DATABASE MANAGEMENT SYSTEM

NOTES

SERIES -4

LinkedIn ATUL KUMAR
7777888822

14/7/2015

Softwares : Oracle, Sybase, SQL server, DB2, mySQL, MS access.

Database System is a collection of inter-related data.

Book : Database System by Korth.
Elmasri and Navati.
Raghwam Krishnan.

What is Database ?

It is a collection of inter-related data and some programs to access those datas.

Database Management System (DBMS).

It is a collection of inter-related data and set of programs to access those data.

Disadvantages of File System.

i) Data redundancy and inconsistency.

ii) Difficulties in accessing the data.

iii) Data isolation. → joining.

iv) Integrity problem. → some constants.

v) Atomicity problem → all will run correct or not.

vi) Concurrent access anomalies → in db when one task is there, other have to wait.

Atul Kumar
7717888822

Database Administrator (DBA)
↓ head of
DB.

Security problem.

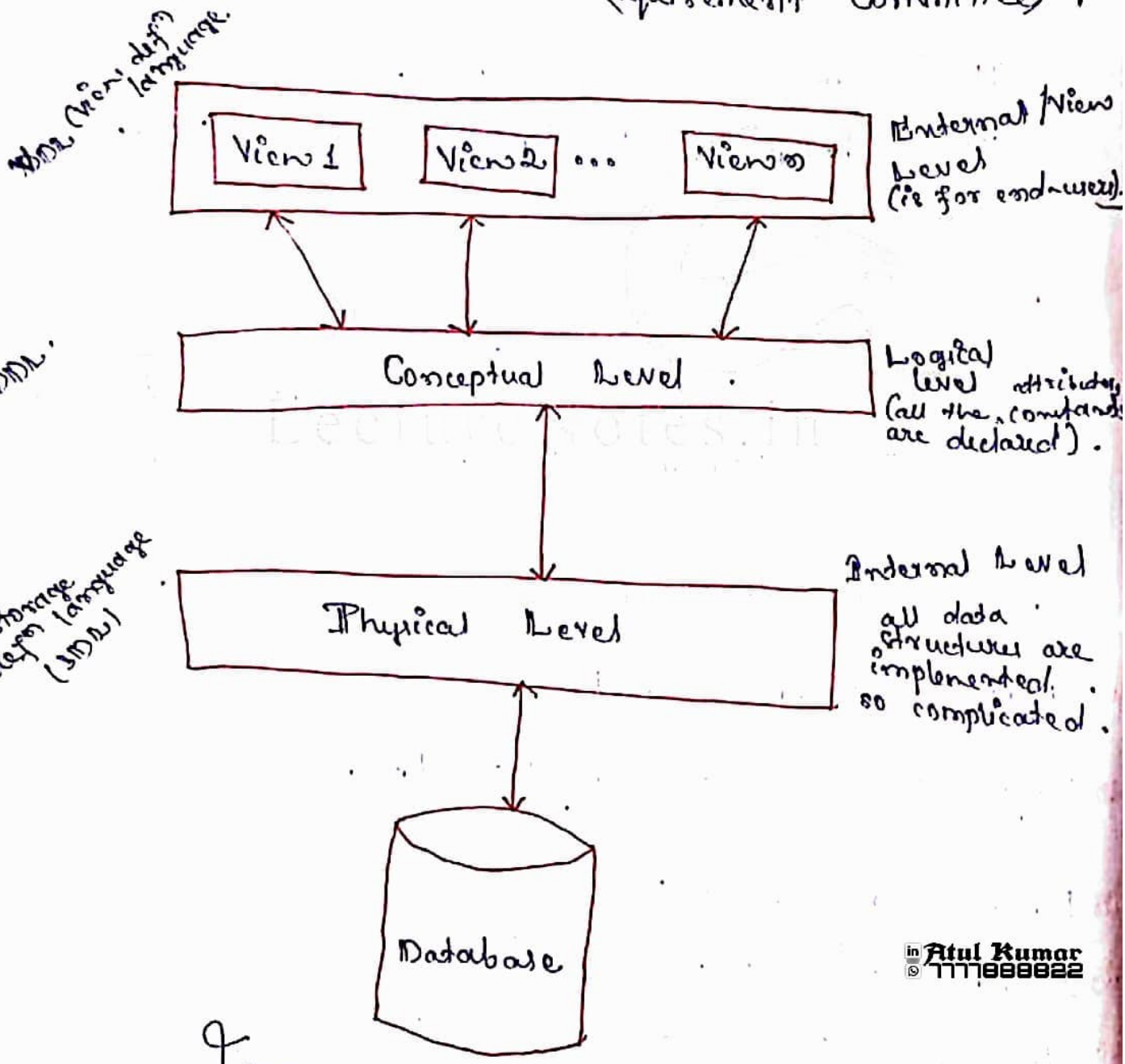
15/7/2015.

View of Data.

make the prgm user-friendly.

DB is designed in 3 levels : known as three schema/ANSI - SPARC (American National Standard Institute - Standard Planning and Requirement Committee).

↓ overall design of database.



Atul Kumar
7777888822

Three Schema

Three Schema Architecture.

i) Physical Level.

*- Physical level describes how the data are actually stored. It describes storage space allocation for data and indices. It describes record description for storage. It describes data compression and encryption techniques.

ii) Logical Level.

*- Logical level describes what data are stored in the database & what relationship exists among those data. It describes all entities and their attributes and relationship.

It describes the constraints on the data.

*- Security and integrity information.

iii) External Level.

*- It describes only part of the database.

What is Schema and instance.

The overall description or design of the database is known as schema. (intension).

Eg: employee schema
student schema.

Atul Kumar
7777888822

Data is stored at a particular moment is called instance. (extension).

- Schema can be divided into :
  - i) physical schema → describing physical level.
  - ii) conceptual schema → describing conceptual level.

Sub-Schema.

- A database may have several schema at view level. Sometimes called sub-schema.

Data Independency.

(Fig).
if we want to add, we hv to change the application prgm.

- The capacity to change the schema at one level of the database schema without having change the schema at the higher level.

It is of 2 types :

i) Logical Data independency

It is the capacity to change the conceptual schema without having to change external schema or application program.

Ex. Addition or removal of entities or their

Atul Kumar
7777888822

(ii) Physical Data Independency.

It is the capacity to change the internal schema without having to change the conceptual schema.

Eg: Modifying indices or hashing algorithm.

17/7/2015.

Advantages and disadvantages of DBMS:

Advantages:

(i) Minimal data redundancy.

(ii) Data consistency.

(iii) Data independency.

(iv) Data integrity.

(v) Greater control.

(vi) Reduced maintenance.

Disadvantages.

(i) Software is very costly.

(ii) Cost of hardware and software.

LectureNotes.in

Atul Kumar
7777888822

Database User.

(i) Naive user
(ii) Application programmer.
(iii) Sophisticated user.
(iv) Specialised user.

(i) Naive User.

*- These are the unsophisticated users to interact with the system by invoking one of the application program that have been written previously.

Atul Kumar
7777888822

(ii) Application Programmer.

*- These are the computer professionals to write application programs.

LectureNotes.in

(iii) Sophisticated Users.

*- Sophisticated users interact with the system without writing the programs, instead they form their request in a database query language.

(iv) Specialised User.

*- Specialised users write specialised database application that do not fit into traditional

Ex: CAD user (Computer Aided Design).

Role of database administrator (DBA):

i) Schema definition.

It creates the original database schema by executing the set of definition language in data definition language (DDL).

DDL is used to create the schema.

Ex: Student schema.

Roll no number(3)

name varchar2(10).

DOB date

Address varchar2(30)

Create table student (table name)

(roll no number(3) ;

name varchar2(10) ;

DOB date ;

address varchar2(30)) ;

It will show table created.

To check : desc (table name);

ii) Storage structure and access method definition.

iii) Granting of autorization for data access.

iv) Routine maintenance.

*- Periodically, backing of data either onto tapes or onto remote servers to prevent lose of

LectureNotes.in
Atul Kumar
7777888822

data in case of any disaster.

* Ensuring that free disc space is available for normal operation and upgrading the as required.

* Monitoring jobs running on the database ensuring that performance is not degraded by every expensive task submitted by some users.

Database Languages.

DDL.	DML.
DDL compiler checks and creates the table.	To manipulate data in the table.
All the information are stored in system catalog or data dictionary. System catalog contains meta data (data about data).	Insertion, deletion, updation.

Atul Kumar
7777888822

DDL. (Data Definition Language)

A language that allows the DBA or user to describe and name the entities, attributes and relationship required for the application together with any associated integrity and security constraints

DML. (Data Manipulation Languge).

A language that provides a set of operation to support basic data manipulation operations on the data held in the database.

It includes insertion, modification, retreival, deletion.

*- DML can be catagorized into 2 types :

(i) Procedural DML.

It require a user to specify what data are needed, and how to get those datas.

(ii) Non-Procedural DML.

It require a user to specify what data are needed without specifying how to get those data.

Atul Kumar
7777888822

Data Model.

It is a collection of concepts that can be used to describe the structure of database.

*- Structure of database means data types, relationship and constraints that should hold the data. (provides the necessary means to acheive the abstraction).

*- The data model can have 3 components :

(i) structural part.

It consists of set of rules according to which database can be constructed.

(ii) Manipulating part.

Defining the basic types of operations that are allowed on data.

(iii) set of integrity rules.

It ensures that the data is accurate.

Categorisation.

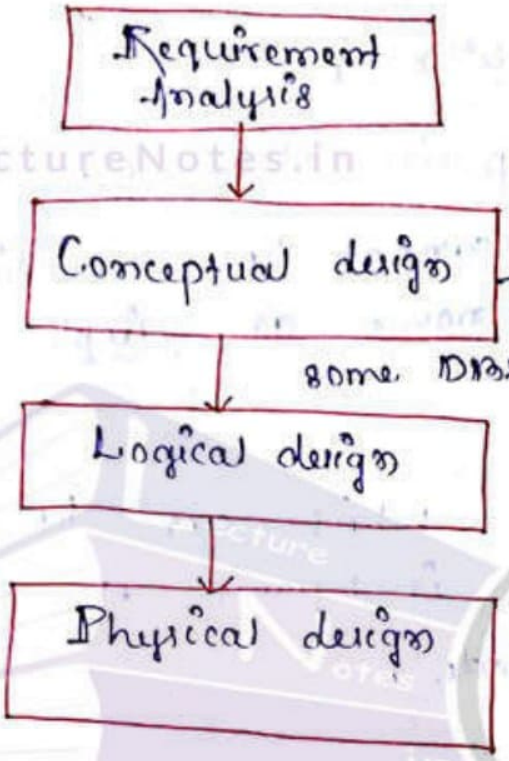
Atul Kumar
7117888822

Data Model

- Object based
  - → E-R model
  - → semantic model
  - → functional model
- Record based
  - → network
  - → hierarchical
  - → Relational
- Physical Data model
  - → unifying model
  - → frame mo

collection of data
representation in diagram → E-R model
use schema.

Different Steps of Database Design.



E-R Model (Entity-Relation Model).

Entity.

An entity is a thing or object that is distinguishable from all other objects.

Ex: Student, customer, employee.

Atul Kumar
7778888822

Student

Property attribute

Roll no	name	address	DOB	Division

→ 1 entity.

1 record is one entity

Entity Set.

It is a set of entities of same type that share the same properties or attributes. They have common properties, but value is different.

Different types of attributes.

1. Simple and Composite attribute.

The attribute cannot be sub-divided into sub-parts is known as simple attribute.

Ex: Roll no.

The attribute which can be sub-divided into sub-parts is known as composite attribute.

Ex: Name.

Atul Kumar
7777888822
LectureNotes.in

2. Single and Multi-value attribute.

The attribute having single value is known as single attribute.

Ex: Roll no., regd. no.

The attribute having multiple value is known as multi-value attribute

Ex: Ph. no., Address.

3. Null Value :

The value is unknown or the value is not applicable for that attribute.

Ex: Commision.

21/7/2015.

4. Derived Descriptive attribute :

Ex. from DOB, we can know age.

stored attribute. derived attribute.

The values of this type of attribute can be derived from the values of other related attribute of other entities.

Relationship.

Customer

Acc. no	Name	DOB
a-1	Jones	1-5-80

Loan

Loan no.	amount
L-101	1,00000

Relationship :

Borrower

Jones	L-101

Atul Kumar
7778888822

A relationship is an association among several entities.

Relationship Set

It is a set of relationship of same type.

It is a mathematical relation on $n \geq 2$ entity set

If $E_1, E_2, \ldots, E_n$ are entity set, then relationship set (R) is subset of

$$R \subset \{(e_1, e_2, \ldots, e_n) \mid e_1 \in E_1, e_2 \in E_2, e_3 \in E_3 \ldots e_n \in E_n\}.$$

Constraints on E-R Model

Atul Kumar
7778888822

Two types of constraints :

i) mapping cardinality or cardinality ratio,

ii) participation constraint.

i) Mapping Cardinality

It is expressed as the number of entities to which another entity can be associated via a relationship set.

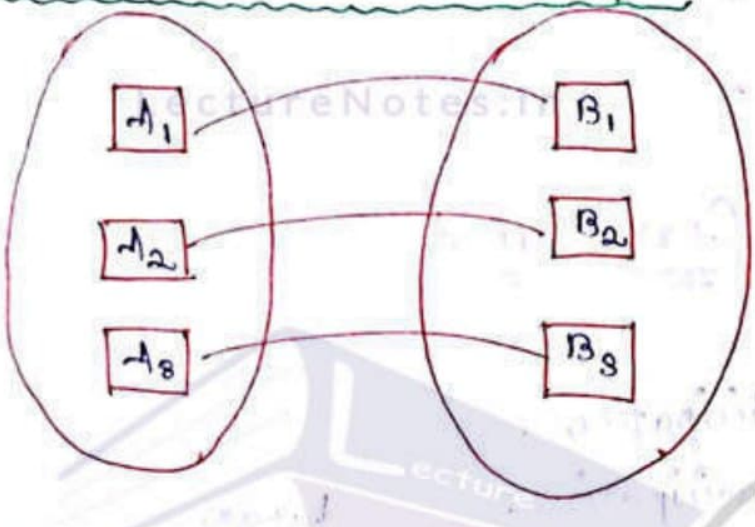
*- For binary, relationship set are the entity set A and B, the mapping

cardinality can be one of the following :

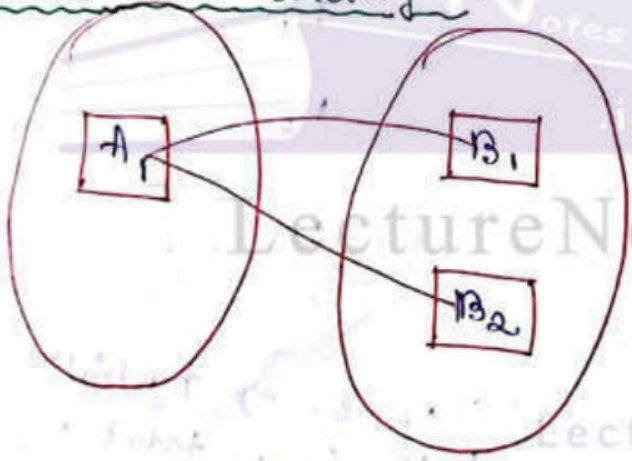
i) one - to - one
ii) one to many
iii) many to one
iv) many to one many .

how many entity set with rel^n is degree .
2 entity → degree 2
3 3 .

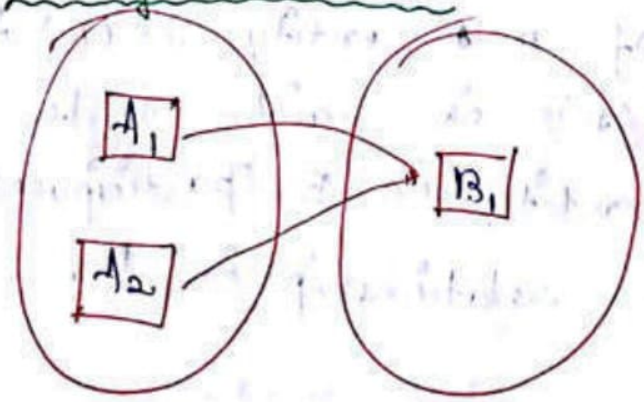
ii) One - to - one relationship .



iii) One to many .



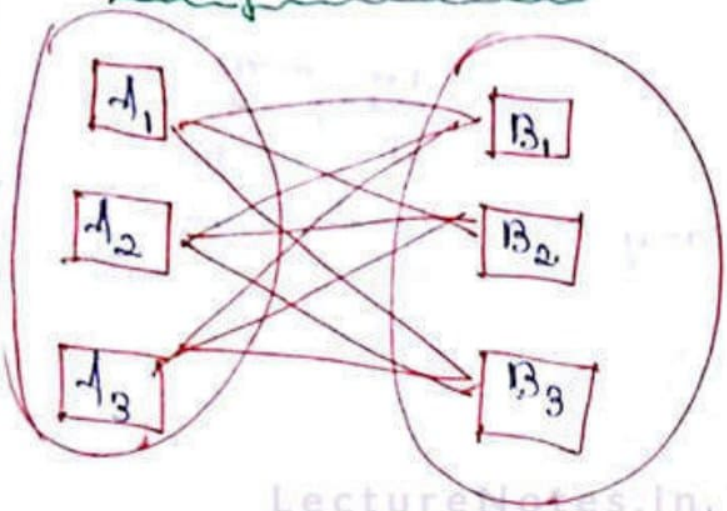
iii) Many to One .



LectureNotes.in

Atul Kumar
7777888822

iv) Many to many :



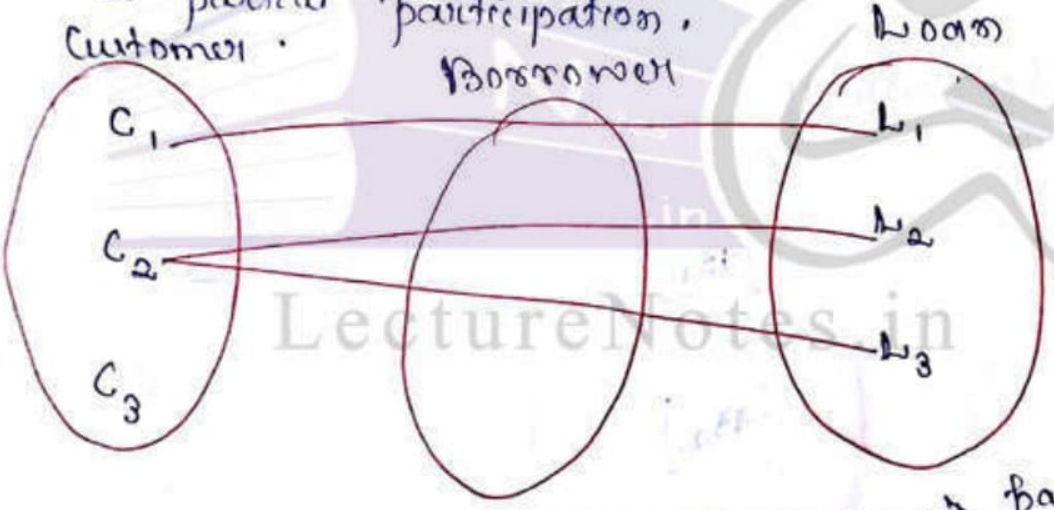
LectureNotes.in

ii) Participation Constraint :

It can be

i) total participation.

ii) partial participation.



Atul Kumar
7778888822

C to L → partial
L to C → total.

i) Total Participation :

The participation of an entity set (E) in a relationship set (R) is said to be total if every entity in E participates in atleast one relationship in R.

ii) Partial Participation :

If only some of the entities in E participation of participates in relationship R is said to be partial.

5. Descriptive Attribute :

Attribute derived from relationship set or a relationship set may have attribute known as description attribute.

Keys :

*- It helps uniquely to identify an entity from an entity set.

*- The types of keys are :

super key
candidate key
primary key.
alternate key
composite key.

Super key :

One ~~Two~~ or more attributes uniquely identify an entity from an entity set, then it is called super key.

Ex: {RollNo, name, address}.

LectureNotes.in

Atul Kumar
7778888822

Candidate Key :

Minimal super key is the candidate key.

Ex: {Roll no}, {name, address}.

Primary Key :

There are several candidate keys in an entity set. The database designer choose one of the candidate key as the primary key whose value is rarely changed.

Ex: {Roll no}

Alternate Key :

There are several candidate keys present in an entity set. One of the key is chosen as primary key and other keys are known as alternate keys.

Composite Key :

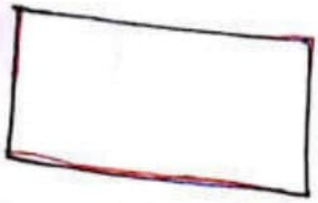
Two or more ~~candidate~~ keys attributes use to make a candidate key is known as composite key.

Atul Kumar
7777888822

22/7/2015.

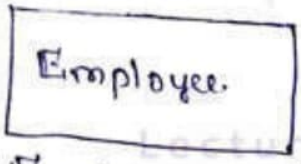
E-R Diagram :

Symbols :

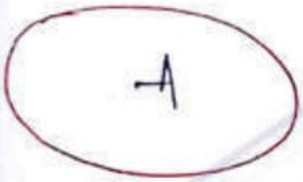


: Entity set.

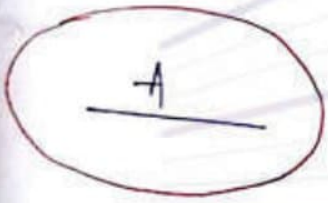
Ex:



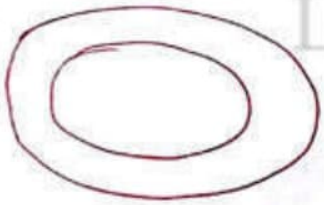
Employee entity set.



: Attribute.



: Primary key



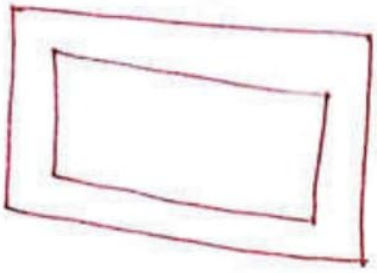
: Multi-value attribute.

: Relationship set.

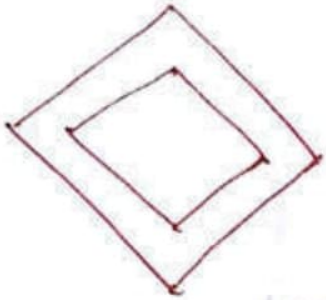
: Derived attribute.

LectureNotes.in

Atul Kumar
7778888822

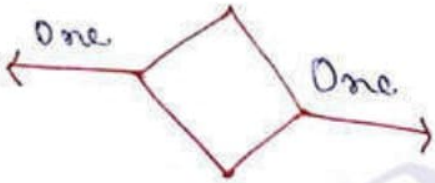


: Weak entity set.



