MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY JAIPUR

Autumn Semester 2017-2018

Lab Sheet : JOINED RELATIONS

GOAL:

- *What are joined Relations?*
  - *Brief overview of joined relations*
- *Getting familiar with different join operations*
  - *Inner join*
  - *Left outer join*
  - *Natural inner join*
  - *Right outer join*
  - *Full outer join*

Exercise:

i. **Determine the names of sailors who are older than the oldest sailor with a rating of 10?**

ii. **Determine the names of sailors who have reserved all boats?**

iii. **Determine the average age of sailors with a rating of 10?**

iv. **Determine the name and the age of the oldest sailor?**

v. **Count the number of sailors?**

vi. **For each red boat, find the number of reservations for this boat?**

WHAT ARE JOINED RELATIONS:

- **Brief Overview of joined relations:**

  SQL provides not only the basic Cartesian-product mechanism for joining tuples of relations found in its earlier versions, but, SQL also provides various other mechanism of joining relations including condition joins and natural joins. These additional operations are typically used as subquery expressions in the **FROM** clause.

  Each of the variants of the join operations in SQL consists of a **join type and a join condition**. The join condition defines which tuples in the two relations match and what attributes are present in the result of the join. The join type defines how tuples in each relation that do not match any tuple in the other relation (based on the join condition).

Following tables gives us some join type and join condition:

Join Type
Inner join
Left Outer Join
Right Outer Join
Full Outer Join

Join Condition
natural **on** <predicate> **using** <A_1,A_2,$A_{3...}$,A_n>

To understand how join works ,Let's create two table :

```
Create table loan(
        loanNo  varchar2(10),
      branch_name   varchar2(20),
        amount  number(5),
        constraint  loan_pk primary key (loanNo)
   );

 Create table borrower (
      customer_name varchar2(20),
      loanNo  varchar2(10),
      constraint  borrower_pk  primary key (loanNo)
);

Create table depositor (
      customer_name   varchar2(20),
      account_number  varchar2(20),
      constraint depositor_pk primary key (customer_name)
   );
```

Now insert following data into this tables:

LOAN

loanNo	branch_Name	amount
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700
L-155	Washigton	4500

BORROWER

Customer_name	loanNo
Jones	L-170
Smith	L-230
Hayes	L-155

DEPOSITOR

Customer_name	Account_number
Jones	HDBC2044
Smith	HDBC2055
Hayes	HDBC2066
Chris	HDBC2077
Adam	HDBC2088

Getting Familiar With Different Join Operations:

- **Inner Joins:**

  We illustrate the use of inner joins through an example:

  Select * from (loan l inner join borrower b on l.loanno =b.loanno);

  //observe the output carefully specifically field names and values of resultant.

  The expression computes the theta join of the loan and the borrower relations, with the join condition being loan.loan-number = borrower.loan-number. The attributes of the result consist of the attributes of the left-hand-side relation followed by the attribute of the right-hand-side relation.

- **Left outer joins:**

  We illustrate the use of inner joins through an example:

  Select * from (loan l left outer join borrower b on l.loanno = b.loanno);

  //observer the output carefully

  We can compute the left outer join operation logically as follows. First, compute the result of the inner join as before. Then, for every tuple t in the left-hand-side relation loan that does not match any tuple in the right-hand-side relation borrower in the inner join, add a tuple r to the result of the join: The attributes of tuple r that are derived from the left-hand-side relation are filled in with the values from tuple t,and the remaining attributes of r are filled with null values.

- **Natural inner joins:**

  we consider an example of the natural join operation:

  select * from (loan natural inner join borrower);

  //observer the output carefully and compare with previous outputs

  This expression computes the natural join of the two relations. The only attribute name common to loan and borrower is loan-number.

  Difference between natural join and inner join is that the attribute loan-number appears only once in the result of the natural join but it comes twice in case of inner join.

- **Right outer join:**

  we consider an example of the right outer join operation:

  select * from (loan l right outer join borrower b on l.loanno =b .loanno);

  //observer the output carefully and compare with left-outer join

  The right outer join is symmetric to the left outer join. Tuples from the right-hand- side relation that do not match any tuple in the left-hand-side relation are padded with nulls and are added to the result of the right outer join.

