**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**Database Management System**

**Assignment # 6**

**Submitted by:**

Biraj Poudyal  
013BSCCSIT015

**Submitted to:**

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

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

* **Join**

Join is a combination of a Cartesian product followed by a selection process Join is a special form of cross product of two tables. It is a binary operation that allows combining certain selections and a Cartesian product into 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.

* + - **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. It is denoted as under:

## 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 |

## Left Outer 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 |

## Right Outer 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.

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

* + - **Inner join**

The most frequently used and important of the joins is the **INNER JOIN**. They are also referred to as an EQUIJOIN.

The INNER JOIN creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate. The query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row.

* + - * **Rename operation**

It is a mechanism to rename both relations and attributes as-clause can appear in both the select and from clauses: old-name as new-name.Example.

select distinct cname, borrower.loan# as loan\_id from borrower, loan where borrower.loan# = loan.loan# and bname= ``SFU"

* + - * **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)

* + - * **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**
    - **Natural join operation**