University Of Pune

Database Assignments and Mini Project for Designing Backend using Software Engineering techniques

S. Y. B. Sc. (Computer Science)

CS -224

SEMESTER I & II

Name ____

College Name ____

Roll No. _______________________ Division ______________________

Academic Year ____________

Prepared By

Ms. Reena Bharathi

Ms. Deepali Joshi

Reviewed By

Dr. Shailaja C. Shirwaikar

Table of contents

Introduction

1. About the work book

This workbook is intended to be used by S. Y. B. Sc (Computer Science) students for the Database Assignments in semester I and mini project for Designing backend using Software Engineering techniques in semester II..

The objectives of this book are

1) Defining clearly the scope of the course
2) Bringing uniformity in the way the course is conducted across different colleges
3) Continuous assessment of the course
4) Bring in variation and variety in the experiments carried out by different students in a batch
5) Providing ready reference for students while working in the lab
6) Catering to the need of slow paced as well as fast paced learners

2. How to use this workbook

The workbook is divided into two sections. Section 1 is related to Database Assignments and they are to be carried out using PostgreSQL in Linux environment

The database Syllabus is divided into seven assignments. The assignments comprise of activities to be carried out on one or more of the six to eight databases. The students have to create, populate database and then perform the activities specified in each of the assignments. A pool of databases will get created as student progresses through the assignments and these databases can be repeatedly used in subsequent assignments. It is mandatory that students create all the databases and also use them in one or other assignment.

Each Assignment has a set A and set B and may contain more sets. Each set contains tasklists related to one database. The tasks are a mix of simple and difficult tasks. Students should complete tasks on specified number of sets. Depending on time availability student should complete more sets.

2.1 Instructions to the students

Please read the following instructions carefully and follow them.

1) Students are expected to carry this book every time they come to the lab for computer science practicals.

2) Students should prepare oneself before hand for the Assignment by reading the relevant material.

3) Instructor will specify which problems to solve in the lab during the allotted slot and student should complete them and get verified by the instructor. However student should spend additional hours in Lab and at home to cover as many problems as possible given

in this work book.

4) Students should complete the Forms related to Mini Project using pencil and discuss with the instructor/guide and then finalize them before the submission date specified by the instructor/guide

5) Students will be assessed for each exercise on a scale from 0 to 5

i) Not done	0
ii) Incomplete	1
iii) Late Complete	2
iv) Needs improvement	3
v) Complete	4
vi) Well Done	5

2.2. Instruction to the Instructors

1) Explain the assignment and related concepts in around ten minutes using white board if required or by demonstrating the software.

2) Choose appropriate problems to be solved by students.

3) Fill the blanks in queries giving specific values which can vary from student to student.

4) Make sure that students follow the instruction as given above. Students should be encouraged to solve more sets than specified.

4) You should evaluate each assignment carried out by a student on a scale of 5 as specified above by ticking appropriate box.

5) The value should also be entered on assignment completion page of the respective Lab course.

2.3. Instructions to the Lab administrator

You have to ensure appropriate hardware and software is made available to each student.

The operating system and software requirements on server side and also client side are as given below:

1) Server and Client Side - (Operating System) Fedora Core Linux

2) Database server – PostgreSQL 7.0 or above

Assignment Completion Sheet

Lab Course II			
Section I – Database Assignments			
Sr. No	**Assignment Name**	**Marks (out of 5)**	**Signature**
1	Simple Queries		
2	Nested Queries, using aggregate functions		
3	Queries using Views		
4	Stored Function		
5	Cursors		
6	Exception Handling		
7	Triggers		
Total (out of 35)			
Total (Out of 10)			

Lab Course II			
Section II – Mini Project for Designing Backend using Software Engineering Techniques			
Sr. No	**Assignment Name**	**Marks (out of 5)**	**Signature**
1	Problem definition , scope		
2	Feasibility study		
3	Gathering Data Requirements and Functional Requirement		
4	ER Diagrams		
5	Designing the normalized Database		
6	Designing queries related to Functional requirements		
Total (out of 30)			

This is to certify that Mr/Ms ____ has successfully completed the course work for Lab Course II (Mini Project) and has scored ___ Marks out of 30.

Instructor **Head, Dept. Of Comp. Sc.**

Internal Examiner **External Examiner**

Section I – Database Assignments

SQL is a strongly typed language; implying that any piece of data represented by postgreSQL has an associated data type.

PostgreSQL supports a wide variety of built-in data types, and it also provides an option to the users to add new data types to PostgreSQL , using the CREATE TYPE command. Table lists the data types officially supported by PostgreSQL. Most data types supported by PostgreSQL are directly derived from SQL standards. The following table contains PostgreSQL supported data types for your ready reference

Category	Data type	Description
Boolean	boolean, bool	A single true or false value.
Binary types	bit(n)	An n-length bit string (exactly n)
	bit varying(n), varbit(n)	A variable n-length bit string (upto
Character Types	character(n)	A fixed n-length character string
	char(n)	A fixed n-length character string
	character varying(n)	
	varchar (n)	
	text	A variable length character string
Numeric types	smallint, int2	A signed 2-byte integer
	integer, int, int4	A signed, fixed precision 4-byte
	bigint, int8	A signed 8-byte integer, up to 18
	real, float4	A 4-byte floating point number
	float8, float	An 8-byte floating point number
	numeric(p,s)	An exact numeric type with
Currency	money	A fixed precision, U.S style
	serial	An auto-incrementing 4-byte
Date and time types	date	The calendar date(day, month
	time	The time of day
	time with time zone	the time of day, including time
	timestamp(includes time	
	interval	An arbitrarily specified length

Assignment I - Simple Queries

Solve atleast ONE from the following sets

Set A

Person –Area Database

Consider the relation Person (pnumber, pname, birthdate, income), Area(aname, area_type). An area can have one or more persons living in it, but a person belongs to exactly one area. The attribute 'area_type' can have values either 'urban' or 'rural'.
Create the relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).
Insert sufficient number of appropriate records.

