Dhirubhai Ambani Iinstitute of Information and Communication Technology
Gandhinagar

P.M. Jat

Lab Manual
IT615 Database Management Systems
Autumn’2015

Practicing and Learning Pedagogy

Following is learning sequence in lab of this course, and is aligned with class room teachings -

1. Querying relational databases through Relational Algebra vis-à-vis SQL
2. Conceptual Modeling using ER Modeling along with enhanced features of aggregation, generalization/specializations
3. Functional Dependencies and Normalizations
4. Views, Stored Procedures, and Triggers
5. Embedded SQL
6. Transaction Processing
7. Indexes
8. Query Execution and Optimizations

Labs of the course are conducted in two modes-

1. **Lab exercises**. For practicing techniques for numbered 1, and 6 to 8 in above list, a set of lab exercises are given.
2. **Individual Mid-size group projects**. Techniques 2 through 6 are done by this route.

DBMS Server used for lab work: PostgreSQL[1].

[1] http://www.postgresql.org/

List of Lab Exercise

Lab01	Get Started
	IT615 Database Management System

Focus of lab work in this week is to get familiarity with environment, and learn to create tables (relations) using SQL.

Do following tasks on remote server for which details have been given to you. Your TAs have access to your database and will be able to see what you have done. Try to finish your work by the 6 PM of your lab day.

1. Look at following pages and understand basics of creating tables and using CREATE TABLE statement -
   http://intranet.daiict.ac.in/~pm_jat/postgres/html/ddl-basics.html
   http://intranet.daiict.ac.in/~pm_jat/postgres/html/ddl-constraints.html

2. Look at following pages and learn about basic data types supported by postgreSQL
   http://intranet.daiict.ac.in/~pm_jat/postgres/html/datatype.html
   (Look at only 8.1.1, 8.1.2, 8.1.3, 8.3, 8.5.1)

Start pgAdmin, connect to postgreSQL server; open SQL window, and perform following actions.

3. Create Schema named Company using following command in SQL window.
   ```
   CREATE SCHEMA company;
   ```

4. Set this as default work schema by issuing following command-
   ```
   SET SEARCH_PATH TO company;
   ```

5. Create tables as per DDL given in figure below.

6. Add some tuples to the company database by executing statements in script
   http://intranet.daiict.ac.in/~pm_jat/insert_company.sql

