DBMS Assignment 3

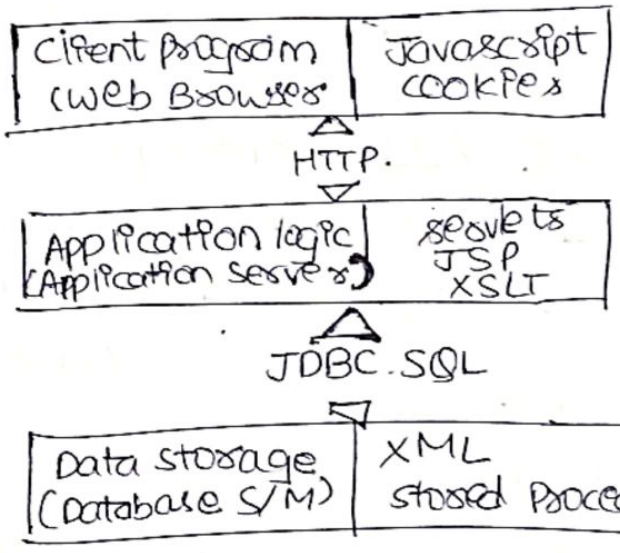
Vivek Raj. V.
1DB18IS098
5th sem 'B' sec

1) Explain 3 tier architecture with neat diagram.

A: The thin-client 2 tier architecture essentially separates presentation issues from the rest of the application. The 3 tier architecture goes one step further & also separates application logic from data management:

- Presentation tier
- Middle tier
- Data Management Tier

Different technologies have been developed to enable distribution of the three tiers of an application across multiple h/w platforms & different physical sites.



Technologies for the 3 tiers

2) Define stored procedure. Explain creating & calling of stored procedure with an example.

A: Stored procedure:

It is a set of logical group of SQL statements which are grouped to perform a specific tasks.

Creating a simple procedure:

Consider the following schema:

Student (usn: string, sname: string)

Let us now write a stored procedure to retrieve the count of students with sname 'Akshay'.

```
create or replace procedure ss
is
stu_cnt int;
```

```
begin
  select count(*) into stu_cnt from students where sname
                                        = 'AKSHAY';
dbms_output.put_line("the count of student is: "|| stu_cnt);
end ss;
```

Stored procedures can also have parameters. These parameters have to be valid SQL types & have one of 3 different modes: IN, OUT or INOUT.

- IN parameters are arguments to the stored procedure.
- OUT parameters are returned from the stored procedure; it assigns values to all OUT parameters that the user can process.
- INOUT parameters combine the properties of IN & OUT parameters: They contain values to be passed to the stored procedures & the stored procedure can set their values as return values

e.x: CREATE PROCEDURE AddInventory (
IN book_isbn CHAR(10), IN addedQty INTEGER)
UPDATE BOOKS SET qty_in_stock = qty in_stock + addedQty
WHERE BookIsbn = isbn

Calling stored procedures:

Stored procedures can be called in interactive SQL with the CALL statement:

CALL storedProcedureName (arg1, arg2, ..argN);

3) Define normal form. Explain 1NF, 2NF & 3NF with suitable example.

A: Normal form:

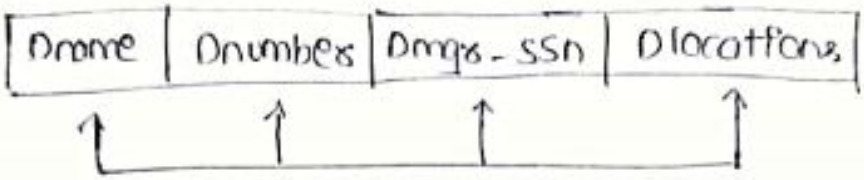
It is of a relation refers to the highest normal form condition that it meets & hence indicates the degree to which it has normalized.

