ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System**

**Theory Lab Assignment #6**

**SUBMITTED BY:**

**Siddhant Rimal**

**013BSCCSIT039**

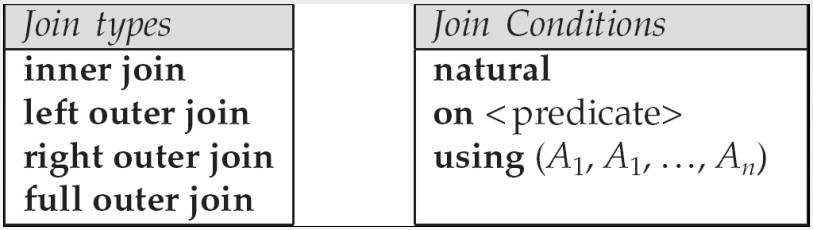
**SUBMITTED TO**

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

Submission Date: September 10th 2015

JOIN

Join operations take two relations and return as a result another relation.  
■ These additional operations are typically used as subquery expressions 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.



**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)

r s 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

**Rename Operation:**

* It can be used in two ways :
  + return the result of expression E in the table named *x*.
  + return the result of expression E in the table named *x* with the attributes renamed to A1, A2,…, An.
  + It’s benefit can be understood by the solution of the query “ Find the largest account balance in the bank”

It can be solved by following steps:

1. Find out the relation of those balances which are not largest.
   1. Consider Cartesion product of Account with itself i.e. Account × Account
   2. Compare the balances of first Account table with balances of second Account table in the product.
   3. For that we should rename one of the account table by some other name to avoid the confusion
   4. It can be done by following operation

*ΠAccount.balance (σAccount.balance < d.balance(Account× ρd(Account))*

* 1. So the above relation contains the balances which are not largest.

1. Subtract this relation from the relation containing all the balances i.e . *Πbalance (Account)*.
2. So the final statement for solving above query is

*Πbalance (Account)- ΠAccount.balance (σAccount.balance < d.balance(Account× ρd(Account))*

Assignment Operation

The assignment operation (←) provides a convenient way to express  
complex queries.  
● Write query as a sequential program consisting of  
4  
a series of assignments  
4 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  
temp1 ← ∏R­S (r )  
temp2 ← ∏R­S ((temp1 x s ) – ∏R­S,S(r ))  
result = temp1 – temp2  
● The result to the right of the ← is assigned to the relation variable on  
the left of the ←.  
● May use variable in subsequent expressions.

**Division Operation:**

* denoted by ÷ is used for queries that include the phrase “for all”.
* For example “Find customers who has an account in all branches in branch city Agra”. This query can be solved by following statement.

*ΠCustomer-name. branch-name (* )÷*Πbranch-name (σBranch-city=”Agra”(Branch)*

* The division operations can be specified by using only basic operations as follows: Let r(R) and s(S) be given relations for schema R and S with

r ÷ s = ΠR-S(r) - ΠR-S ((ΠR-S (r) × s) - ΠR-S,S (r))

* The division operations can be specified by using only basic operations as follows: Let r(R) and s(S) be given relations for schema R and S with

r ÷ s = ΠR-S(r) - ΠR-S ((ΠR-S (r) × s) - ΠR-S,S (r))

* Additional Operations

Set-Intersection Operation

Set intersection: v = r ∩ s  
● when a tuple is inserted in r we check if it is present in s, and if so  
we add it to v.  
● If the tuple is deleted from r, we delete it from the intersection if it  
is present.  
● Updates to s are symmetric  
● The other set operations, union and set difference are handled in a  
similar fashion.

**Natural Join** ()

* Forms Cartesian product of its two arguments, performs selection forcing equality on those attributes that appear in both relations
* For example consider Borrower and Loan relations, the natural join between them will automatically perform the selection on the table returned by Borrower × Loan which force equality on the attribute that appear in both Borrower and Loan i.e. Loan-no and also will have only one of the column named Loan-No.
* That means = *σBorrower.Loan-no = Loan.Loan-no* (Borrower × Loan).
* The table returned from this will be as follows:

Eliminate rows that does not satisfy the selection criteria “*σBorrower.Loan-no = Loan.Loan-no”* from Borrower × Loan =

|  |  |  |  |
| --- | --- | --- | --- |
| **Borrower.Cust-name** | **Borrower.Loan-no** | **Loan.Loan-no** | **Loan.Amount** |
| Ram | L-13 | L-13 | 1000 |
| ~~Ram~~ | ~~L-13~~ | ~~L-30~~ | ~~20000~~ |
| ~~Ram~~ | ~~L-13~~ | ~~L-42~~ | ~~40000~~ |
| ~~Shyam~~ | ~~L-30~~ | ~~L-13~~ | ~~1000~~ |
| Shyam | L-30 | L-30 | 20000 |
| ~~Shyam~~ | ~~L-30~~ | ~~L-42~~ | ~~40000~~ |
| ~~Suleman~~ | ~~L-42~~ | ~~L-13~~ | ~~1000~~ |
| ~~Suleman~~ | ~~L-42~~ | ~~L-30~~ | ~~20000~~ |
| Suleman | L-42 | L-42 | 40000 |

And will remove one of the column named Loan-no.

* i.e. =

|  |  |  |
| --- | --- | --- |
| **Cust-name** | **Loan-no** | **Amount** |
| Ram | L-13 | 1000 |
| Shyam | L-30 | 20000 |
| Suleman | L-42 | 40000 |