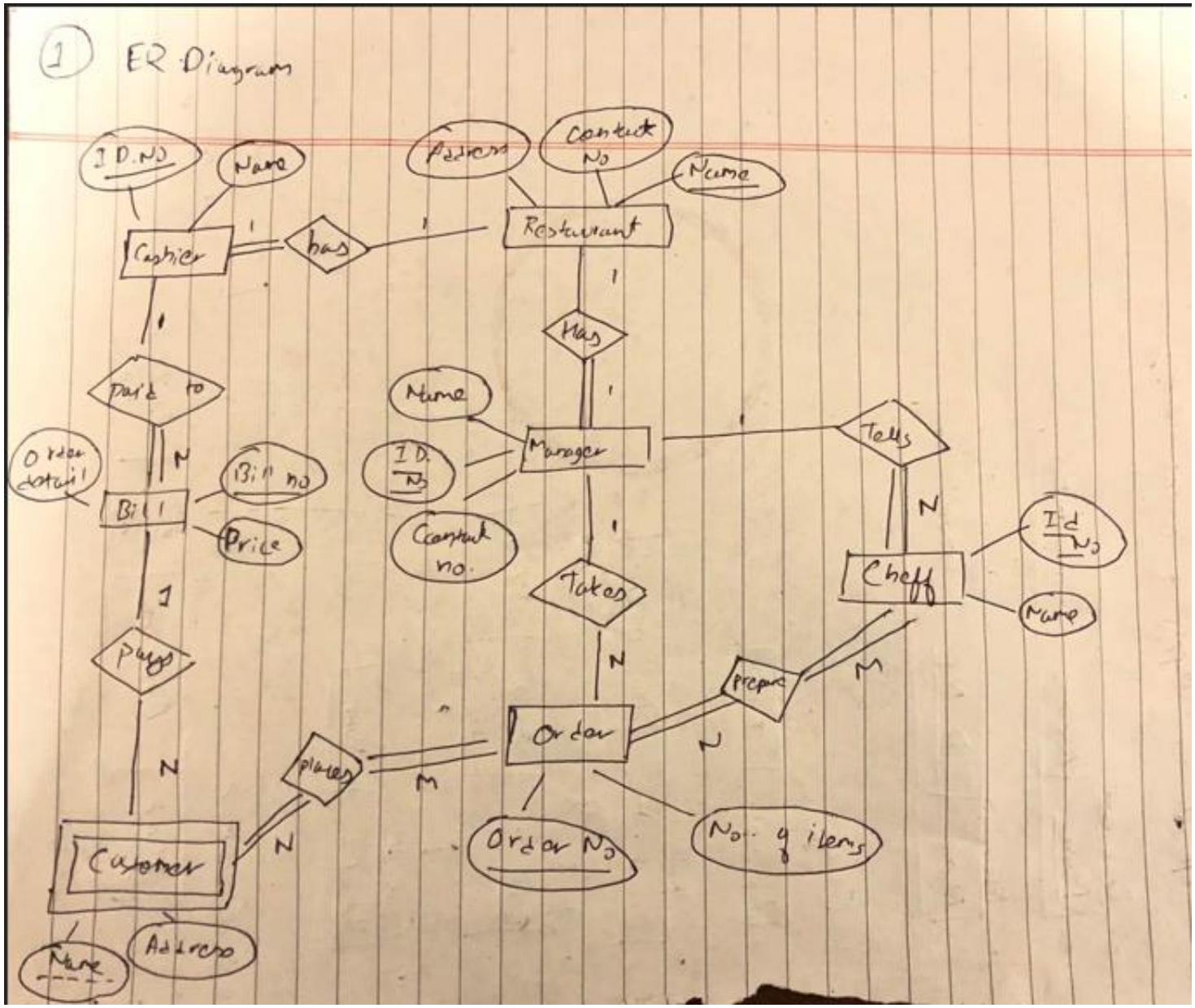
Course: DBMS

Digital Assignment

Done by: Prithak Gajurel

Registration Number: 20BCE2921

1. Draw the ER diagram of the database maintained to run a restaurant. Convert the ER diagram into corresponding tables. [10]



now,

Mapping the ER diagram to table

Step I: Mapping Strong Entity:

- Restaurant

Name	Address	Contact number

- Manager

Name	ID. No	Contact no.

- Cheff

Id No	Name

- Order

Order No	No. of items

- Bill

Bill no.	Order detail	Price

- Cashier

Id. No	Name

Step II : Mapping of weak entity

- Restaurant

Name	Address	Contact No.