Solve the Queries

1. List the names of all people living in '_______' area.
2. List the details of all people whose names start with the alphabet '__'.
3. List the names of all people whose birthday falls in the month of_______.
4. Count the people who are born on_______.
5. Count the people whose income is below______.
6. List the names of all people whose income is between ______and _______.
7. List the names of people with average income.
8. List the details of the people, sorted by person name.
9. Transfer all people living in 'Pune' to 'Mumbai'.
10. Delete information of people staying in 'urban' area.

Set B

Movie Database

Movies(M_name, release_year, budget)
Actor(A_name, role, charges, A_address)
Producer(producer_id, name, P_address)
Each actor has acted in one or more movies. Each producer has produced many movies and each movie can be produced by more than one producers. Each movie has one or more actors acting in it, in different roles.
Create the relations accordingly, so that the relationship is handled properly and the relations are in normalized form(3NF).
Insert sufficient number of appropriate records.

Solve the Queries:
1. List the names of actors who have acted in at least one movie, in which '_______' has acted.
2. List the names of the movies with the highest budget.
3. List the names of actors who have acted in the maximum number of movies.
4. List the names of movies, produced by more than one producer.
5. List the names of actors who are given with the maximum charges for their movie.
6. List the names of producers who produce the same movie as '________'.
7. List the names of actors who do not live in _________or _________ city.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Assignment 2 - Nested Queries, using aggregate functions

Solve atleast THREE from the following sets

Set A
Bank database
Consider the following database maintained by a Bank. The Bank maintains information about its branches, customers and their loan applications.

Following are the tables:
BRANCH (**BID** INTEGER, BRNAME CHAR (30), BRCITY CHAR (10))
CUSTOMER (**CNO** INTEGER, CNAME CHAR (20), CADDR CHAR (35), CITY CHAR(20))
LOAN_APPLICATION (**LNO** INTEGER, LAMTREQUIRED MONEY, LAMTAPPROVED MONEY, L_DATE DATE)

The relationship is as follows:
BRANCH, CUSTOMER, LOAN_APPLICATION are related with ternary relationship.

TERNARY (**BID INTEGER, CNO INTEGER, LNO INTEGER**).

Solve the Queries

1. Find the names of the customers for the “Aundh” branch.
2. List the names of the customers who have received loan less than their requirement.
3. Find the maximum loan amount approved.
4. Find out the total loan amount sanctioned by “Deccan “branch.
5. Count the number of loan applications received by “M.G.ROAD” branch.
6. List the names of the customer along with the branch names who have applied for loan in the month of September.

Set B
Student- Teacher database
Consider the following database maintained by a school. The school maintains information about students and the teachers. It also gives information of the subject taught by the teacher.

Following are the tables:
STUDENT (**SNO** INTEGER, S_NAME CHAR(30), S_CLASS CHAR(10), S_ADDR CHAR(50))
TEACHER (**TNO** INTEGER, T_NAME CHAR (20), QUALIFICATION CHAR (15),EXPERIENCE INTEGER)
The relationship is as follows:
STUDENT-TEACHER: M-M with descriptive attribute SUBJECT.

Solve the queries
1. Find the minimum experienced teacher.
2. Find the number of teachers having qualification “Ph. D.”.
3. List the names of the students to whom “Mr. Patil” is teaching along with the subjects he is teaching to them.
4. Find the subjects taught by each teacher.

5. List the names of the teachers who are teaching to a student named “Suresh”.
6. List the names of all teachers along with the total number of students they are teaching.

Set C

Project-Employee database

Consider the database maintained by a company which stores the details of the projects assigned to the employees.

Following are the tables:
PROJECT (**PNO** INTEGER, P_NAME CHAR(30), PTYPE CHAR(20),DURATION INTEGER)
EMPLOYEE (**ENO** INTEGER, E_NAME CHAR (20), QUALIFICATION CHAR (15), JOINDATE DATE)
The relationship is as follows:
PROJECT - EMPLOYEE: M-M Relationship , with descriptive attributes as start_date (date), no_of_hours_worked (integer).

Solve the Queries

1. Find the names of the employees starting with ’A’.
2. Find the details of employees working on the project “System”.
3. Find the employee numbers of the employees, who do not work on project “Robotics”.
4. Get employee number of the employee, who works on at least one project that employee number ‘2000’ works on.
5. List the names of the first three employees in alphabetical order.
6. Find the names of the employees who have worked for more than three years.

Set D

Business trip database

Consider the business trip database that keeps track of the business trips of salesman in an office.
Following are the tables:
SALESMAN (**SNO** INTEGER, S_NAME CHAR (30), START_YEAR YEAR, DEPTNO VARCHAR (10))
TRIP (**TNO** INTEGER, FROM_CITY CHAR (20), TO_CITY CHAR (20), DEPARTURE_DATE DATE, RETURN DATE)
DEPT (**DEPTNO** VARCHAR (10), DEPT_NAME CHAR(20))
EXPENSE (**EID** INTEGER, AMOUNT MONEY)
The relationship is as follows
DEPT-SALESMAN 1 TO M
SALESMAN - TRIP 1 TO M
TRIP - EXPENSE 1 TO 1

1. Give the details for trips that exceed Rs. 10,000 in expenses.
2. List the salesman numbers and names of the salesmen who made trips to Calcutta.
3. Delete all the trips made by department “computer” having expenses more than Rs. 15000.
4. Find the departments from which the salesmen have done highest number of trips.
5. Find the total expenses incurred by the salesman “Mr. Patil”.