First Normal form:

- Defined to disallow multivalued attributes, composite attributes & their combinations.
- It states that the domain of an attribute must include only atomic values & that the value of any attribute in a

tuple must be a single value from the domain of that attribute.
- 1NF disallows relations within relations or relations as attribute values within tuples.
- The only attribute values permitted by 1NF are single atomic values.
- Consider the DEPARTMENT relation schema shown in figure below.

(i) DEPARTMENT



- Primary key is Dnumber
- we assume that each department can have a no; of locat
- The DEPARTMENT Schema & a sample relation state are shown in figure below:

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	(Bellaire, Sugarland, Houston
Administration	4	987654321	(Stamford
Headquarters	1	888665555	(Houston)

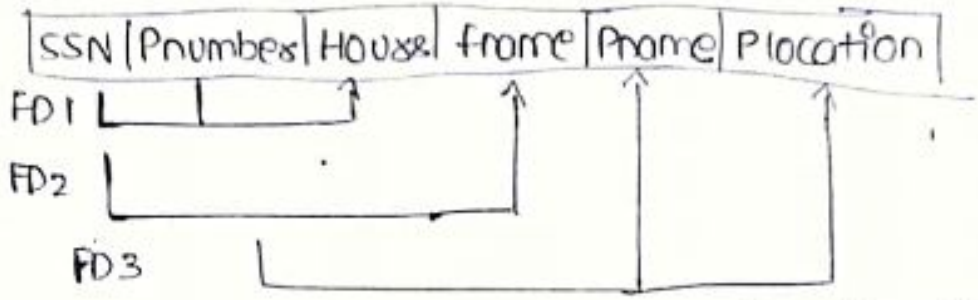
- As we can see, this is not in 1NF because Dlocations is not an atomic attribute as illustrated by the first tuple in figure.
- There are 2 ways we can look at the Dlocations attribute:
→ The domain of Dlocation contains atomic values, but some tuples can have a set of these values. In this case, Dlocations is not functionally dependent on the primary key Dnumber.
→ The domain of Dlocations contains set of values & hence is nonatomic. In this case, Dnumber → Dlocations because each set is considered a single member of the attribute domain
⇒ In either case, the DEPARTMENT relation is not in 1

Second Normal form:

- Second Normal form (2NF) is based on the concept of full functional dependency.

- A functional dependency $X \rightarrow Y$ is a full functional dependency if removal of any attribute A from X means that the dependency does not hold any more, that is; for any attribute $A \in X$, $(X - \{A\})$ does not functionally determine Y
- A functional dependency $X \rightarrow Y$ is a partial dependency if some attribute $A \in X$ can be removed from X & the dependency still holds; that is; for some $A \in X$, $(X - \{A\}) \rightarrow Y$

EMP_PROJ



- In the above figure, {SSN, Pnumber} → Hours is a full dependency (neither SSN → Hours nor Pnumber → Hours hold)
- {SSN, Pnumber} → Fname is partial because SSN → Ename holds.
- Definition: A relation schema R is in 2NF if every nonprime attribute A in R is fully functionally dependent on the primary key of R.

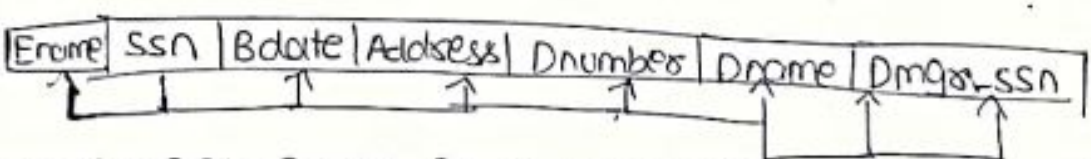
Third Normal form:

- Transitive functional dependency

A fun^c dependency $X \rightarrow Y$ in a relation schema R is a transitive dependency if there exists a set of attribute Z that are neither a primary nor a subset of any key of R (candidate key) & both $X \rightarrow Y$ & $Y \rightarrow Z$ holds.