- Manager

Name	ID No	Contact no

- Chef

Id. No	Name

- Order

Order No.	No. of items

- Bill

Bill no.	Order details	Price

- Cashier

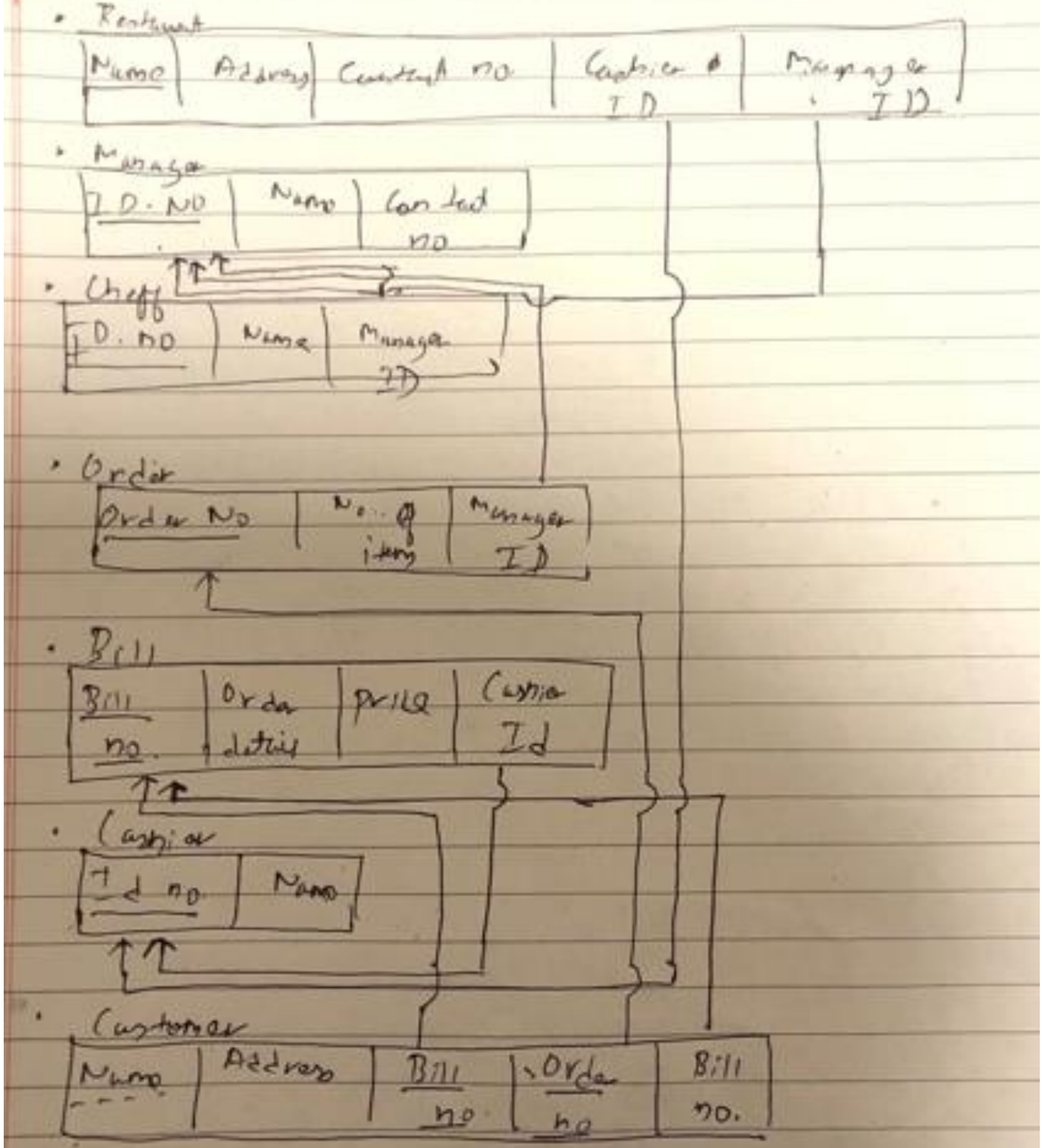
Id no	Name

- Customer

Name	Address	Bill no.	Order no.

[This is a weak entity]

Step 4: Mapping of M:1 : N :



Step 5:

Mapping M:N relationship

- Restaurant

Name	Address	Contact no.	Cashier Id	Manager Id

- Manager

I.D. No	Name	Contact no.