6. Increase the expenses of all the trips by Rs. 5000.

Set E
Warehouse Database

CITIES(**CITY** CHAR(20),STATE CHAR(20))
WAREHOUSES(**WID** INTEGER,WNAME CHAR(30),LOCATION CHAR(20))
STORES(**SID** INTEGER,STORE_NAME CHAR(20), LOCATION_CITY CHAR(20))
ITEMS(**ITEMNO** INTEGER,DESCRIPTION TEXT,WEIGHT DECIMAL(5,2), COST DECIMAL(5,2))
CUSTOMER(**CNO** INTEGER, CNAME CHAR(50),ADDR VARCHAR(50), CU_CITY CHAR(20))
ORDERS(**ONO** INT,ODATE DATE)

The relationship is as follows
CITIES-WAREHOUSES 1 TO M
WAREHOUSES - STORES 1 TO M
CUSTOMER – ORDERS 1 TO M
ITEMS – ORDERS M TO M relationship with descriptive attribute ordered_quantity
STORES-ITEMS M TO M RELATION with descriptive attribute quantity

Solve the following queries.

1. Find the item that has minimum weight.
2. Find the different warehouses in “Pune”.
3. Find the details of items ordered by a customer “Mr. Patil”.
4. Find a Warehouse which has maximum stores.
5. Find an item which is ordered for minimum number of times.
6. Find the details orders given by each customer.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Assignment 3 - Views

Views can be thought of as stored queries, which allow us to create a database object that functions very similarly to a table, but whose contents dynamically reflect the selected rows. Views are very flexible; you may use a view to address common simple queries to a single table, as well as for complicated ones which may span across several tables.

Creating a View : CREATE [OR REPLACE] [TEMP | TEMPORARY] VIEW *name* [(*column_name* [, ...])] AS *query*

CREATE VIEW defines a view of a query. The view has no physical existence, but is dynamically created whenever it is referenced in a query.

CREATE OR REPLACE VIEW replaces an existing view of same name. The new query must use the same column names in the same order and with the same data types, but it may add additional columns to the end of the list.

Schema name (CREATE VIEW myschema.myview) must be specified to create a view in schema other than the current schema. The TEMPORARY or TEMP parameter is specified to create Temporary views, however If any of the tables referenced by the view are temporary, the view is created as a temporary view (whether TEMPORARY is specified or not). Temporary views exist in a special schema, so a schema name cannot be given when creating a temporary view. View names should be distinct.

Name:Name (optionally schema-qualified) of a view to be created.

column_name: An optional list of names to be used for columns of the view. If not given, the column names are deduced from the query.

Query: A SELECT command which will provide the columns and rows of the view.

Example to create a view consisting of all comedy films

```
CREATE VIEW comedies AS

   SELECT *

   FROM films

   WHERE type = 'Comedy';
```

ALTER VIEW statement is used to change the definition of a view.

ALTER VIEW *name* ALTER [COLUMN] *column* SET DEFAULT *expression*

ALTER VIEW *name* ALTER [COLUMN] *column* DROP DEFAULT

DROP VIEW statement is used to remove a view

DROP VIEW [IF EXISTS] name [, ...] [CASCADE | RESTRICT]

<u>Solve any THREE of the following Sets :</u>

<u>Set A</u>
Using the Bank database
1. Create a view which contains the details of all customers who have applied for a loan more than Rs.100000.

2. Create a view which contains details of all loan applications from the ‘Sadashiv peth’ branch.
3. Write the following Queries, on the above created views :
   a. List the details of customers who have applied for a loan of Rs. 500000.
   b. List the details of loan applications from Sadashiv peth , where loan amount is > Rs 50000.
   c. List the details of Loan applications, with the same loan amount.

<u>Set B</u>
Using Project-Employee database
1. Create a view over the employee table which contains only employee name and his qualification and it should be sorted on qualification.
2. Create a view containing the project name, project type and start date of the project and should be sorted by start date of the project.
3. Write the following Queries, on the above created views :
   a. List different qualifications of employees.
   b. List the name and type of projects started on 1st April 2014.
   c. List the names of employees who are qualified as Engineers.

<u>Set C</u>
Using the Business trip database.
1. Create a view to list the details of all salesman from ‘ Western ‘ department.
2. Create a view to list all salesman’s name his trip details and his expenses for the trip.
3. Write the following Queries, on the above created views :
   a. List the details of salesmen from ‘Western’ department, whose start year is 2005.
   b. List the names of salesmen from ‘Western’ department, for whom their total trip expense is > Rs 100000.
   c. List the names of salesmen who have gone on a trip to “Mumbai” city.

<u>Set D</u>
Using the Warehouse database.
1. Create a view containing details of all the stores of a Warehouse named ’Spares’.
2. Create a view to list the details of all customers who have placed orders on the date ’03-10-2013’ .
3. Write the following Queries, on the above created views :
   a. ‘List the names of stores of ‘Spares ’ warehouse, located at Pune.
   b. List the names of customers from Pune city, who have placed orders on 03-10-2013
   c. List the orders placed by “Mr. Joshi”.

