**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**Database Management System**

**Assignment # 6**

**Submitted by:**

Anlisha Maharjan  
013BSCCSIT006

**Submitted to:**

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

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

**JOIN**

A SQL 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.

**NATURAL JOIN**

The natural join is arguably one of the most important operators since it is the relational counterpart of logical AND. In particular, natural join allows the combination of relations that are associated by a [foreign key](https://en.wikipedia.org/wiki/Foreign_key). More formally the semantics of the natural join are defined as follows:

R \bowtie S = \left\{ t \cup s \ \vert \ t \in R \ \land \ s \in S \ \land \ \mathit{Fun}(t \cup s) \right\}

It is usually required that *R* and *S* must have at least one common attribute, but if this constraint is omitted, and *R* and *S* have no common attributes, then the natural join becomes exactly the Cartesian product. The natural join can be simulated with Codd's primitives as follows. Assume that *c*1,...,*cm* are the attribute names common to *R* and *S*, *r*1,...,*rn* are the attribute names unique to *R*and *s*1,...,*sk* are the attribute unique to *S*. Furthermore assume that the attribute names *x*1,...,*xm* are neither in *R* nor in *S*. In a first step we can now rename the common attribute names in *S*:

T = \rho_{x_1/c_1,\ldots,x_m/c_m}(S) = \rho_{x_1/c_1}(\rho_{x_2/c_2}(\ldots\rho_{x_m/c_m}(S)\ldots))

Then we take the Cartesian product and select the tuples that are to be joined:

P = \sigma_{c_1=x_1,\ldots,c_m=x_m}(R \times T) = \sigma_{c_1=x_1}(\sigma_{c_2=x_2}(\ldots\sigma_{c_m=x_m}(R \times T)\ldots))

Finally we take a projection to get rid of the renamed attributes:

U = \pi_{r_1,\ldots,r_n,c_1,\ldots,c_m,s_1,\ldots,s_k}(P)

The natural join is a special case of equi-join which is itself a special case of inner join as described in [Natural join](https://en.wikipedia.org/wiki/Join_(SQL)#Natural_join_in_inner_join).

**RIGHT JOIN**

A right outer join (or right join) closely resembles a left outer join, 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 outer join returns 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 outer join (or simply 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 outer 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. The result of the join can be defined as the outcome of first taking the Cartesian product (or Cross join) of all records in the tables (combining every record in table A with every record in table B) and then returning all records which satisfy the join predicate.

SQL specifies two different syntactical ways to express joins: "explicit join notation" and "implicit join notation". Although "implicit join notation" was deprecated in 1992, and its use is not considered a best practice, database systems still support it.

**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 equal sign (=) is the only Transact-SQL assignment operator. In the following example, the @MyCounter variable is created, and then the assignment operator sets @MyCounter to a value returned by an expression.DECLARE @MyCounter INT;SET @MyCounter = 1;

**DIVISION OPERATION:**

**ADDITIONAL OPERATION:**