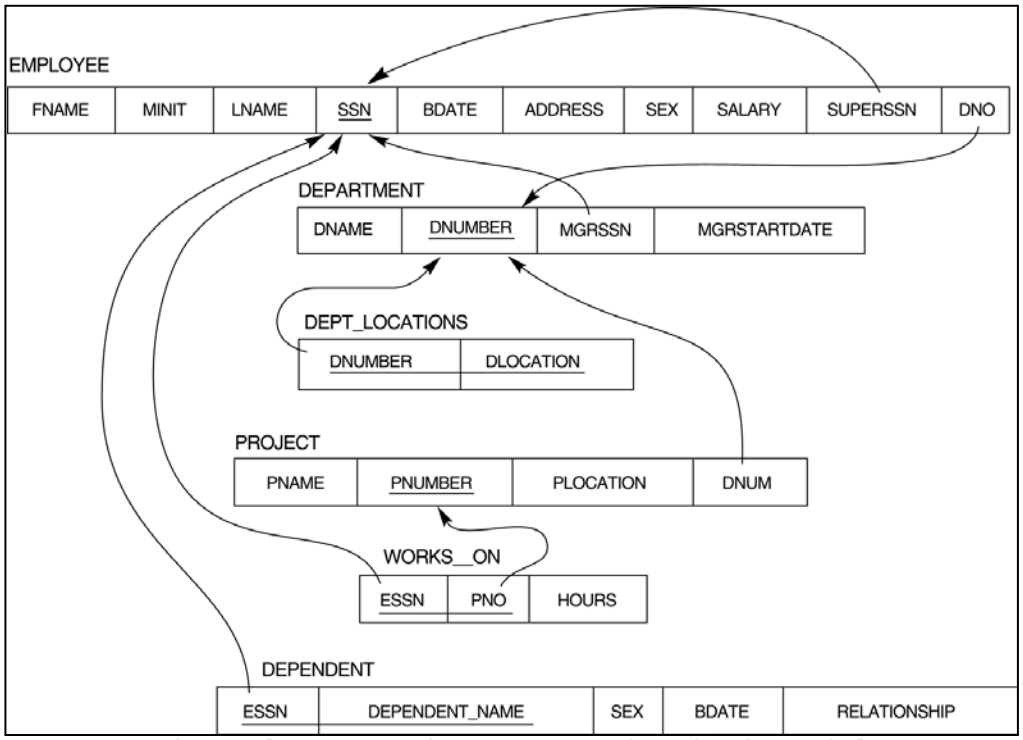

Figure-1 [Source: Database Systems, Elmasri and Navathe]

```
CREATE TABLE employee (
      fname VARCHAR(20),
      minit CHAR(1),
      lname VARCHAR(20),
      ssn DECIMAL(9,0),
      bdate DATE,
      address VARCHAR(30),
      sex CHAR(1),
      salary DECIMAL(10,0),
      superssn DECIMAL(10,0),
      dno SMALLINT,
      PRIMARY KEY (ssn),
      FOREIGN KEY (superssn) REFERENCES employee(ssn)
            ON DELETE SET DEFAULT ON UPDATE CASCADE
);

CREATE TABLE department (
      dname VARCHAR(20),
      dno SMALLINT,
      mgrssn DECIMAL(9,0),
      mgrstartdate DATE,
      PRIMARY KEY (dno),
      FOREIGN KEY (mgrssn) REFERENCES employee(ssn)
            ON DELETE SET DEFAULT ON UPDATE CASCADE);

CREATE TABLE dept_locations(
      dno SMALLINT,
      dlocation VARCHAR(20),
      FOREIGN KEY (dno) REFERENCES department on delete cascade,
      PRIMARY KEY (dno,dlocation)
```

```
);

CREATE TABLE project(
      pname VARCHAR(20),
      pno SMALLINT,
      plocation VARCHAR(20),
      dno SMALLINT,
      PRIMARY KEY (pno),
      FOREIGN KEY (dno) REFERENCES department(dno)
);

CREATE TABLE works_on(
      essn DECIMAL(9,0),
      pno SMALLINT,
      hours DECIMAL(5,1),
      FOREIGN KEY (essn) REFERENCES employee(ssn)
            ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY (pno) REFERENCES project(pno)
            ON DELETE CASCADE ON UPDATE CASCADE,
      PRIMARY KEY (essn,pno)
);

CREATE TABLE dependent(
      essn DECIMAL(9,0),
      dependent_name VARCHAR(20),
      sex CHAR(1),
      bdate DATE,
      relationship VARCHAR(20),
      FOREIGN KEY (essn) REFERENCES employee(ssn)
            ON DELETE CASCADE ON UPDATE CASCADE,
      PRIMARY KEY (essn,dependent_name)
);

ALTER table employee add FOREIGN KEY (dno) REFERENCES
department(dno);
```

Figure-2

Lab02	Operations on Relations #1
	IT615 Database Management System

Focus of lab work in this week is to learn and practice Relational Algebra and SQL. Always follow the sequence of Relational Algebra and then SQL while writing queries.

1. Write down queries for following queries on company schema created in last week-
   a. List particulars of Employees for department no = 1.
   b. List particulars of all Employees working at 'Headquarter'.
   c. List the Department Name and Name of its Manager.
   d. List Employees directly supervised by 'Franklin' (fname)
   e. List the names of all employees in department no = 5 who work more than 10 hours per week on the ProductX project.

2. Understand schema of DA-IICT academic scenario from DDL script http://intranet.daiict.ac.in/~pm_jat/acad_ddl.sql, and draw relational schema diagram for the same.

