Department of
Computer Science and Engineering

Title: Implementation of Relational Databases (Join Function)

Database System Lab
CSE 210

Green University of Bangladesh

1 Objective(s)

- We learned about the need to normalize to make it easier to maintain the data.Though this makes it easier to maintain and update the data, it makes it very inconvenient to view and report information.
- Through the use of database joins we can stitch the data back together to make it easy for a person to use and understand.

2 Problem analysis

Before we begin let's look into why you have to combine data in the first place. SQLite and other databases such as Microsoft SQL server and MySQL are relational databases. These types of databases make it really easy to create tables of data and a facility to relate (join or combine) the data together.

As requirements are cast into table designs, they are laid up against some best practices to minimize data quality issues. This process is called normalization and it helps each table achieve singular meaning and purpose.

For instance, if I had a table containing all the students and their classes, then wanted to change a student's name, I would have to change it multiple times, once for each class the student enrolled in.

We can easily produce these details with the help of JOIN function.

MySQL JOIN functions

JOIN function	Description
Cross Joins	return all combinations of rows from each table.
Inner joins	return rows when the join condition is met.
Outer joins	return all the rows from one table, and if the join condition is met, columns from the other.
Left Outer Join	Return all rows from the "left" table, and matching rows from the "right" table.
Right Outer Join	Return all rows from the "right" table, and matching rows from the "left" table.
Full Join	Return all rows from an inner join, when no match is found, return nulls for that table.

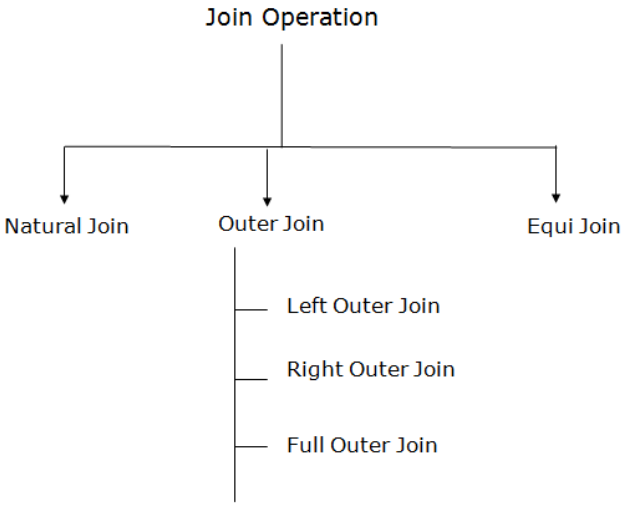


Figure 1: Employees Table Information

2.1 Join Function

- **Cross Joins** Cross joins return all combinations of rows from each table. So, if you're looking to find all combinations of size and color, you would use a cross join. Join conditions aren't used with cross joins. It pure combinatory joy.

- **Inner joins:** Inner joins return rows when the join condition is met. This is the most common Database join. A common scenario is to join the primary key of once table to the foreign key of another.

  This is used to perform "lookup," such are to get the employee's name from their employeeID.

- **Outer joins:** Outer joins return all the rows from one table, and if the join condition is met, columns from the other. They differ from an inner join, since an inner join wouldn't include the non-matching rows in the final result.

  Consider an order entry system. There may be cases where we want to list all employees regardless of whether they placed a customer order. In this case an outer join comes in handy.

  When using an outer join all employees, even those not matching orders, are included in the result.

3 Procedure (Implementation in MySQL)

1. **Create a Data**

   - **Create a table student:**

```
CREATE TABLE  student(
s_id             int(11)            NOT NULL         AUTO_INCREMENT,
FirstName        varchar(255)       NOT NULL,
LastName         varchar(255 )      NOT NULL,
Address          varchar(255 )      NOT NULL,
dept_name        enum( 'CSE', 'EEE' , ' TEX' )      DEFAULT NULL,
AdmissionDate    datetime           NOT NULL         DEFAULT current_timestamp(),
PRIMARY KEY(S_ID)
);
```

   - **Insert values into student table:**

```
INSERT INTO student (s_id, FirstName, LastName, Address, dept_name)
VALUES           (142002015, 'Zeseya' , 'Sharmin' , 'Dhaka' , 'CSE'),
                  (142002001, 'Sakib' , 'Hasan' , 'Natore' , 'CSE'),
                  (162002002, 'Asef', 'Tajwar' , 'Rangpur' , 'EEE'),
                  (162002003, 'Maruf', 'Hasan', 'Barisal', 'EEE'),
                  (172082002, 'Ashek' , 'Farabi', 'Gazipur' ,'TEX'),
                  (173002003, 'Ismile' , 'Hasan' , 'Barisal' , 'TEX');
```