- Example:

EMP_DEPT



→ SSN - DMGRSSN is a transitive FD since SSN → DNUMBER & DNUMBER → DMGRSSN hold.
→ Dnumber is neither a key itself nor a subset of the key of EMP_DEPT.
→ SSN - ENAME is non-transitive since there is no set of attribute X where SSN → X & X → ENAME.

Definition: A relation schema is in third normal form (3NF) if it is 2NF and no non prime attribute A is transitively dependent on the primary key

4) Discuss insertion, deletion & modification anomalies. Why are they considered bad? Illustrate with example.

A Insertion Anomalies.

It can be differentiated into 2 types, illustrated by the following examples based on EMP_DEPT relation:

1) To insert a new employee tuple into EMP_DEPT, we must include either the attribute values for the dept that the employee works for NULL

2) It is difficult to insert a new dept that has no employees as yet in the EMP_DEPT relation. The only way to do this is to place NULL values in the attributes, for employee

Deletion Anomalies:

- The problem of deletion anomalies is related to the second insertion anomaly situation just discussed.
- If we delete from EMP_DEPT an employee tuple that happens to represent the last employee working for a particular dept, the info concerning that dept is lost from the data.
- This problem does occur in the database because DEPT tuples are stored separately.

Modification Anomalies:

- In EMP_DEPT, if we change the value of one of the attributes of a particular department - say, the manager of department 5 - we must update the tuples of all employees who work in that dept; otherwise the database will become inconsistent.
- If we fail to update some tuples, the same dept will be shown to have 2 different values for manager in different employee tuples which would be wrong..

They are considered bad because they lead to:

- It is difficult to maintain consistency of data in database
- It leads to redundant data.
- It causes unnecessary updates of data.
- Memory space will be wasted at the stored level.

5) Explain 4 informal guidelines that may be used as measure to determine the quality of relation schema diagram.

A: 4 informal guidelines:

- Making sure that the semantics of attributes is clear in the schema.
- Reducing the redudant information in tuples.
- Reducing the NULL values in tuples.
- Disallowing the possibility of generating spurious tuples.

Guideline 1:

- Design a relation schema so that it is easy to explain its meaning.
- Do not combine attributes from multiple entity types & relationship types into a single relation.
- If a relation schema corresponds to one entity type or one relationship for it is straightforward to interpret & to explain its meaning.
- If the relation corresponds to a mixture of multiple entities & relationships, semantic ambiguities will result & the relation can't be easily explained.

Guideline 2:

- Design the base relation schemas so that no insertion deletion or modification anomalies are present in the relation.
- If any anomalies are present note them clearly & make sure that the programs that update the database will operate correctly.
- The second guideline is consistent with & in a way a restatement of the first guideline.
- These guidelines may sometimes have to be violated in order to improve the performance of certain queries.

Guideline 3:

- As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL.

- If NULLs are unavoidable make sure that they apply in exceptional cases only & do not apply to a majority of tuples in the relation.
- Using a space efficiently & avoiding joins with NULL values are the 2 overriding criteria that determine whether to include the columns that may have NULLs in a relation or to have a separate relation for those columns with the appropriate key columns.

Guideline 4:
Design relation schemas so that they can be joined with equality conditions on attributes that are appropriately related pairs in a way that guarantees that no spurious tables are generated.
- Avoid relations that contain matching attributes that are not combinations because joining on such attributes may produce spurious tuples.

6) Write an algorithm for finding a minimal cover 'F' for a set of functional dependencies 'E'. find the minimal cover for the giving set of FD's G: {A - BCDE, CDF}.
A: Input: A set of functional dependencies E
1) set F = E
2) Replace each functional dependency $X \to \{A1, A2, \ldots An\}$ in f by the n functional dependencies $X \to A1$, $X \to A2$, ... $X \to An$
3) For each functional dependency $X \to A$ in f for each attribute B that is an element of X if $\{f - \{X \to A\}\} \cup \{(X - \{B\}) \to A\}$ is equivalent to F then replace $X \to A$ with $(X - \{B\}) \to A$ in f.
4) For each remaining functional dependencies $X \to A$, in F if $\{f - \{X \to A\}\}$ is equivalent to f, then remove $X \to A$ from F.