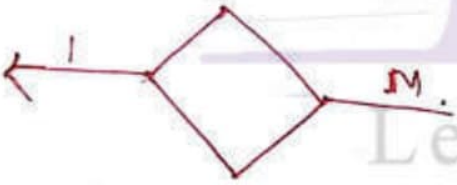
: Identifying relationship set for weak entity set.

LectureNotes.in



: One to one.

: Many to one.



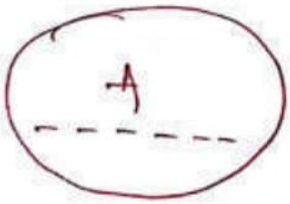
: One to many.

LectureNotes.in

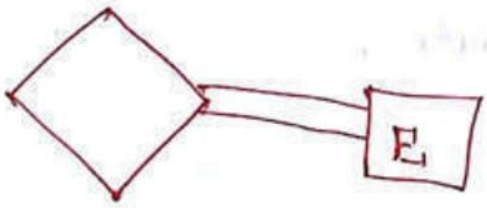
: Many to many.

LectureNotes.in

Atul Kumar
7777888822

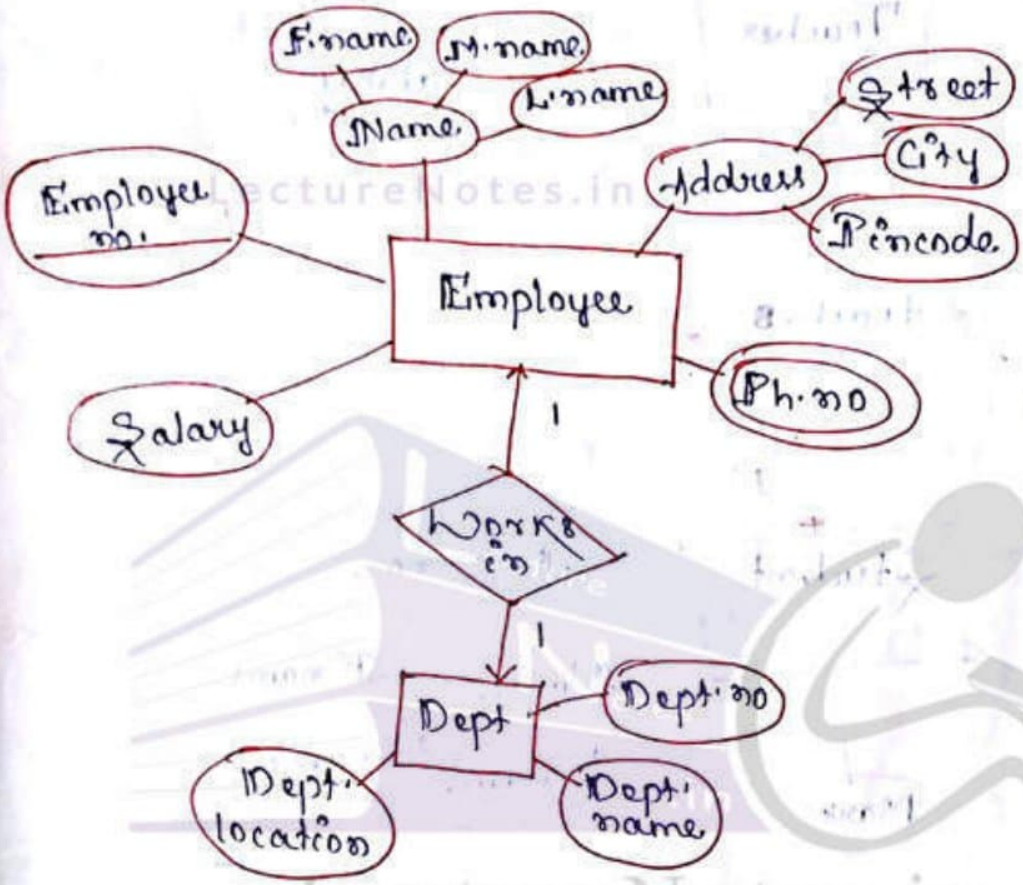


: Partial key or discriminator of weak entity set.



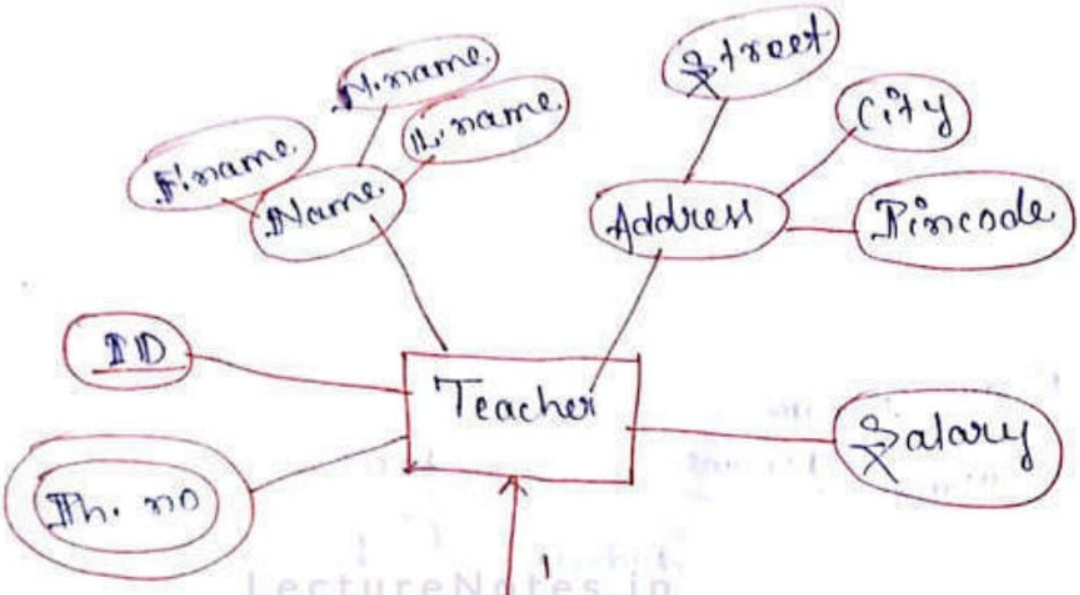
: Total participation.

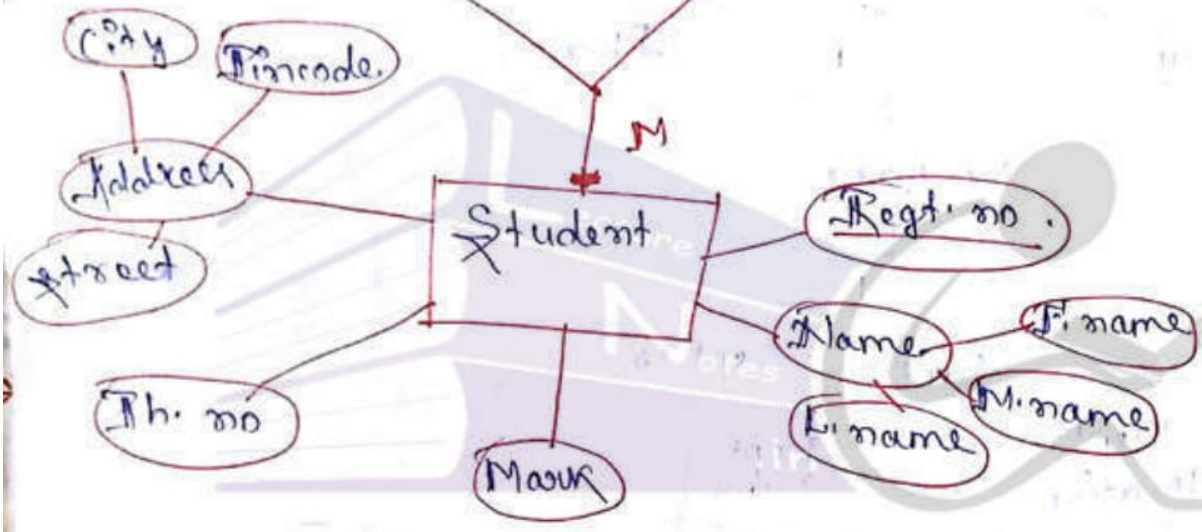
Draw an E-R diagram for employee.



LectureNotes.in

Atul Kumar
7778888822

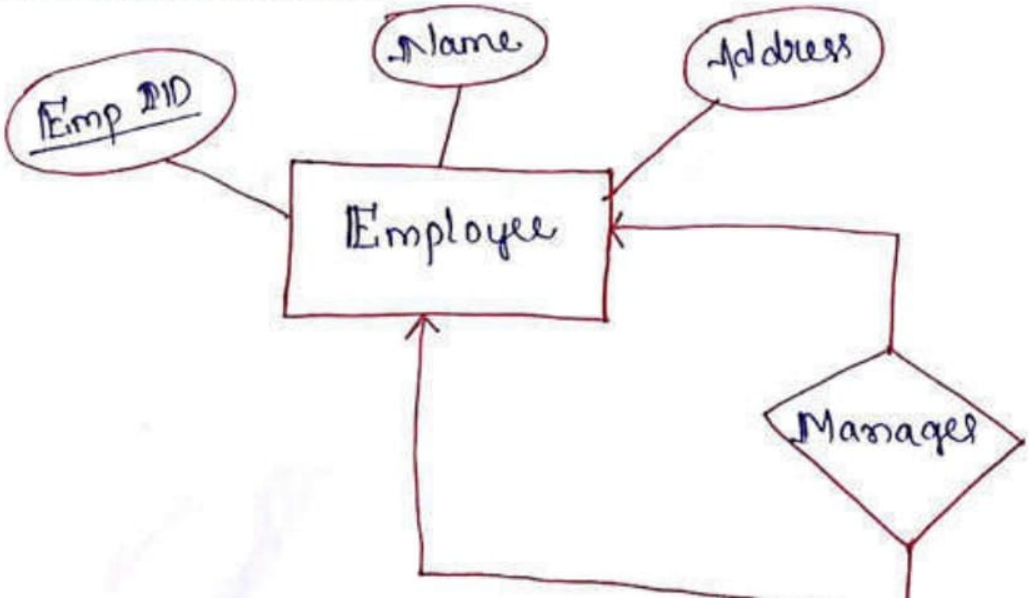




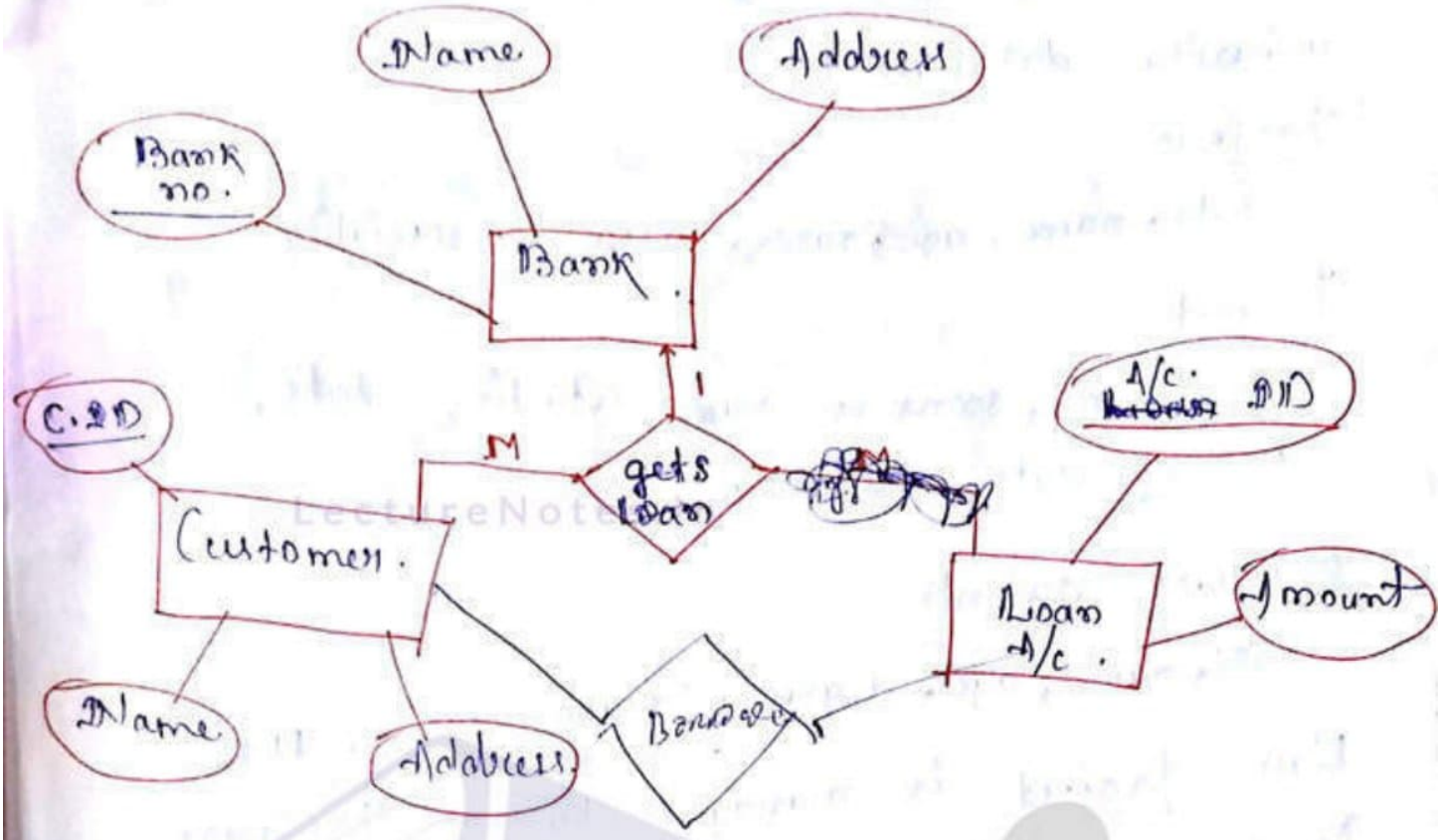
LectureNotes.in

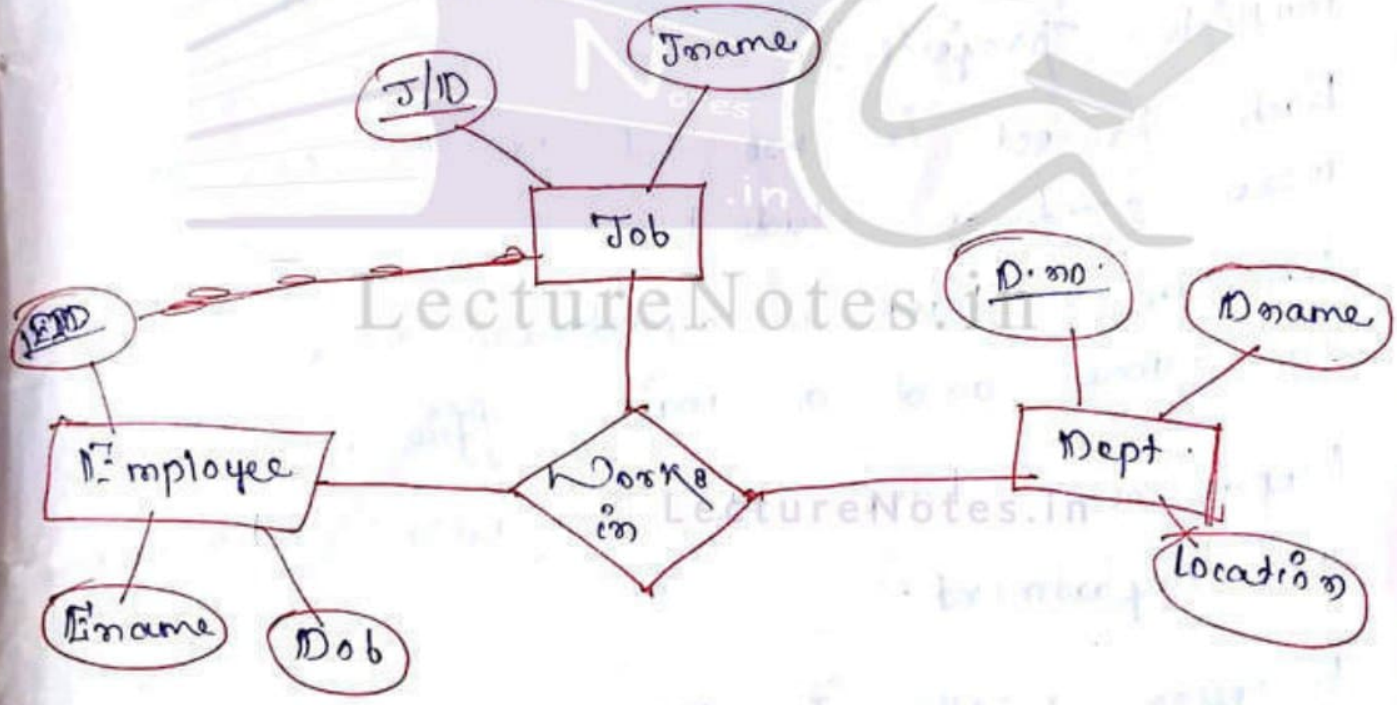
[One teacher teaches many student]

Unary entity set with a relationship.



Atul Kumar
7778888822





Q3 entity set ?

Atul Kumar
7717888822

Consider the following inf^n for the university database :

professor

SSN, name, age, rank, research speciality.

project.

project no., sponsore name, starting date, ending date, budget.

graduate students.

SSN, name, age, degree program (MS or PhD).

Each project is managed by a professor.

Professor can manage or works on multiple projects.

Each project is worked on by one or more graduate students.

Department have a department no., dept. name and a main office.

Department have a professor who runs the department.

Professor works in one or more departments and for each department, they work in a time percentage is associated with their job.

Graduate students have one major

Atul Kumar
7777888822

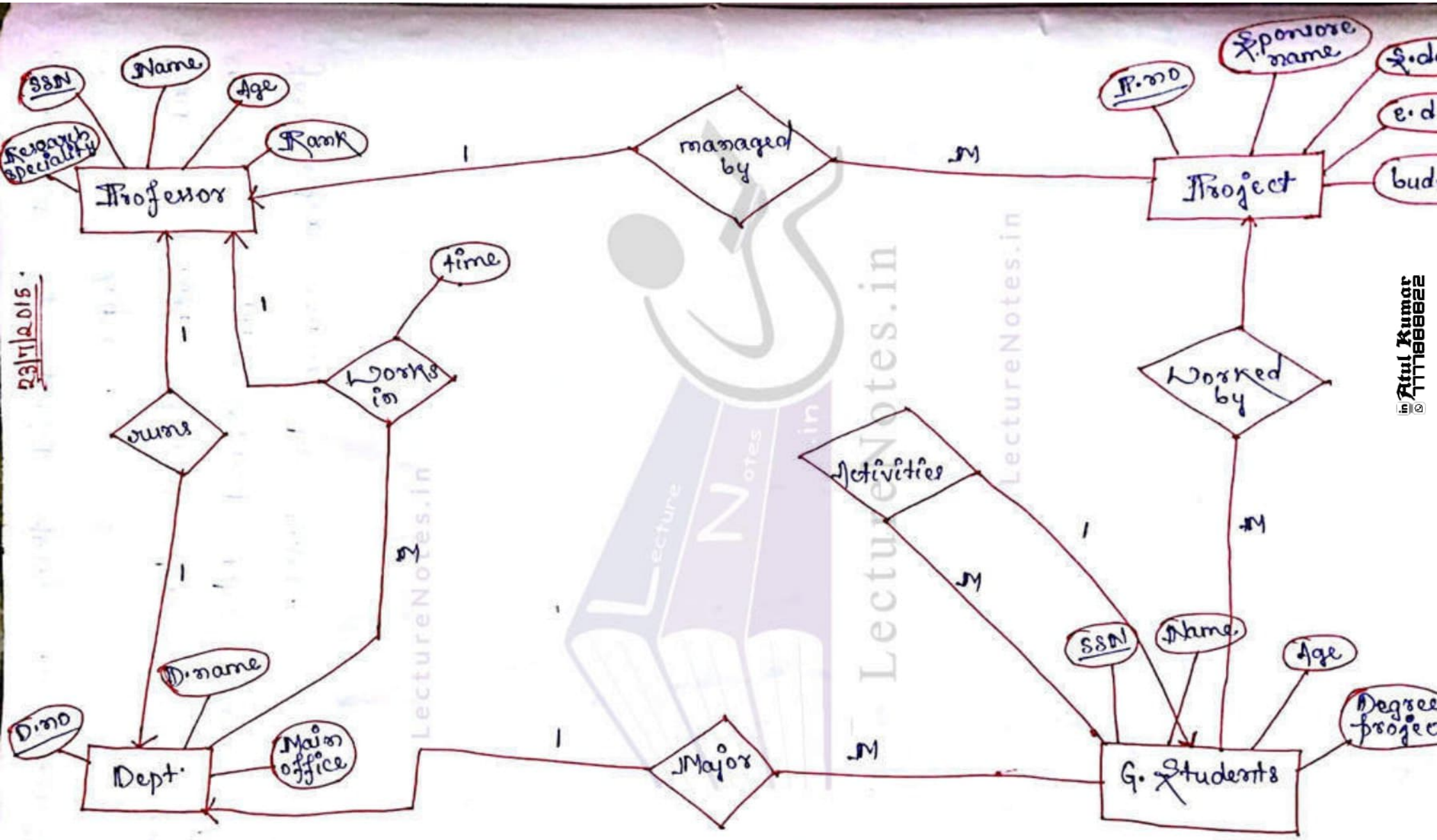
department in which they are working on their degree.

Each graduate student has another more senior graduate student who advices him for what course he takes.

LectureNotes.in

Atul Kumar
7777888822

23/7/2015.



Atul Kumar 7779888822

Q.

(i) Patients are identified by SSN and their name, address and age must be recorded.

(ii) Doctors are identified by SSN. For each doctor, the name, speciality and years of experience must be recorded.

(iii) Each pharmaceutical company is identified by name and ~~mus~~ has a phone number.

(iv) For each drug, the tradename and formula must be recorded. Each drug is sold to a given pharmaceutical company and the tradename identify the drug uniquely from among the product of the company.

(v) Each pharmacy has a name, address and phone number.

(vi) Each patient has a primary physician. Every doctor has atleast one patient.

(vii) Each pharmacy sells several drugs and has price for each. A drug could be sold at several pharmacies and the price could vary from one pharmacy to another.

