**DBS HAND IN 3**

**Exercise 1**

1.

CREATE VIEW view1 AS SELECT \* FROM works\_on;

SELECT \* FROM view1;

2.

CREATE VIEW vie2(PROJ\_Name, PROJ\_No, PROJ\_Location, DEPART\_No, hours)

AS SELECT p.pname, p.pnumber, p.plocation, p.dnum, SUM(wo.hours)

FROM Project p, Works\_on wo

WHERE p.pnumber = wo.pno

GROUP BY p.pnumber, p.pname

ORDER BY p.pnumber;

SELECT \* FROM view2;

3.

CREATE VIEW view3(EMP\_No, EMP\_Name, PROJ\_No, PROJ\_Name, Hours, Cost)

AS SELECT e.ssn, e.fname || ' ' || e.lname, p.pnumber, p.pname, wo.hours, wo.hours \* 300

FROM Employee e, Works\_on wo, Project p

WHERE e.ssn = wo.essn AND wo.pno = p.pnumber;

SELECT \* FROM view3;

4.

CREATE VIEW view4(DEPART\_Name, MANAGER\_Name, MANAGER\_Salary)

AS SELECT d.dname, e.fname || ' ' || e.lname, e.salary

FROM Department d, Employee e

WHERE d.mgrssn = e.ssn;

SELECT \* FROM view4;

5.

CREATE VIEW view5(EMP\_Name, SUPERVISOR\_Name, EMP\_Salary)

AS SELECT e1.fname || ' ' || e1.lname, e2.fname || ' ' || e2.lname, e1.salary

FROM Employee e1, Employee e2, Department d

WHERE e1.dno = d.dnumber AND d.dname = 'Research' AND e1.superssn = e2.ssn;

SELECT \* FROM view5;

6.

CREATE VIEW view6(PROJ\_Name, DEPART\_Name, No\_of\_EMP, Hours\_per\_week)

AS SELECT p.pname, d.dname, COUNT(\*), SUM(wo.hours)

FROM Project p, Department d, Works\_on wo

WHERE p.dnum = d.dnumber AND p.pnumber = wo.pno

GROUP BY p.pname, d.dname;

SELECT \* FROM view6;

7.

CREATE VIEW view7(PROJ\_Name, DEPART\_Name, No\_of\_EMP, Hours\_per\_week)

AS SELECT \*

FROM view6 v6

WHERE v6.No\_of\_EMP > 1;

SELECT \* FROM view7;

8.

CREATE VIEW view8(EMP\_Name)

AS SELECT e1.fname || ' ' || e1.lname

FROM Employee e1, Employee e2, Employee e3

WHERE e1.superssn = e2.ssn AND e2.superssn = e3.ssn AND e3.ssn = '888665555';

SELECT \* FROM view8;

9.

CREATE VIEW view9(DEPART\_Name, No\_of\_EMP)

AS SELECT d.dname, COUNT(e.dno)

FROM Department d, Employee e

WHERE e.dno = d.dnumber

GROUP BY d.dname

HAVING AVG(e.salary) > 30000;

SELECT \* FROM view9;

10.

Create a view which contains the project name, project number, department name, department location for the projects with an average of worked hours per employee bigger than 20

CREATE VIEW view10(PROJ\_Name, PROJ\_No, DEPART\_Name)

AS SELECT p.pname, p.pnumber, d.dname

FROM Project p, Department d, Works\_on wo

WHERE p.dnum = d.dnumber AND p.pnumber = wo.pno

GROUP BY p.pname, p.pnumber, d.dname

HAVING AVG(wo.hours) > 20;

SELECT \* FROM view10;

**Exercise 2**

1.