- **Full outer join**

  we consider an example of the full outer join operation:

  select * from (loan l full outer join borrower b on l.loanno=b.loanno);

  //observer the output carefully and compare with previous outputs

  The full outer join is a combination of the left and right outer-join types. After he operation computes the result of the inner join, it extends with nulls tuples from the left-hand-side relation that did not match with any from the right-hand-side, and adds them to the result. Similarly, it extends with nulls tuples from the right-hand-side relation that did not match with any tuples from the left-hand-side relation and adds them to the result.

- **How to use various above joined relations in a SQL statements:**

  Example: Find all customers who have an account but no loan at the bank?

  SELECT depositor.customer-name
  FROM (depositor **left outer join** borrower
  on depositor.customer-name = borrower.customer-name)
  WHERE borrower.customer-name is null;

  Example: Find all customers who have either an account or a loan (but not both) at the bank?

  SELECT customer-name
  FROM (depositor **natural full outer join** borrower)
  WHERE account-number is null **OR** loan-number is null;

Exercise 1 : Create a database for an Organization as per the details given below.

Table Name	Attribute Name	Data Type	Constraint
department	dnumber	Integer	Primary key
	dname	varchar	Not null
	MGRssn	integer	Not null
	MGRstartdate	integer	Not null
dependent	Essn	integer	Primary key
	dependent_name	varchar	Primary key
	sex	varchar	Not null
	bdate	varchar	Not null
	relationship	varchar	Not null
dep_location	dnumber	integer	Primary key, Foreign key
	dlocation	varchar	Primary key
employee	first_name	varchar	Not Null
	MINIT	varchar	Not Null
	last_name	varchar	Not null
	ssn	integer	Primary key
	bdate	varchar	Not Null
	address	varchar	Not Null
	sex	varchar	Not Null
	salary	integer	Not Null
	SUPERssn	integer	Null
	dno	integer	Not Null, Foreign key
Project	pname	varchr	Not null
	pnumber	integer	Primary key
	plocation	varchar	Not null
	dnum	integer	Not null
works_on	Essn	integer	Primary key, Foreign key
	pno	integer	Primary key, Foreign key
	hours	varchar	null

Insert at least 10 sample tuples in each table and write the SQL queries for the following:

a) Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.
b) Retrieve the names of employees who have no dependents.
c) Retrieve the name and address of every employee who works for the 'Research' department.
d) For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.
e) For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.

Exercise 2 : Create a database for a Banking System as per the details given below.

Table Name	Attribute Name	Data Type	Constraint
account	account_number	integer	Primary key
	branch_name	varchar	Not null
	balance	integer	Not null
borrower	customer_name	varchar	Primary key, Foreign key
	loan_number	integer	Primary key, Foreign key
branch	branch_name	varchar	Primary key
	assets	integer	Not null
	branch_city	varchar	Not null
customer	customer_name	varchar	Primary key
	street	varchar	Null
	city	varchar	Not null
loan	loan_number	integer	Primary key
	branch_name	varchar	Not null
	amount	integer	Not null
depositor	account_number	integer	Primary key, Foreign key
	customer_name	varchar	Primary key, Foreign key

Insert at least 10 sample tuples in each table and write the SQL queries for the following:

i. Find the names of all branches in the loan relation.
ii. Find all loan numbers for loans made at the SBI branch.
iii. Find the loan numbers where loan amounts is between $9000 and $100,000.
iv. Find the Cartesian product of borrower and loan.
v. Find the name, loan number and loan amount of all customers having a loan at the Perryridge branch.
vi. Find the name, loan number and loan amount of all customers; rename the column name loan_number as loan_id.
vii. Find the customer names and their loan numbers for all customers having a loan at some branch.
viii. Find the names of all branches that have greater assets than some branch located in Jaipur.
ix. Find the names of all customers whose street includes the substring “pur”.
x. List in alphabetic order the names of all customers having a loan in Jaipur branch.
xi. Find all customers who have a loan, an account, or both.
xii. Find the average account balance at the ICICI branch.
xiii. Find the number of tuples in the customer relation.
xiv. Find the number of depositors in the bank.
xv. Find the number of depositors for each branch.
xvi. Find the names of all branches where the average account.
xvii. Find all customers who have both an account and a loan at the bank.
xviii. Find all customers who have a loan at the bank but do not have an account at the bank.
xix. Find all customers who have both an account and a loan at the Perryridge branch.
xx. Find all branches that have greater assets than some branch located in Brooklyn.