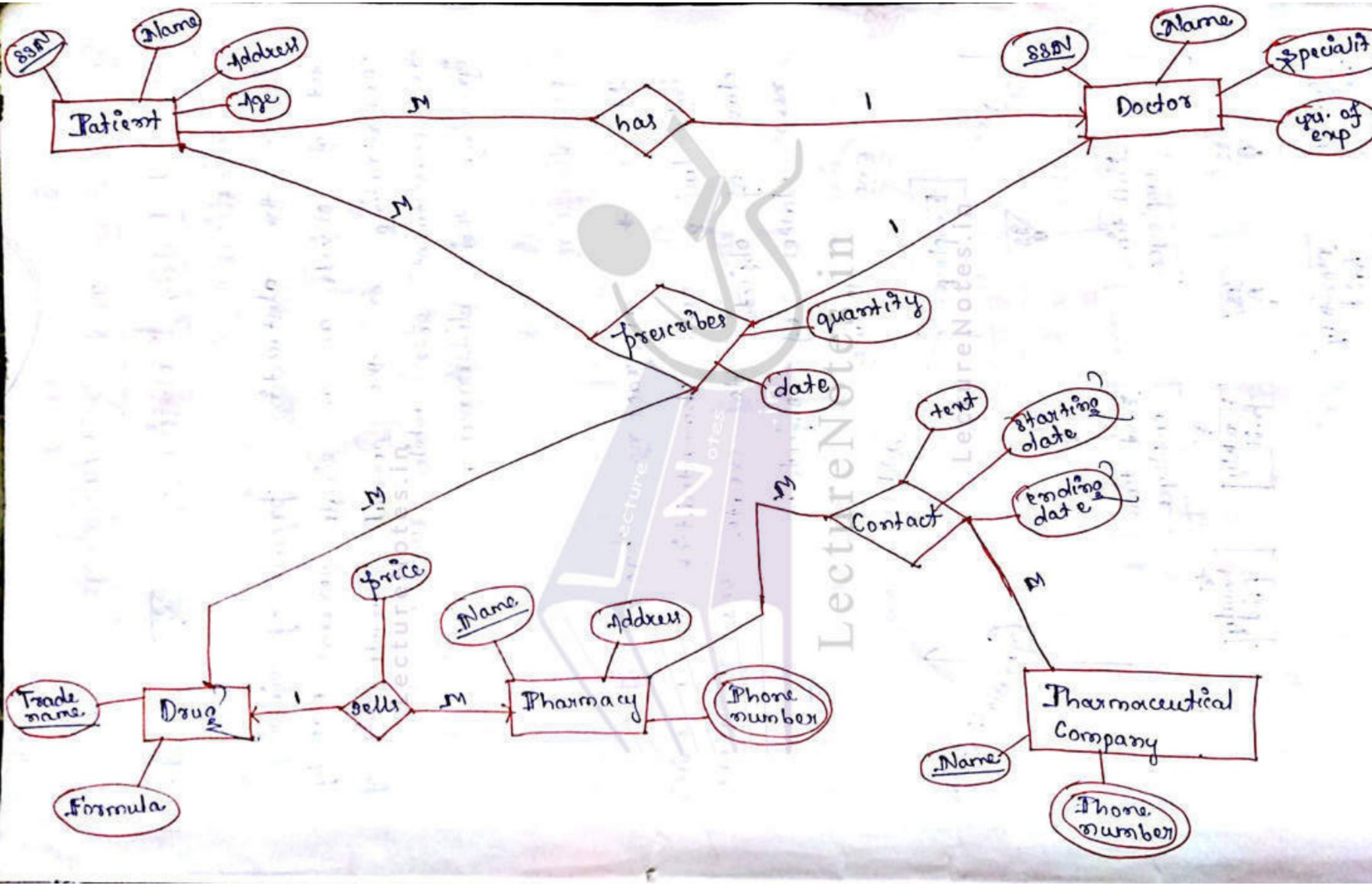
(viii) Doctor prescribes drugs for patients. A doctor can prescribe one or more drugs for several patients and a patient could obtain prescriptions for several doctor. Each prescription

Atul Kumar
7778888822

has a date and a quantity associated with it.

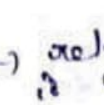
ex Pharmaceutical companies have long long contacts with pharmacies. A pharmaceutical company can contact with several pharmacies and pharmacy can contact with several pharmaceutical companies. For each contact you must have to store starting date, ending date and text of contact.

Lecture Notes .in
Atul Kumar
7777888822
LectureNotes.in
LectureNotes.in


SSN
Name
Address
Age
Patient
M
has
1
SSN
Name
Specialit
Doctor
yrs. of exp
M
1
prescribes
quantity
date
text
Starting date
ending date
M
Contact
M
M
price
Name
Address
Trade name
Drug
1
sells
M
Pharmacy
Phone number
Pharmaceutical Company
Name
Formula
Phone number

Generalisation and Specialisation.

Extended Entity Relation.



→ relationship is used.

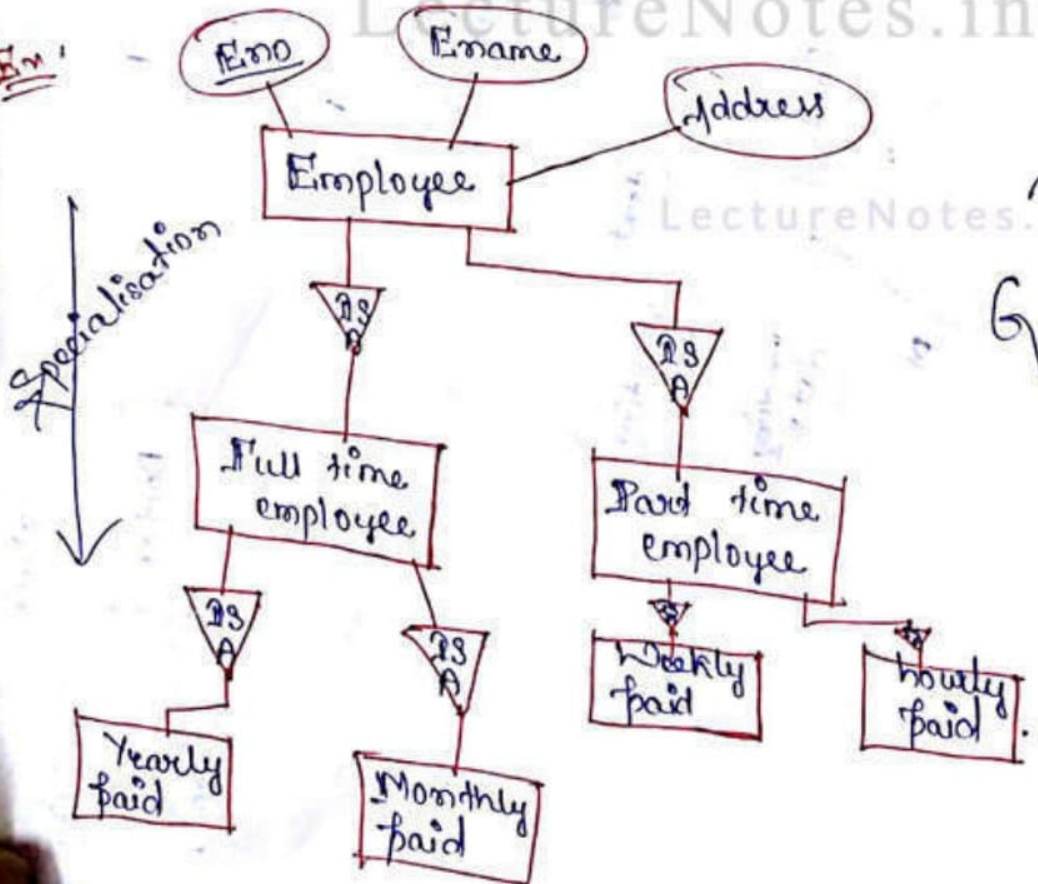
Generalisation:

It is the abstracting process of viewing set of objects as a single general class by concentrating on the general characteristics of the constituent sets while supressing or ignoring their differences.

Specialisation.

It is the abstracting process of introducing new characteristics to an existing class of objects to create one or more new classes of objects.

Ex:



LectureNotes.in

Atul Kumar
7777888822

Properties of Generalisation & Specialisation.

1. Attribute Inheritance.

*- A property of higher level & lower level entities created by generalisation and specialisation is called attribute inheritance.

*- The attribute of higher level entity set are set to be inherited by lower set entity set.

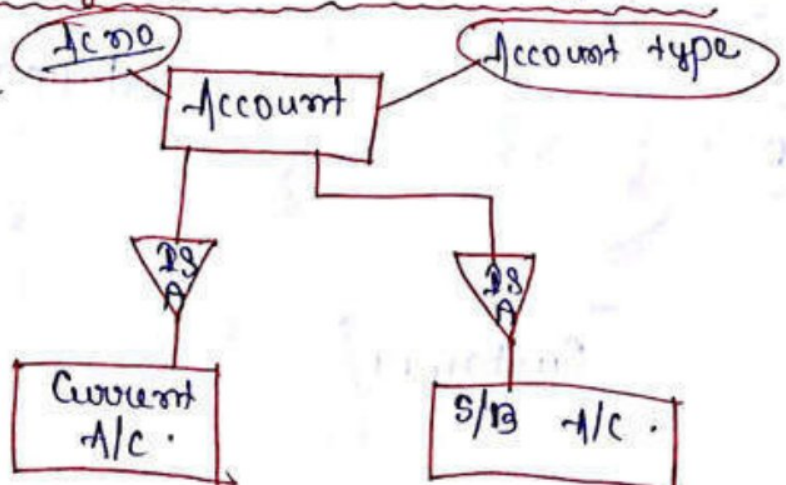
*- Lower level entity set with distinct features that apply within a particular ~~lower~~ level entity set.

*- In a hierarchy, an entity set may be involved as a lower level entity set in only one IS-A relationship. That diagram is called single inheritance.

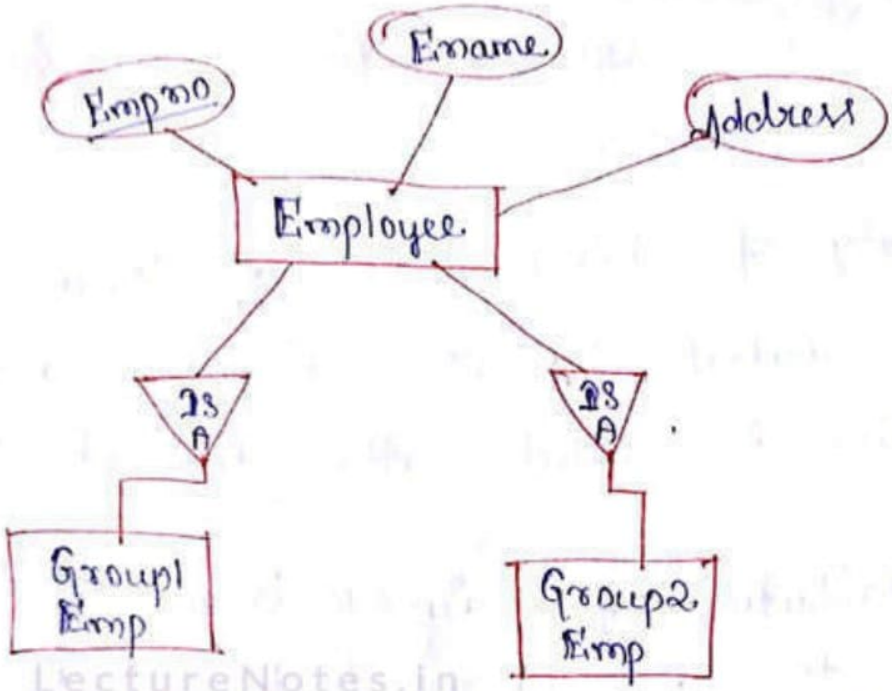
Constraints on Generalisation and Specialisation.

1. Condition defined and user defined.

Condition defined.



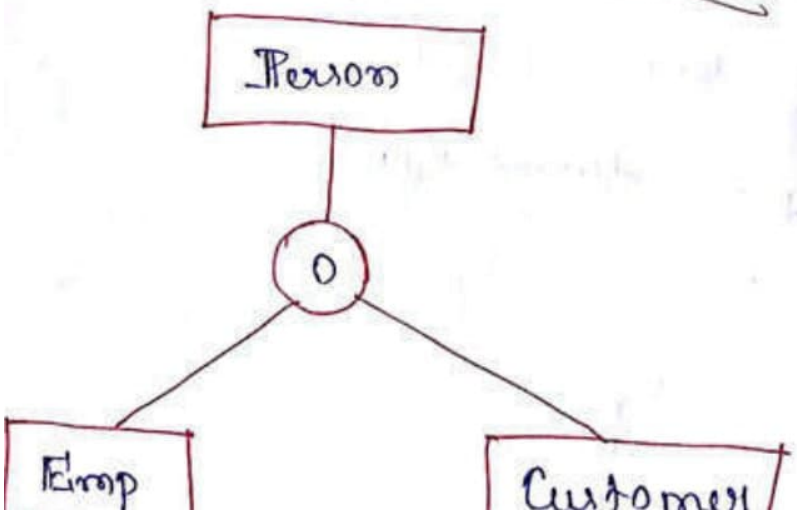
User defined.



- In condition defined, lower level entity set membership is evaluated on the basis of whether or not an entity satisfies an explicit condition.

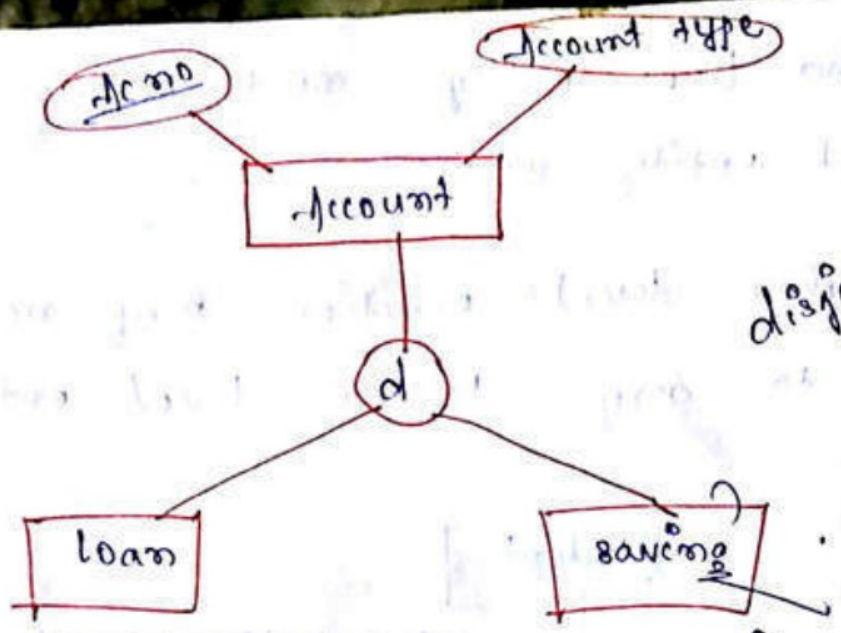
- User-defined entity set are not constrained by a membership condition rather than the database user assign entities to a given entity set.

Disjoint and overlapping.



overlapping.

Atul Kumar
7778888822

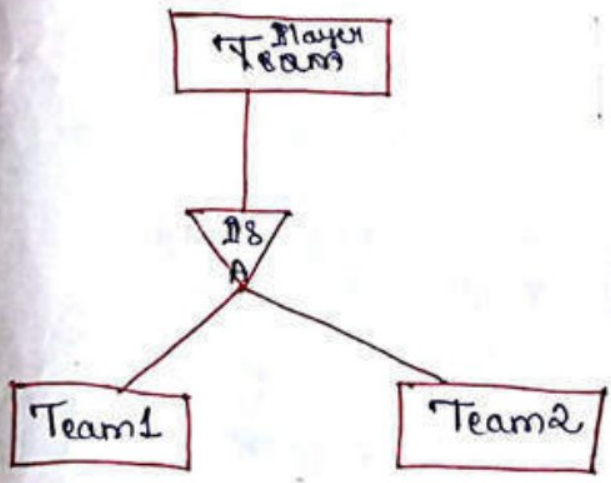


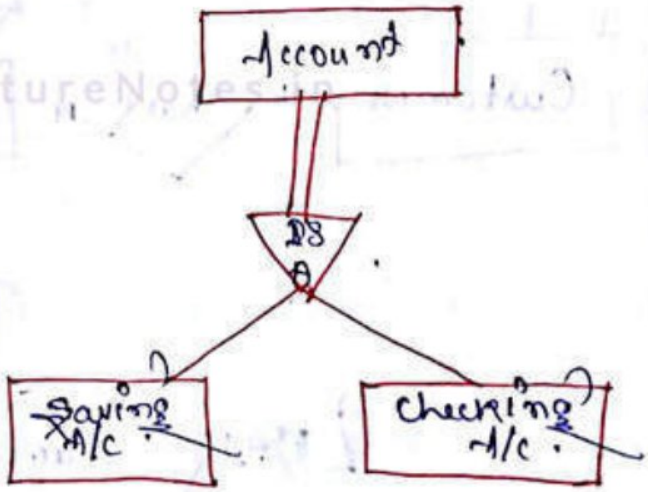
disjoint.

* Disjoint requires that an entity belongs to no more than one lower level entity set.

*Imp! Same entity may belong to more than one lower level entity set within a single generalisation.

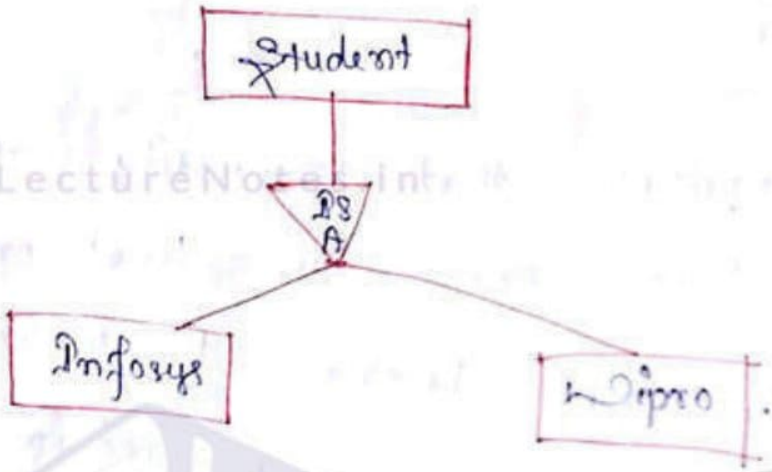
3. Total generalisation and specialisation and Partial generalisation and specialisation.





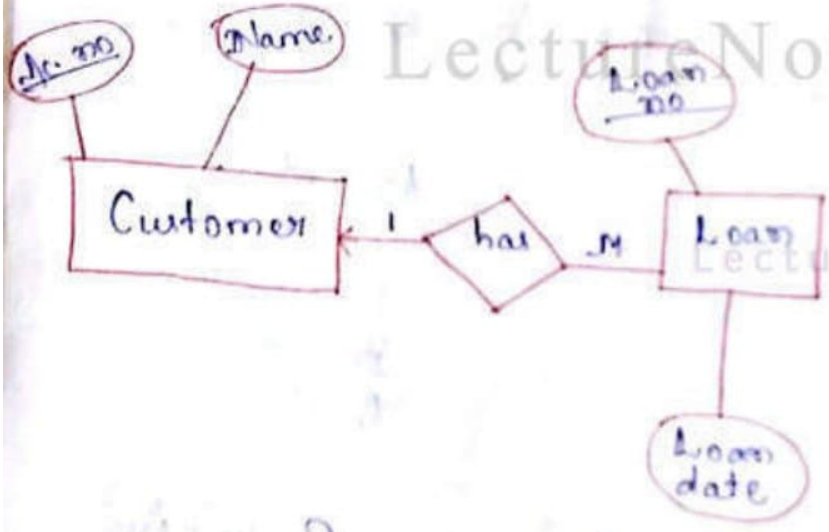
* Total:
Each higher level entity must belong to a lower level entity set.

* Partial:
Some higher level entities may not belong to any lower level entity set.



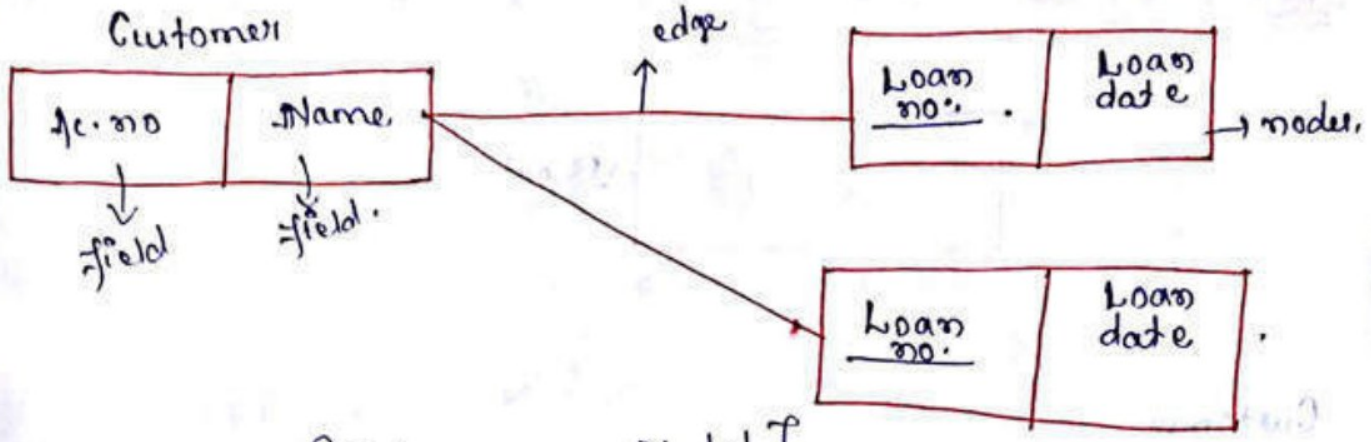
29/7/2015.

Network Model.



ER diagram

LectureNotes.in
Atul Kumar
7778888822



[Network Model].

A network database consists of collection of records connected to one another through links.

*- The record in many respects is similar to an entity in E-R model.