   Also, and write down queries for following queries-
   a. Retrieve all students of B.Tech.(CS) batch 2007 having CPI >= 7.0
   b. List ID, and Name of students who have registered in 'MT101' in 'Autumn' of 2008
   c. List Courses offered by Instructor 'P M Jat' in 'Winter' of 2010

   You can test run your SQL queries on sample database stored in a database named as public (schema **acad**) on your postgresql server.

3. Get introduced with following relational schema, given as following -

   Suppliers(sid:int, sname:varchar(20), city:varchar(20))
   Parts(pid:int, pname:varchar(20), color:varchar(10))
   Supplies(sid:int, pid:int, cost:int)
   -- **sid** and **pid** are FKs referring into **Suppliers** and **Parts** respectively

   Write down queries in relational algebra and SQL for following queries-
   a. Find the names of suppliers who supply some red part.
   b. Find **sid** of suppliers who supply some red or green part
   c. Find the **sid** of suppliers who supply some red part or are located at 'Delhi'.

   You can test your SQL queries on sample database stored in a database named as public (schema **supplyparts**) on your postgresql server.

Lab03	Operations on Relations #2
	IT615 Database Management System

This week also continue working on "Operations on Relations" in Algebra and SQL.
Important Note: Do it in Relational Algebra first, try to convert algebra expressions into SQL.

1. Considering acad scenario, write down queries in relational algebra and SQL for following queries-
   a. List IDs of students who have taken courses in 'MT101' or have taken 'MT104'
   b. List IDs of students who have taken courses in 'MT101' and also have taken 'MT104'
   c. List IDs of students who have taken courses in 'MT101' but have not taken 'MT104'
   d. List IDs of students who have taken courses in 'MT104' but have not taken 'MT101'
   e. List ID, and Name of students who have taken courses in 'MT101' and also have taken 'MT104' of 2007 batch
   f. List name and dname of employee who does not work on any project
   g. List student have scored AA in all courses in Semester Autumn 2008
   h. List student have scored AA or AB in all courses in Semester Autumn 2008
   i. List all courses offered by Prof 'P. M. Jat' from Autumn 2007 to Summer 2011
   j. List prog-ids and batches of students who have taken all courses offered by Prof 'P. M. Jat' from Autumn 2007 to Summer 2011
   k. List IDs of students of B.Tech.(CS) batch 2007 having SPI >= 6 in all semesters
   l. List IDs of students of B.Tech.(CS) batch 2007 not having any F grade

   An instance of acad schema has been created on your postgresql server (Database name: public and schema name acad). We are in the process of adding more data

2. Considering following set of relations from database of MyFaceBook -

   user(uid:int, name:varchar, email:varchar, joindate:date, city:varchar)
   -- represents all users/members registered with MyFaceBook
   city(cityname: varchar, country: varchar)
   -- based on assumption that city name is unique !
   friend(id:int, fid:int, since:date, messages_in:int, messages_out:int)
   -- Semantics of attributes: **id** has a set of friends identified by **fid** since **since**, **message_in** and **message_out** are counts of incoming and outgoing communications respectively, between **id** and **fid**.
   --FKs: **id** and **fid** of friend refers to **user(uid)**, and **city** of user refers to **city(cityname)**

   a. List friends (name, email) of uid 12345, that are older than one year
   b. List common friends (name, email) of uid 12345 and uid 23456
   c. Lids users (name, email) of user having all friends of uid 12345
   d. Lids users (name, email) of friends of uid 12345 with whom 12345 had no communication either way)
   e. List friends (name, email) of friends of uid 12345, that are from same country as of uid 12345

f. List friends (name, email) of friends of uid 12345, that are not from same country as of uid 12345
g. List users (name, email) that have no friends from outside of their own country

Lab04	Operations on Relations #3
	IT615 Database Management System

This week also continue working on "Operations on Relations" in Algebra and SQL. Important Note: Do it in Relational Algebra first and then translate into SQL. Submit Algebraic solutions to your TAs before leaving lab.