<u>Set E</u>
Using the Student- Teacher database.
1. Create a view containing details of all the teachers teaching the subject ‘Mathematics’.
2. Create a view to list the details of all the students who are taught by a teacher having experience of more than 3 years .
3. Write the following Queries, on the above created views :
   a. List the name of the most experienced teacher for “Mathematics”.
   b. List the names of students of ‘S.Y.B.Sc’ class, who are taught by a teacher having more than 3 years experience.

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Assignment 4 - Stored Functions

Stored functions are user defined functions, that are created using the CREATE FUNCTION statement . The functions, thus created, are called stored functions because they are stored as database objects within the PostgreSQL database. The CREATE FUNCTION command names the new function, states its arguments and return type.

Creating a stored function:

```
CREATE FUNCTION identifier (arguments) RETURNS type AS ‘
`DECLARE
   Variable–declarations;
    [……]
BEGIN
    Statement;
    [……..]
END ;
‘ LANGUAGE ‘plpgsql’;
```

Argument Variables:

PL/pgSQL functions can accept argument variables of different types. Function arguments allow a user to pass information into a function, that the function may need.

Each function argument that is received by a function is incrementally assigned to an identifier that begins with the dollar ($) sign and is labeled with the argument number. Thus the identifier $1 is used for the first argument, $2 is used for the second argument and so on an so forth.

PL/pgSQL allows us to create variable aliases. Aliases are created using the ALIAS keyword and it gives us the ability to provide an alternate variable to use when referencing argument variables. All aliases should be declared within the DECLARE section of a block before they are used.

Returning variables:

PL/pgSQL functions must return a value that matches the data type specified as the return type during the function definition. Values are returned from a function using the RETURN statement.

Attributes:

PL/pgSQL provides variable attributes that basically assists or helps the database programmer , when working with database objects. These attributes are %TYPE and %ROWTYPE.
The %TYPE attribute : Used to declare a variable with the data type of a referenced database object (a table column). Syntax :
Variable_name table_name.column_name%TYPE ;
The %ROWTYPE attribute : Used to declare a PL/pgSQL record variable to have the same structure as the rows in a table , that we specify in the function block. Syntax :
Variable_name table_name%ROWTYPE ;

Controlling Program Flow:
Most programming languages provides different ways of controlling the flow of execution, within a program. PL/pgSQL also provides various statements using which a programmer can control the way actions will be executed within a PL/pgSQL function code. PL/pgSQL provides conditional statements and control loops for the same.

	Statement	Syntax
1	The IF/THEN statement	If < Condition> Then statement; [..] Else Statement; [..] End if;
2	**Nested If-else if**	If <Condition1> Then Statement; [..] Else if <Condition 2> Then Statement; [….] Else if <Condition 3> Then Statement; [……] Else Statement; [….] End if; End if; End if;
3	**Loops**	LOOP Statement; [………] EXIT [label] WHEN <condition> END LOOP;
4	**While Loop**	While Condition loop Statement; […….] End loop;
5	**For Loop**	For Loop–Index In {Reverse} expression1 ..... expression2 Loop Statement; [……..] End loop;
6	**For Loop (iterate through a query resultset)**	For {record_variable | %rowtype_variable } IN select_statement LOOP Statement; [………] END LOOP

Practice Examples :

a. A simple stored function to demonstrate a basic loop :

```
Create function loop_demo( ) returns integer as ‘
Declare
Cube–num integer := 2;
Begin
Loop
    Cube–num := Cube–num + 2;
    Exit When Cube–num > 20;
End loop;
Return cube-num;
End; ‘ language ‘plpgsql’;
```

b. A stored function that returns the names of students from a given address.

```
Create function for_demo(varchar ) returns text as ‘
Declare
     studaddr  alias for $1;
      /* declare a variable to hold student names and set its default value to a new line. */
     stud_info   text := ‘’\n’’;
  --declare a variable to hold rows from the student table.
     row_data student%Rowtype;
Begin
    --iterate through the results of a query.
     For row_data in select * from student
             where s_addr = studaddr  order by name
      Loop
         /* insert the name of the matching student into the stud_info variable */
          stud_info :=  stud_info || row_data.name || ‘’\n’’;
       end loop;
        -- return the list of students .
         Return stud_info;
     End;  ‘ language ‘plpgsql’;
```

Solve atleast THREE sets from the sets given below:

SET A
Using Bank Database
a) Write a function that returns the total number of customers of a particular branch.
(Accept branch name as input parameter.)
b) Write a function to find the maximum loan amount approved.

SET B:
Using Project-Employee database
a)Write a function to accept project name as input and returns the number of employees working on the project.
b)Write a function to find the number of employees whose date of joining is before ‘03/10/2010’

SET C

Business trip database

a)Write a PL/pgsql function to find a business trip having maximum expenses.
b)Write a PL/pgsql function to count the total number of business trips from ‘Pune’ to ‘Mumbai’.