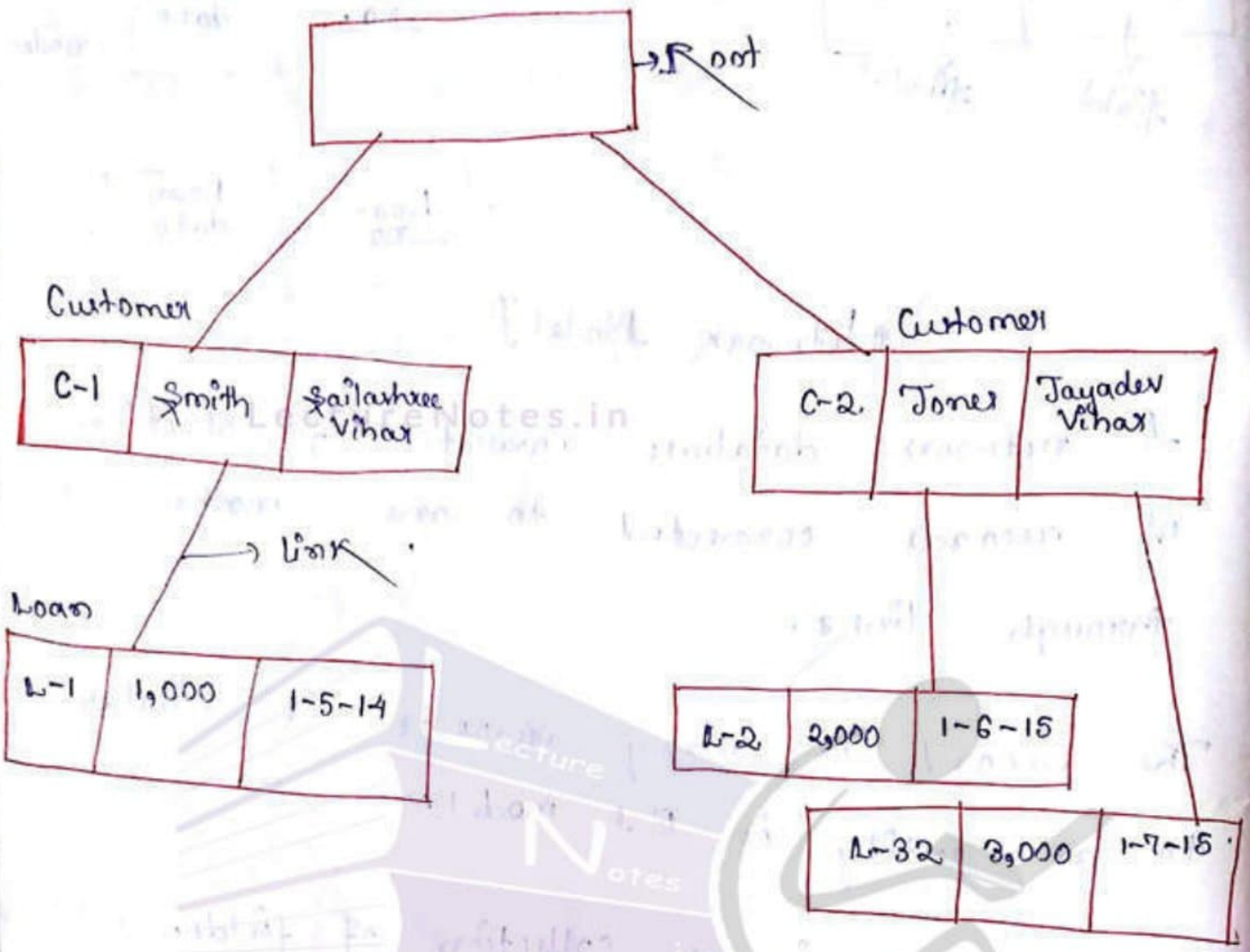
*- Each record is a collection of fields. Each of which contains only one data value.

*- A link is an association between precisely two records.

Disadvantages.

1. Querying is significantly more complicated.
2. It increases the burden on programmer for database design as well as data manipulation as compared to relational model.

Hierarchical Model :



A hierarchical database consists of collection of records that are connected to each other through links.

* Each record is a collection of field. Each of which contains only one data value.

* A link is an association between precisely two records.

Atul Kumar
7778888822

Disadvantages

1. Restriction to many to one links makes the design of database more complicated.
2. The lack of declarative querying facilities and need for navigation of pointer to access needed information make querying rather complex.

Relational Model

Given by E.F Codd.

Employee

attribute →

E.no.	Ename	Address	DOB
Cell			
D_1	D_2	D_3	D_4

D_1, D_2, D_3, D_4 : domains

A relation is a table with columns and rows.

It is based on the mathematical concept of relation which is represented physically as a table.

record → tupple.

Tupple

A tupple is a row of relation.

How to find the degree of a relation

The number of attributes containing a relation is called degree.

How to find cardinality of a relation.

No. of tupples contained in a relation is known as cardinality.

Properties of Relation.

1. The relation has name that is distinct from all the names in the relationship schema.
2. Each cell of the relation contains exactly one single value.
3. Each attribute has distinct names.
4. Each tupple is distinct. There are no duplicate tupples.
5. The order of attribute has no significance.
6. The order of tupple has no significance.

LectureNotes.in

Atul Kumar
7777888822

Keys : super key.
primary key.
candidate key.
alternate key.
composite key.
foreign key.

Foreign Keys :

Dept · primary key

D.no	D.name	D.location
10	MCA	x
20	CS	Y
30	IT	Z

Employee · foreign key

E.no	E.name	D.no
1	Smith	10
2	Jones	20
3	Peterson	10
4	Raj	30

```
Create table dept.
(Dno        number(2) primary key,
Dname       varchar2(3),
Dlocation   varchar2(1)) ;
```

```
select e.ename, d.dlocation from emp e, dept d
where e.dno = d.dno .
```

```
Create table Employee
(Eno number(3) primary key,
Ename varchar2(10),
Dno number(2), foreign key,
reference Dept(Dno)) .
```

An attribute or set of attributes within one relation that matches for candidate key of some (possibly the same) relation, is known as **foreign key**.

Employee

Eno	Ename	DOB	Employee manager emp. no (mgreno)
1	Smith	1-5-85	2
2	Jones	1-6-99	–
3	Gopal	6-9-15	2
4	Ram	8-5-14	3

: Select e.ename, s.ename from emp e, emp s where e.mgreno = s.eno;

Output:

Smith	Jones
Gopal	Jones
Ram	Gopal

LectureNotes.in

Atul Kumar
7778888822

3/8/2015.

Constraints in Relational Model.

1: Primary Key Constraint (Entity Integrity Constant).

Create table.

Employee(Eno number(2) primary key;
Ename varchar2(10),
Address varchar2(30));

Employee. (cannot have duplicate value).

Eno.	Ename	Address
1		
2		
3		

*- In a relation, no attribute of a primary key can be null.

(ii) Foreign/Referential Integrity, Entity Constraint.

Employee

Eno	Ename	Dno
1	Smith	10
2	Tony	20
3	Ram	10
4	Gopal	30
5	Sita	—

Dept.

Dno.	Dname	DLocation
10	CS	x
20	IT	y
30	MCA	z

Foreign key value must match to the primary key.

The domain or data type must match between primary key and foreign key.

Create table Employee
(Eno number(3),
Ename varchar2(10),
Dnumber number

Atul Kumar
7777888822

Dept (Dno));

*- If a foreign key exists in a relation, either the foreign key value must match a candidate key value of some tuple in its home relation or the foreign key value must be fully null.

*- The attribute in foreign key (R1) have same domain as primary key attribute of R2.

*- The value of foreign key in tuple (T1) of current state (r1) of (R1) either occurs as the value of primary key for some tuple (T2) in the current state of (R2) or is null.

r1 → relation state, R1 → Table.

(iii) Not Null Constraint

```
Create table Employee
(Eno number(3),
Ename varchar2(10) not null,
Address varchar2(30)) ;
Age number(2) check (age > 21));
```

4/8/2015.

(iv) Check Constraint.

Student

roll number (2)

mark number (3)

check (mark ≤ 100).

(v) Domain Constraint.

It specify that within each tuple, the value of each attribute must be an atomic value from the domain.

Codd's 12 Rule.

1. Information Rule.

*- All information is explicitly specifically & logically represented in tables as data values.

2. Rule of guranteed access.

*- Every item of data must be logically addressable with the help of a table name, primary key value and column name.

3. System treatment of null values.

*- The attributes must be able to support null values. (these values are different form zeroes).

to representing missing or inapplicable information.

4. The database description rule :

- A description of database is maintained using the logical structures with which data was defined by the RDBMS. These are accessible to the users with appropriate authority and store in the data dictionary.

5. Comprehensive data sub-languages :

- According to this rule, the system must support the following :
  - (i) Data definition
  - (ii) View definition.
    (Create view v,
    as select eno, ename from emp ;)
  - (iii) Data manipulation.
  - (iv) Integrity constraints.
  - (v) Authorisation.
  - (vi) Transaction management operation.

data control language.
DCL
(grant, revoke)

6. View Updating Rule.

- All views that are theoretically updatable, must also be updatable by the system.

7. Insert and update Rule.

*- This rule implies that all the data manipulation commands must be operational on a relation rather than on set of rows having on a single row.

8. Physical Independence Rule.

*- Application programs must remain unchanged when any changes are made in storage representation or access methods.

9. Logical Independence Rule.

*- The changes that are made should not affect the user's ability to work with the data.

*- The changes can be splitting the table into many more table.

10. Integrity Independence Rule.

*- The ~~and~~ integrity constraints should be stored in the system catalogue or in the database as a table.

11. Distribution.

*- The system must be able to access or manipulate the data that is distributed in other system.

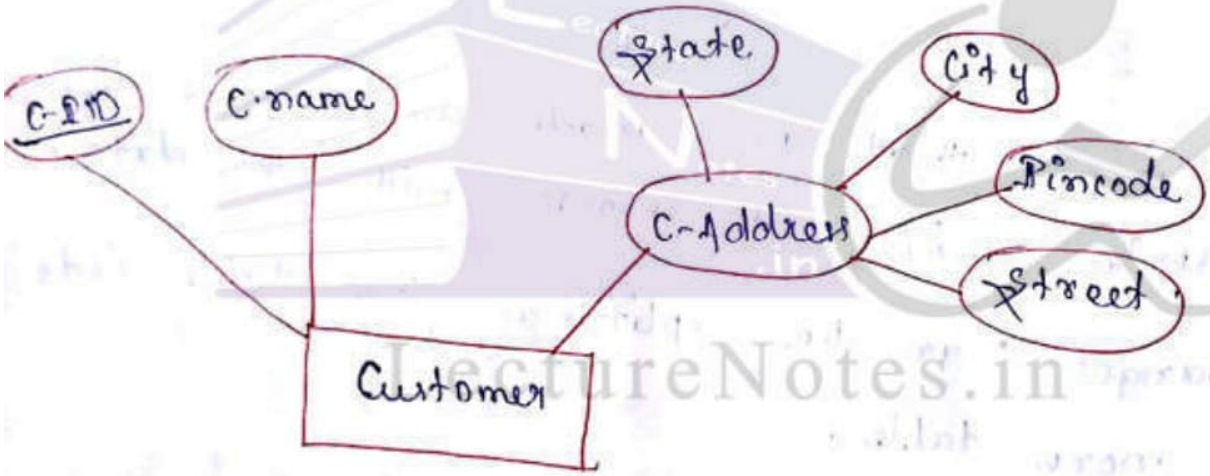
Non-Subversion Rule.

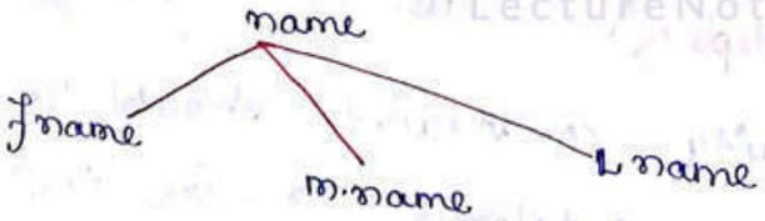
- It states that the different levels of language cannot bypass the integrity constraints.

5/8/2015.

Conversion of E-R Diagram to Tables.

Conversion of composite attributes to tables.





Atul Kumar
7777888822

Table :

Customer

C-ID	C-name	Street	City	State	Pincode

Create table :

```
Create table
Customer
(C-ID number(3) primary key,
C-name varchar2(10),
Street varchar2(10),
Pincode number(8));
```

2. Conversion of multi-value attribute :

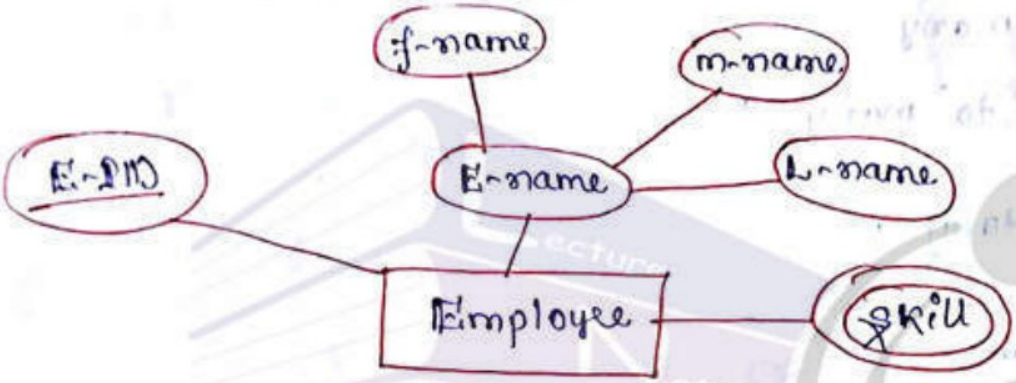


Table :

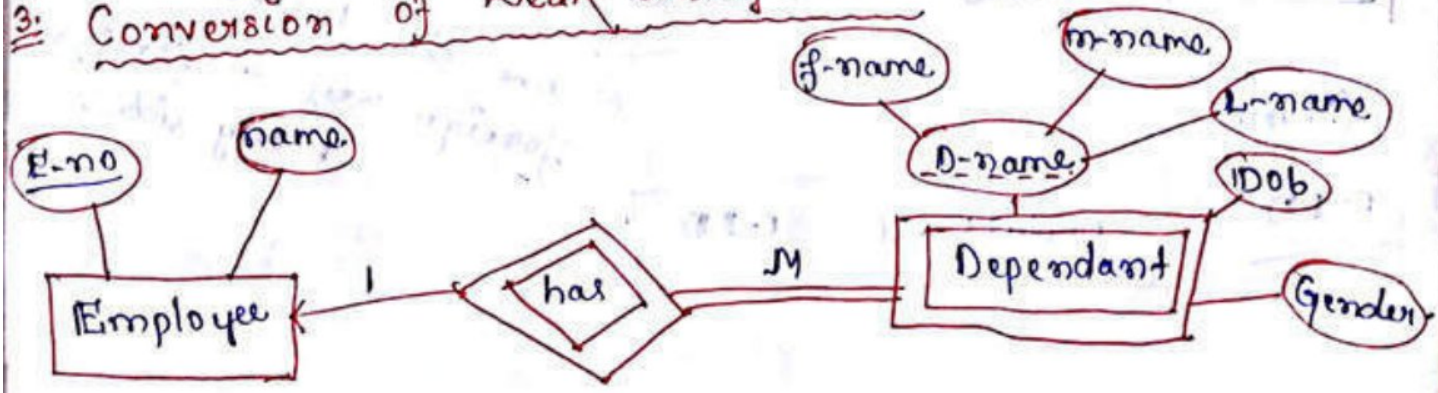
Employee

E-ID	fname	mname	Lname

Skill

E-ID	Skill

3. Conversion of weak entity set :



Employee

Eno	Ename

Dependent

fname	mname	lname	Eno	DOB	Gender

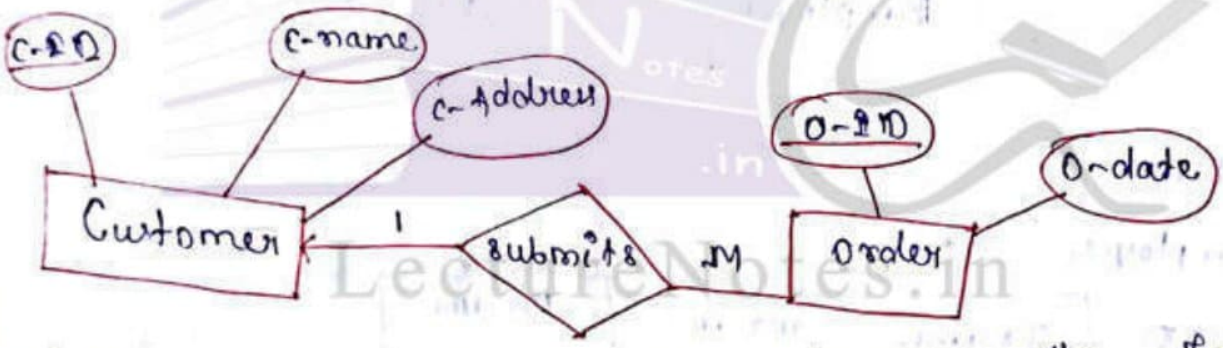
4. Conversion of binary relationship.

i) One to ~~many~~ one.

ii) Many to many.

iii) ~~Man~~ One to many.

i) One to Many.



*- Primary key of one side becomes the foreign key of many sides.

Customer

C-ID	C-name	C-address

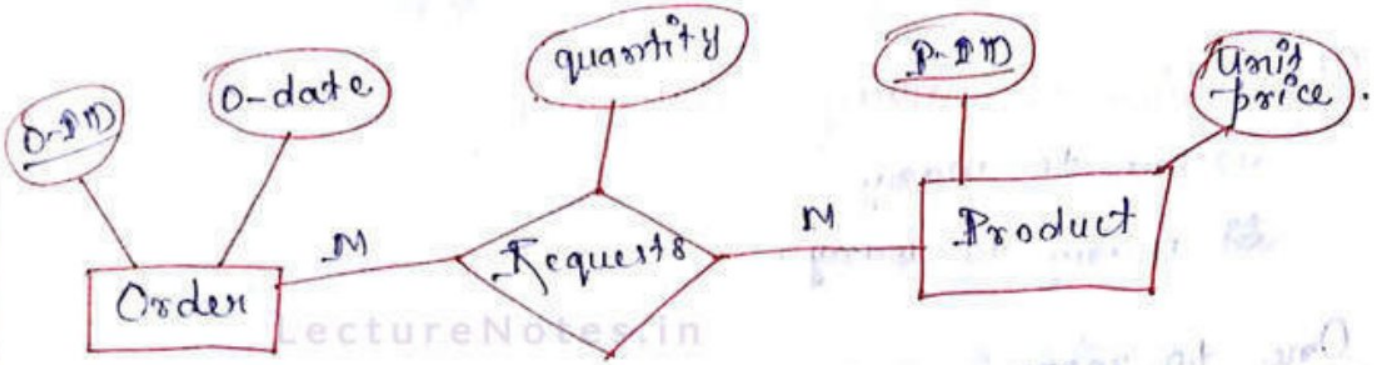
Order

O-ID	O-date	C-ID

Primary key of one side becomes foreign key of foreign many side.

Atul Kumar
7717888822

(ii) Many to many :



Order

O-ID	O-date

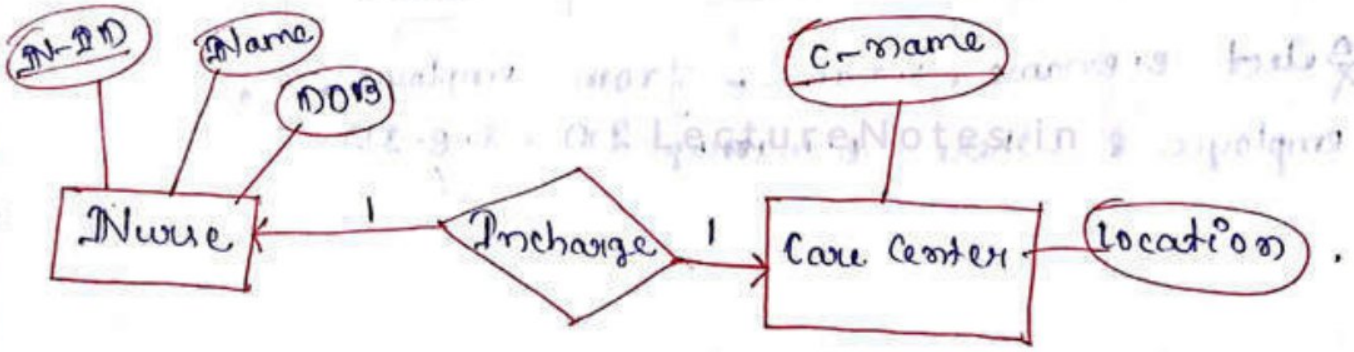
Product

P-ID	Unit price

Request

O-ID	P-ID	Request Quantity

(iii) One to One :



Nurse

N-ID	Name	DOB

Care Center

C-name	location	N-ID

LectureNotes.in

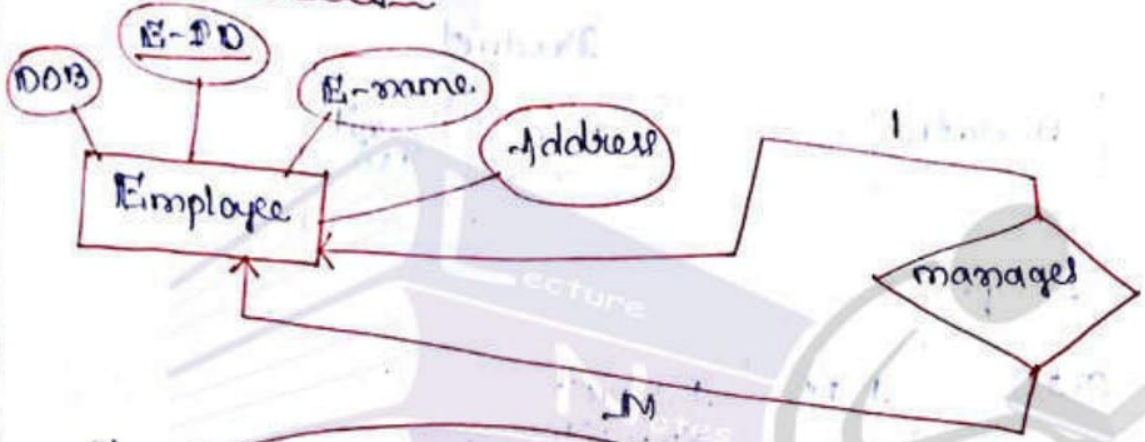
Select Nurse name from Nurse, Care center where C-name = "Apollo" and Nurse N-ID = Carecenter. N-ID.

5. Conversion of unary relationship.

i) one to many

ii) many to many.

i) One to many.



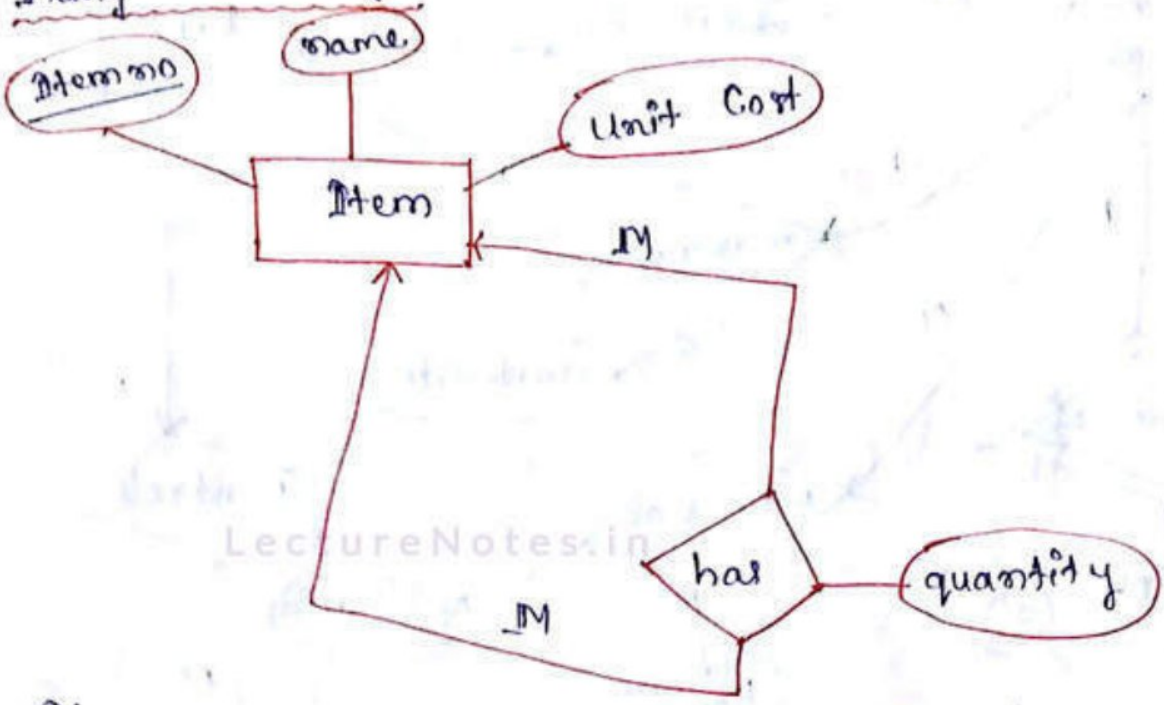
Employee

E-ID	Ename	Dob	Address	Manager ID

Select e.ename, s.name from employee e, employee s where e.manager ID = s.e-ID.