CREATE TABLE Log\_works\_on(

ESSN CHAR (9) NOT NULL,

PNO INTEGER NOT NULL,

HOURS\_now INTEGER,

HOURS\_before INTEGER,

DAY\_TIME TIMESTAMP);

CREATE FUNCTION Log\_for\_works\_on() RETURNS TRIGGER AS $BODY$

BEGIN

IF(tg\_op = 'INSERT') THEN INSERT INTO Log\_works\_on(ESSN, PNO, HOURS\_now, DAY\_TIME)

VALUES(NEW.ESSN, NEW.PNO, NEW.HOURS, NOW());

RETURN NEW;

END IF;

IF(tg\_op = 'UPDATE') THEN INSERT INTO Log\_works\_on(ESSN, PNO, HOURS\_now, DAY\_TIME)

VALUES(NEW.ESSN, NEW.PNO, NEW.HOURS, OLD.HOURS, NOW());

RETURN NEW;

END IF;

IF(tg\_op = 'DELETE') THEN INSERT INTO Log\_works\_on(ESSN, HOURS\_before, DAY\_TIME)

VALUES(OLD.ESSN, OLD.HOURS, NOW());

RETURN NEW;

END IF;

RETURN NULL;

END;

$BODY$ LANGUAGE plpgsql;

CREATE TRIGGER Log\_insert BEFORE INSERT ON Works\_on FOR EACH ROW

EXECUTE PROCEDURE Log\_for\_works\_on();

CREATE TRIGGER Log\_update BEFORE UPDATE ON Works\_on FOR EACH ROW

EXECUTE PROCEDURE Log\_for\_works\_on();

CREATE TRIGGER Log\_delete AFTER DELETE ON Works\_on FOR EACH ROW

EXECUTE PROCEDURE Log\_for\_works\_on();

2.

CREATE FUNCTION prevent\_insert() RETURNS TRIGGER AS $func$

DECLARE pcount integer;

BEGIN

SELECT COUNT(\*) into pcount

FROM Project

WHERE dnum = new.dnum;

IF pcount >= 3 THEN

RAISE EXCEPTION 'You cannot add more than 3 projects for a department';

END IF;

RETURN NEW;

END;

$func$ LANGUAGE plpgsql;

CREATE TRIGGER trigger2 BEFORE INSERT ON Project

FOR EACH ROW

EXECUTE PROCEDURE prevent\_insert();

3.

CREATE FUNCTION prevent\_insert\_wo() RETURNS TRIGGER AS $func$

DECLARE pnocount integer;

BEGIN

SELECT COUNT(\*) into pnocount

FROM Works\_on

WHERE essn = new.essn;

IF pnocount >= 4 THEN

RAISE EXCEPTION 'You cannot add more than 4 projects for an employee';

END IF;

RETURN NEW;

END;

$func$ LANGUAGE plpgsql;

CREATE TRIGGER trigger3 BEFORE INSERT ON Works\_on

FOR EACH ROW

EXECUTE PROCEDURE prevent\_insert\_wo();

4.

CREATE TABLE Log\_department(

DNAME VARCHAR (20) UNIQUE,

DNUMBER INTEGER NOT NULL,

MGRSSN CHAR (9),

MGRSTARTDATE DATE,

DAY\_TIME TIMESTAMP);

CREATE FUNCTION Log\_for\_department() RETURNS TRIGGER AS $BODY$

BEGIN

IF(tg\_op = 'INSERT') THEN INSERT INTO Log\_department(DNAME, DNUMBER, MGRSSN\_now, MGRSTARTDATE\_now, DAY\_TIME)

VALUES(NEW.DNAME, NEW.DNUMBER, NEW.MGRSSN, NEW.MGRSTARTDATE, NOW());

RETURN NEW;

END IF;

IF(tg\_op = 'UPDATE') THEN INSERT INTO Log\_department(DNAME, DNUMBER, MGRSSN\_now, MGRSTARTDATE\_now, DAY\_TIME)

VALUES(NEW.DNAME, NEW.DNUMBER, NEW.MGRSSN, OLD.MGRSSN, NEW.MGRSTARTDATE, OLD.MGRSTARTDATE, NOW());

RETURN NEW;

END IF;

IF(tg\_op = 'DELETE') THEN INSERT INTO Log\_department(DNAME, DNUMBER, MGRSSN\_before, MGRSTARTDATE\_before, DAY\_TIME)

VALUES(OLD.DNAME, OLD.DNUMBER, OLD.MGRSSN, OLD.MGRSTARTDATE, NOW());

RETURN NEW;

END IF;

RETURN NULL;

END;

$BODY$ LANGUAGE plpgsql;

CREATE TRIGGER Log\_insert BEFORE INSERT ON Department FOR EACH ROW

EXECUTE PROCEDURE Log\_for\_department();

CREATE TRIGGER Log\_update BEFORE UPDATE ON Department FOR EACH ROW

EXECUTE PROCEDURE Log\_for\_department);

CREATE TRIGGER Log\_delete AFTER DELETE ON Department FOR EACH ROW

EXECUTE PROCEDURE Log\_for\_department();

5.