SET D

Railway Reservation Database

Consider a railway reservation Database of passengers. Passengers reserve berths of a bogie of trains. The bogie capacity of all the bogies of a train is same.
1. TRAIN (TRAIN_NO INT, TRAIN_NAME VARCHAR(20), DEPART_TIME TIME , ARRIVAL_TIME TIME, SOURCE_STN VARCHAR(20) , DEST_STN VARCHAR (20), NO_OF_RES_BOGIES INT , BOGIE_CAPACITY INT)

2. PASSENGER (PASSENGER_ID INT, PASSENGER_NAME VARCHAR(20), ADDRESS VARCHAR(30), AGE INT , GENDER CHAR)
Relationship is as follows:
TRAIN _PASSENGER : M-M with descriptive attributes as follows :
TICKET (TRAIN_NO INT , PASSENGER_ID INT, TICKET_NO INT COMPOSITE KEY, BOGIE_NO INT, NO_OF_BERTHS INT , DATE DATE , TICKET_AMT DECIMAL(7,2),STATUS
CHAR)
The status of a particular berth can be ‘W‘ (waiting) or ‘C‘ (confirmed).

a)Write a PL/pgsql function to calculate the ticket amount paid by all the passengers on 13/5/2009 for all the trains.
b) Write a PL/pgsql function to update the status of the ticket from “waiting“ to “confirm“ for passenger named “Mr. Jadhav”

SET E

Bus transport Database

Consider the following Database of Bus transport system . Many buses run on one route. Drivers are allotted to the buses shiftwise.
Following are the tables:
BUS (BUS_NO INT , CAPACITY INT , DEPOT_NAME VARCHAR(20))
ROUTE (ROUTE_NO INT, SOURCE CHAR(20), DESTINATION CHAR(20), NO_OF_STATIONS INT)
DRIVER (DRIVER_NO INT , DRIVER_NAME CHAR(20), LICENSE_NO INT, ADDRESS CHAR(20), D_AGE INT , SALARY FLOAT)
The relationships are as follows:
BUS_ROUTE : M-1
BUS_DRIVER : M-M with descriptive attributes Date of duty allotted and Shift — it can be 1(Morning) or 2 (Evening).
Constraints :1. License_no is unique. 2. Bus capacity is not null.
a)Write a PL/pgsql function to find out the name of the driver having maximum salary.
b)Write a PL/pgsql function to accept the bus_no and date and print its allotted driver.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Assignment 5 : Cursors

PL/SQL Cursors provide a way to select multiple rows of data from the database and then to process each row individually. Using a cursor, we can traverse up and down a result set and retrieve only those rows which are explicitly requested. Cursors basically help an application to efficiently use a static result set.

Declaring Cursor Variables

All access to cursors in PL/pgSQL goes through cursor variables, which are always of the special data type refcursor. We can declare a cursor variable either as a bound cursor variable or an unbound cursor variable.

a. To create an unbound cursor variable , just declare it as a variable of type refcursor.

b. To create a bound cursor variable, use the following cursor declaration syntax

name CURSOR [(*arguments*)] FOR *query*;

where *arguments*, if specified, is a comma-separated list of pairs '*name datatype'* that define names to be replaced by parameter values in the given query. The actual values to substitute for these names will be specified later, when the cursor is opened (parameterized cursors).

Opening Cursors

Before a cursor can be used to retrieve rows, it must be opened. PL/pgSQL has three forms of the OPEN statement, two of which use unbound cursor variables while the third uses a bound cursor variable.

Syntax for OPEN FOR query

OPEN unbound_cursorvar FOR query;

Syntax for OPEN FOR EXECUTE

OPEN unbound_cursorvar FOR EXECUTE query_string USING expression [, ...]];

Syntax for Opening a Bound Cursor

OPEN *bound_cursorvar* [(*argument_values*)];

Using Cursors

Once a cursor has been opened, it can be manipulated with the statements described below.

FETCH

Syntax :

FETCH [*direction* { FROM | IN }] *cursor* INTO *target*;

The *direction* clause can be any of the following variants :

NEXT, PRIOR, FIRST, LAST, ABSOLUTE *count*, RELATIVE *count*, FORWARD, or BACKWARD. Default is NEXT.

MOVE : MOVE repositions a cursor without retrieving any data.

Syntax :

MOVE [*direction* { FROM | IN }] *cursor*;

The *direction* clause can be any of the variants NEXT, PRIOR, FIRST, LAST, ABSOLUTE *count*, RELATIVE *count*, ALL, FORWARD [*count* | ALL], or BACKWARD [*count* | ALL].

CLOSE : CLOSE closes the portal underlying an open cursor. This can be used to release resources earlier than end of transaction, or to free up the cursor variable to be opened again.

Syntax :

CLOSE *cursor*;

Practice Examples :

Example 1 :

Consider a relation, Employee (eno, ename, deptno,salary). We write a function, to Define a cursor to print the details of the employee along with commission earned for each employee. Commission is 20% of salary for employees of dept no = 5; its 50% of salary for employees of dept no = 8; its 30% of salary for employees of dept no = 10.

```
Create function cursor_demo( ) returns integer as ‘
Declare
Emp–rec Employee%rowtype
C–emp cursor for Select * from employee;
Comm Number (6,2);
Begin

Loop
Fetch C–emp into emp–rec;
 If emp–rec.deptno = 5 then
   Comm:= emp–rec.salary  * 0.2;
Else if emp–rec.deptno = 8 then
   Comm := emp–rec.salary * 0.5;
Else
    If emp–rec.deptno = 10 then
        Comm := emp–rec.salary * 0.3;
          End if;
        End if;
        Endif;
      Raise notice ‘’emp–rec.ename||emp–rec.deptno||emp–rec.salary||comm. ‘’;
       Exit when not found;
End loop;
Close C–emp;
End; ‘ language ‘plpgsql’;
```