- Chef

I.D. no	Name	Manager ID

- Orda

Order no.	No of items	Manager ID

- Bill

Bill no.	Order detail	price	Cashier I D

- Cashier

Id no.	Name

- Customer

Name	Address	Bill no.	Order no.	Bill no.

- Placed

order.no	Customer name

- prepare

Order no.	Chef ID

2. Describe heuristic query optimization along with an example. [10]

(2)

Many different relational algebra expressions and hence many different query trees can be semantically equivalent, that is they can represent the same query and produce the same results.

The query parser will typically generate a standard initial query tree to correspond to the SQL query without doing any optimization.

A canonical query tree represents a relational algebra expression that is very inefficient if executed directly, because of Cartesian product (x) operations.

The heuristic query optimizer will transform this initial query tree into an equivalent final query tree that is efficient to execute.

The optimizer must include rules for equivalence among relational algebra expressions that can be applied to transform the initial tree into the final, optimized query tree.

The way how a query tree is transformed using heuristics is given below:

1) Break up 'select' operations with conjunctive conditions into a cascade of SELECT operations.

2) Using the commutativity of SELECT with other operations, move each SELECT operation as far down in the tree as is permitted by the attributes involved in the select condition.

3) Using commutativity and associativity of binary operations, rearrange the leaf nodes of the tree so that the leaf node relations with the most restrictive SELECT operations are executed first in the query tree representation.

4) Combine a CARTESIAN PRODUCT operation with a subsequent SELECT operation in the tree into a JOIN operation, if the condition represents a join condition.

5) Break down and move lists of projection attributes down the tree as far as possible by creating new PROJECT operations as needed.

6) Identify sub-trees that represent groups of operations that can be executed by a single algorithm.

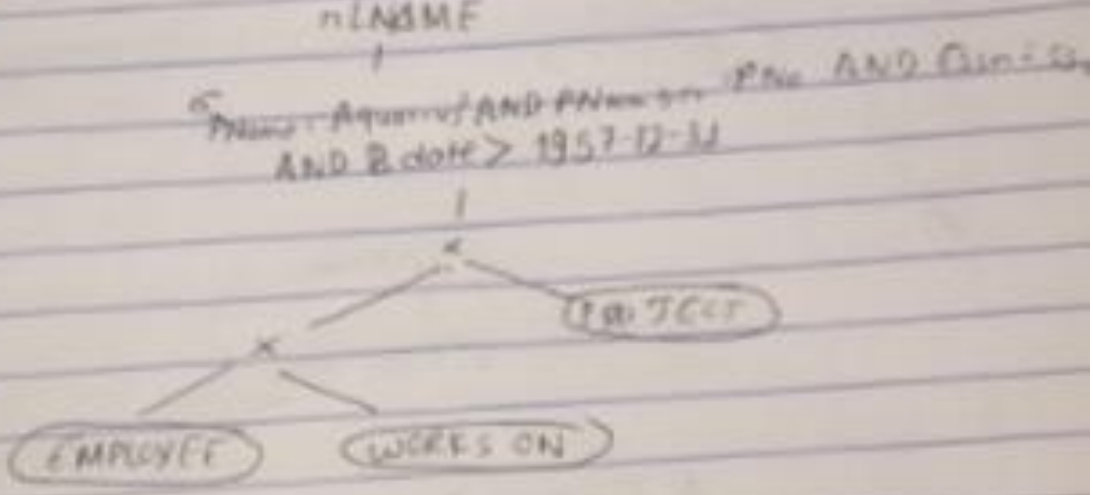
example:

Query: Find the last names of employees born after 1957 who work on project named 'Aquarius'.

