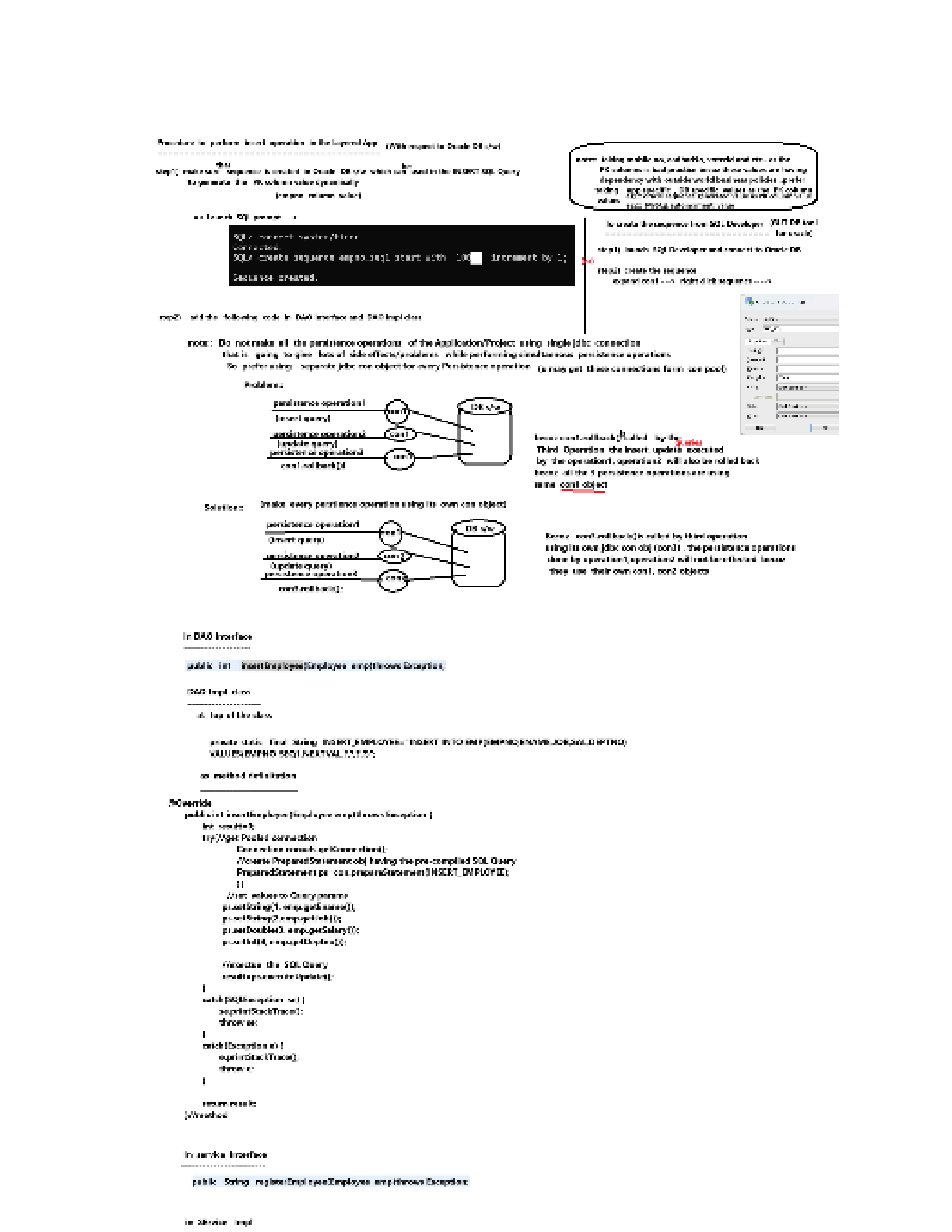

Procedure to perform insert operation in the Layered App (With respect to Oracle DB s/w)