4. Considering following set of relations aimed to store sales related data of a trading company (underlined are PKs of relations)-

```
Customer(CustNo:int, Name: varchar(20))
Item(Code:int, Name:varchar(20), Category:int,
SalePrice:int, Stock:int, ReOrderLevel:int,
AveragePurchasePrice:int)
Sales(InvNo:int, InvDate:Date, CustomerNo:int)
   FK: CustomerNo references into Customer
SalesDetails(InvNo:int, ItemCode:int, Qty:int, Rate:int)
   FKs: InvNo references into Sales; ItemCode references
Items
```

   h. List items requires to buy more (that is items have stock below their reorder level)

   i. What is sale of given date (total of sales amount all invoices of the date)
      a. Total sale amount for a day can be computed by sum (qty * price) from SalesDetails relation for invoices that are having the date in their InvDate attribute
   j. List items having aggregated profit more than 30%
      a. i.e. computationally, Average of (sale_price) >= 1.3 * AveragePurchasePrice
   k. Most valuable customer of the year (customer giving max profit in given year)
      a. Hint: SQL provides LIMIT option on result to be returned from a SELECT
   l. Top 10 selling items (in terms of numbers) of the year
   m. Top 10 profitable items for the given year.

5. Considering acad schema, write down queries in relational algebra and SQL for following queries-
   a. Retrieve Name of faculty who took more than one course in a semester, along with the details of courses and semester
   b. Retrieve all students (StudentID, Name, TotalCreditTaken) for given Batch, Academic Year, Semester (Autumn|Winter|Summer), and ProgID
   c. Retrieve all students (StudentID, Name, TotalCreditTaken) for B.Tech.(CS) batch 2007 who have taken less than 10 or more than 20 Credits in current semester.
   d. Retrieve all students (Id and name) who got more than three F grades.

Projects	Individual Database Projects
	IT615 Database Management System

To get hands on, it is expected that you design and implement database system for a relatively large and real-life situation.

For project, you will work in groups, and permitted group size is 3 to 5 (not less, not more).

Project Evaluation Parameters:

1. Size and Complexity of Schema.
2. Realistic to the real-word
3. Complexity of queries answered
4. Goodness of the solutions

Project Milestones -

1. **Write a description of the scenario**. [Tentative Submission Date: Middle of September] There is no standard format for scenario description, idea is you must be able to state the scope of database, and capture all data requirements from the description. Sample scenario is being provided for your reference.

   Important Note: In order to identity scope of the database project; try to understand database in users perspective that what purpose this database will serve to the user, what questions of user, database will be able to answer, and so forth. Do not think in terms of table or entities.

2. Draw **ER Diagram** for the scenario [Tentative Submission Date: September end]. ER diagram should include Cardinality and participation constraints. Use Ternary Relationship, Generalization/Specializations judiciously; use them only when you do not find any other way out.

   Use software Dia for creating ERDs. The software is in software folder in my lecture folder.

3. Create **relational schema diagram** from your ER diagram. Identify all Functional Dependencies in your Database and prove that all relations, you have created are in BCNF. If not in BCNF decompose it to bring in BCNF, if you cannot, then give reason, why you can't. [Tentative Submission Date: October 3rd week]

4. **Implement the database** as separate schema on the PostgreSQL server. Save DDL scripts for creating schema. You will have to submit schema diagram and DDL scripts of your project database. [Tentative Submission Date: October 3rd end]

5. **Retrieval queries on your project database**. You should include as many queries as your scenario may demand. I expect each scenario should have about 10-15 GOOD queries. Test your queries on sample database (Populate your database with sample data

through INSERT statements and save INSERT statements in a script file)
[Tentative Submission Date: November Middle]

6. **Stored procedures and Triggers** in your project scenario. Identify some functionality and constraints that can be better accomplished by stored procedures and triggers. [Tentative Submission Date: November Middle]

