**ST. XAVIER’S COLLEGE**

**Maitighar, Kathmandu**



DATABASE MANAGEMENT SYSTEM

theory Assignment #6

**Submitted by:**

Suraj Nepal

013BSCCSIT042

**Submitted to:**

Er. Sanjay Kr. Yadav

Lecturer, St. Xavier’s college

Department of Computer Science

**Date of Submission:**

August 10, 2015

Types of Joins

Join is a special form of cross product of two tables. It is a binary operation that allows combining certain selections and a Cartesian productinto one operation. The

join operation forms a Cartesian product of its two arguments, performs a selection

forcing equality on those attributes that appear in both relation schemas, and finally

removes duplicate attributes. Following are the different types of joins: -

1. Theta Join

2. Equi Join

3. Semi Join

4. Natural Join

5. Outer Joins

Theta Join:

In theta join we apply the condition on input relation(s) and then only those selected

rows are used in the cross product to be merged and included in the output. It means

that in normal cross product all the rows of one relation are mapped/merged with all

the rows of second relation, but here only selected rows of a relation are made cross

product with second relation.

Equi­Join:

This is the most used type of join. In equi­join rows are joined on the basis of values

of a common attribute between the two relations. It means relations are joined on the

basis of common attributes between them; which are meaningful. This means on the

basis of primary key, which is a foreign key in another relation. Rows having the

same value in the common attributes are joined. Common attributes appear twice in

the output. It means that the attributes, which are common in both relations, appear

twice, but only those rows, which are selected. Common attribute with the same name

is qualified with the relation name in the output. It means that if primary and foreign

keys of two relations are having the same names and if we take the equi ­ join of both

then in the output relation the relation name will precede the attribute name.

Natural Join:

This is the most common and general form of join. If we simply say join, it means the

natural join. It is same as equi­join but the difference is that in natural join, the

common attribute appears only once. Now, it does not matter which common attribute

should be part of the output relation as the values in both are same.

Left Outer Join:

In left outer join all the tuples of left relation remain part of the output. The tuples that

have a matching tuple in the second relation do have the corresponding tuple from the

second relation. However, for the tuples of the left relation, which do not have a

matching record in the right tuple have Null values against the attributes of the right

relation.

Right Outer Join:

In right outer join all the tuples of right relation remain part of the output relation,

whereas on the left side the tuples, which do not match with the right relation, are left

as null. It means that right outer join will always have all the tuples of right relation

and those tuples of left relation which are not matched are left as Null.

Outer Join:

In outer join all the tuples of left and right relations are part of the output. It means

that all those tuples of left relation which are not matched with right relation are left

as Null. Similarly all those tuples of right relation which are not matched with left

relation are left as Null.

Semi Join:

In semi join, first we take the natural join of two relations then we project the

attributes of first table only. So after join and matching the common attribute of both

relations only attributes of first relation are projected.

elational database systems are expected to be equipped with a query language that can assist its users to query the database instances. There are two kinds of query languages − relational algebra and relational calculus.

Relational Algebra

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either **unary** or **binary**. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

The fundamental operations of relational algebra are as follows −

* Select
* Project
* Union
* Set different
* Cartesian product
* Rename

We will discuss all these operations in the following sections.

Select Operation (σ)

It selects tuples that satisfy the given predicate from a relation.

**Notation** − σ*p*(r)

Where **σ** stands for selection predicate and **r** stands for relation. *p* is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like − =, ≠, ≥, < ,  >,  ≤.

**For example** −

σ*subject = "database"*(Books)

**Output** − Selects tuples from books where subject is 'database'.

σsubject = "database" and price = "450"(Books)

**Output** − Selects tuples from books where subject is 'database' and 'price' is 450.

σsubject = "database" and price = "450" or year > "2010"(Books)

**Output** − Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

Project Operation (∏)

It projects column(s) that satisfy a given predicate.

Notation − ∏A1, A2, An (r)

Where A1, A2 , An are attribute names of relation **r**.

Duplicate rows are automatically eliminated, as relation is a set.

**For example** −

∏subject, author (Books)

Selects and projects columns named as subject and author from the relation Books.

Union Operation (∪)

It performs binary union between two given relations and is defined as −

r ∪ s = { t | t ∈ r or t ∈ s}

**Notion** − r U s

Where **r** and **s** are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold −

* **r**, and **s** must have the same number of attributes.
* Attribute domains must be compatible.
* Duplicate tuples are automatically eliminated.

∏ author (Books) ∪ ∏ author (Articles)

**Output** − Projects the names of the authors who have either written a book or an article or both.

Set Difference (−)

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

**Notation** − **r** − **s**

Finds all the tuples that are present in **r** but not in **s**.

∏ author (Books) − ∏ author (Articles)

**Output** − Provides the name of authors who have written books but not articles.