Atul Kumar
7777888822

ii) Many to many :



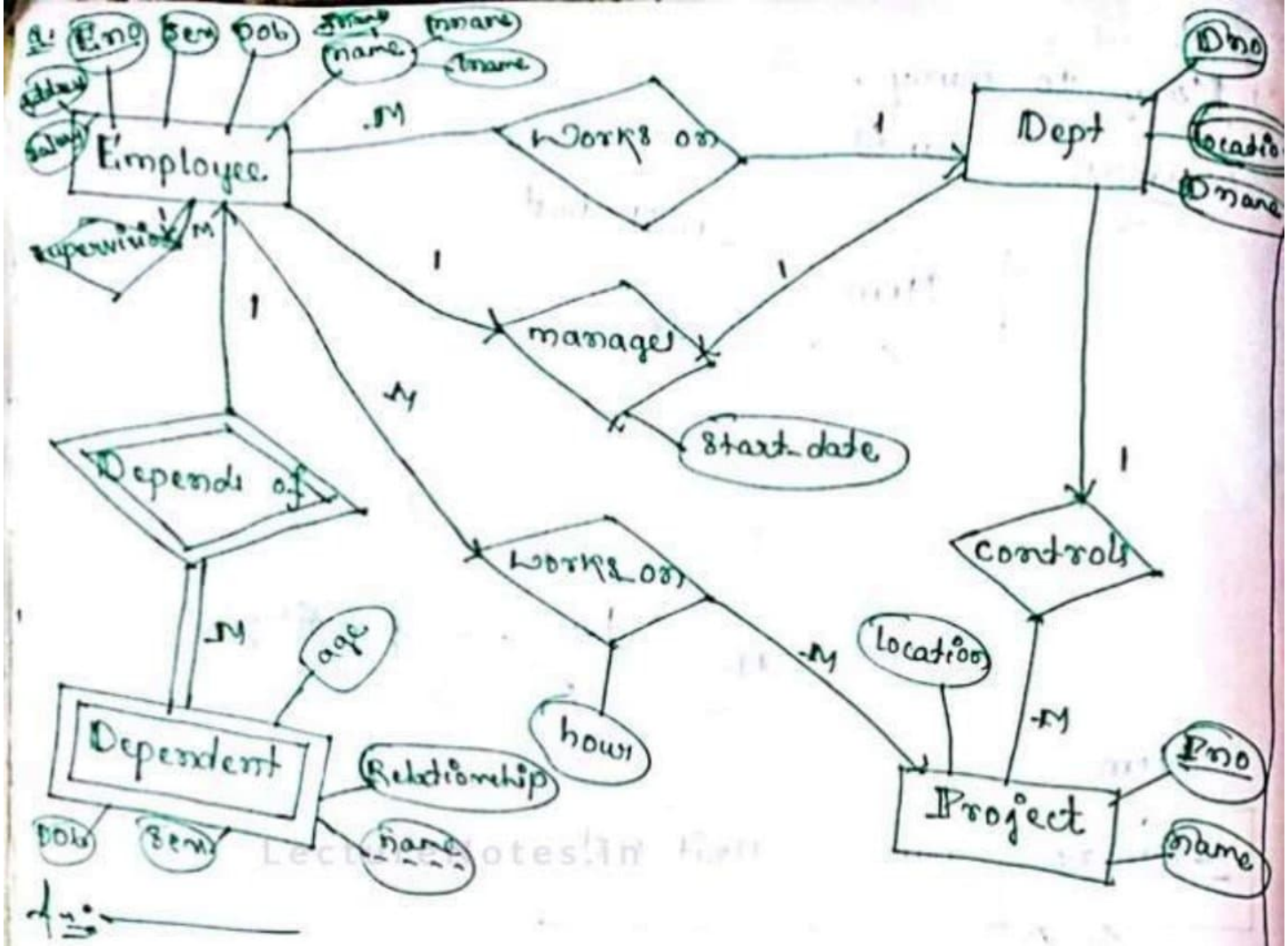
Item

Item no	name	Unit cost

Product

Item-no	Product-no

LectureNotes.in

Q: 

Ans:-

Employee

Eno	fname	mname	lname	Address	salary	sex	DOB	Dno
					Supervisor Eno			

Dept

Dno	Dname	Eno	start date

Dept-Location

Dno	location

Dependent

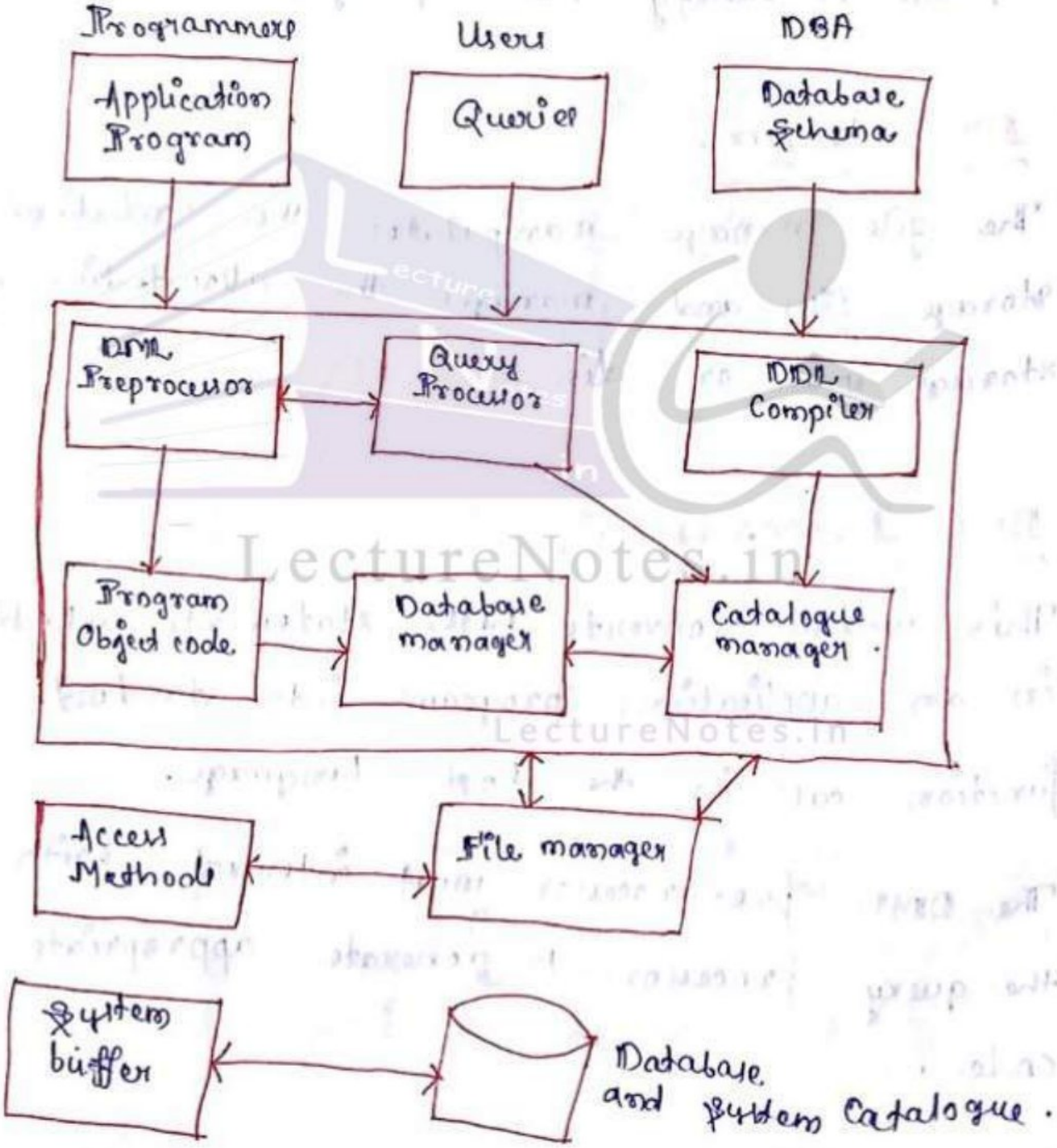
Dno	name	relationship	sex	dob	age

LectureNotes.in

Atul Kumar
7777888822

11/8/2015.

System Structure of DBMS.



(ii) Query Processor.

*- It transforms the queries into a series of low level instructions directed to the database manager.

(iii) Database Manager.

*- The database manager accepts queries and examines the external and conceptual schema to determine what conceptual records are required to satisfy the requerty.

(iii) File Manager.

*- The file manager manipulates the underlined storage files and manages the allocatedion of storage space or disc.

(iv) DML Pre-processor.

*- This module converts DML statements embedded in an application program into standard function call in the host language.

*- The DML pre-processor must interact with the query processor to generate appropriate

LectureNotes.in

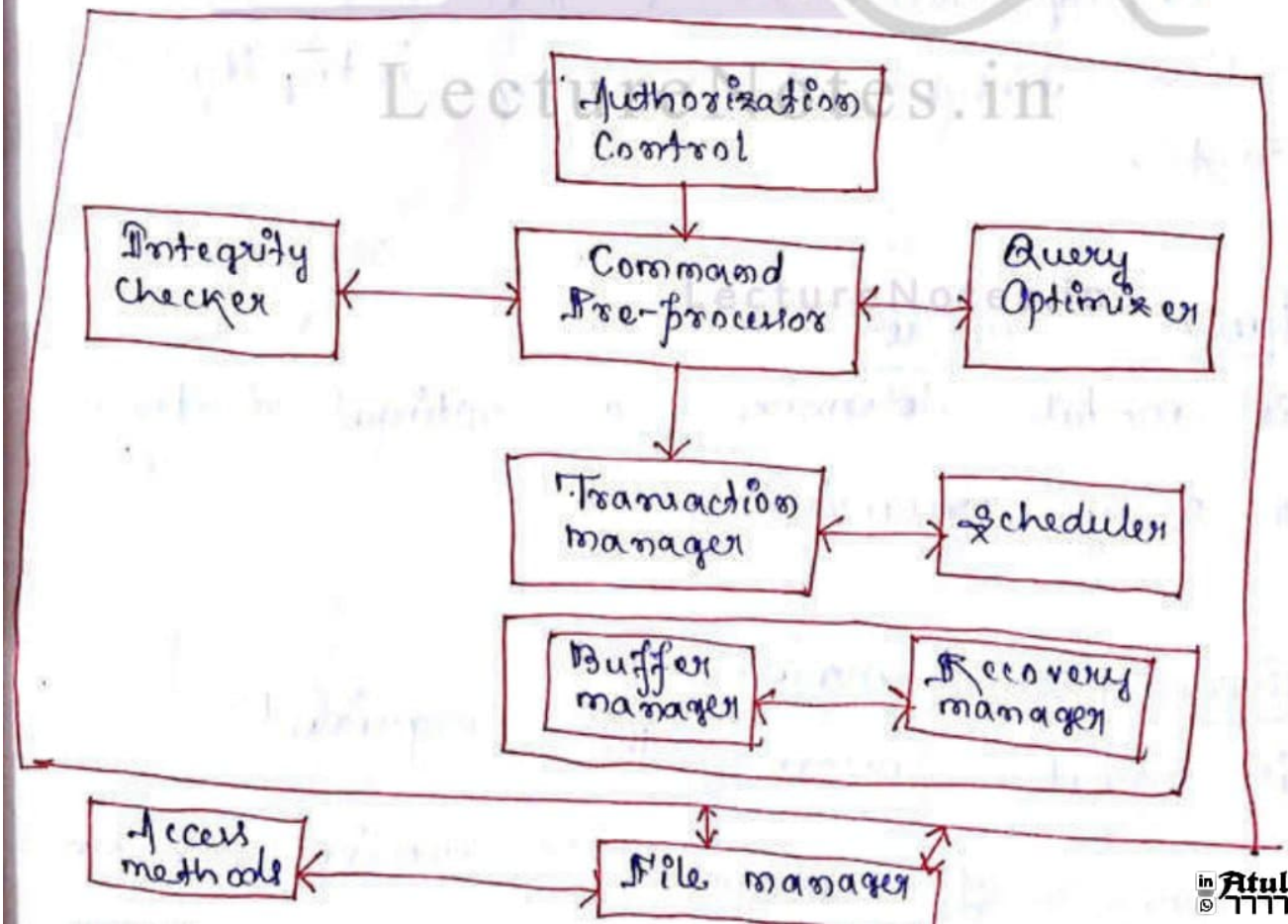
DDL Compiler.

- The DDL compiler converts DDL statements into a set of tables containing meta-data. These tables are stored in system catalogue.

Catalog Manager.

- The catalogue manager manages access to and maintains the system catalogue.
- The system catalogue is accessed by DBMS compiler.

Different Components of Database Manager.



Atul Kumar
7777888822

12/8/2015.

(i) Authorization Control :

*- This module checks that the ~~new~~ user has necessary authorisation to carry out required operation.

(ii) Command Pre-Processor :

*- Once the system has checked that the user has authority to carry out operation.

*- The control is passed to the command pre-processor.

(iii) Integrity Checker :

*- For an operation that changes the database, the integrity checker checks that the requested operation satisfys all necessary integrity constants.

(iv) Query Optimizer :

*- This module determines an optimal strategy for query execution.

(v) Transaction Manager :

*- This module performs the required processing of operation, it receives from

transaction.

vi) Scheduler.

*- This module is responsible for ensuring that concurrent operations on the database proceeds without conflicting with one another.

vii) Recovery Manager.

*- This module ensures that the database remains in a consistent state in presence of failure.

viii) Buffer Manager:

*- This module is responsible for the transfer of data between main memory and secondary storage such as disk.

*- The recovery manager and buffer manager are sometimes reffered to as data manager.

LectureNotes.in

Atul Kumar
7778888822

Query Language

Query.

A query is a statement requesting the retrieval of information.

Query Language.

A query language is a language in which user request information from the database.

- Procedural Query Language
  - Ex. Relational Algebra
- Non-Procedural Query Language.
  - Ex. Tuple Relational Calculus (TRC).
    Domain Relational Calculus (DRC).

Procedural Query Language.

*- The user instructs the system to perform sequence of operations on the database to compute the desired result.

Ex. Relational algebra.

Non-Procedural Query Language.

*- The user describes the desired information without giving a specific

procedure for obtaining that information.

Relational Algebra.

It is a procedure query language.

Operations of Relational Algebra.

- Selection (σ)
- Left outer join ($\bowtie$)
- Right outer join ($\bowtie$)
- Full outer join ($\bowtie$)
- Division ($\div$)
- Assignment ($\leftarrow$)

$=$, $>$, $<$, $>=$, $<=$, $\neq$, $!=$, $\wedge$ (and), $\vee$ (or)

- selection (σ)
- Projection (π)
- Cartesian product ($\times$)
- Union ($\cup$)
- Set difference ($-$)
- Intersection ($\cap$)
- Rename (ρ)
- Join ($\bowtie$)
- Natural join ($*$)

Selection (σ)

Syntax: $\sigma_P(R)$.

Used for selecting the tuple and rows.

In sql, select is used for selecting rows & columns, but here it is used for selecting rows.

Employee

Eno	name	Salary
1	Jones	1000
2	Smith	2000

Find all the details of the employee whose salary = 1000.

(select * from emp where salary = 1000;).

Ans:-

$$\sigma_{salary = 1000}(Emp).$$

$$\sigma_{salary > 1000 \wedge Eno = 2}(Emp)$$

$$\sigma_{salary > 1000 \wedge Ename = 'Smith'}(Emp).$$

iii) Projection operator (π).

Used to select the columns.

Syntax: $\pi_{A1, A2, \ldots, An}(R)$.

Find the employee name whose salary is 2000.

(select name from employee where salary = 2000;)

Ans:-

$$\pi_{ename}\left(\sigma_{salary = 2000}(Employee)\right).$$

Find employee number and salary of the employee whose ename = Jones.

$$\pi_{eno, salary}\left(\sigma_{ename = 'Jones'}(Employee)\right)$$

13/8/2015.

Cartesian Product (x).

R x S

R

A	B	C
a	b	c
d	e	f
g	h	i

S

D	E	F
J	K	L
m	n	o
p	q	r

A	B	C	D	E	F
a	b	c	J	K	L
a	b	c	m	n	o
a	b	c	p	q	r
d	e	f	J	K	L
d	e	f	m	n	o
d	e	f	m	n	o
g	h	i	p	q	L
g	h	i	m	n	o
g	h	i	p	q	r

Emp

Emp no	Ename	Dno
1	Smith	10
2	Jones	20
3	Blake	

Dept.

Dno	Dname	Dlocation
10	MCA	x
20	CS	y

Q. Find the employee working in MCA department.

(Select emp.ename from emp, Dept where emp.Dno = Dept.Dno and Dname = 'MCA';)

$\pi_{emp.ename}(\sigma_{emp.Dno = Dept.Dno \wedge Dept.Dname = 'MCA'}(emp \times dept))$.

Emp. no	Emp. ename	Emp. Dno.	Dept. Dno	Dept. name	Dept. Location
1	Smith	10	10	MCA	x
1	Smith	10	20	CS	y
1	Smith	10	30	IT	z
2	Jones	20	10	MCA	x
2	Jones	20	20	CS	y
2	Jones	20	30	IT	z
3	Blake	10	10	MCA	x
3	Blake	10	20	CS	y
3	Blake	10	30	IT	z
4	Peter	30	10	MCA	x
4	Peter	30	20	CS	y
4	Peter	30	30	IT	z

(Select * from emp, Dept where emp.Dno = Dept.Dno);

Gives which employee belongs to which department.

$\sigma_{emp.Do = Dept.Do}(emp \times dept)$.

$\sigma_{Dept.Dname = 'MCA'} (\sigma_{emp.Dno = Dept.Dno} (Emp \times Dept))$

(Select * from emp, dept where emp.Dno = Dept.Dname and Dept.dname = 'MCA' ;)

$\pi_{emp.ename} (\sigma_{Dept.Dname = 'MCA'} (\sigma_{emp.Dno = Dept.no} (emp \times Dept)))$.

Set Operations ($\cup, \cap, -$)

Union (R∪S)

*- For set operations to be valid, two conditions has to be satisfied that is called as union compatibility.

(i) The relation R and S must be of same arity. i.e., they must have same number of attributes.

(ii) The domain of ith attribute of R and the domain of ith attribute of S must be same for all i.

Borrower

Ename	Loan no.
Ha Ha Ha.... !!	Gift from your best friend. guess who?

Depositor.

Ename	Ac. no.
what is this?	why is this table blank ?? :P

Q- Find the employee name either they have taken the loan or they have an account in the bank or both.

A:-

(select cname from borrower union select cname from depositor;)

π_{cname} (Borrower) $\cup$ π_{cname} (depositor).

14/8/2015.

Intersection ($\cap$).

Borrower

cname	Loan no
Phir ek cir ?	Phir Kya cir ?

Depositor.

cname	Ac. no

Q. Find the customer name those who have taken the loan and have account in the bank.

A:—

π_{cname} (Borrower) $\cap$ π_{cname} (Depositor).

Set difference (−).

Q. Find the customer name those who have an account, but they have not taken any loan.

A:—

π_{cname} (Depositor) − π_{cname} (Borrower).

LectureNotes.in

(v) Rename (ρ).

Student

Roll no	Name	Mark
1	Smith	40
2	Jones	50
3	Blake	60

ρ_d (Student).

Roll no	Name	Mark
1	Smith	40
2	Jones	50
3	Blake	60

Q. Find the maximum mark in the class.

Student Roll	Student name	Student mark	d roll	d name	d mark
1	Smith	40	1	Smith	40
1	Smith	40	2	Jones	50
1	Smith	40	3	Blake	60
2	Jones	50	1	Smith	40
2	Jones	50	2	Jones	50
2	Jones	50	3	Blake	60
3	Blake	60	1	Smith	40
3	Blake	60	2	Jones	50
3	Blake	60	3	Blake	60.

- Projection : No duplicate values are allowed.

To rename attribute

$\rho_{a \to a'}$ (R).

Atul Kumar
7777888822

$\pi_{student.name,\ student.mark}\ (\sigma_{student.mark < d.mark}\ (student \times \rho_d(student)))$

For less mark.

$\pi_{mark}(student) - \pi_{student.mark}\ (\sigma_{student.mark < d.mark}\ (student \times \rho_d(student)))$

For less mark:

$\pi_{mark}(student) - \pi_{student.mark}\ (\sigma_{student.mark > d.mark}\ (student \times \rho_d(student)))$.

(vi) Assignment Operator.

$r_1 \leftarrow \pi_{cname}(Borrower) \cup r_2 \leftarrow \pi_{cname}(Depositor)$

$\Rightarrow r_1 \cup r_2$.

18/8/2015.

(vii) Join.

θ join
equi join
natural join
left outer join
right outer join
full outer join

$\sigma_{Emp.Dno = Dept.Dno}\ (Emp \times Dept) \equiv Emp \bowtie_{Emp.Dno = Dept.Dno} Dept$.

$\sigma_P (R \times S) \equiv R \bowtie_P S$, P → condition or predicate.

= symbol → equi join.

i) θ Join.

The θ join operation defines relation that contains tupples satisfying the predicate (P) from the cartesian product of R and S.

* The predicate (P) is the form of

$$\boxed{\underset{\text{relation}}{R} \cdot a_i \; \theta \; S \cdot b_i ,}$$

Where θ may be one of the comparison operator like <, ≤, >, ≥, =, ≠.

equi join is also a θ join.

ii) Equi Join

When predicate contains "=" symbol, it is known as equi join.

Emp

Eno	Ename	Dno
1	Smith	10
2	Jones	20
3	Blake	10
4	Peter	30

Dept

Dno	Dname	Location
10	MCA	x
20	IT	y
30	CS	z

Q: Find the employee name who belongs to MCA.

To match which employee belongs to which department.

$Emp \bowtie_{Emp.Dno = Dept.Dno} Dept.$

(select * from emp e, Dept d where e.dno = d.dno;)

Eno	Ename	Dno	Dno	Dname	Location
1	Smith	10	10	MCA ~~10~~	M
2	Blake	10	10	MCA ~~10~~	M
3	Jones	20	20	IT ~~20~~	Y
4	Peter	30	30	CS	Z

To match which employee belongs to MCA.

