ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System**

**Theory Lab Assignment #6**

**SUBMITTED BY:**

**Saurav Bajracharya**

**013BSCCSIT035**

**SUBMITTED TO**

|  |  |
| --- | --- |
| **Er. Sanjay Kr. Yadav**  **( Lecturer )** |  |
| **Department of Computer Science** | |

Submission Date: September 10th2015

JOIN

Join operations take two relations and return as a result another relation.  
■ These additional operations are typically used as subqueryexpressions in the from clause  
■ Join condition – defines which tuples in the two relations match, and  
what attributes are present in the result of the join.  
■ Join type – defines how tuples in each relation that do not match any  
tuple in the other relation (based on the join condition) are treated.

**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 ⋈θR2

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.

**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.

**Eg:**

SELECT cFirstName, cLastName, orderDate

FROM customers NATURAL JOIN orders;

**Left Join**

This join returns all the rows from the left table in conjunction with the matching rows from the right table. If there are no columns matching in the right table, it returns NULL values.

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.

**Eg:**

SELECT user.name, course.name

FROM `user`

LEFT JOIN `course` on user.course = course.id;

**Right Join**

This join returns all the rows from the right table in conjunction with the matching rows from the left table. If there are no columns matching in the left table, it returns NULL values.

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.

**Eg:**

SELECT user.name, course.name

FROM `user`

RIGHT JOIN `course` on user.course = course.id;

**Inner Join**

In this kind of a JOIN, we get all records that match the condition in both the tables, and records in both the tables that do not match are not reported.

In other words, INNER JOIN is based on the single fact that: ONLY the matching entries in BOTH the tables SHOULD be listed.

Note that a JOIN without any other JOIN keywords (like INNER, OUTER, LEFT, etc) is an INNER JOIN.

**Eg:**

SELECT cFirstName, cLastName, orderDate

FROM customers INNER JOIN orders

USING (custID);

**Natural­ Join Operation**  
■ Let r and s be relations on schemas R and S respectively.  
Let r and s be relations on schemas R and S respectively.   
Then, r s is a relation on schema R ∪S obtained as follows:

Consider each pair of tuples tr from r and ts from s.

If tr and ts have the same value on each of the attributes in R∩S, add a tuple t to the result, where

t has the same value as tr on r

t has the same value as ts on s

Example:

R = (A, B, C, D)

S = (E, B, D)

Result schema = (A, B, C, D, E)

rs is defined as:  
∏r.A, r.B, r.C, r.D, s.E (σr.B = s.B∧r.D = s.D (r x s))

Types:

* + - Right Join
    - Left Join
    - Inner Join

**The Assignment Operation**

It is convenient at times to write a relational-algebra expression by assigning parts of it to temporary relation variables. The assignmentoperation, denoted by ←, works like assignment in a programming language. To illustrate this operation, consider the definition of the natural-join operation. We could write *r \_ s* as:

*temp*1 ← *R* × *S*

*temp*2 ← \_*r.A*1 =*s.A*1 ∧*r.A*2 =*s.A*2 ∧*...*∧*r.An* =*s.An* (*temp*1)

*result* = *\_R* ∪ *S* (*temp*2)

The evaluation of an assignment does not result in any relation being displayed to the user. Rather, the result of the expression to the right of the ← is assigned to the relation variable on the left of the ←. This relation variable may be used in subsequent expressions.

With the assignment operation, a query can be written as a sequential program consisting of a series of assignments followed by an expression whose value is displayed as the result of the query. For relational-algebra queries, assignment must always be made to a temporary relation variable. Assignments to permanent relations constitute a database modification. Note that the assignment operation does not provide any additional power to the algebra. It is, however, a convenient way to express complex queries.

**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**.

Consider the query:

**select** *name*, *course id*

**from** *instructor*, *teaches*

**where** *instructor*.*ID*= *teaches*.*ID*;

The result of this query is a relation with the following attributes:

*name*, *course id*

The names of the attributes in the result are derived from the names of the attributes in the relations in the **from** clause.

We cannot, however, always derive names in this way, for several reasons:

First, two relations in the **from** clause may have attributes with the same name, in which case an attribute name is duplicated in the result. Second, if we used an arithmetic expression in the **select** clause, the resultant attribute does not have a name. Third, even if an attribute name can be derived from the base relations as in the preceding example, we may want to change the attribute name in the result. Hence, SQL provides a way of renaming the attributes of a result relation

**Division operation**

The division operator of relational algebra, “÷”, is defined as follows. Let *r* (*R*) and *s*(*S*) be relations, and let *S* ⊆ *R*; that is, every attribute of schema *S* is also in schema *R*. Then *r* ÷ *s* is a relation on schema *R* − *S* (that is, on the schema containing all attributes of schema *R* that are not in schema *S*). A tuple *t* is in *r* ÷ *s* if and only if both of two conditions hold:

* *t* is in *\_R*−*S*(*r* )
* For every tuple *ts* in *s*, there is a tuple *tr* in *r* satisfying both of the following:

1. *tr* [*S*] = *ts* [*S*]
2. *tr* [*R* − *S*] = *t*

**Additional operation**

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.

* + **Set- intersection operation**

Find tuple 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.

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

* + **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**