   - **Create a table department:**

```
CREATE TABLE  'department'(
dept_id          int(11)            NOT NULL        AUTO_INCREMENT,
dept_name        enum( 'CSE', 'EEE', 'TEX')         DEFAULT NULL,
dept_location    varchar(255 )      NOT NULL,
PRIMARY KEY(dept_id)
);
```

   - **Insert values into department table:**

```
INSERT INTO department (dept_id, dept_name, dept_location)
VALUES          (101, ‘CSE’, ‘Building-2’),
                (102, ‘EEE’, ‘Building-2’),
                (103, ‘TEX’, ‘Building-1’);
```

- **Create another table course_registrstion:**

```
CREATE TABLE ‘course_registration’(
reg_serial      int(11)         NOT NULL        AUTO_INCREMENT,
course_code     varchar(255)    NOT NULL,
Course_title    varchar(255 )   NOT NULL,
dept_id         int(11 )        NOT NULL,
s_id            varchar(255 )   NOT NULL,
PRIMARY KEY(reg_serial)
);
```

- **Insert values into course_registration table:**

```
INSERT INTO course_registration(course_code,Course_title,dept_id,s_id)
VALUES          (‘CSE 311’, ‘Computer Networks’,101,142002015),
                (‘CSE 311’, ‘Computer Networks’,101,142002001),
                (‘EEE 301’,‘Electrical Circuit’,201,162002002),
                (‘TEX 201’, ‘Aparales’, 301,172002002),
                (‘CSE 312’, ‘Computer Networks Lab’,101,142002015),
                (‘CSE 207’, ‘Algorithm’,101,142002001);
```

- **join_table:**

```
SELECT s_id
FROM student
UNION
SELECT s_id
FROM course_registration;
```

- **join_table:**

```
SELECT s_id
FROM student
UNION ALL
SELECT s_id
FROM course_registration;
```

2. **Join, Inner Join, Left Join, Right Join, Where, Group by:**

- **INNER JOIN example**

```
SELECT student.s_id, student.FirstName, student.LastName
FROM student
INNER JOIN course_registration ON student.s_id = course_registration.s_id;
```

- **INNER JOIN with WHERE clause**

```
SELECT student.s_id, student.FirstName, student.LastName
FROM student
INNER JOIN course_registration ON student.s_id = course_registration.s_id
WHERE course_registration.s_id = 142002015;
```

- **Multiple Inner Join**

```
SELECT student.s_id, student.FirstName, student.dept_name, department.dept_id
FROM student
INNER JOIN department ON student.dept_name = department.dept_name
```

- **Multiple Inner Join**

```
SELECT student.s_id, student.FirstName, student.dept_name, department.dept_id,
course_registration.course_code
FROM student
INNER JOIN department ON student.dept_name = department.dept_name
INNER JOIN course_registration ON department.dept_id = course_registration.dept_id;
```

- **INNER JOIN using GROUP BY for eliminating duplicate records.**

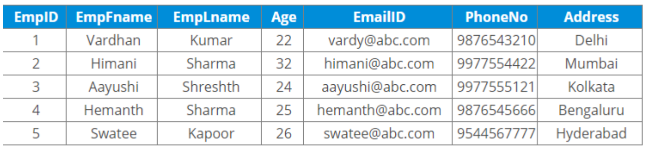
```
SELECT student.s_id, student.FirstName, student.dept_name, department.dept_id,
course_registration.course_code
FROM student
INNER JOIN department ON student.dept_name = department.dept_name
INNER JOIN course_registration ON department.dept_id = course_registration.dept_id
GROUP BY s_id;
```

4 Discussion & Conclusion

In the following experiment we dig into the various join types, explore Database joins involving more than one table, and further explain join conditions, especially what can be done with non-equijoin conditions.

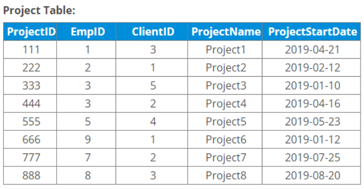
5 Lab Task (Please implement yourself and show the output to the instructor)

- Task-1:



EmpID	EmpFname	EmpLname	Age	EmailID	PhoneNo	Address
1	Vardhan	Kumar	22	vardy@abc.com	9876543210	Delhi
2	Himani	Sharma	32	himani@abc.com	9977554422	Mumbai
3	Aayushi	Shreshth	24	aayushi@abc.com	9977555121	Kolkata
4	Hemanth	Sharma	25	hemanth@abc.com	9876545666	Bengaluru
5	Swatee	Kapoor	26	swatee@abc.com	9544567777	Hyderabad

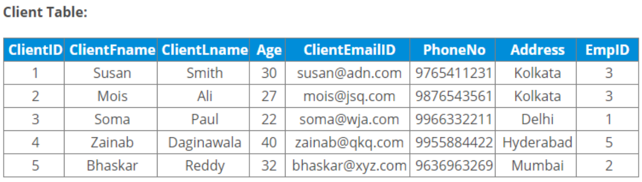
Figure 2: Project Table Information



Project Table:

ProjectID	EmpID	ClientID	ProjectName	ProjectStartDate
111	1	3	Project1	2019-04-21
222	2	1	Project2	2019-02-12
333	3	5	Project3	2019-01-10
444	3	2	Project4	2019-04-16
555	5	4	Project5	2019-05-23
666	9	1	Project6	2019-01-12
777	7	2	Project7	2019-07-25
888	8	3	Project8	2019-08-20

Figure 3: Project Table Information



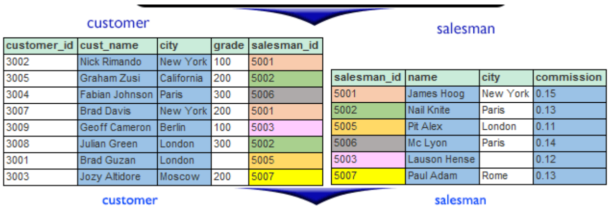
Client Table:

ClientID	ClientFname	ClientLname	Age	ClientEmailID	PhoneNo	Address	EmpID
1	Susan	Smith	30	susan@adn.com	9765411231	Kolkata	3
2	Mois	Ali	27	mois@jsq.com	9876543561	Kolkata	3
3	Soma	Paul	22	soma@wja.com	9966332211	Delhi	1
4	Zainab	Daginawala	40	zainab@qkq.com	9955884422	Hyderabad	5
5	Bhaskar	Reddy	32	bhaskar@xyz.com	9636963269	Mumbai	2

Figure 4: Client Table Information

1. Create these tables in a company database
2. Write a SQL query for all the JOIN operation
3. Location count

- Task 2:



customer

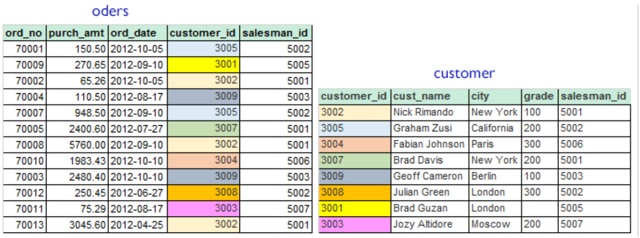
customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3005	Graham Zusi	California	200	5002
3004	Fabian Johnson	Paris	300	5006
3007	Brad Davis	New York	200	5001
3009	Geoff Cameron	Berlin	100	5003
3008	Julian Green	London	300	5002
3001	Brad Guzan	London		5005
3003	Jozy Altidore	Moscow	200	5007

salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5003	Lauson Hense		0.12
5007	Paul Adam	Rome	0.13

Figure 5: Customer and Salesman table

6 Lab Exercise (Submit as a report)



oders

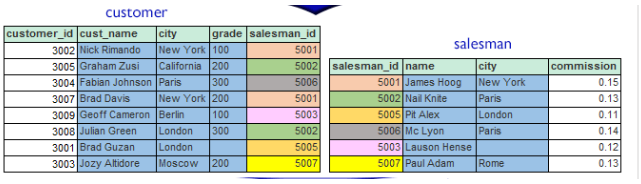
ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.50	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.50	2012-08-17	3009	5003
70007	948.50	2012-09-10	3005	5002
70005	2400.60	2012-07-27	3007	5001
70008	5760.00	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.40	2012-10-10	3009	5003
70012	250.45	2012-06-27	3008	5002
70011	75.29	2012-08-17	3003	5007
70013	3045.60	2012-04-25	3002	5001

customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3005	Graham Zusi	California	200	5002
3004	Fabian Johnson	Paris	300	5006
3007	Brad Davis	New York	200	5001
3009	Geoff Cameron	Berlin	100	5003
3008	Julian Green	London	300	5002
3001	Brad Guzan	London		5005
3003	Jozy Altidore	Moscow	200	5007

Figure 6: Customer and Salesman table

1. Write a SQL statement to find the details of a order i.e. order number, order date, amount of order, which customer gives the order and which salesman works for that customer and commission rate he gets for an order.



customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3005	Graham Zusi	California	200	5002
3004	Fabian Johnson	Paris	300	5006
3007	Brad Davis	New York	200	5001
3009	Geoff Cameron	Berlin	100	5003
3008	Julian Green	London	300	5002
3001	Brad Guzan	London		5005
3003	Jozy Altidore	Moscow	200	5007

salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5003	Lauson Hense		0.12
5007	Paul Adam	Rome	0.13

Figure 7: Customer and Salesman table

2. Write a SQL statement to make a list in ascending order for the customer who works either through a salesman or by own.

3. Attach with query codes and with output screenshots in the report.

7 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.