=====

that

be

step1) make sure sequence is created in Oracle DB s/w which can used in the INSERT SQL Query to generate the PK column value dynamically

(empno column value)

=> Launch SQL prompt --->

```
SQL> connect system/tiger
Connected.
SQL> create sequence empno_seq1 start with
```

note:: taking mobile no, aadharNo, voterId and etc.. as the PK columns is bad practice becoz these values are having dependency with outside world business policies ..prefer taking app specific, DB specific values as the PK column eg1:: oracle sequence generated value as PK column value values eg2:: MySQL autoincrment value

```
100
increment by 1;
```

(pr)

==========

========

To create the sequence from SQL Developer (GUT DB tool for oracle) step1) launch SQL Developer and connect to Oracle DB step2) create the sequence

```
Sequence created.
```

expand con1 ---> right click sequence ---->

Create Sequence

step2) add the following code in DAO Interface and DAO Impl class

note:: Do not make all the persistence operations of the Application/Project using single jdbc connection that is going to give lots of side effects/problems while performing simultaenous persistence operations So prefer using separate jdbc con object for every Persistence operation (u may get these connections form con pool)

Problem::

Schema: SYSTEM Name: ENO_SEQ

Properties DDL

Start With:

1

Increment:

1

Min Value:

1

Solution::

In DAO Interface

persistence operation1

DB s/w

con 1

(insert query)

persistence operation2

con1

becoz con 1.rollback(called by the

(update query)

persistence operation3

con 1

con1.rollback()l

(make every perstience operation using its own con object)

persistence operation1

fcon 1)

(insert query)

persistence operation2

con 2

(update query)

persistence operation3

con3

con3.rollback();

DB s/w

Max Value:

10000

Cache:

<Not Specified>

Cache Size:

Cycle:

<Not Specified>

Order:

<Not Specified>

Help

queries Third Operation the insert, update executed by the operation1, operation2 will also be rolled back becoz all the 3 persistence operations are using same con1 object

Becoz con3.rollback()is called by third operation

using its own jdbc con obj (con3), the persistence operations done by operation1,operation2 will not be effected becoz they use their own con1, con2 objects

public int insertEmployee (Employee emp) throws Exception;

DAO Impl class

at top of the class

private static final String INSERT_EMPLOYEE="INSERT INTO EMP(EMPNO,ENAME,JOB,SAL, DEPTNO) VALUES(EMPNO_SEQ1.NEXTVAL,?,?,?,?)";

as method definitation

@Override

public int insertEmployee(Employee emp) throws Exception {

int result=0;

try(//get Pooled connection

}

Connection con=ds.getConnection();

//create PreparedStatement obj having the pre-compiled SQL Query PreparedStatement ps=con.prepareStatement(INSERT_EMPLOYEE);

X{

//set values to Query params ps.setString(1, emp.getEname());

ps.setString(2,emp.getJob());

ps.setDouble(3, emp.getSalary()); ps.setInt(4, emp.getDeptno());

//exectue the SQL Query

result=ps.executeUpdate();

catch (SQLException se) {

}

se.printStackTrace();

throw se;

catch (Exception e) {

e.printStackTrace();

throw e;

}

return result;

}//method

In service Interface

public String registerEmployee(Employee emp) throws Exception;

