ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System**

**Assignment # 6**

**SUBMITTED BY:**

**Apil Neupane**

**013BSCCSIT008**

**SUBMITTED TO:**

|  |  |
| --- | --- |
| **Mr. Sanjay Kr. Yadav**  **Lecturer** |  |
| **Department of Computer Science** | |

Submission Date: 10thSeptember 2015

**Join:**

SQL Join is used to fetch data from two or more tables, which is joined to appear as single set of data. SQL Join is used for combining column from two or more tables by using values common to both tables. **Join** Keyword is used in SQL queries for joining two or more tables. Minimum required condition for joining table, is **(n-1)** where **n**, is number of tables. A table can also join to itself known as, **Self Join**.

**Join** 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.

The different Types of join are:

1. Theta join
2. Natural Join

**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.

|  |  |  |
| --- | --- | --- |
| Student | | |
| SID | Name | Std |
| 101 | Alex | 10 |
| 102 | Maria | 11 |

|  |  |
| --- | --- |
| Subjects | |
| Class | Subject |
| 10 | Math |
| 10 | English |
| 11 | Music |
| 11 | Sports |

Student\_Detail: STUDENT **⋈Student.Std = Subject.Class** SUBJECT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student\_detail | | | | |
| SID | Name | Std | Class | Subject |
| 101 | Alex | 10 | 10 | Math |
| 101 | Alex | 10 | 10 | English |
| 102 | Maria | 11 | 11 | Music |
| 102 | Maria | 11 | 11 | Sports |

## 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.

|  |
| --- |
| **Syntax For Natural Join** |
| SELECT \*  from table-name1  NATURAL JOIN  table-name2; |

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 |

**Types of Natural Join:**

* **INNER JOIN**: Returns all rows when there is at least one match in BOTH tables

|  |
| --- |
| **Syntax For Inner Join** |
| SELECT column-name-list  from table-name1  INNER JOIN  table-name2  WHERE table-name1.column-name = table-name2.column-name; |

* **LEFT JOIN**: Return all rows from the left table, and the matched rows from the right table

|  |
| --- |
| **Syntax For Left Join** |
| SELECT column-name-list  from table-name1  LEFT OUTER JOIN  table-name2  on table-name1.column-name = table-name2.column-name; |

* **RIGHT JOIN**: Return all rows from the right table, and the matched rows from the left table

|  |
| --- |
| **Syntax For Right Join** |
| select column-name-list  from table-name1  RIGHT OUTER JOIN  table-name2  on 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*.

**Notation** − *ρ* x (E)

Where the result of expression **E** is saved with name of **x**.

The rename operator returns an existing relation under a new name. ρA(B) is the relation B with its name changed to A. For example, find the employees in the same Department as employee 3.

 ρemp2.surname,emp2.forenames (

 σemployee.empno = 3 ^ employee.depno = emp2.depno ( employee × (ρemp2employee) )

 )

## Assignment Operator

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 thePRODUCT table, raising the existing prices by 2 percent:

UPDATE product SET prod\_price\_n

= prod\_price\_n \* 1.02 (10 row(s) affected)

And the same operator would be used for comparing values when used, for example, in the WHERE clause of an SQL statement:

UPDATE product SET prod\_price\_n

= prod\_price\_n \* 1.02 WHERE prod\_id\_n = 1880 (1 row(s)

affected)

This statement assigns a 2 percent increase to a product whose ID is 1880; in the same query, the equals operator (=) is used in its assignment and comparison capacity at the same time.

**Division Operation:**

Letr(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 present in r.  
  
Example:

|  |  |  |
| --- | --- | --- |
| Relation or table "r":- | Relation or table "s":- | Therefore, r ÷ s |
| Code: | Code: | Code: |
| +--------+-------+  | A | B |  +--------+-------+  | a | 1 |  | b | 2 |  | a | 2 |  | p | 3 |  | p | 4 |  +--------+-------+ | +------+  | B |  +------+  | 2 |  | 3 |  +------+ | +------+  | A |  +------+  | b |  | a |  | p |  +------+ |

**Additional Operation:**

1. **Set Intersection Operation:**

**The set intersection operation: -**finds tuples 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.

Code:

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

1. **Natural Join Operation:**

**The 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**