$\sigma_{Dname = 'MCA'} (Emp \bowtie_{Emp.Dno = Dept.no} Dept)$

To Know only name.

$\pi_{ename} (\sigma_{Dname = 'MCA'} (Emp \bowtie_{Emp.Dno = Dept.Dno} Dept))$

(select ename from emp e, Dept d where e.dno = d.dno and Dname = 'MCA')

π_{mark} (student) − $\pi_{student.mark}$ (~~σ student.mark < d.mark student.name~~ (student ⋈ student.mark < d.mark. ρ_d (student)));

For student ⋈ ρ_d (student)
student.mark < d.mark.

student roll no	student name	student mark	d. roll	d. name	d. mark
1	Smith	40	2	Jonel	50
1	Smith	40	3	Blake	60
1	Smith	40	4	Peter	70
2	Jonel	50	3	Blake	60
2	Jones	50	4	Peter	70
3	Blake	60	4	Peter	70

Natural Join (*) → common attribute is eliminated.

Emp * Dept
Emp.Dno
= Dept.Dno

The natural join is an equi join of the two relation R and S over all common attribute (X).

*- One occurrence of each common attribute is eliminated from the result.

Eno	Ename	Dno	Dname	Dlocation

selection, projection, rename → unary

19/8/2015

(iv) Left Outer Join (⟕)

matching + extra in left rel^n

R1

Ename	Street	City
Smith	x	BBSR
Jones	y	Cuttack
Ram	z	Puri

R2

Ename	Pincode	Lane number
Blake	751012	10
Ram	751014	20
Smith	751016	30

$(R1 * R2$
$R1.Ename = R2.Ename)$

Ename	Street	City	Pincode	Lanenumber
Smith	x	BBSR	751016	30
Ram	z	Puri	751014	20

natural join.

$R1 \bowtie_{R1.Ename = R2.Ename} R2$

Ename	Street	City	Pincode	Lane number
Smith	x	BBSR	751016	30
Ram	z	Puri	751014	20
Jones	y	Cuttack	null	null

(select * from R1, R2 where R1.Ename (+) = R2.Ename ;)

or (select * from R1 left outer join R2 on R1.ename = R2.ename ;)

c) Right Outer Join ($\bowtie$). match + extra in right rel^n.

$R1 \bowtie R2$
R1.ename = R2.ename.

Ename	Street	City	Pincode	Lane number
Smith	x	BBSR	751016	30
Ram	z	Puri	751014	20
Blake	null	null	751012	10

(Select * from R1, R2 where R1.Ename (+) = R2.Ename;)

or (select * from R1 right outer join R2 on R1.ename = R2.ename ;)

Full outer join.

match + extra of both.

R1 ⋈ R2

R1.ename
= R2.ename.

Ename	Street	City	Pincode	Lane number
Smith	x	BBSR	751016	30
Ram	z	Puri	751014	20
Jones	y	Cuttack	null	null
Blake	null	null	751012	10

(Select * from R1 full outer join R2 on R1.ename = R2.ename;)

Different Aggregate Functions.

max, min, count, average.

<grouping attribute> $\mathcal{F}$ (relation)

↓ script "f" or G

function name (attribute name).

Ex: $\mathcal{F}_{max(salary)}$ (Emp) → to find maximum salary.

(select max (sal) from emp;)

Deptno, max(salary) (Imp) ⟶ to find max salary for each department.

(select max(salary) from emp group by deptno;)

Q. T1

P	Q	R
10	a	5
15	b	8
25	a	6

T2

A	B	C
10	b	6
25	c	3
10	b	5

(a) $T_1 \bowtie T_2$

T1.P = T2.A.

P	Q	R	A	B	C
10	a	5	10	b	6
10	a	5	10	b	5
25	a	6	25	c	3

LectureNotes.in

(b) T1 ⋈ T2 .

T1.Q = T2.B .

P	Q	R	A	B	C
15	b	8	10	b	6
15	b	8	10	b	5

(c) T1 ⟕ T2 .

T1.P = T2.A .

P	Q	R	A	B	C
10	a	5	10	b	6
10	a	5	10	b	5
25	a	6	25	c	3
15	b	8	null	null	null

(d) T1 ⟖ T2

T1.Q = T2.B .

P	Q	R	A	B	C
15	b	8	10	b	6
15	b	8	10	b	5
null	null	null	25	c	3

LectureNotes.in

2) T1 ∪ T2 .

P	Q	R	A	B	C
10	a	5			
15	b	8			
25	a	6			
10	b	6			
25	c	3			
10	b	5			

3) T1 ⋈ T2

T1.P = T2.A

and T1.R = T2.C .

P	Q	R	A	B	C
10	a	5	10	b	5

LectureNotes.in

Generalised Projection :

Student

Roll no	Tmark	Msecured
1	30	6
2	30	7
3	30	8

$\pi_{Roll\ no,\ (Tmark - msecured)}$ Student as 'new_mark'.

Output :—

Roll no	New mark
1	24
2	23
3	22

This operation extends the projection operation by allowing arithmetic operation to be used in the project list.

Syntax :

$$\boxed{\pi_{f_1, f_2, \ldots, f(n)} (R)}$$

where R → relation

each $f_1, f_2, \ldots, f_n$ is the arithmetic expression.

every all → divisions operator

Q. Emp

Eno	fname	mname	Lname	Sal	Dob	Address	Dno	super Eno.

Dept

Dname	Dnumber	MgrEno	MgrDate

Project

Pname	Pnumber	Plocation	Dnum

Works_on

EEno	Pno	Hours.

Dependent

EEno	Dependent name	Sex	Dob	Relation-ship

(i) Find the name and address of all employee who works for MCA department.

(ii) For every project located in BBSR find the project no, controlling dept. no, dept manager, last name & dob.

(iii) Find the names of employee who works on all the project controlled by dept. no = 5.

(iv) List the names of all employees with 2 or more dependant.

(v) Retrieve the name of the employee who have no dependent.

(vi) List the name of managers who have atleast one dependant.

(i)

$$\pi_{fname,\ mname,\ lname,\ address}\left(\sigma_{Dname = \text{'MCA'}}\left(Emp \bowtie_{Emp.Dno = Dept.no} Dept\right)\right)$$

(ii)

$$\sigma_{Plocation = \text{'BBSR'}}\left(\left(Project \bowtie Dept\right) \bowtie Emp\right)$$

(iii)

$$\pi_{Emp.fname,\ Emp.mname,\ Emp.lname}\left(\sigma_{Dependant.name > 2}\left(Emp \bowtie_{Emp.Eno = Dependant.EEno} Dependant\right)\right)$$

(iv)

$$\pi_{Emp.fname,\ Emp.mname,\ Emp.lname}\left(\sigma_{Dependant.name = \text{'null'}}\left(Emp \bowtie_{Emp.Eno = Dependant.Eno} Dependant\right)\right)$$

(vi)

$$\pi_{Emp.fname,\ Emp.mname,\ Emp.lname}\left(\sigma_{Dependant.name \geq 1}\left(\sigma_{mgrsno}\left(Emp \bowtie_{Emp.Dno = Dept.Dno} Department\right) \bowtie_{Emp.Eno = Dependant.EEno} Dependant\right)\right)$$

LectureNotes.in

Atul Kumar
7777888822

21/8/2015.

1. $x_1 \leftarrow \pi_{Dnumber}(\sigma_{Dname = 'MCA'}(Dept))$.

$\pi_{fname, mname, lname, address}(Emp \bowtie_{Emp.Dno = x_1.Dnumber} x_1)$

LectureNotes.in

or.

$\pi_{fname, mname, lname, address}(\sigma_{Dname = 'MCA'}(Emp \bowtie_{Emp.Dno = Dept.Dno} Dept))$

2. $x_1 \leftarrow \pi_{Dnum, Pno}(\sigma_{Plocation = 'BBSR'}(Project))$

$x_2 \leftarrow \pi_{mgrsno, Dnum, Pno}(Dept \bowtie_{Dept.Dno = x_1.Dnum} x_1)$.

$\pi_{Pno, Dnum, lname, Dob}(Emp \bowtie_{Emp.eno = x_2.mgrsno} x_2)$.

LectureNotes.in

or. $x_1 \leftarrow \pi_{Pno, Dno, Mgrsno}(\sigma_{Plocation = 'BBSR'}(Project \bowtie_{Project.Dnum = Dept.Dnum} Dept))$

$\pi_{lname, dob, Pno, Dno, mgrsno}(Emp \bowtie_{Emp.Eno = x_1.mgrsno} x_1)$.

4. $r_1 \leftarrow {}_{EEno}\mathcal{F}_{\text{count (Dependent name) as no of dependents}}$

$r_2 \leftarrow \pi_{EEno}(\sigma_{\text{no of dependent} \geq 2}(r_1))$.

$\pi_{fname,\ mname,\ lname}(Emp \bowtie_{Emp.eno = r2.EEno} r_2)$.

5. $r_1 \leftarrow \pi_{Eno}(Emp) - \pi_{EEno}(Dependent)$.

$\pi_{fname,\ mname,\ lname}(Emp \bowtie_{Emp.Eno = r1.Eno} r_1)$.

6. $r_1 \leftarrow \pi_{EEno}(\pi_{EEno}(Dependent) \cap \pi_{mgreno}(Department))$

$\pi_{fname,\ mname,\ lname}(Emp \bowtie_{Emp.eno = r1.EEno} r_1)$.

LectureNotes.in

25/8/2015.

(iii) Division (÷).

Q. Find the customer name who have an account in all the SBI branch, BBSR.

Branch (bname, bcity, assets)

Depositor (cname, acno)

Account (acno, bname, balance).

Ans:

R

A	B
a_1	b_1
a_1	b_2
a_2	b_1
a_3	b_4
a_4	b_2
a_5	b_1
a_5	b_2

S

B
b_1
b_2

if

b_1
b_2
b_3

$R \div S$ = null.

$R \div S = a_1, a_5$.

Without ÷ use π X–

$T_1 \leftarrow \pi_A(R)$.

T_1

A
a_1
a_2
a_3
a_4
a_5

LectureNotes.in

$T_1 \times S$

A	B
a_1	b_1
a_1	b_2
a_2	b_1
a_2	b_2
a_3	b_1
a_3	b_2
a_4	b_1
a_4	b_2
a_5	b_1
a_5	b_2

$((T_1 \times S) - R)$

"—" present in A and in B

A	B
a_2	b_2
a_3	b_1
a_3	b_2
a_4	b_1

$T_2 \leftarrow \pi_A((\pi \times S) - R))$

T_2

a_2
a_3
a_4

$T_1 - T_2$.

a_1
a_5

LectureNotes.in

Branch

Bname	bcity	assets
DRC	BBSR	x
Infocity	BBSR	y
Link Road	CTC	z
CRP	BBSR	P

Depositor.

cname	Acno
Ram	A-101
Smith	A-103
Ram	A-105
Ram	A-104
Jones	A-102

Account.

Ac. no	Bname	Balance
A-101	DRC	10000
A-102	CRP	20000
A-103	Link road	30000
A-104	Infocity	40000
A-105	CRP	50000

Depositor ⋈ Account.

D. acno = Ac. acno

Acno	cname	bname	balance
A-101	Ram	DRC	10,000
A-102	Jones	CRP	20,000
A-103	Smith	Link road	30,000
A-104	Ram	Infocity	40,000
A-105	~~CRP~~ Ram	CRP	50,000

LectureNotes.in

$T_1 \leftarrow \pi_{bname, cname}$ (Depositor $\bowtie_{D.acno = Ac.ac.no}$ Account)

cname	bname
Ram	DRC
Jones	CRP
Smith	Lindrood
Ram	Infocity
Ram	CRP

$T_2 \leftarrow \pi_{bname}(\sigma_{bcity = 'BBSR'}(Branch))$

bname
DRC
Infocity
CRP

$T_1 \div T_2$

Ram

Delete Operation

delete one record or tupple.

Syntax : $\boxed{R \leftarrow R - E}$,

where R is the relation
E is the expression.

Q. Delete the Smith recorder from Depositor.

Depositor ← Depositor − σ_{cname} (Depositor) .

Insert Operation .

Syntax : $R \leftarrow R \cup E$

To insert into customer table :

Customer ← Customer ∪ {a-101, "Jones", 10,000} .

Customer

Acno	cname	Balance

Update Operation .

Syntax : $R \leftarrow \pi_{f_1, f_2, f_3, \ldots, f_n}(R)$.

where each f_i is the ith attribute of R .

Account ← $\pi_{acno, bname, balance + 100}$ (Account) .

Relational Calculus :

Tuple → row in a record.

i) Tuple relational calculus.

ii) Domain relational calculus.

This relational calculus, the name comes from a branch of symbolic logic called predicate calculus.

Predicate :

It is a truth value function with arguments.

*- When we substitute values for the arguments, the function yields an expression called a proposition. which can be either true or false.

(i) Tuple Relational Calculus :

It is based on tuple variable.

Tuple Variable :

A tuple variable is a variable that ranges over a name relation.

Syntax of TRC :

$\boxed{\{t \mid cond^n(t)\}}$, where t is a tuple variable

Employee

Eno	Ename	salary
1	Smith	10,000
2	Jones	20,000
3	Blake	30,000

Q. Find all the details of the employee whose salary = 10,000.

$$\left(\sigma_{salary=10,000} (\text{Employee}) \right)$$

(select * from employee where salary = 10,000;)

$\{t \mid employee(t) \text{ and } t.salary = 10,000\}$

or

$\wedge$

Q. Find the employee name whose salary < 20,000.

$\{t.ename \mid employee(t) \wedge t.salary < 20,000\}$.

For two variables

$\{t.eno, t.ename \mid employee(t) \text{ and } t.salary < 20,000\}$

* There are two types of quantifiers:

i) There exist ($\exists$)

ii) for all or universal quantifier ($\forall$).

Free Variable → used before bar (|).

If a variable is not quantified by ∃ and ∀ symbols, then it is called free variable.

Bound Variable → after |.

If a variable is quantified by ∃ or ∀ symbol, then it is called bound variable.

Employee

Eno	Ename	Salary	Dno	mgrno
1	Smith	10,000	10	2
2	Jones	20,000	20	null
3	Blake	30,000	10	2

Dept

Dno	Dname	Dlocation
10	MCA	x
20	IT	y
30	CS	z

$$\left(\pi_{ename} \left(\sigma_{Dname = 'MCA'} \left(Emp \bowtie_{Emp.Dno = Dept.Dno} Dept \right) \right) \right)$$

(Select e.ename from emp e inner join Dept d on e.dno = d.dno where d.dname = 'MCA';)

$$\left\{ e.ename \mid emp(e) \text{ and } (\exists d)\ Dept(d) \text{ and } e.dno = d.dno \text{ and } d.dname = 'MCA' \right\}$$

Q. Find the employee name and corresponding manager name.

select s.ename, t.ename from employee s, employee t where ~~employee s self join~~ ~~employee t~~ s.mgrno = t.eno;

$$\left(\pi_{\substack{\text{Emp.ename,} \\ \text{d.ename}}} \left(\text{Emp} \bowtie_{\substack{\text{emp.ename} \\ \text{emp.mgrno} \\ = \text{d.eno}}} f_d(\text{Emp}) \right) \right)$$

$$\{ t.\text{ename}, s.\text{ename} \mid \text{employee}(t) \text{ and } \text{employee}(s) \text{ and } t.\text{mgrno} = s.\text{eno} \}$$

existential quantifier, universal quantifier

Conversion of $\exists$ to $\forall$ and vice versa. 1/9/2015

1. $(\exists x)(F(x)) \equiv \sim(\forall x)(\sim F(x))$.

2. $(\forall x)(F(x)) \equiv \sim(\exists x)(\sim F(x))$.

3. $(\exists x)(F_1(x) \wedge F_2(x)) \equiv \sim(\forall x)(\sim F_1(x)) \vee (\sim(F_2(x))$.

4. $(\forall x)(F_1(x) \wedge F_2(x)) \equiv \sim(\exists x)(\sim F_1(x)) \vee \sim(F_2(x))$.

Q. Find the employee name those who have no dependant.

Employee

Eno	Ename	Address

Dependent

Eno	Dependent name	DOB	Relationship

Ans:

for dependent

$\{e.ename \mid employee(e) \wedge ((\exists d)\ Dependent(d)$ and $e.eno = d.eno)\}$

not dependent

$\{e.ename \mid employee(e) \wedge (\sim(\exists d)\ Dependent(d)$ and $e.eno = d.eno)\}$

$\{e.ename \mid employee(e) \wedge ((\forall d) \sim(Dependent(d)) \vee \sim(e.eno = d.eno))\}$

Q.

Borrower

Cname	Loan no

Depositor

Cname	Ac. no

Find the customer name those who have either a loan or both from the bank.

Atul Kumar
7777888822

A:

{e.cname, d.cname | Borrower(e) or Depositor(d)}.

Find the customer name those who have taken a loan and have account in bank.

A:

{e.cname, d.cname | Borrower(e) and Depositor(d)}.

Domain Relational Calculus (DRC).

declare variable for each attribute.

Syntax :

$$\boxed{\{x_1, x_2 \ldots x_n \mid \text{cond}^n(x_1 \ldots x_n)\}}$$

where $x_1, x_2, \ldots, x_n$ are domain variables that ranges over the domains.

Employee

a	b	c
Eno	Ename	Salary

Find all the details of employee whose Salary = 5000.

Ans:

$\{abc \mid employee(abc) \wedge c = 5000\}$

Find employee number, employee name whose salary < 10,000.

Ans:

$\{ab \mid employee(\exists c)\ employee(abc) \wedge c < 10{,}000\}$.

Find employee number whose salary > 5000.

$\{a \mid (\exists b)(\exists c)\ employee(abc) \wedge c > 5000\}$.

$(\exists b, c)$

Dept. a b c

Dno	Dname	Dlocation

Emp d e f g

Eno	Ename	Dno	Mgr eno

Find the employee who belongs to the MCA department.

Ans:

$\{e \mid (\exists a)(\exists b)(\exists f)\ dept(abc) \wedge emp(def) \wedge b = \text{'MCA'} \wedge (a = f)\}$.

Emp i j k

Eno	Ename	Dno	Mgr no

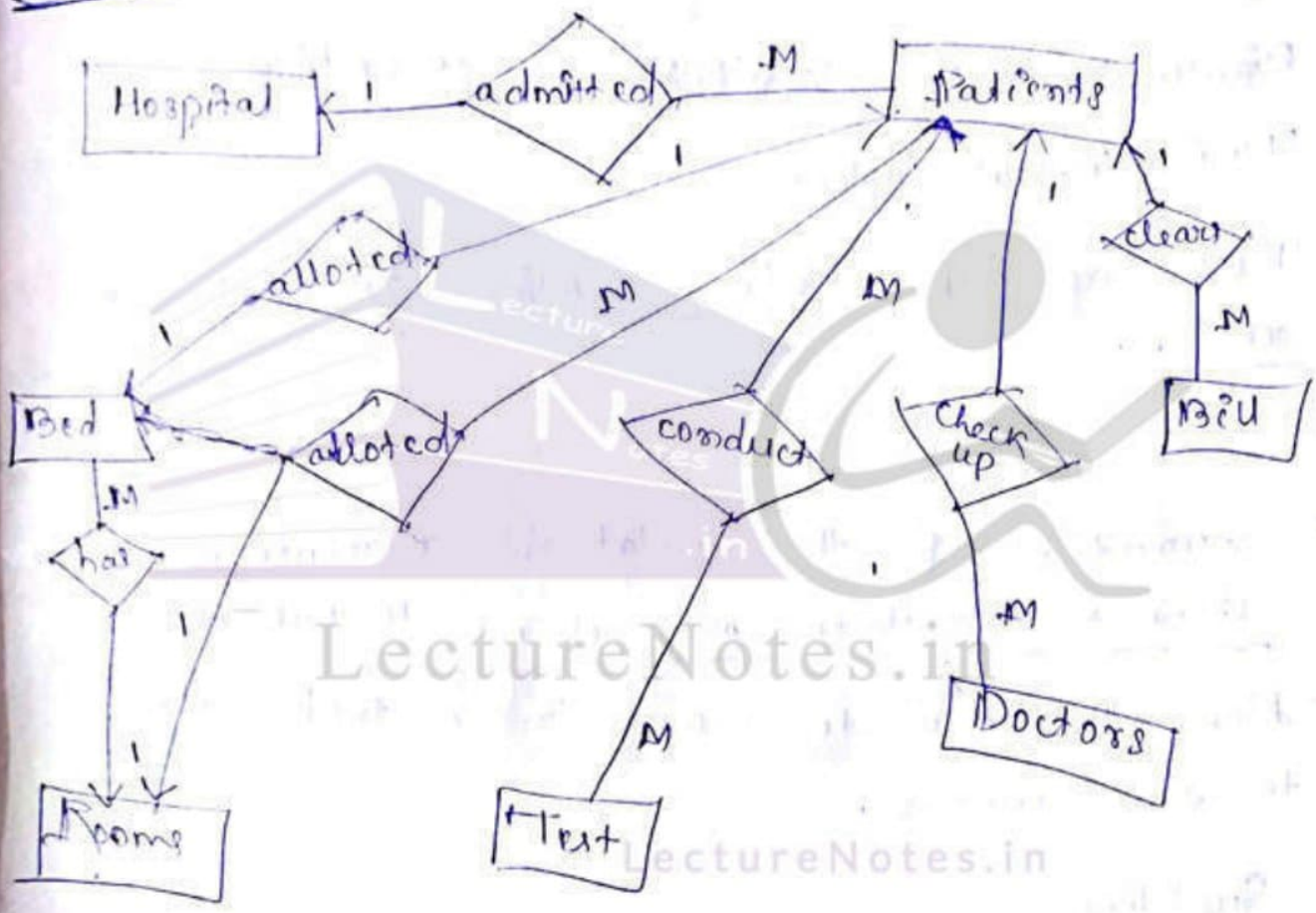
Q. Find employee name and corresponding manager name.

Ans:—

$\{e.x.a \mid (\exists a)(\exists f)\ dept(abc) \wedge emp(defg) \wedge$

$\{e\ i \mid (\exists g)(\exists h)\ emp(defg) \wedge emp(hijk) \wedge (g=h)\}$.

Internal.



LectureNotes.in

Atul Kumar
7777888822

Relational Database Design.

Rules.

1. Semantics of the relational attributes.
2. Reducing duplicate values in tuples.
3. Reducing the null values in tuples.
4. Disallowing the possibility of generating spurious (extra) tuples.

These are called design guidelines for relations schema.

1. Semantics of the relation attributes. (how the attributes are related to each other).

*- How the attribute values in a tuple related to one another.

Guidelines.

1. Design a relation schema so that it is easy to explain. Do not combine attributes from multiple entity set, relationship set into a single entity set.

2.

LectureNotes.in

2. Reducing duplicate values in tuples.

Emp- Dept

Eno	Ename	DOB	Dno	Dname	Dlocation
1	Akanksha	10/1/1994	10	CS	X
2	Ananya	19/5/1994	20	IT	Y
3	Arpita	11/4/1994	10	CS	X
4	Sumanda	2/5/1994	30	ET	Z
5	Sukhi	1/6/1994.	20	IT	Y.