in Service Impl

@Override

public String register Employee (Employee emp)throws Exception { //use DAO

```
int result=empDAO.insertEmployee(emp);
return result==0?"Employee not registred":"Employee is registered";
Controller class
public String processEmployee(Employee emp) throws Exception{ //use service
String resultMsg=empService.registerEmployee(emp); return resultMsg;
}
Client App's main(-) method
//get IOC container
ApplicationContext ctx-SpringApplication.run(BootlocProj03 LayeredAppMiniProjectApplication.class, args);
//get Controller class obj ref
EmployeeOperationsController controller=ctx.getBean("empController",
EmployeeOperationsController.class);
//read input from enduser
Scanner sc=new Scanner(System.in);
System.out.println("Enter name::");
String name=sc.next();
System.out.println("Enter desg::");
String desg=sc.next();
System.out.println("Enter salary::");
double salary=sc.nextDouble();
System.out.println("Enter deptno (10,20,30,40,...)");
int deptno=sc.nextInt();
//create Employee class object
Employee emp=new Employee();
emp.setEname(name);emp.setJob(desg);emp.setDeptno(deptno); emp.setSalary(salary);
try {
}
String resultMsg=controller.processEmployee(emp);
System.out.println(resultMsg);
catch (Exception e) {
}
e.printStackTrace();
((ConfigurableApplicationContext) ctx).close();
step4) Run the Client app
MGR HIREDATE
SAL
COMM
DEPTNO
```

CSV_STATUS

100 raja

CLERK

8000

10

101 karan

hyd

5000

20

How best we can design Primary Key column of the DB table

=====:

======

candidate key column

=>The column of db table that holds unique values and can be used

to identify the whole record is called candidate key column

eg:: aadhar no, voterid, pan no, driving license number, passport number, mobile no and etc.

Natural key column

=> The candidate key column whose values are inter linked with outside world

business policies and expected from the end users is called natural key column

=> This column values can not be generate the underlying DB s/w or underlying Application.. must be collected from

End users of the App

Limitations of taking Natural key column as PK column

==========

==========

=> these values are quite lengthy values, So they need more memory towards insertion of the record

are

=> These values interlinked with outside world business policies, so any

change in business or govt policies they may effect main db tables,dependent

db tables and their relevant java code. (java classes)

=> Expected from endusers, if enduser fail to supply these values then the record

insertion fails (The Business operation will be disturbed)

Surrogate key column

====

=>The candidate key column whose value is generated by underlying DB s/w

or underlying software App dynamically at run time is called Surrogate key

column

=> These values are not expected from endusers and not linked with outside world business and govt

policies

note:: Candidate key column, natural key column and surrogate key column and etc.. are logical keys (These are not physical keys (constraints) like PK,FK, UK,NNK and etc..

eg1:: Oracle sequence genernated value as PK col values

eg2:: MySQL Autoincrement constraint generated values

eg3:: Hibenrate /spring data jpa generators generated values

PK :: Primary Key

FK:: Foreign key

UK :: Unique key NNK: Not Null Key

Advantages of taking Surrogate key column as Primary key column

==========

==========

=>These values allocate less memory

==========

=> These values are not linked with outside world business policies or govt policies, So they will not

java

be changed and does not db tables data and relevant source code

=> Not Expected from end users, if enduser fails to give one or two other columns then also we can complete record insertion process

eg:: oracle sequence generated values for PK column

MySQL auto increment generated values for PK colum

Hibernate generator values (sequence, hilo, seqhilo, native and etc..)

JPA Generator values (AUTO, SEQUENCE, IDENTITY, UUID, TABLE,...)

usecases of taking surrogate key column as PK Colum

==========

=> Generating Transaction Id dynamically by Bank App

=> Generating registration number for student dynamically

=> Generating registration number for employee dynamically

GUI DB Tools

==========

=> As developers are not so comfortable with SQL Queries, they prefer using separate GUI Db table for DB Operations which can be operated using GUI Screens

=> eg:: SQL Developer for Oracle (separate installation)

eg:: MySQL Workbench for MySQL (Built-in with MySQL DB s/w)

eg:: SQL Yog For MYSQL (seperate installation)

eg:: TOAD for oracle (separate installation)

eg::

TOAD for MySQL (separate installation)

In Lombok API

==========

@NoArgrConstructor gives Zero Param constructor @AllArgsConstructor gives parameterized constructor by involving all the properties/member variables @RequiredArgsConstructor gives parameterized constructor only by involving the properties/member variables that are annotated with @NonNull

OK