7) Consider the following schema of order database.
SALESMAN (SalesmanId, Name, City, commission)
CUSTOMER (CustId, CustName, City, Grade, SalesmanId)
ORDERS (Ord No, PurchaseAmt, Ord Date, Cust Id, SalesmanId)

i) Find the name & no's of all salesman who has more than one customes

ii) Count the customes with grade above Bangaloxe's average.

iii) List all the salesman details whose first name is 'John'.

A: Table creation & schema diagram

Salesman

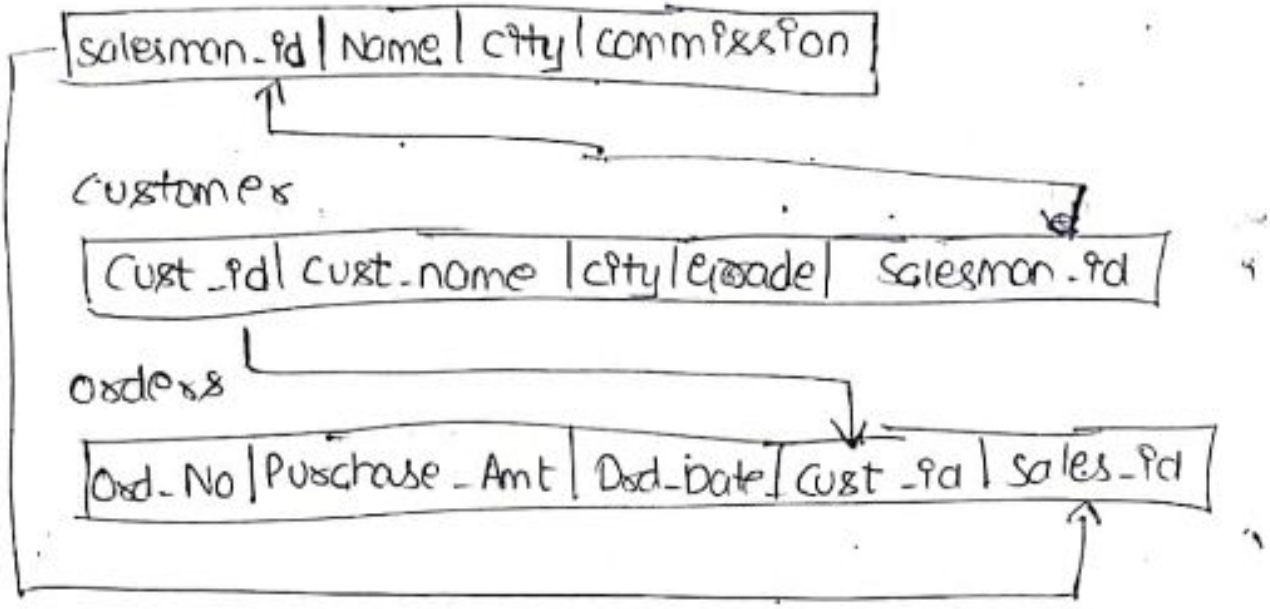


Table creation:

```
CREATE TABLE salesman (salesman_id numbes(4), name
varchar(20), city varchar2(20), commission varchar2
(20), primary key (salesman_id));
create table customes1 (cust_id numbes(4), cust_name
varchar 2(20), city varchar2(20), grade numbes(3),"
primary key (cust_id), salesman_id references
salesman (salesman_id) on delete set NULL);
create Table orders (ord_no numbes(5), purchase_amt
numbes(10,2), ord_date date, primary key (ord_no),
cust_id references customes1 (customes_id) on delete
cascade, salesman_id references salesman (salesman
_id) on delete cascade);
```

Table descriptions:

```
DESC SALESMAN;
SQL > DESC SALESMAN;
```

Name	Null?	Type
SALESMAN_ID	NOT NULL	NUMBER(4)
NAME		VARCHAR2(15)
CITY		VARCHAR2(15)
COMMISSION		NUMBER(3,2)