CREATE FUNCTION prevent\_insert\_emp() RETURNS TRIGGER AS $func$

BEGIN

IF new.salary <10000 THEN

RAISE EXCEPTION 'You cannot add an employee having a salary less than 10000';

END IF;

RETURN NEW;

END;

$func$ LANGUAGE plpgsql;

CREATE TRIGGER trigger5 BEFORE INSERT ON Employee

FOR EACH ROW

EXECUTE PROCEDURE prevent\_insert\_emp();

**Exercise 3**

1.

**public** **static** **void** createTable() **throws** Exception

{

**try**

{

Connection conn = *getConnection*();

PreparedStatement create = conn.prepareStatement(

"CREATE TABLE IF NOT EXISTS Book(id INTEGER NOT NULL, name VARCHAR(50), author VARCHAR(50), PRIMARY KEY(id));");

create.executeUpdate();

}

**catch**(Exception e)

{

System.***out***.println(e);

}

**finally**

{

System.***out***.println("Function complete.");

}

}

2.

**public** **static** **void** insertIntoTable() **throws** Exception

{

**try**

{

Connection conn = *getConnection*();

PreparedStatement insert = conn.prepareStatement("INSERT INTO Book(id, name, author) VALUES (1,'Ulysses', 'James Joyce),"

+ "(2, 'Pride and Prejudice', 'Jane Austen'),"

+ "(3, 'The Immortals', null);");

insert.executeUpdate();

}

**catch**(Exception e)

{

System.***out***.println(e);

}

**finally**

{

System.***out***.println("Insert completed");

}

}

3.

**public** **static** **void** updateTable() **throws** Exception

{

**try**

{

Connection conn = *getConnection*();

PreparedStatement update = conn.prepareStatement("UPDATE Book SET author = 'Tamora Pierce' WHERE id = 3;");

update.executeUpdate();

}

**catch**(Exception e)

{

System.***out***.println(e);

}

**finally**

{

System.***out***.println("Update completed");

}

}

4.

**public** **static** **void** deleteContent() **throws** Exception

{

**try**

{

Connection conn = *getConnection*();

PreparedStatement delete = conn.prepareStatement("TRUNCATE Book;");

delete.executeUpdate();

}

**catch**(Exception e)

{

System.***out***.println(e);

}

**finally**

{

System.***out***.println("Delete completed");

}

}

5.

**public** **static** **void** dropTable() **throws** Exception

{

**try**

{

Connection conn = *getConnection*();

PreparedStatement drop = conn.prepareStatement("DROP TABLE Book;");

drop.executeUpdate();

}

**catch**(Exception e)

{

System.***out***.println(e);

}

**finally**

{

System.***out***.println("Delete completed");

}

}

6.

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**public** **class** Main {

**public** **static** **void** main(String[] args) **throws** Exception

{

*createTable*();

*insertIntoTable*();

*updateTable*();

*deleteContent*();

*dropTable*();

}

**public** **static** **void** createTable() **throws** Exception {…}

**public** **static** **void** insertIntoTable() **throws** Exception {…}

**public** **static** **void** updateTable() **throws** Exception {…}

**public** **static** **void** deleteTable() **throws** Exception {…}

**public** **static** **void** dropTable() **throws** Exception {…}

**public** **static** **Connection** getConnection() **throws** Exception

{

**try**

{

String driver = "org.postgresql.Driver";

String url = "jdbc:postgresql://localhost:5432/postgres";

String username = "username";

String password = "password";

Class.*forName*(driver);

Connection con = DriverManager.*getConnection*(url, username, password);

System.***out***.println("Connected");

**return** con;

}

**catch**(Exception e)

{

System.***out***.println(e);

}

**return** **null**;

}

}