Example 2 :
Consider the following relations,
Dept(dno, dname) Employee(eno, dno, ename, salary) and the relation between Dept and Employee is one to many(1:M). Now we write a PL/pgSQL function to print the list of employees, department wise. Here we use 2 cursors, the 1st Cursor retrieves the department Information for each department and the 2nd cursor, to which dept number is sent as a parameter retrieves the employees for that department.

```
Create function parameterizedcursor_demo( ) returns integer as ‘
Declare
  d–Info cursor for Select * from dept;
E–Info cursor (dnum dept.dno%type) for
Select * From Emplyee Where dno = dnum;
```

```
X number := 0;
Begin
For drec In d–Info
Loop
   Raise notice ‘’ drec.dno || drec.dname’’;
  For erec In E–Info (drec.dno)
    Loop
      Raise notice ‘’erec.eno || erec.ename || erec.salary’’;
      X := X + 1;
    End loop;
Raise notice “Total  employees for:||drec.dname||‘is %’|| X’’;
End loop;
Return (X);
End; ‘ language ‘plpgsql’;
```

Solve atleast THREE sets from the sets given below :

SET A

Using the Warehouse database

a) Write a stored function using cursors to accept a city from the user and to list all warehouses in the city.

b) Write a stored function using cursors to find the list of items whose cost is between Rs.5000 to 10000

SET B

Company –Person database

Company(Name varchar(30),address (50),city varchar(20), phone varchar(10), share _value money)

Person(pname varchar(30),pcity varchar (20))

Company_Person are related with M to M relationship with descriptive attribute No_of_shares. Integer

a) Write a stored function using cursors to transfer the shares owned by ‘Sachin ‘ to ‘Rahul’.

b) Write a stored function using cursors to print the total number of distinct investors along with its total invested value.

SET C

Student –Marks database

Student (rollno integer,name varchar(30),address varchar(50),class varchar(10))

Subject(Scode varchar(10),subject name varchar(20))

student and subject are related with M-M relationship with attributes marks scored.

Create a RDB in 3NF for the above and solve the following.

a) Write a stored function using cursors, to accept a address from the user and display the name,subject and the marks of the students staying at that address.

b) Write a stored function using cursors which will calculate total and percentage of each student

SET D

Using railway reservation database

a) Write a stored function using cursors to find the confirmed bookings of all the trains on 18-05-2009

b) Write a stored function using cursors to find the total number of berths not reserved for all the trains on 18-05-2009.

SET E

Using bus driver database

a) Write a stored function using cursors to display the details of a driver,

(Accept driver name as input parameter).

b) Write a stored function using cursors to display the details of the buses that run on routes 1,2 (Use two different cursors for route_no = 1 and route_no = 2).

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Assignment 6 : Handling errors and Exceptions

The RAISE statements raise errors and exceptions during a PL/pgSQL function's execution..A Raise statement is also given the level of error it should raise and the string error message it should send to postgreSQL. The string can also be embedded with variables and expressions, that one needs to list along with the error message. The percent (%) sign is used as the place holder for the variables that are inserted into the string.

The syntax of the RAISE statement is as follows :

RAISE level ''message string '' [, identifier [….]];

The three possible values for the RAISE statement's level are as follows:

Value	**Explanation**
DEBUG	Debug level statements send the specified text as a debug message to the PostgreSQL log.
NOTICE	Notice level statements send the specified text as a Notice;
EXCEPTION	Exception level statements send the specified text as an ERROR. The exception level also causes the current transaction to be aborted.

Practice examples :

Example 1
In this example , the first raise statement gives a debug level message. The second and third raise statements send a notice to the user. The fourth raise statement displays an error and throws an exception, causing the function to end and the transaction to be aborted.

```
 Create function raise_demo ( ) returns integer as '
Declare
    Int_var integer: =1;
Begin
     -- raise a debug level message
      Raise debug ''the raise demo function began'';
Int_var := int_var+1;
--raise a notice stating the change in value of variable
Raise notice '' variable int_var's value is now % .'',int_var;
--raise an exception
Raise exception ''variable % changed. Transaction aborted.'',int_var;
Return 1;
End;
'language 'plpgsql';
```

Example 2
The function given below changes the value of a variable, and on change it raises an exception and stops the function execution.

```
CREATE FUNCTION raise_test () RETURNS integer AS '
DECLARE
-- Declare an integer variable for testing.
an_integer INTEGER = 1;
BEGIN
-- Raise a debug level message.
RAISE DEBUG "The raise_test() function began.";
an_integer = an_integer + 1;
-- Raise a notice stating that the an_integer variable was changed,
-- then raise another notice stating its new value.
RAISE NOTICE "Variable an_integer was changed.";
RAISE NOTICE "Variable an_integer's value is now %.",an_integer;
-- Raise an exception.
RAISE EXCEPTION "Variable % changed. Transaction aborted.",an_integer;
RETURN 1;
END;
' LANGUAGE 'plpgsql';
```

Solve atleast TWO sets from the sets given below :

SET A
Using Bank Database
a) Write a stored function to print the total number of customers of a particular branch. (Accept branch name as input parameter.) In case the branch name is invalid, raise an exception for the same.
b) Write a stored function to increase the loan approved amount for all loans by 20%. In case the initial loan approved amount was less than Rs 10000, then print a notice to the user, before updating the amount .

SET B
Using Project-Employee database
a)Write a stored function to accept project name as input and print the names of employees working on the project. Also print the total number of employees working on that project. Raise an exception for an invalid project name.
b)Write a stored function to decrease the Hours_worked by 2 hours, for all projects in which employees from department no 2 is working. Raise an exception , in case the hours_worked becomes = 0 , after updation.

SET C
Using Bus transport Database
a)Write a stored function to print the names of drivers working on both shifts on ‘20/04/2014’.

b)Write a stored function to accept the bus_no and date and print its allotted drivers. Raise an exception in case of invalid bus number.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Assignment 7 : Triggers.

A trigger defines a function which occurs before or after , an action on a table. A trigger is implemented through PL/pgSQL, C or any other functional language that PostgreSQL can use to define a function..