DESC CUSTOMER1;
SQL > DESC CUSTOMER1;

Name	Null?	Type
CUSTOMER_ID	NOT NULL	NUMBER(4)
CUST_NAME		VARCHAR2(15)
CITY		VARCHAR2(15)
GRADE		NUMBER(3)
SALESMAN_ID		NUMBER(4)

DESC ORDERS;
SQL > DESC ORDERS;

Name	Null?	Type
ORD_NO	Not Null	NUMBER(5)
PURCHASE_AMT		NUMBER(10,2)
ORD_DATE		DATE
CUSTOMER_ID		NUMBER(4)
SALESMAN_ID		NUMBER(4)

Insertion of values to tables

```
INSERT INTO SALESMAN VALUES(1000,'JOHN','BLR','25%');
INSERT INTO SALESMAN VALUES(2000,'RAVI','BLR','20%');
INSERT INTO SALESMAN VALUES(3000,'KUMAR','MYSURU','15%');
INSERT INTO SALESMAN VALUES(4000,'SMITH','DELHI','30%');
INSERT INTO SALESMAN VALUES(5000,'HARSHA','DLH','15%');
```

```
Insert into cust1 values(10,'Preethi','Blr',100,1000);
Insert into cust1 values(11,'Vikas','Mlr',300,1000);
```

```
Insert into cust values(12 'Bob' ,chennai, 400, 2000);
Insert into cust values(13, 'Chet', 'BLR', 200, 2000);
Insert into cust values(14, 'Mindy' 'BLR', 400, 3000);

Insert into orders values (50, 5000, '04-May-17', 10,1000);
Insert into orders values (51,450, '20-Jan-17', 10, 2000);
Insert into orders values (52,1000, '24-Feb-17', 13, 2000);
Insert into orders values (53, 3500, '13-Apr-17', 14, 3000);
Insert into orders values (54, 550, '9 May-17', 12, 2000);
```

```
SELECT * from SALESMAN;
```

SALESMAN-ID	NAME	CITY	COMMISSION.
1000	John	BLR	25%
2000	Ravi	BLR	20%
3000	Kumar	MYSURU	15%
4000	Smith	DELHI	30%
5000	Harsha	HYD	15%

cust_id	cust_name	city	grade	salesman_id
10	Preethi	BLR	100	1000
11	Vikas	MLR	300	1000
12	Bob	Chennai	400	2000
13	chet	BLR	200	2000
14	Mindy	BLR	400	30000

```
Select * from orders
```

ORD_NO	PURCHASE_AMT	ORD_DATE	CUST_ID	SALESMAN_ID
50	5000	4-May-17	10	1000
51	450	20-Jan-17	10	2000
52	1000	24-Feb-17	13	2000
53	3500	13-Apr-17	14	3000
54	550	9-May-17	12	2000

Queries:

i) SELECT salesman.pid, Name from salesman A where 1 < (SELECT COUNT(*) from CUST1 where salesman.pid = A.salesman.pid);

Salesman_pid	Name
1000	John
2000	Ravi

ii) select Grade, count(distinct cust_pid) from cust1 group by grade having grade > (select Avg(grade) from cust1 where city = 'BLR');

GRADE	COUNT(DISTINCT CUST_ID)
300	1
400	2

iii) select salesman.salesman_id, name, cust_name, commission from salesman, customer1 where salesman.city = customer1.city union select salesman_id, name, 'NO MATCH', commission from salesman where not city = Any (select CITY FROM customer1) order by 2 DESC;

Salesman_id	Name	CUST_NAME	COMMISSION
4000	Smith	NO MATCH	30%
2000	Ravi	Chethan	20%
2000	Ravi	Namratha	20%
2000	Ravi	preethi	20%
3000	Kumar	No match	15%
1000	John	Chethan	25%
1000	John	Namatha	25%
1000	John	Preethi	25%
5000	Harsha	No MATCH	15%.