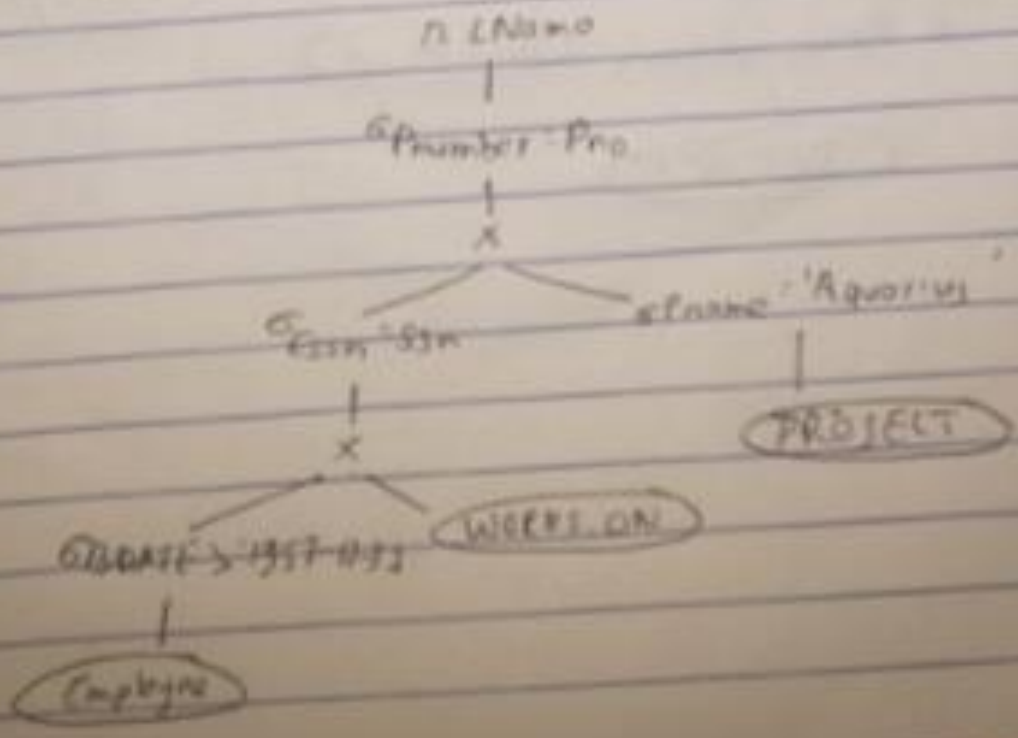
SQL

```
SELECT LNAME
FROM EMPLOYEE, WORKS_ON, PROJECT
WHERE PNAME = 'Aquarius' AND PNUMBER = PNO AND
ESSN = SSN AND BDATE > '1957-12-31'
```

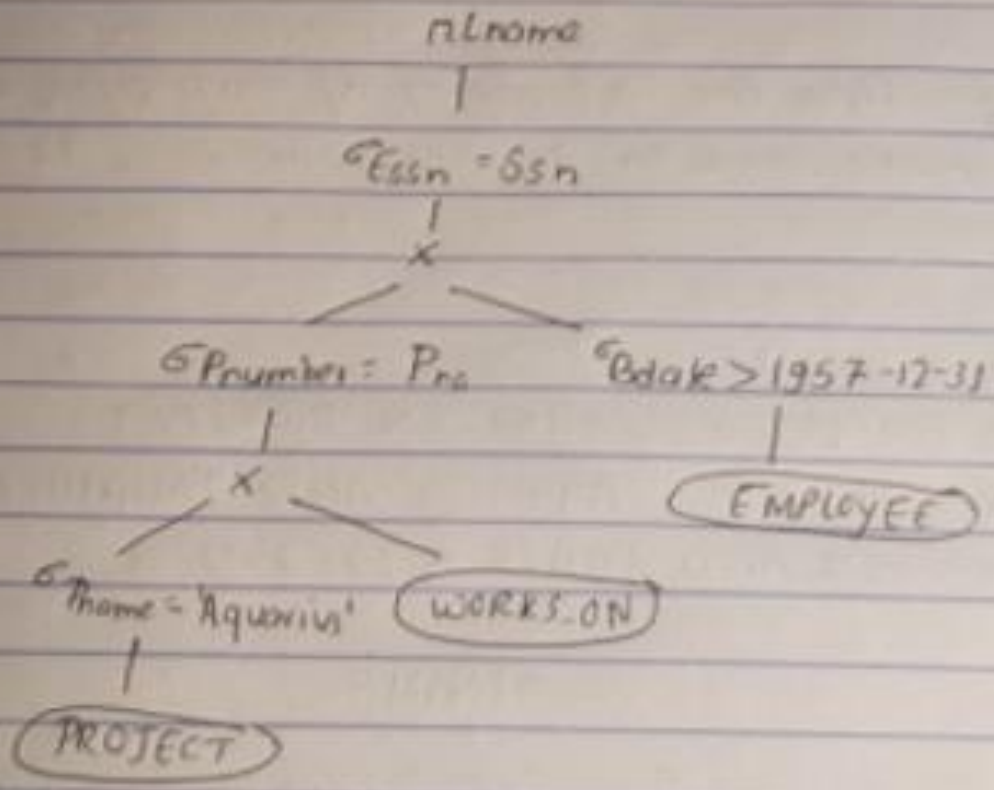
a)