A trigger is a PL/pgSQL block that is associated with a table, stored in a database and executed in response to a specific data manipulation event. Triggers can be executed or fired in response to the following events.

a. A row is inserted into table
b. A row in a table is updated.
c. A row in a table is deleted.

Syntax for defining a database trigger

```
Create  Trigger trigger–name
{ Before | After} {event [ or event …]} ON table–name
 for each { Row | statement}
execute procedure fucntionname ( arguments) ;
```

A trigger procedure is created with the CREATE FUNCTION command, declaring it as a function with no arguments and a return type of trigger.
Special variables created automatically, on call to a trigger function, are as follows :

Variable Name	Purpose and contents
NEW	Holds the new database row for INSERT/UPDATE operations in row-level triggers
OLD	Holds the old database row for UPDATE/DELETE operations in row-level triggers.
TG_NAME	Contains the name of the trigger actually fired.
TG_WHEN	Specifies the trigger timing. Contains a string of BEFORE or AFTER,
TG_LEVEL	Specifies the trigger type. Contains a string of either ROW or STATEMENT
TG_OP	Specifies the trigger operation. Contains a string of INSERT, UPDATE, or DELETE.
TG_RELID	Contains the object ID of the table that caused trigger invocation.
TG_TABLE_NAME	Contains the name of the table that caused the trigger invocation.
TG_TABLE_SCHEMA	Contains the name of the schema of the table that caused trigeer invocation
TG_NARGS	Number of arguments given to the trigger procedure
TG_ARGV[ ]	Contains the arguments for the trigger statement.

Practice Examples :

Example 1

This trigger ensures that any time a row is inserted or updated in the table, the current user name and time are stamped into the row. And it checks that an employee's name is given and that the salary is a positive value.

```
CREATE TABLE employee (
   ename text,
   esalary integer,
   last_date timestamp,
   last_user text
);


 CREATE FUNCTION emp_timestamp() RETURNS trigger AS ‘
   BEGIN
      -- Check that empname and salary are given
      IF NEW.ename IS NULL THEN
         RAISE EXCEPTION 'empname cannot be null';
      END IF;
      IF NEW.esalary IS NULL THEN
         RAISE EXCEPTION '% cannot have null salary', NEW.ename;
      END IF;
      IF NEW.esalary < 0 THEN
         RAISE EXCEPTION '% cannot have a negative salary', NEW.ename;
      END IF;
      -- Remember who changed the payroll when
      NEW.last_date := current_timestamp;
      NEW.last_user := current_user;
      RETURN NEW;
   END;
‘ LANGUAGE ‘plpgsql’;

CREATE TRIGGER emp_stamp BEFORE INSERT OR UPDATE ON employee
FOR EACH ROW EXECUTE PROCEDURE emp_stamp();
```

Solve atleast THREE sets from the sets given below :

SET A

Using Item_supplier Database

Item(itemno integer,Itemname varchar(20),quantity integer)
Supplier(SupplierNo,Supplier name,address,city)
Item_sup(item_no integer ,Supplier_no integer,Rate Money)

Item and supplier are related with many to many relationship .Rate is descriptive attribute.

a. Write a trigger before update on rate field, If the difference in the old rate and new rate to be entered is more than Rs 2000/ . Raise an exception and display the corresponding message
b. Write a trigger before insert or update on rate field, If the rate to be entered is zero then. Raise an exception and display the message “Zero rate not allowed”.

SET B :

Student –Marks database

Student (rollno integer,name varchar(30),address varchar(50),class varchar(10))
Subject(Scode varchar(10),subject name varchar(20))
student and subject are related with M-M relationship with attributes marks scored.

a) Write a trigger before deleting a student record from the student table. Raise a notice and display the message "student record is being deleted"
b) Write a trigger to ensure that the marks entered for a student, with respect to a subject is never < 10 and greater than 100.

SET C

News paper database

Newspaper(name varchar(20), language varchar(20),Publisher varchar(20),cost money)
Cities(pincode varchar(6), city varchar(20), state varchar(20))
Newspaper & Cities M to M relationship with descriptive attribute daily_required integer

a) Calculate the length of pincode. Write a trigger which will fire before insert on the cities table which check that the pincode must be of 6 digit. If it is more or less then it display the appropriate message.
b) Write a trigger which will prevent deleting cities from Maharatra state.

SET D

Using Railway Reservation Database

a) create a trigger to validate train arrival time must be less than train departure time.
b) Write a trigger which will be activated before changing the status field in the ticket table and print a message to the user.

SET E

Using Bus Transportation database

a) Define a trigger after insert or update the record of driver if the age is between 18 and 50 give the message "valid entry" otherwise give appropriate message.
b)Define a trigger after delete the record of bus having capacity < 10. Display the message accordingly

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Section II – Mini Project for Designing Backend using Software Engineering techniques

The Mini project is to be carried out using following steps and document them by filling the forms provided herewith

Step 1 – Form the team of two students

Step 2 – Identify the problem that involves data processing and finalize the project after discussing with your teacher guide. (Form 1)

Step 3 – Understand the problem, study the existing system and prepare a problem description and present drawbacks of existing system and scope of the proposed System(Form 2)

Step 4 – Carry out Feasibility study of proposed system (Form 3)

Step 5 – Identify users of the System. Gather Data requirements and Functional requirements. Prepare Data Flow Diagram (Form 4)

Step 6 – Identify Entities and their Attributes and draw Entity Relationship Diagrams(Form 5)

Step 7 - Design the Database (Form 6)