7. Create a simple console based program (No GUI required) using embedded SQL in C that accesses your project database [Tentative Submission Date: November Middle]

8. Final document containing ERD and relational schema with normalization proofs. SQL statements of all the queries including DDL INSERT queries. Also submit code of your stored procedures and console application. [Tentative Submission Date: November Middle]

9. Final Presentation and Viva: Dates to be announced. For final viva and presentations, you are to bring hard copies as following –
    a. ERD (2 copies)
    b. Relational Schema Diagram (2 copies)
    c. Note: Relational schema should be atleast in 3NF; BCNF desirable.
    d. Set of FDs (no MVDs) normalization proofs. (1 copy).
    e. If any relation is not in BCNF, give reason why it could not be decomposed into BCNF
    f. SQL Scripts of SELECT queries (1 copy)
    g. Source Code of your stored procedures and console application. (1 copy)

Lab05	Transaction Processing (Source: taken from Prof S Sudarshan's[2] Assignments and modified to company schema)
	IT615 Database Management System

In this lab, you experience, different isolation levels in SQL in general and postgreSQL in particular. Have two client windows connected to your postgresql server connected to company schema. Windows1 and Window2 refer to these clients respectively in following explanations.

Exercise-1

Window1: begin;
Window1: update employee set salary = 45000 where ssn = '123456789';
Window2: begin;
Window2: select * from employee where ssn = '123456789';

Look at the value of salary.
Can you figure out why you got the result that you saw?
What does this tell you about concurrency control in PostgreSQL?

Window1: commit;
Window2: select * from employee where ssn = '123456789';

Look at the value of salary again. Why is it so?

Windows2 now shows updated salary of the employee.
Window2: commit;

Exercise-2

Window1: begin;
Window1: update employee set salary = 45000 where ssn = '123456789';
Window2: begin;
Window2: update employee set salary = salary+3000 where ssn = '123456789';

Hold for a moment, and see result of this query

Window1: commit;
Window2: commit;
Window2: select salary from employee where ssn = '123456789';

Why do you get the result what you see? What if T1 aborts?

[2] http://www.cse.iitb.ac.in/~sudarsha/

Exercise-3

Window1: begin;
Window1: update employee set salary = 45000 where ssn = '123456789';
Window2: begin;
Window2: update employee set salary = (select min(salary) from employee where ssn = '123456789')+3000 where ssn = '123456789';

Hold for a moment, and see result of this query

Window1: commit;
Window2: commit;
Window2: select * from employee where ssn = '123456789';

Why do you get the result what you see?
What isolation level you observe?

Exercise-4

Set isolation to serializable by running following statement after begin in both windows, and rerun exercise-3 again

set transaction isolation level serializable;

Exercise-5

Window1: select ssn, salary from employee where ssn in('888665555', '123456789');
note the results
Window1: begin;
Window1: set transaction isolation level serializable;
Window2: begin;
Window2: set transaction isolation level serializable;

Window 1: update employee set salary = (select salary from employee where ssn = '888665555') where ssn = '123456789';

Window 2: update employee set salary = (select salary from employee where ssn = '123456789') where ssn = '888665555';

Window1: commit;
Window2: commit;

Window1: select ssn, salary from employee where ssn in('888665555', '123456789');

Compare the results. Is this equivalent to any serializable schedule?

What is wrong, and what is solution?

Now, just before the update statement in window 1, add the line

select salary from employee where ssn = '888665555' for update;

Do you see any change?

Lab06	Understand Query Execution Plans
	IT615 Database Management System

In this lab you experience query execution plan and algorithms that are selected for query execution by query optimizer.

Here, you use Explain statement of SQL or Visual Query Execution Plan viewer like pgAdmin[3] from postgres, or PICASSO[4] from IISc, Bangalore.

[3] http://www.pgadmin.org/
[4] http://dsl.serc.iisc.ernet.in/projects/PICASSO/