**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**DATABASE MANAGEMENT SYSTEM**

**THEORY ASSIGNMENT #06**

**Submitted by:**

Rojesh Tamrakar

013BSCCSIT032

**Submitted to:**

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

Lecturer

Department of Computer Science

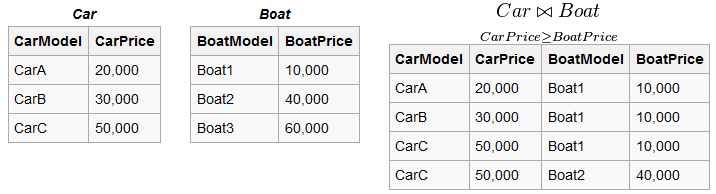
Date of submission: 10th September, 2015

1. **Join**

* The most used operator in the relational algebra
* Allows us to establish connections among data in different relations, taking advantage of the "value-based" nature of the relational model
* Two main versions of the join:
* **Natural join**: takes attribute names into account;
* **Theta join.**
* Both join operations are denoted by the symbol ⋈
  1. **Theta Join**

A theta join allows for arbitrary comparison relationships (such as ≥). An equijoin is a theta join using the equality operator. A natural join is an equijoin on attributes that have the same name in each relationship. It is denoted by θ.

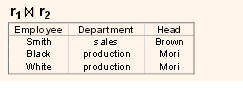
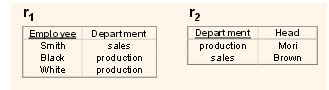
Consider tables Car and Boat which list models of cars and boats and their respective prices. Suppose a customer wants to buy a car and a boat, but she does not want to spend more money for the boat than for the car. The θ-join on the relation CarPrice ≥ BoatPrice produces a table with all the possible options.



* 1. **Natural Join**

Natural join is a binary operator that is written as (R⋈S) where R and S are relations. The result of the natural join is the set of all combinations of tuples in R and S that are equal on their common attribute names. For an example consider the tables Employee and Dept and their natural join. It’s types are:

* Inner join
* Right join
* Left join

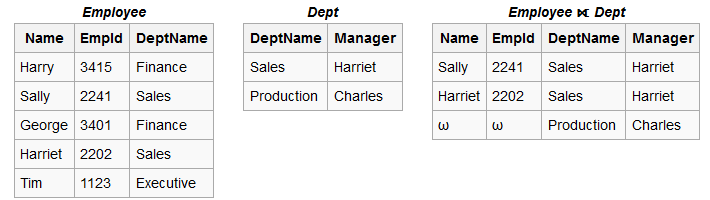


* + 1. **Inner Join**

An SQL INNER JOIN return all rows from multiple tables where join condition is met. Notice that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in "Customers" table. The relationship between two tables above is the "CustomerID" column.

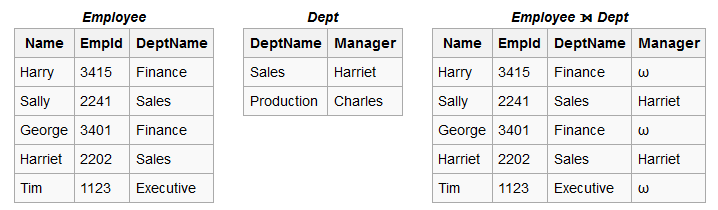
* + 1. **Right Join**

The right outer join of relations *R* and *S* is written as *R* ⟖ *S*. The result of the right outer join is the set of all combinations of tuples in *R* and *S* that are equal on their common attribute names, in addition to tuples in *S* that have no matching tuples in *R*. For example consider the tables *Employee* and *Dept* and their right outer join:

****

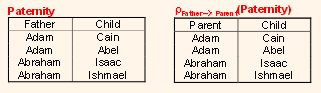
* + 1. **Left Join**

The left outer join is written as *R* ⟕ *S* where *R* and *S* are relations. The result of the left outer join is the set of all combinations of tuples in *R* and *S* that are equal on their common attribute names, in addition (loosely speaking) to tuples in *R* that have no matching tuples in *S*. For example consider the tables *Employee* and *Dept* and their left outer join:

****

1. **Rename Operation**

* This is a unary operator which changes attribute names for a relation without changing any values.
* Renaming removes the limitations associated with set operators
* For example, ρFather→Parent(Paternity)

****

1. **Assignment Operation**
2. **Division Operation**
3. **Additional Operation**
   1. **Set Intersection**
   2. **Natural Join Operation**