Problems :

(i) insertion anamolies

(ii) Deletion anamolies

(iii) Updation anamolies

(i) Insertion Anomolies :

It may not possible to store certain information unless some other unrelated information is stored as well.

Ex: Suppose we want to insert information for a new department, but we cannot insert according to entity - entity constraint as primary key cannot be null.

(ii) Deletion Anamolies.

It may not possible to delete certain information without some other relation unrelated information as well.

(iii) Updation Anamolies.

If one copy of such repeated data is updated and inconsistency of created unless all copies are similarly updated.

Guidelines.

1. Design the schema so that no insertion, deletion or modification anamolies are present in the relation.

3. Reducing null values in tuples.

Problems.

(i) Wastage of space at storage level.

(ii) Difficult to understand the meaning of attributes with join operation.

(iii) Difficulty in applying aggregate function such as count, average, etc.

Guidelines

1. As far as possible avoid placing attribute in a base relation whose values may frequently be null.

4. Disallowing the possibility of generating spurious tuples.

Emp_Proj

Eno	Pno	Pname	Ename	Plocation
101	1	P_1	Ram	BBSR
104	1	P_1	Gopal	BBSR
101	2	P_2	Ram	Cuttack
102	3	P_3	Hari	Puri

R_1

Eno	Pno	Pname	Plocation
101	1	P_1	BBSR
104	1	P_1	BBSR
101	2	P_2	Cuttack
102	3	P_3	Puri

R_2

Ename	Plocation
Ram	BBSR
Gopal	BBSR
Ram	Cuttack
Hari	Puri

101 P

After natural join

R_1 * R_2

R_1.location

R_2.location

Eno	Pno	Pname	Ename	Plocation
101	1	P_1	Ram	BBSR
101	1	P_1	Ram	BBSR
104	1	P_1	Gopal	BBSR
104	1	P_1	Ram	BBSR
101	2	P_2	Gopal Ram	CTC
102	3	P_3	Hari	PURI

Using foreign key, & primary key

R_2 R_1

Eno	Pno	Ename
101	1	Ram
104	1	Gopal
101	2	Ram
102	3	Hari

Pno	Pname	Plocation
1	P_1	BBSR
2	P_2	Cuttack
3	P_3	Puri

(select * from R_1, R_2 where R_1.Pno = R_2.Pno),

Result : ——— 4 tuples.

(same question relation.

4/9/2015.

Guidelines:

Try to join the relation with the help of primary key and foreign key.

Functional Dependency (FD)

describes the relationship betn the attributes.

A determines B, A, B ⊂ R.
A → B

t1[A] = t2[A],
then t1[B] = t2[B].

R

	A	B	C	D
t_1				
t_2				

A Functional dependency (FD) is denoted by

$$A \xrightarrow{\text{determines}} B$$

between two sets of attributes A and B that are subset of relation (R) specifies a constraint on the possible tuples that can form a relation state r of R.

*- The constraint is that any tuples t_1 and t_2 in small r that have

$$t_1[A] = t_2[A],$$

they must also have

$$t_1[B] = t_2[B],$$

Ex: Eno, Pno → hour

*- A functional dependency can be trivial or non-trivial.

Condition for trivial functional Dependency.

A → B,

when B ⊂ A → trivial dependency.

{Eno, Ename} → Ename

Condition for non-trivial.

If B ⊄ A → non-trivial.

Q: Consider the following relation:

A	B	C
10	b1	c1
10	b2	c2
11	b4	c1
12	b3	c4
13	b1	c1
14	b3	c4

A → B
B → C ✓
C → B ✗
B → A
C → A.

LectureNotes.in

Closure of F and F+ .

set of functional dependency .

We can determine other functional dependencies from it .

Ex. $F = \{A \to B, B \to C\} \longrightarrow A \to C$.

$F^+ = \{A \to B, B \to C, A \to C\}$.

F is a set of functional dependancy . The set of all dependancies that include F as well as all the dependencies that can be inferred (derived) from F is called closure of F or F^+ .

Armstrong Axioms .

Basic rule (RAT) → Reflexive, augumentation, transitive .

1. Reflexive Rule .

If $x \supseteq y$, $\supseteq$ → superset .

then $x \to y$.

2. Augmentation Rule .

$\{x \to y\} \models \{xz \to yz\}$.

3. Transitive Rule.

$\{x \to y, y \to z\} \models \{x \to z\}$.

4. Decomposition Rule.

$\{x \to yz\} \models x \to y$

5. Union / Additive Rule.

$\{x \to y, x \to z\} \models x \to yz$.

6. Pseudo-transitive Rule.

$\{x \to y, wy \to z\} \models \{wx \to z\}$.

$x \to y$

By rule 2,

$\Rightarrow wx \to wy$

By rule 3,

$wy \to z$.

$\Rightarrow wx \to z$.

LectureNotes.in

3. As we know $x \to y$, $y \to z$ (rule 3) holds.

$\Rightarrow x \to z$ holds.

5. By rule 4,

$xz \to yz$

8/9/2015.

Attribute Closure.

Step-1:

Closure = x;

Step-2:

Repeat until there is no change

{

if there is an functional dependancy (FD) $U \rightarrow V$ in F (set of functional dependencies) such that

$U \subseteq$ Closure,

then set: Closure = Closure $\cup$ V

}

Q: F = {{Eno} → {Ename}

{pnumber} → {pname, plocation}

{Eno, pnumber} → {Hours}}

Find $\{Eno\}^+$, $\{pnumber\}^+$, $\{Eno, pnumber\}^+$.

Ans:

a) Eno = x. (Closure = Eno). U = Eno, V = Ename

$\{Eno\}^+ = \{Eno, Ename\}$. U $\subseteq$ Closure

Closure = {Eno, Ename}.

pnumber = x

$\{pnumber\}^+ =$

then, $\underset{U}{\{pnumber\}} \rightarrow \underset{V}{\{pname, plocation\}}$.

U $\subseteq$ closure.

b)

Closure = pnumber.

Closure = {pnumber, pname, plocation}.

{Eno, pnumber} $\subseteq$ closure.

$\{Eno\}^+ = \{Eno, Ename\}$.

$\{Pnumber\}^+ = \{pnumber, pname, plocation\}$.

$\{Eno, Pnumber\}^+ = \{Eno, Ename, pnumber, pname, plocation, hours\}$.

Q. Consider a relation scheme Emp_Dept and the following set of functional dependency (G)

$G = \{\{Eno\} \rightarrow \{Ename, dob, address, dnum\}$.

$\{dnum\} \rightarrow \{dname, dmgrsno,\}$.

Calculate the closure $\{Eno\}^+$ and $\{dnum\}^+$ with respect to G.

Ans:

$\{Eno\}^+ = \{Eno, Ename, dob, address, dnum, dname, dmgrsno\}$

$\{dnum\}^+ = \{dnum, dname, dmgrsno\}$.

Q. Consider the relation schema

$R = \{A, B, C, D, E, F, G, H, I, J\}$.

$F = \{AB \rightarrow C$

$A \rightarrow DE$

$B \rightarrow F$

$F \rightarrow GH$

$D \rightarrow IJ\}$

AB^+, A^+, B^+, F^+, D^+.

Atul Kumar
7778888822

Ex:- key

$(AB)^+ = \{A, B, C, D, E, F, G, H, I, J\}$

$A^+ = \{A, D, E, I, J\}$

$B^+ = \{B, F, G, H\}$

$F^+ = \{F, G, H\}$

$D^+ = \{D, I, J\}$.

Equivalence of functional dependency.

Given, $F = \{ \quad \}$, $G = \{ \quad \}$

Prove $F^+ = G^+$.

By finding F cover G

G cover G.

Procedure to find F cover G.

We can determine F cover G by determining α^+ w.r.t F for each functional dependancy α determines γ $(\alpha \rightarrow \gamma)$ in G, then checking whether this α^+ includes the attributes in γ. If this is the case for every functional dependency, then F cover G.

We can determine F and G are equivalent by checking F cover G and G cover F.

Q. $F = \{A \to C, AC \to D, E \to AD, E \to DH\}$.

$G = \{A \to C, E \to AH\}$. Find $\{A\}^+$ w.r.t F. Then, check y of G is present in y of A^+ is present or not.

To prove $F^+ = G^+$.

A :-

F covers G.

$\{A\}^+ = \{A, \underline{C}, \underline{D}\}$ by G.

Then, check y is in F is in G's y, $F^+ = G^+$.

$\{AC\}^+ = \{A, C, D\}$.

Equivalent.

$\{E\}^+ = \{E, \underline{A}, \underline{H}, C, D\}$. (equivalent).

G covers F.

$\{A\}^+ = \{A, \underline{C}, D\}$. (equivalent).

$\{E\}^+ = \{E, A, D, C, H\}$ (equivalent).

~~G covers F.~~

$\{AC\}^+ = \{A, C, \underline{D}\}$

$\{E\}^+ = \{E, \underline{A}, \underline{H}, C, \underline{D}\}$.

G covers F.

Thus, $F^+ = G^+$.

LectureNotes.in

9/9/2015.

Minimal Set of Functional Dependencies.

A minimal cover of a set of functional dependencies E is a minimal set of dependencies (F) that is equivalent to E.

Algorithm.

1. Set F = E

2. Replace each FD by $x \rightarrow \{A_1, A_2, \ldots, A_n\}$ in F by the 'n' functional dependencies of the type
$x \rightarrow A_1, x \rightarrow A_2, \ldots, x \rightarrow A_n$.

3. For each FD, $x \rightarrow A$ in F,
~~$x \rightarrow A$~~
in F; for ~~each~~ ~~attr~~
for each attribute B that is an element of x, if
$\{\{F - \{x \rightarrow A\}\} \cup \{\{x - \{B\}\} \rightarrow A\}\}$ is equivalent to F,
then replace $x \rightarrow A$ with $(x - \{B\}) \rightarrow A$ in F.

4. For each remaining functional dependency $x \rightarrow A$ in F,
if $\{F - \{x \rightarrow A\}\}$ is equivalent to F,
then remove $x \rightarrow A$ from F.

LectureNotes.in

$AB \to C$ $\quad$ $B \to$ extraneous $\quad$ $A \to B$

$A \to C$ $\quad$ $B \to C$

$A \to C$ → can be removed

(2).

Ex. $E = \{x \to ABC$, $P \to EID\}$ $\longrightarrow$ $x \to A$, $x \to B$, $x \to C$, $P \to E$, $P \to D$.

(3)

Ex. $F = \{\underset{x}{AB} \to \underset{A}{C}, C \to D\}$.

$F - \{x \to A\}$. (remove one attribute of x)

After this $\{A \to C, C \to D\}$

$G = \{C \to D, A \to C\}$.

Show equivalent (F covers G, G covers F).

If not equi, remove the other attribute and see.

LectureNotes.in

* After (3), remove any FD of F,

$F = \{A \to C, C \to D\}$.

After removing let $C \to D$ (any can remove).

$G = \{A \to C\}$.

Then show $F = G$.

Then, remove that FD from F.

$A \to B$
$B \to C$
$A \to D$

$E = \{A \to BC$
$B \to C$
$AB \to C$
$AC \to D\}$.

$E = \{A \to BC, B \to C, AB \to C, AC \to D\}$.

$E = \{A \to B, A \to C, B \to C, \cancel{A \to C, B \to C}, \cancel{A \to D, C \to D}, AB \to C, AC \to D\}$.

$= \{A \to B, A \to C, B \to C, A \to D, \cancel{C \to D}, AB \to C\}$.

$= \{A \to B, A \to C, B \to C, AB \to C, AC \to D\}$.

$E = \{A \to B, A \to C, B \to C, \cancel{A \to D, C \to D, A \to C}, AB \to C, AC \to D\}$.

$G = \{A \to B, A \to C, B \to C, A \to C, AC \to D\}$.

E COVERS G

$\{A^+\} = \{A, B, C, D\}$.

$\{B^+\} = \{B, C\}$.

$\{C^+\} = \{A, C, B, D\}$.

G COVERS E

$\{A^+\} = \{A, B, C, D\}$.

$\{B^+\} = \{B, \underline{C}\}$.

$\{AB^+\} = \{A, B, \underline{C}, D\}$.

$\{AC^+\} = \{A, C, \underline{D}\}$.

$E = \{A \to B, A \to C, B \to C, AB \to C, AC \to D\}$.

$G = \{A \to B, A \to C, B \to C, AB \to C, A \to D\}$.

E COVERS G.

$\{A^+\} = \{A, \underline{B}, \underline{C}, D\}$.

$\{B^+\} = \{B, \underline{C}\}$.

$\{AB^+\} = \{A, B, \underline{C}, D\}$.

LectureNotes.in

G covers E

$\{A^+\} = \{A, B, C\}$

$\{B^+\} = \{B\}$

1. $E = \{A \to BC$

$B \to C$

$AB \to C$

$AC \to D\}$.

2. $E = \{A \to B$

$A \to C$

$B \to C$

$AB \to C$

~~$B \to C$~~.

$AC \to D$

~~$B \to D$~~ $\}$.

remove one att. from left hand side.

3. $G = \{A \to B$

~~$A \to C$~~

$B \to C$

$A \to C$

$AC \to D\}$.

Remove B.

G covers E.

$\{A^+\} = \{A, \underline{B}, \underline{C}, \underline{D}\}$.

$\{B^+\} = \{B, \underline{C}\}$.

$\{AB^+\} = \{A, B, \underline{C}, D\}$.

$\{AC^+\} = \{A, C, B, \underline{D}\}$.

E covers G.

$\{A^+\} = \{A, \underline{B}, \underline{C}, D\}$.

$\{B^+\} = \{B, \underline{C}\}$.

LectureNotes.in

$\{A^+\} = \{A, B, C, D\}$

(equ^n)

Replace $AB \to C$ with $A \to C$.

$E = \{A \to B$
$A \to C$
$B \to C$
$A \to C$
$AC \to D\}$

LectureNotes.in

$E_1 = \{A \to B$
$A \to C$
$B \to C$
$AC \to D\}$

Remove A

$E_1 = \{A \to B$
$B \to C$
$A \to C$
$C \to D\}$

E contains G

LectureNotes.in

$\{A^+\} = \{A, B, C, D\}$

$\{B^+\} = \{B, C\}$

$\{C^+\} = \{C\}$ (not equal)

LectureNotes.in

G_1 contains E.

$\{A^+\} = \{A, B, C, D\}$

$\{B^+\} = \{B, C, D\}$

$\{C^+\} = \{C, D\}$

So, remove C.

$E = \{A \to B, A \to C, B \to C, A \to C, AC \to D\}$ $\Rightarrow$ $E = \{A \to B, A \to C, B \to C, AC \to D\}$.

$G = \{A \to B, A \to C, B \to C, A \to D\}$.

G cover E.

$\{A^+\} = \{A, \underline{B}, \underline{C}\}$.

$\{B^+\} = \{B, \underline{C}\}$.

$\{AC^+\} = \{A, C, B, \underline{D}\}$.

E cover G.

$\{A^+\} = \{A, \underline{B}, \underline{C}, D\}$.

$\{B^+\} = \{B, \underline{C}\}$.

~~$\{A^+\} =$~~.

equivalent.

$E = \{A \to B, A \to C, B \to C, A \to D\}$.

4. Remove $B \to C$.

$G = \{A \to B, A \to C, \cancel{B \to C}, A \to D\}$.

LectureNotes.in

Atul Kumar
7777888822

G cover E.

$\{A^+\} = \{A, B, C, D\}$.

$\{B^+\} = \{B\}$. (not present).

So, remove $A \to C$.

$G = \{A \to B$
$B \to C$
$A \to D\}$.

G cover E.

$\{A^+\} = \{A, B, C, D\}$.

$\{B^+\} = \{B, C\}$.

E cover G.

$\{A^+\} = \{A, B, C, D\}$.

$\{B^+\} = \{B, C\}$.

equivalent.

So, $E = \{A \to B$
~~$A \to C$~~
$B \to C$.
$A \to D\}$.

Another method.

$E = \{A \to BC$
$B \to C$
$AB \to C$
$AC \to D\}$.

$G = \{A \to B$
$A \to C$
$B \to C$
$AB \to C$

LectureNotes.in

By augumentation rule,

$$G = \{ A \to B,\ A \to C,\ AA \to AC \quad \text{— (2)},\ A \to AC,\ AC \to D \quad \text{— (3)},\ A \to D,\ B \to C,\ AB \to C \}.$$

As $A \to D$ is there, so remove $AC \to D$,

$$G = \{ A \to B,\ A \to C,\ B \to C,\ A \to D,\ AB \to C \}.$$

As, $A \to B$
$B \to C$
$A \to C$

So, remove $A \to C$,

$$G = \{ A \to B,\ AA \to AB,\ A \to AB,\ AB \to C,\ A \to C \}.$$

So, remove $AB \to C$.

$\Rightarrow$ $G = \{ A \to B,\ B \to C,\ A \to D \}.$

LectureNotes.in

Normalization. → to remove duplicate values.

It is the process of organising the attributes and tables of a relational database to minimise data redundancy, with the help of functional dependencies and keys.

Techniques to ~~seem~~ implement normalization :

(i) 1 Normalization form (NF).
(ii) 2NF
(iii) 3NF
(iv) BCNF (Boyce and Codd NF).

} depends on functional dependency

(v) 4NF → based on multi-valued dependency
(vi) 5NF → based on join dependency (Project join normal form (PJNF)).

10/9/2015.

1st normalize then decomposition, then check these two properties :
(i) dependency preservation
(ii) lossless join preservation property (non-additive join).

(i) Dependency Preservation.

It ensures that each FD is represented in some individual resulting relation after decomposition.

iii) Loss Less join Property.

It guarantees that the spurious tuple generation problem does not occur with respect to the relation schema created after decomposition.

4. 1 NF. (a relation is already in 1NF).

The only attribute values permitted by 1NF are single atomic values.

Dept.

Dname	Dnumber	Dmgrno	Dlocation
CS	10	1	L_1, L_2
IT	20	2	L_3
MCA	30	3	L_4

→ not in 1NF.

1st Technique to normalize.

R_1

Dnumber	Dlocation
10	L_1
10	L_2
20	L_3
30	L_4

R_2

Dnumber	Dname	Dmgrno
10	CS	1
20	IT	2
30	MCA	3

LectureNotes.in

2nd Technique.

R3

Dname	Dnumber	Dmgrno	Dlocation
CS	10	1	L1
CS	10	1	L2
IT	20	2	L3
MCA	30	3	L4

3rd Technique

R4

Dname	Dnumber	Dmgrno	Dlocation1	Dlocation2
CS	10	1	L1	L2
IT	20	2	L3	Null
MCA	30	3	L4	Null.

2. 2NF.

2NF is based on fully functional dependency.

Prime Attribute.

If an attribute is the part of a key, then that attribute is known as prime attribute.

Non-Prime Attribute.

If an attribute is not the part of a key, then that attribute is known as non-prime attribute.

LectureNotes.in

Eg. Roll name, address, dob salary

↓ candidate Key. candidate key

So, name is a prime attribute.
address is a prime attribute.
roll is a prime attribute.

LectureNotes.in

11/9/2015.

Fully Functional Dependency.

A functional dependency $x \to y$ is full functional dependency if removal of any attribute from x means that the dependency doesn't hold any more.

Partial Dependency.

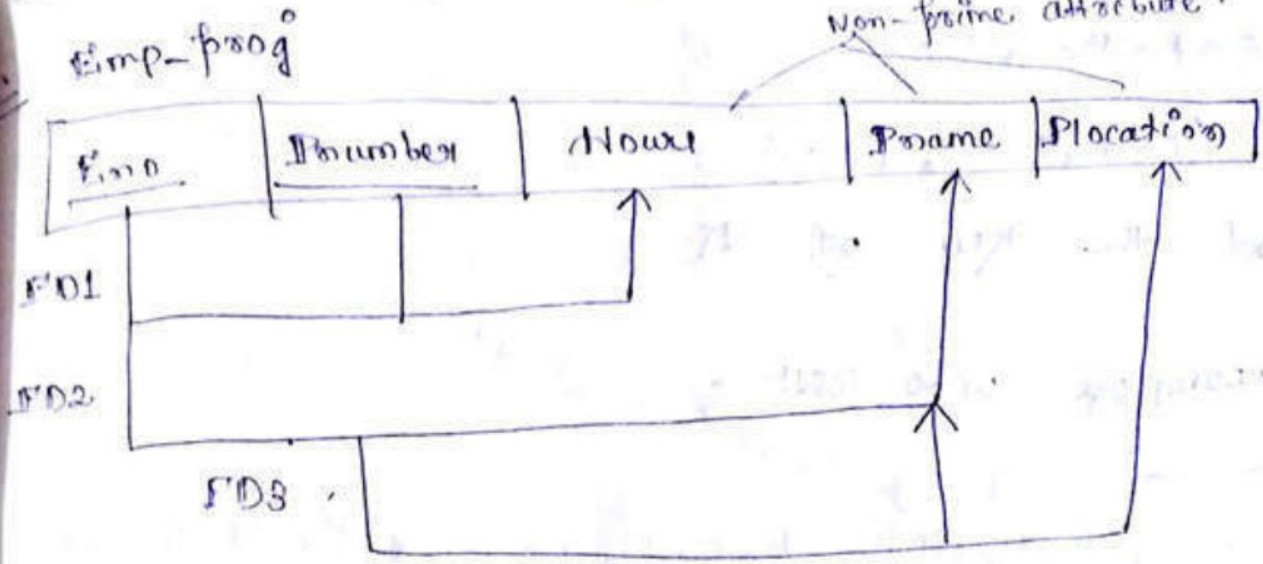
A functional dependency $x \to y$ is partial dependency if some attribute $a \in x$ can be removed from x and the dependency still holds.

LectureNotes.in

2NF.

A relation schema (R) is in second normal form if every non-prime attribute (A) in R is not partially dependent on key of R.

Emp-proj



Check whether the relation is in 2NF.

Hours is fully dependent on Eno, Pnumber.

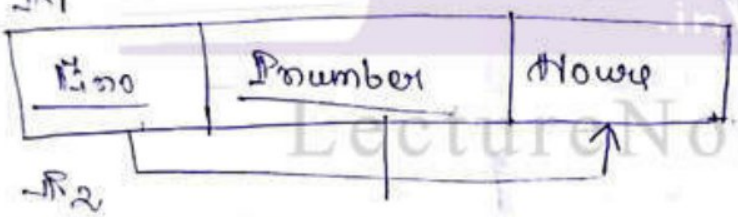
Pname is not fully dependent as it is only dependent of Eno.

Plocation is not fully dependent.

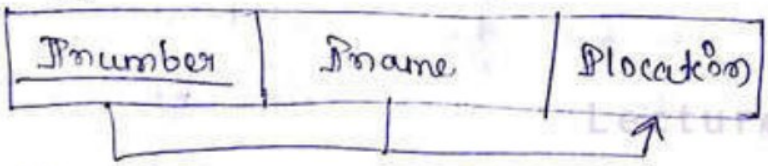
Then, the relation is not in 2NF.