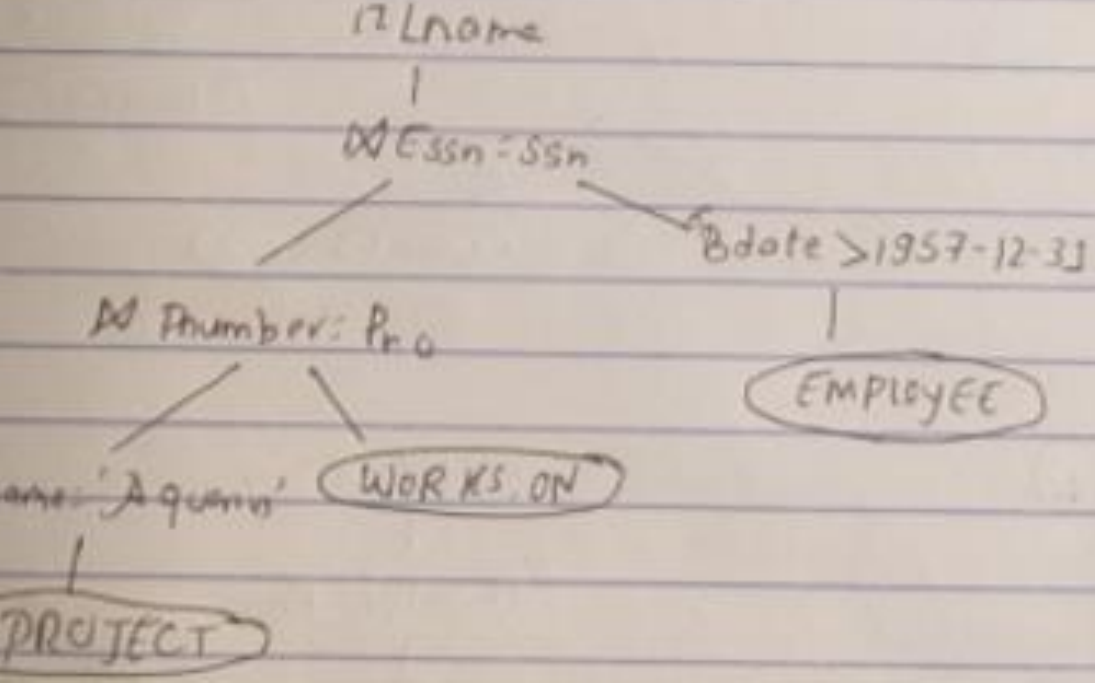
b)

