ST. XAVIER’S COLLEGE

**Maitighar,Kathmandu**

**(Affiliated to Tribhuvan University)**



**Database Management System**

**Theory Assignment #6**

**Submitted By**

Abhishek Tamrakar

B.Sc. CSIT

Year II/IV Semester

013BSCIT003

**Submitted To:**

Er. Sanjay Kumar Yadav

Lecturer,

Department of Computer Science

St. Xavier’s College

Maitighar, Kathmandu

**Submitted On:**

9thSeptember, 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. We will briefly describe various join types in the following sections.
   1. **Theta (θ) 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.

Student

|  |  |  |
| --- | --- | --- |
| SID | Name | Std |
| 101 | Alex | 10 |
| 102 | Maria | 11 |

Subjects

|  |  |
| --- | --- |
| Class | Subject |
| 10 | Math |
| 10 | English |
| 11 | Music |
| 11 | Sports |

Student\_Detail −

STUDENT ⋈Student.Std = Subject.Class SUBJECT

Student\_detail

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SID | Name | Std | Class | Subject |
| 101 | Alex | 10 | 10 | Math |
| 101 | Alex | 10 | 10 | English |
| 102 | Maria | 11 | 11 | Music |
| 102 | Maria | 11 | 11 | Sports |

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.

Courses

|  |  |  |
| --- | --- | --- |
| CID | Course | Dept |
| CS01 | Database | CS |
| ME01 | Mechanics | ME |
| EE01 | Electronics | EE |

HoD

|  |  |
| --- | --- |
| Dept | Head |
| CS | Alex |
| ME | Maya |
| EE | Mira |

Courses ⋈HoD

|  |  |  |  |
| --- | --- | --- | --- |
| Dept | CID | Course | Head |
| CS | CS01 | Database | Alex |
| ME | ME01 | Mechanics | Maya |
| EE | EE01 | Electronics | Mira |

* 1. Right join**( R S )**

⋈

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.

|  |  |  |  |
| --- | --- | --- | --- |
| **Courses HoD** | | | |
| **A** | **B** | **C** | **D** |
| 100 | Database | 100 | Alex |
| 102 | Electronics | 102 | Maya |
| --- | --- | 104 | Mira |

* 1. Left join: (R Left Outer Join S): 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.

|  |  |
| --- | --- |
| Left | |
| A | B |
| 100 | Database |
| 101 | Mechanics |
| 102 | Electronics |

|  |  |
| --- | --- |
| Right | |
| A | B |
| 100 | Alex |
| 102 | Maya |
| 104 | Mira |

Courses Left Outer Join HoD

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | D |
| 100 | Database | 100 | Alex |
| 101 | Mechanics | --- | --- |
| 102 | Electronics | 102 | Maya |

* 1. Inner join

1. **Rename operation (ρ):**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.

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.

1. **Assignment operation:**The assignment operator is one of the most intuitive to use. It assigns a value to a variable. The only confusion in using this operator could stem from its overloading. All RDBMS overload this operator with an additional function — comparison — in the SQL.

The equals operator (=) is used as an assignment in the following SQL query that updates the price (PROD\_PRICE\_N) column in the PRODUCT table, raising the existing prices by 2 percent:

UPDATE product SET prod\_price\_n

= prod\_price\_n \* 1.02 (10 row(s) affected)

And the same operator would be used for comparing values when used, for example, in the WHERE clause of an SQL statement:

UPDATE product SET prod\_price\_n

= prod\_price\_n \* 1.02 WHERE prod\_id\_n = 1880 (1 row(s)

affected)

This statement assigns a 2 percent increase to a product whose ID is 1880; in the same query, the equals operator (=) is used in its assignment and comparison capacity at the same time.

1. **Division operation:**It is denoted as ÷.

Letr(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 operations:** 
   1. Set-intersection operation: finds tuples in both the relations.

It is denoted as **∩**.

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 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))**