laptop, CPU, 500

Laptop, hdd, 300

Laptop, case, 700

Car, wheels, 1000

Can be JOINed to form:

| PNAME | PRICE | CNAME | COST |

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

| Laptop | 1500 | CPU | 500 |

| Laptop | 1500 | hdd | 300 |

| Laptop | 1500 | case | 700 |

| Car | 20000 | wheels | 1000 |

* 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**. Theta Join is denoted by:

R1 ⋈θ R2

A theta join allows for arbitrary comparison relationships

The general theta-join form is:

<Table\_1.Column> relator <Table\_2.Column>

# 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.

A natural join is an equijoin on attributes that have the same name in each relationship.

Additionally, a natural join removes the duplicate columns involved in the equality comparison so only 1 of each compared column remains.

TableA

id firstName lastName

.......................................

1 arun prasanth

2 ann antony

3 sruthy abc

6 new abc

TableB

id2 age Place

................

1 24 kerala

2 24 usa

3 25 ekm

5 24 chennai

# Right Join

Right Join operation gets all records from the RIGHT linked table but if you have selected some columns from the LEFT table, if there is no related records, these columns will contain NULL.

firstName lastName age Place

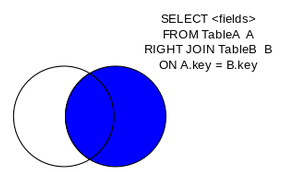
....................................................

arun prasanth 24 kerala

ann antony 24 usa

sruthy abc 25 ekm

NULL NULL 24 chennai



# Left Join

The result set of a Left Outer Join includes all the rows from the left table specified in the LEFT OUTER clause, not just the ones in which the joined columns match. When a row in the left table has no matching rows in the right table, the associated result set row contains null values for all select list columns coming from the right table.

firstName lastName age Place

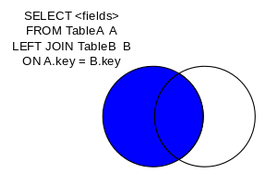
....................................................

arun prasanth 24 kerala

ann antony 24 usa

sruthy abc 25 ekm

new abc NULL NULL



# Inner Join

The Inner Join operation returns rows when there is a match in both tables. These include equi-joins and natural joins.  
Inner Joins use a comparison operator to match rows from two tables based on the values in common columns from each table.

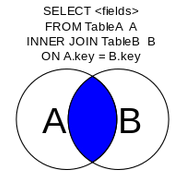
firstName lastName age Place

..............................................

arun prasanth 24 kerala

ann antony 24 usa

sruthy abc 25 ekm



# Rename Operation

The rename operation is used for renaming. It is denoted by **ρ**.

E **:** relational algebra expression   
**ρ x (E):**returns the result of expression E under the name x.  
**ρ x (A1, A2, A3… An) (E):**returns the result of expression E under the name x with attributes renamed to A1, A2, A3… An.

# Assignment Operation

The assignment operation (←) provides a convenient way to express complex queries.

* + Write query as a sequential program consisting of
    - a series of assignments
    - followed by an expression whose value is displayed as a result of the query.
  + Assignment must always be made to a temporary relation variable.

Example: Write *r* ÷ *s* as

*temp*1← ∏*R-S* (*r*)   
 *temp*2 ← ∏*R-S* ((*temp*1 x *s*) – ∏*R-S,S (r*))  
 *result* = *temp*1 – *temp*2

# Division Operation

The division operation is a binary operation, notated as ÷, on relations r(R) and s(S) such that S ⊆ R. Intuitively, division is a “for all” query — it returns the tuples in r that “match” all of the tuples in s.

*R*(*Z*) ÷ *S*(*X*) equals maximal *T*(*Z*- *X*) which satisfies *T*(*Z*- *X*) × *S*(*X*) ⊆ *R*(*Z*)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | R | |  |  | | --- | --- | | **A** | **B** | | a | 1 | | a | 2 | | a | 3 | | b | 2 | | b | 3 | | b | 4 | | c | 1 | | c | 2 | | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | S1 | |  | | --- | | **B** | | 1 | | 2 | | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | R÷S1 | |  | | --- | | **A** | | a | | c | | | |  |  | |  |  |

1. **Additional Operations**
   1. **Set Intersection Operation**

The set intersection operator finds tuples in both the relations. It is denoted by **∩**.

Example:  
Borrower (customer-name, loan-number)  
Depositor (customer-name, account-number)  
Customer (customer-name, street-number, customer-city)  
  
List all the customers who have both a loan and an account.

Code:

**Π customer-name (Borrower) ∩ Π customer-name (Depositor)**

* 1. **Natural Join Operation**

It is a binary operation and a combination of certain selections and a Cartesian product into one operation. It is denoted as |X| .

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**