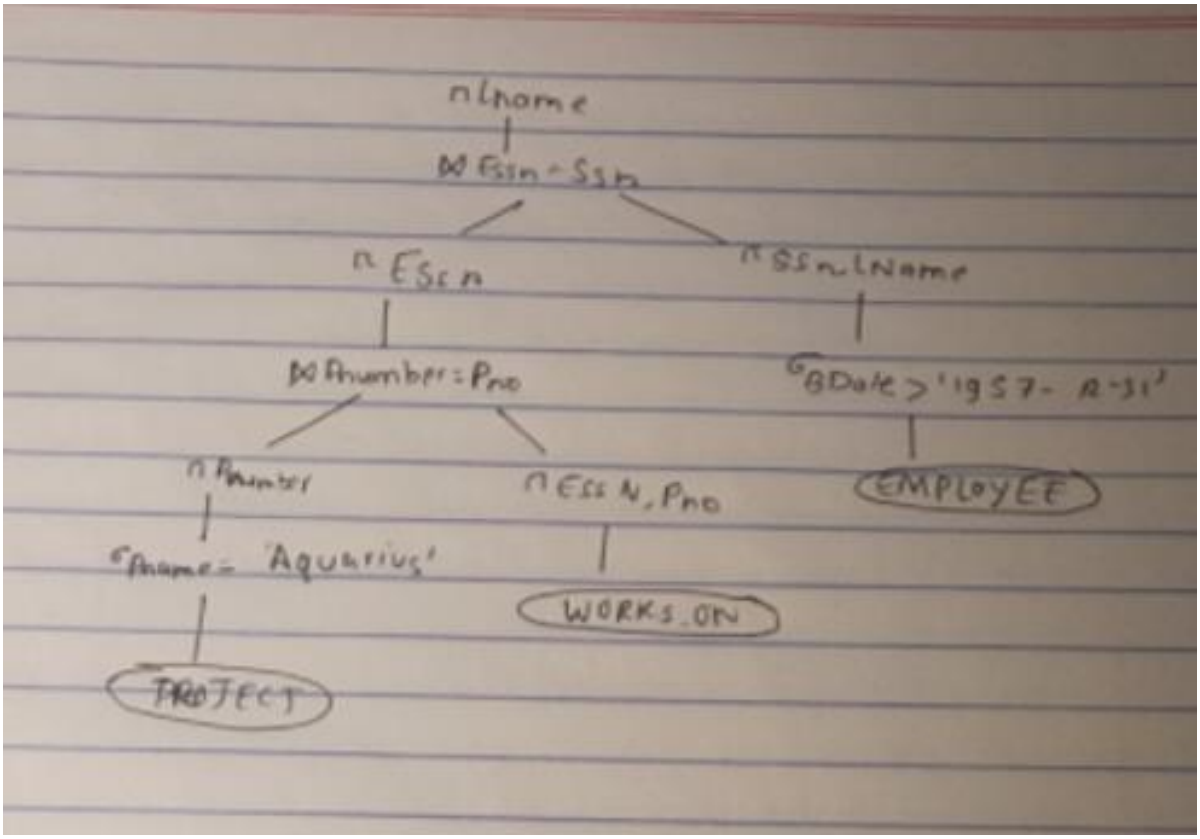


c)


πLname
σEssn = Ssn
×
σPnumber = Pno
σBdate > 1957-12-31
×
EMPLOYEE
σPname = 'Aquarius'
WORKS_ON
PROJECT

d)


πLname
⋈Essn = Ssn
σBdate > 1957-12-31
⋈ Pnumber = Pno
EMPLOYEE
σ Pname = 'Aquarius'
WORKS_ON
PROJECT


πLname
⋈Essn=Ssn
πEssn
πSsn,Lname
⋈Pnumber=Pno
σBDate>'1957-12-31'
πPnumber
πEssn,Pno
EMPLOYEE
σPname='Aquarius'
WORKS_ON
PROJECT

3. Describe the need of 2PL along with an example. [10]

(3)

Using binary locks or read/write locks in transaction doesn't guarantee serializability of schedules on its own. A transaction is said to follow the two-phase locking protocol (2PL) if all locking operations (read_lock, write_lock) precede the first unlock operation in the transaction.

This 2PL protocol guarantees serializability. That is one of the need of 2PL protocol.

Let's take up an example to prove it.

Transactions T1 and T2 as shown in Fig 1 follow two-phase locking protocol because the write_lock(X) operation follows the unlock(Y) operation in T1 and similarly the write_lock(Y) operation follows the unlock(X) operation in T2.

T_1	T_2
read_lock(Y);	read_lock(X);
read_item(X);	read_item(X);
unlock(Y);	unlock(X);
write_lock(X);	write_lock(Y);
read_item(X);	read_item(Y);
X := X+Y;	Y = X+Y;
write_item(X);	write_item(Y);
unlock(X);	unlock(X);

fig 1

So,

Initial values: X = 20, Y = 30

result of serial schedule T_1 followed by T_2 we get

X = 50, Y = 80

And the result of serial schedule T_2 followed by T_1 we get:
$X = 70, \quad Y = 50$

Hence, it does not satisfy conflict serializability

But if we consider the following example in fig 2;

	T_1'	T_2'

The given transactions T_1' and T_2' are in serializable order if we make the order $T_1' \rightarrow T_2'$ or $T_2' \rightarrow T_1'$	read_lock(Y);
	read_item(Y);
	unlock(Y);
		read_lock(X);
		read_item(X);
		unlock(X)
		write_lock(X);
		read_item(X);
		Y = X + Y;
		write_item(Y);
		unlock(Y);

let X = 20, Y = 30 So, X = 70, Y = 50 when $T_1' \rightarrow T_2'$ and X = 70, Y = 50 when $T_2' \rightarrow T_1'$	write_lock(X) read_item(X) X = X + Y; write_item(X) unlock(X)	

Hence, it can be proved that, if every transaction in a schedule follows the two-phase locking protocol, the schedule is guaranteed to be serializable.

2pl protocol is also a the solution for lost update problem and solution for inconsistent summary problem respectively.