**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**Database Management System**

**Theory Assignment #6**

**Submitted by:**

Prashraya Hada  
013BSCCSIT027

**Submitted to:**

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

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

**Join**

It 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 understand the benefits of taking a Cartesian product of two relations, which gives us all the possible tuples that are paired together. But it might not be feasible for us in certain cases to take a Cartesian product where we encounter huge relations with thousands of tuples having a considerable large number of attributes.

**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.

**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. We discuss three types of natural joins below. They are:

1. Right join
2. Left join
3. Inner join

## Right Join: (R Right Outer Join 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. Right Outer Join Syntax is:

select column-name-list

from *table-name1*

RIGHT OUTER JOIN

*table-name2*

on table-name1.column-name = table-name2.column-name;

## 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 Outer Join syntax is:

SELECT column-name-list

from table-name1

LEFT OUTER JOIN

table-name2

on table-name1.column-name = table-name2.column-name;

**Inner Join:**

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the query. Inner Join Syntax is:

SELECT column-name-list

from table-name1

INNER JOIN

table-name2

WHERE table-name1.column-name = table-name2.column-name;

**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*. Successive renames of a variable can be collapsed into a single rename. Rename operations which have no variables in common can be arbitrarily reordered with respect to one another, which can be exploited to make successive renames adjacent so that they can be collapsed.

1. \rho_{a / b}(\rho_{b / c}(R)) = \rho_{a / c}(R)\,\!
2. \rho_{a / b}(\rho_{c / d}(R)) = \rho_{c / d}(\rho_{a / b}(R))\,\!

Notation : *ρ* x (E) where the result of expression **E** is saved with name of **x**.

**Assignment Operation**

The additional operations do not add any power to the relational algebra, but can simplify writing common queries – Set intersection – Natural join – Division – Assignment.

An assignment operator is the operator used to assign a new value to a variable, property, event or indexer element in C# programming language. Assignment operators can also be used for logical operations such as bitwise logical operations or operations on integral operands and Boolean operands.

Unlike in C++, assignment operators in C# cannot be overloaded directly, but the user-defined types can overload the operators like +, -, /, etc. This allows the assignment operator to be used with those types

**Division Algebra**

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 pressent in r.

**Additional Operations**

“Additional operations” refer to relational algebra operations that can be expressed in terms of the fundamentals — select, project, union, set-difference, cartesian-product, and rename. • The compositions of these operations are so lengthy, yet so common, that we define new operations for them, based on the fundamentals. It is kind of a mathematical “syntactic sugar.”

**Set-Intersection**

The set-intersection operation is a binary operation on relations r and s that is denoted by the traditional intersection symbol, ∩. r ∩ s results in all tuples t such that (t ∈ r) ∧ (t ∈ s). Set-intersection is defined in terms of set-difference: r ∩ s = r − (r − s)

Thus, set-intersection must follow the same compatibility rules as set-difference: same arity, corresponding domains.

**Natural Join**

The natural-join operation is a binary operation on relations r(R) and s(S) that is denoted by the symbol ./. Intuitively, a natural-join “matches” the tuples of r with the tuples of s based on attributes that are both in r and s. If we take the relational schemas R and S as sets of attributes, then we can define “attributes that are in both r and s” as R ∩ S = {A1, A2, . . . , An}. With that, we can formally define r ./ s as: r ./ s = ΠR ∪ S(σr.A1 = s.A1 ∧ r.A2 = s.A2 ∧ ... ∧ r.An = s.An (r × s))

Note that R ∪ S removes duplicate attribute names, so r ./ s will only have one attribute Ak ∀Ak ∈ R ∩ S. Natural join is associative — that is, (a ./ b) ./ c = a ./ (b ./ c). • When r and s do not have any common attributes — i.e., R ∩ S = ∅ — then r ./ s = r × s.

**REFERENCES:**

[1] <http://www.tutorialspoint.com/dbms/database_joins.htm>

[2] <http://www.studytonight.com/dbms/joining-in-sql>

[3] <http://www.w3schools.com/sql/sql_join.asp>

[4] <https://mivuletech.wordpress.com/2011/03/22/an-overview-of-relational-algebra-operators-and-their-sql-translations/>

[5] <https://en.wikipedia.org/wiki/Relational_algebra#Rename>

[6] <http://www.go4expert.com/articles/relational-algebra-operations-sql-t21166/>

[7] <http://myweb.lmu.edu/dondi/share/db/relational3.pdf>

[8] <https://www.cs.sfu.ca/CourseCentral/354/jpei/slides/relational-algebra.pdf>