**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**DATABASE MANAGEMENT SYSTEM**

**THEORY ASSIGNMENT#6**

**Submitted by:**

Shreesha Pokharel

013BSCCSIT038

**Submitted to:**

|  |  |
| --- | --- |
| **Er. Sanjay Kumar Yadav** |  |

Lecturer

Department of Computer Science

Date of submission: 10th September, 2015

* **Join**

A **join** clause combines records from two or more tables in a relational database. It creates a set that can be saved as a table or used as it is. A JOIN is a means for combining fields from two tables (or more) by using values common to each. As a special case, a table (base table, view, or joined table) can JOIN to itself in a *self-join*.

A programmer writes a JOIN statement to identify the records for joining. If the evaluated predicate is true, the combined record is then produced in the expected format, a record set or a temporary table.

* + **Theta join**

Theta join combines tuple 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.

**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.

* + - **Right join**

A **right join** closely resembles a left, except with the treatment of the tables reversed. Every row from the "right" table (B) will appear in the joined table at least once. If no matching row from the "left" table (A) exists, NULL will appear in columns from A for those records that have no match in B.

A right join return all the values from the right table and matched values from the left table (NULL in the case of no matching join predicate).

* + - **Left join**

The result of a *left join* for tables A and B always contains all records of the "left" table (A), even if the join-condition does not find any matching record in the "right" table (B). This means that if the ON clause matches 0 (zero) records in B (for a given record in A), the join will still return a row in the result (for that record)—but with NULL in each column from B. A **left join** returns all the values from an inner join plus all values in the left table that do not match to the right table, including rows with NULL (empty) values in the link field.

* + - **Inner join**

An **inner join** requires each record in the two joined tables to have matching records, and is a commonly used join operation in applications but should not be assumed to be the best choice in all situations. Inner join creates a new result table by combining column values of two tables (A and B) based upon the join-predicate. The query compares each row of A with each row of B to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied by matching non-NULL values, column values for each matched pair of rows of A and B are combined into a result row.

* **Rename operation**

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**.

* **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**

It is denoted as ÷.

Let r(R) and s(S) be relations  
  
**r ÷ s: -** the result consists of the restrictions of tuple 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 tuple in s are present in r.

* **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**