Step 8 – Designing queries related to functional requirements (Form 7)

Form No : 1 **Submission Deadline : / /**

S. Y. B. Sc. (Computer Science)

Mini Project

Academic Year (20 - 20)

Project Title:____

____

____

Team members:

1) Name : ____

Roll No . Exam Seat No:

2) Name : ____

Roll No . Exam Seat No:

Project Guide Name:____

Project Guide Signature: _______________

Start Date: / / Completion Date: / /

Form No : 2 **Submission Deadline : / /**

Assignment 8 : Problem definition and scope

Problem Description

Study of Existing system(Manual or computerised)

Drawbacks of Existing system

Scope of the Proposed System

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Submission Date : / / Signature of Instructor/Guide

Form No : 3 **Submission Deadline : / /**

Assignment 9: Feasibility Study

Technical Feasibility

Economical Feasibility

Operational Feasibility

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Submission Date : / / Signature of Instructor/Guide

Form No : 4 **Submission Deadline : / /**

Assignment 10 : Gathering Data Requirements and Functional Requirements

Data Requirements of the System

Identify End Users of the System

Input Data to the System

Output Information from the System

Functional or Processing Requirements of the System

Data Flow Diagrams

Context Level DFD

First level DFD

Explain all the main process in the DFD

1)Process Name: ____

Input data to process: ____

Output data from the process: ____

Data store required: ____

Description of the Process: ____

____

____

____

2) Process Name: ____

Input data to process: ____

Output data from the process: ____

Data store required: ____

Description of the Process: ____

____

____

____

3) Process Name: ____

Input data to process: ____

Output data from the process: ____

Data store required: ____

Description of the Process: ____

____

____

____

4) Process Name: ____

Input data to process: ____

Output data from the process: ____

Data store required: ____

Description of the Process: ____

____

____

____

5) Process Name: ____

Input data to process: ____

Output data from the process: ____

Data store required: ____

Description of the Process: ____

____

____

____

6) Process Name: ____

Input data to process: ____

Output data from the process: ____

Data store required: ____

Description of the Process: ____

____

____

____

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Submission Date : / / Signature of Instructor/Guide

Form No : 5 **Submission Deadline : / /**

Assignment 11 : Entity Relationship Modeling

Identify the entities and the attributes.

1) Entity name:______________________________

Attributes______________________________

____

2)Entity name:______________________________

Attributes______________________________

____

3) Entity name:______________________________

Attributes______________________________

____

4) Entity name:______________________________

Attributes______________________________

____

5)Entity name:______________________________

Attributes______________________________

____

6)Entity name:______________________________

Attributes______________________________

____

____

Entity Relationship diagram

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Submission Date : / / Signature of Instructor/Guide

Form No : 6 **Submission Deadline : / /**

Assignment 12: Designing the Normalized Database

Table1	Relationship	Table2
1)		
2)		
3)		
4)		
5)		
6)		
7)		
8)		
9)		
10)		
11)		
12)		
13)		
14)		
15)		

Table name:____

Field name	Field type	Description
1)		
2)		
3)		
4)		
5)		
6)		
7)		
8)		
9)		
10)		

Table name:____

Field name	Field type	Description
1)		
2)		
3)		
4)		
5)		
6)		
7)		
8)		
9)		
10)		

Table name:____

Field name	Field type	Description
1)		
2)		
3)		
4)		
5)		
6)		
7)		
8)		
9)		
10)		

Table name:____

Field name	Field type	Description
1)		
2)		
3)		
4)		
5)		
6)		
7)		

Table name:____

Field name	Field type	Description
1)		
2)		

3)__________________________	________________________	________________________
4)__________________________	________________________	________________________
5)__________________________	________________________	________________________
6)__________________________	________________________	________________________

Table name:____

Field name	Field type	Description
1) __________________________	________________________	________________________
2)__________________________	________________________	________________________
3)__________________________	________________________	________________________
4)__________________________	________________________	________________________
5)__________________________	________________________	________________________
6)__________________________	________________________	________________________

Table name:____

Field name	Field type	Description
1) __________________________	________________________	________________________
2)__________________________	________________________	________________________
3)__________________________	________________________	________________________
4)__________________________	________________________	________________________
5)__________________________	________________________	________________________
6)__________________________	________________________	________________________

In postgreSQL, create the database on the computer containing all the tables of your system along with the relationship and store the corresponding SQL statements in a file. .

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Submission Date : / / Signature of Instructor/Guide

Form No : 7 **Submission Deadline : / /**

Assignment 13 : Designing queries related to Functional requirements

Queries/ stored functions/ cursor/ triggers to carry out expected functional requirement of the System

1.

Functional Requirement Description

Queries/ stored functions/ cursor/ triggers

Implement the above queries on computer and save it in a file.

2.

Functional Requirement Description

Queries/ stored functions/ cursor/ triggers

Implement the above queries on computer and save it in a file.

3.

Functional Requirement Description

Queries/ stored functions/ cursor/ triggers

Implement the above queries on computer and save it in a file.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Submission Date : / / Signature of Instructor/Guide

Bibliography

1. “Practical PostgreSQL”, O’Reilly Publications

2. www.postgresql.org/docs/7.0/interactive/tutorial.html. (PostgreSQL Manual)

3. www.postgresql.org/docs/8.0/interactive/tutorial.html. (PostgreSQL Manual)

4. “Software Engineering”, by Roger Pressman, 7th Edition

5. Relational Database Management System(MySQL) Lab Book – By Vaishali Bhoite, Shilpa dange, Sampada Vidhvans, Jyoti Yadav, Kalpana Joshi