We need to decompose.

R_1



R_2



R_3

LectureNotes.in

Ex: $R = \{A, B, C, D\}$.

$F = \{C \to A, B \to C, C \to D\}$.

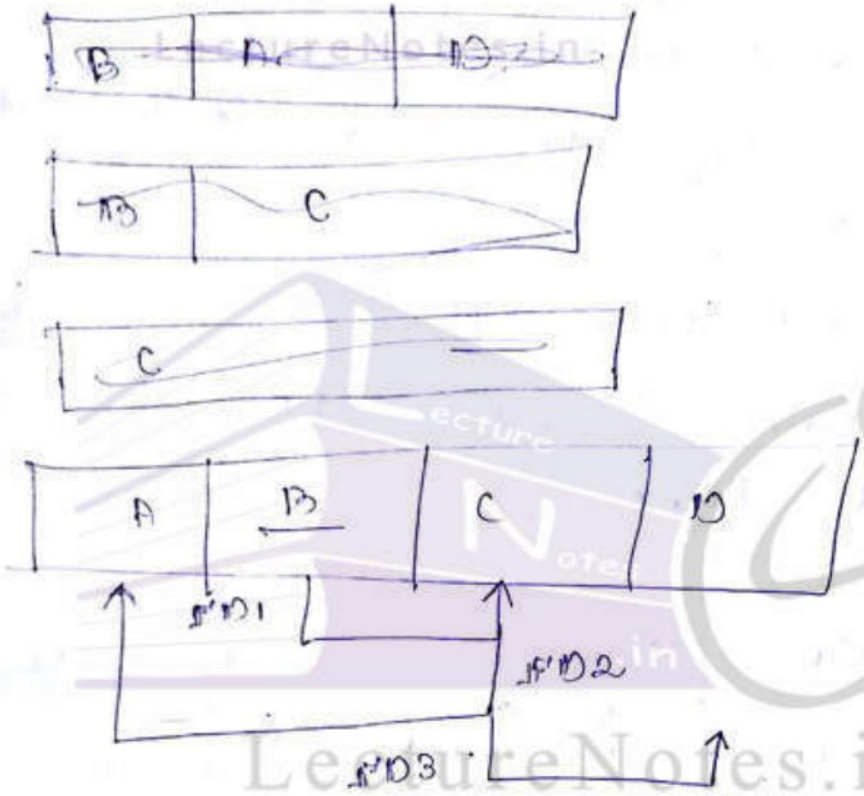
(i) Find the key of R.

(ii) Decompose upto 2NF.

Ans:—

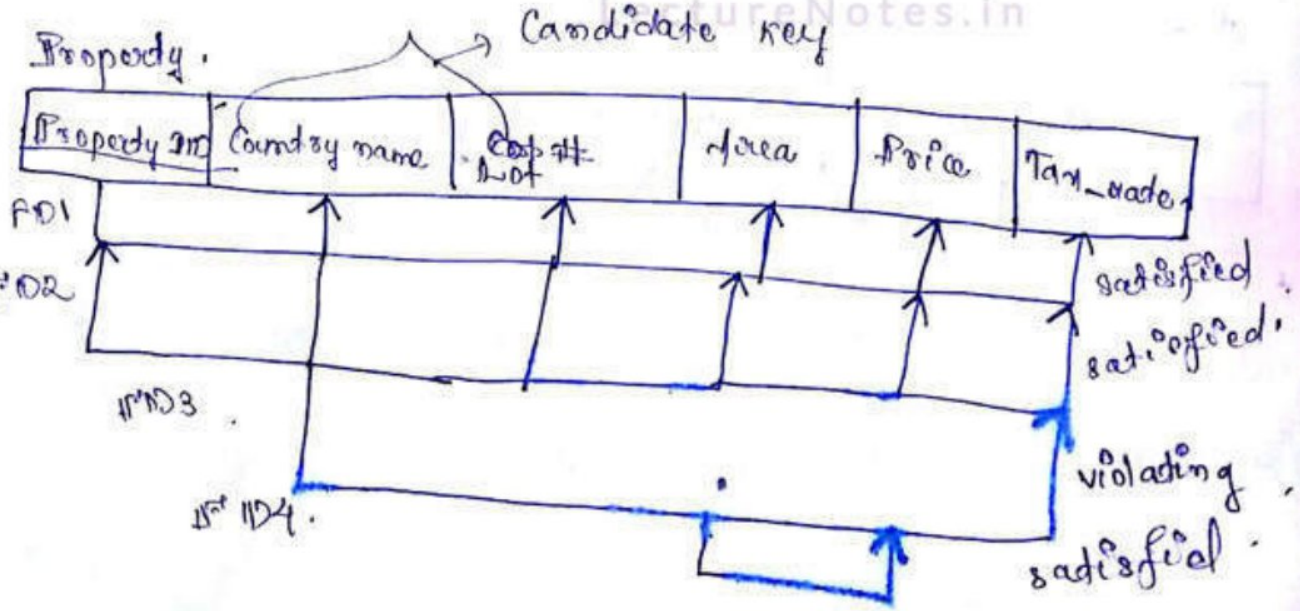
(i) B. ($\because \{B\}^+ = \{B, C, D, A\}$, $\{BC\}^+ = \{B, C, D, A\}$, selected (minimal super key))

(ii)



satisfied 2NF (as the non-prime attributes A, C, D depends on non-prime attributes).

Q.



R2

Countryname	Tax rate

R1

PropertyID	Country name	Lot #	Area	Price

14/9/2015.

3. 3NF.

* 3NF is based on transitivity property.

A relation schema (R) is in 3NF if it satisfies 2NF and no non-prime attribute of R is transitively dependent on primary key.

Ex

Ename	Eno	Dob	Dnumber	Dname	Dmgrsno

x y z

$x \to y$, $y \to z$ $\Rightarrow$ $x \to z$.

(non-prime attributes depends on transitive property).

R1

Ename	Eno	Dob	Dnumber

R2

Dnumber	Dname	Dmgrsno

Generalised Definition of 3NF.

A relation schema (R) is in 3NF if whenever a non-trivial functional dependency

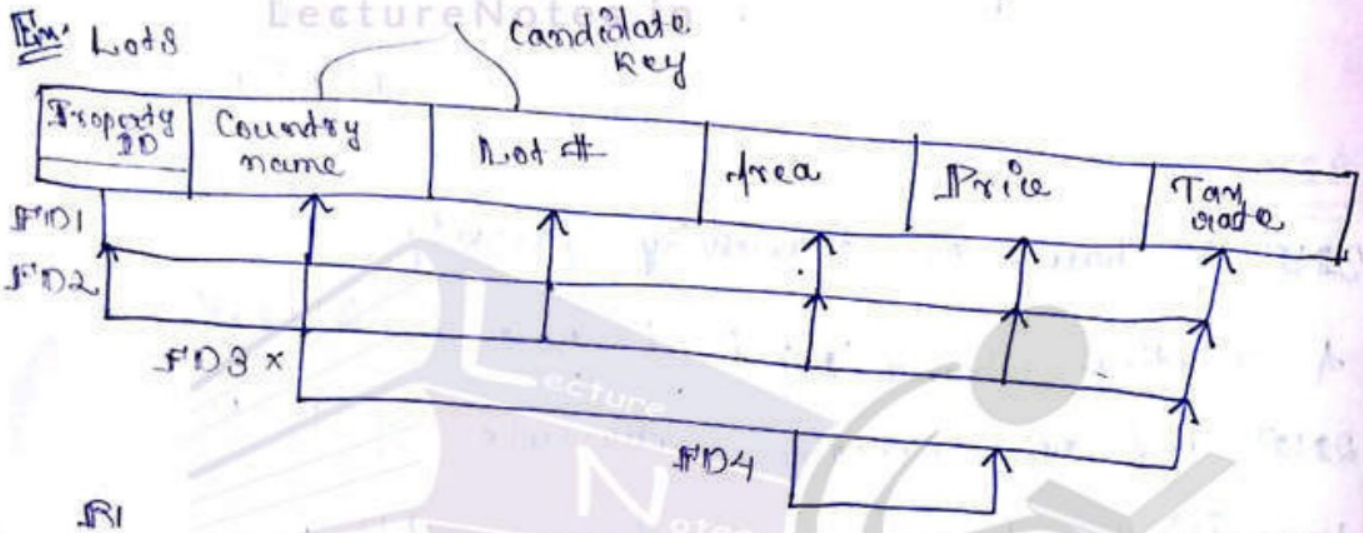
$X \rightarrow A$

holds in R either

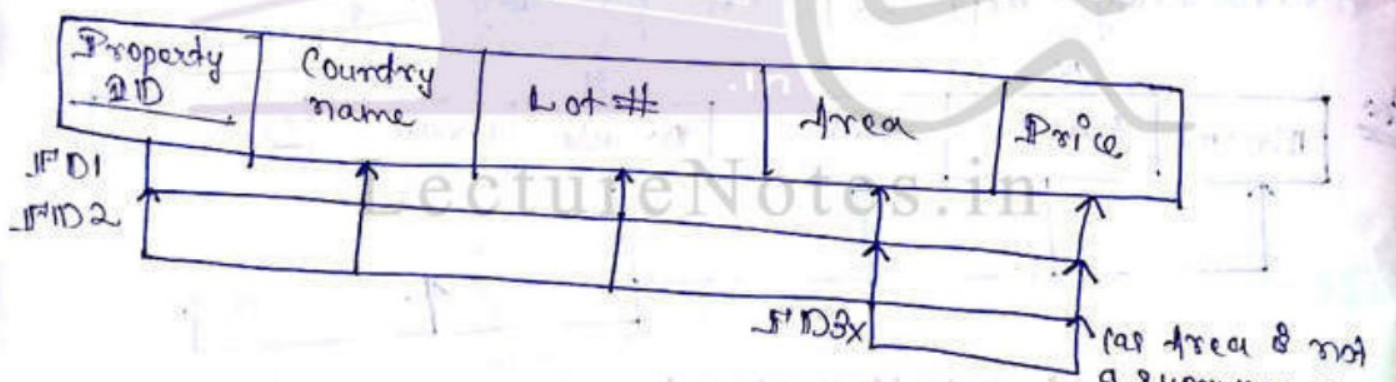
(i) X is a super key of R.

(ii) A is a prime attribute of R.

Ex: Lots

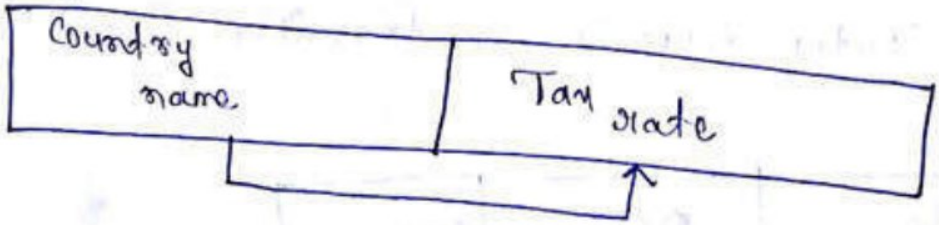


R1



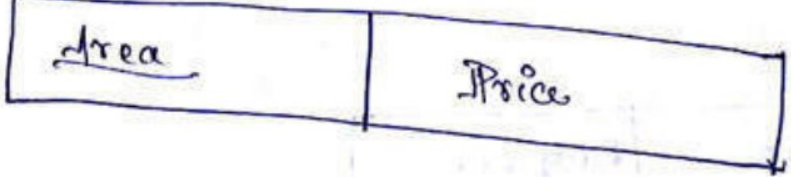
(as Area is not a super key & price is not prime)

R2



Further decomposition :

R2



R_1

Property ID	Country name	Lot #	Area

R_3

Country name	Tax rate.

Ex: R is a relation schema containing attributes A to J. having FD

$F = \{AB \to C$
$A \to DE$
$B \to F$
$F \to GH$
$D \to IJ\}$.

(a) Find the key of R.
(b) Decompose upto 3NF.

Ans:

$\{AB^+\} = \{A, B, C, D, E, F, G, H, I, J\}$.

$\{A^+\} = \{A, D, E, I, J\}$.

$\{B^+\} = \{B, F, G, H\}$,

Key is $\{AB^+\}$.

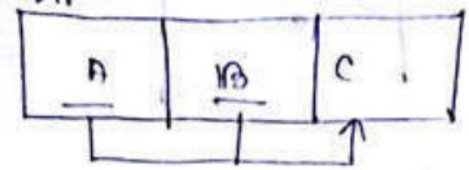
A	B	C	D	E	F	G	H	I	J

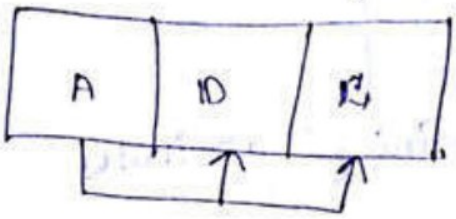
FD1
FD2 × (can only depend on A)
FD3 ×
FD4
FD5

LectureNotes.in

2nd NF.

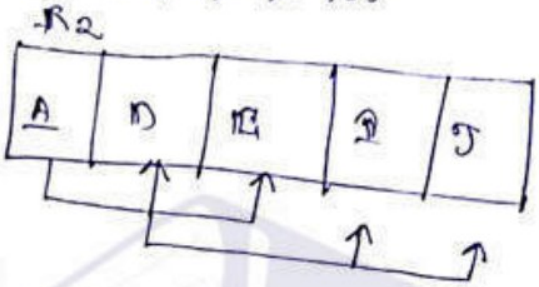
R1

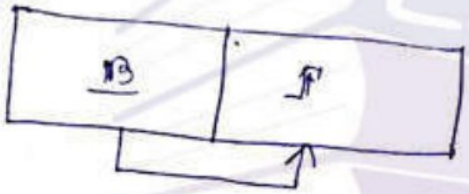




As $\langle A^+ \rangle = \langle A, D, E, I, J \rangle$

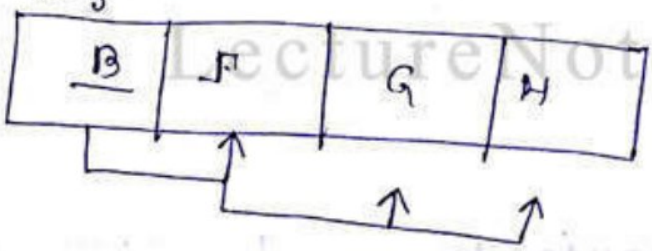
So, R2





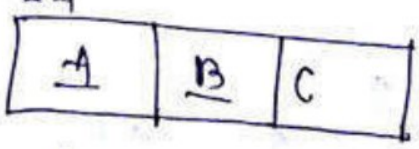
As $\langle B^+ \rangle = \langle B, F, G, H \rangle$.

So, R3

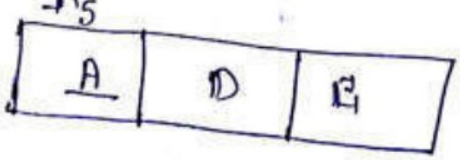


3NF.

R4



R5



LectureNotes.in

R_6

D	D	J

R_7

B	F

R_8

F	G	H

Ex: R is a relation containing attributes A to J.

F.D. F = {AB→C
BD→EF
AD→GH
A→I
H→J}

(i) Find the key of R.

(ii) Decompose upto 3NF.

AB^+ = A, B, C

BDEF

AD I, G, H, J

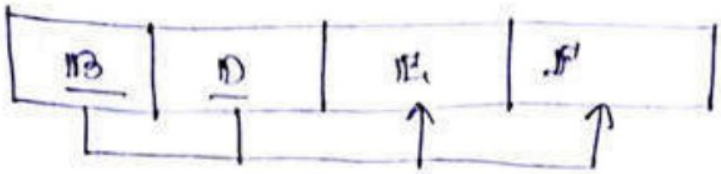
Ans:

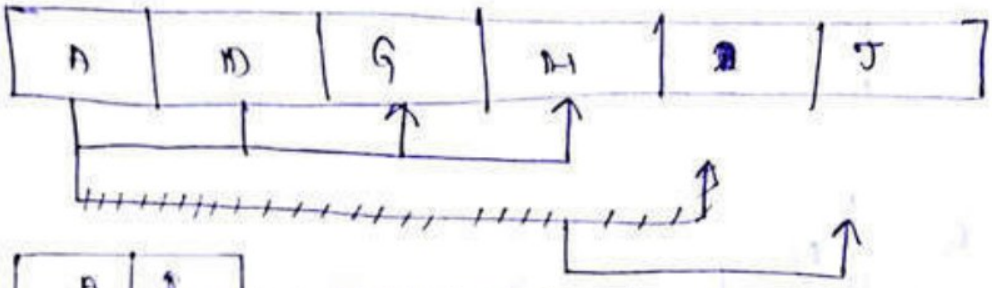
$\{ABD^+\}$ = {A, B, D, I, C, E, F, G, H, J}

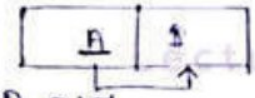
A	B	C	D	E	F	G	H	I	J

2nd NF.

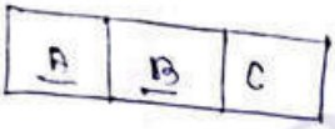
A	B	C

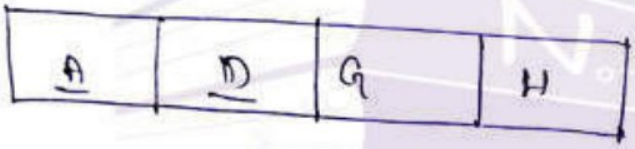


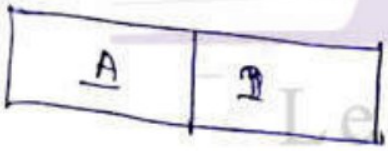


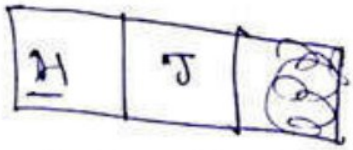


3 NF.









* After decomposition if any of the relation is not containing the key, then we have to create another relation for the key.

LectureNotes.in

R is a relation

$R = \{A, B, C, D\}$.

$F = \{C \to A$
$C \to D$
$B \to C\}$.

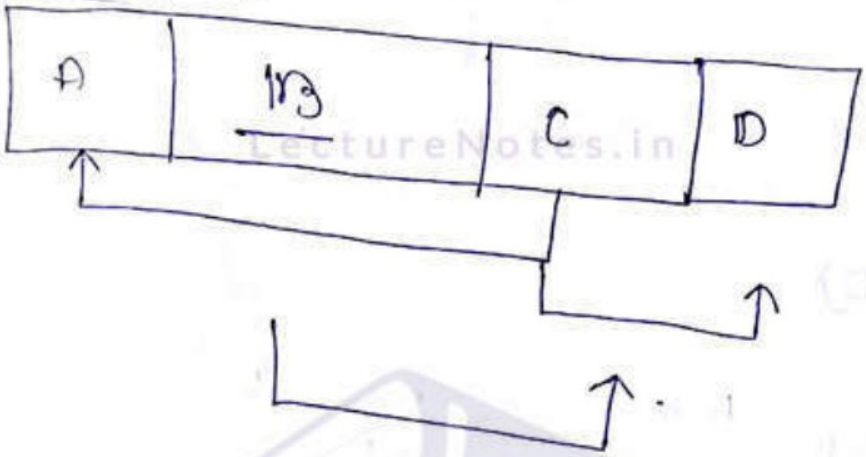
Find key, decompose upto 3NF.

CAD BCAD.

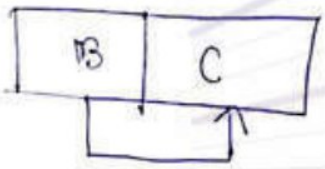
Ans:

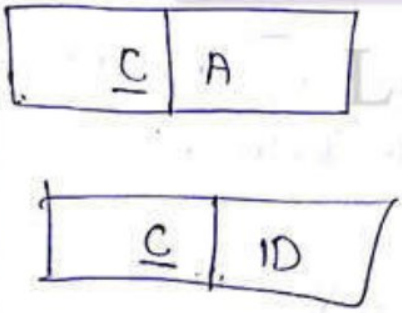
$\{B\}^+$ is the key.

1st & 2ND NF

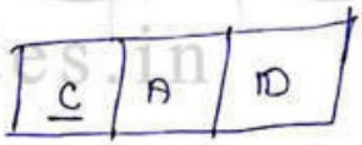


3NF





or



R is a relⁿ

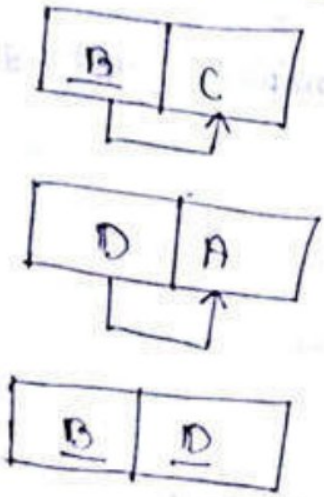
$R = \{A, B, C, D\}$.

$F = \{B \to C$
$D \to A\}$.

B C BC
BD BCAD

$\{BD\}^+$.

2NF & 3NF



Ex: R is a rel^n

$R = \{A, B, C, D\}$

$F = \{ABC \to D$

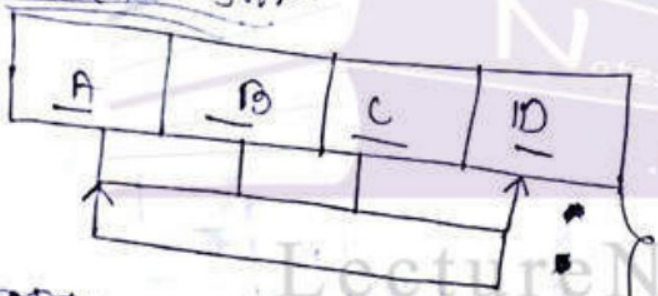
$D \to A\}$

Ans: ABC and BCD is the key.

BCD DA. AD B

AD

1st & 2nd & 3NF.



satisfying as all are prime attributes.

Ex. $R = \{A, B, C, D, E, F, G, H, I, J\}$.

$F = \{AB \to C$

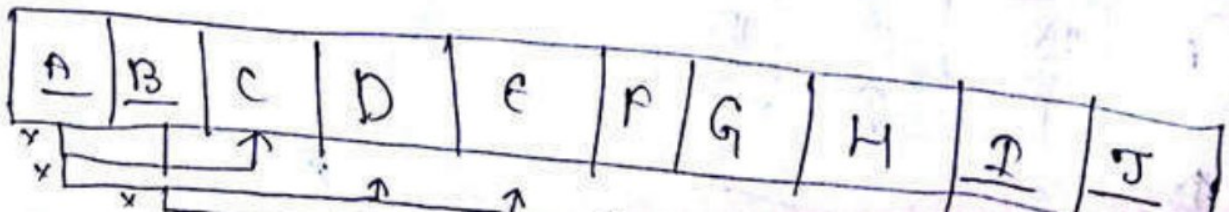
$A \to DE$

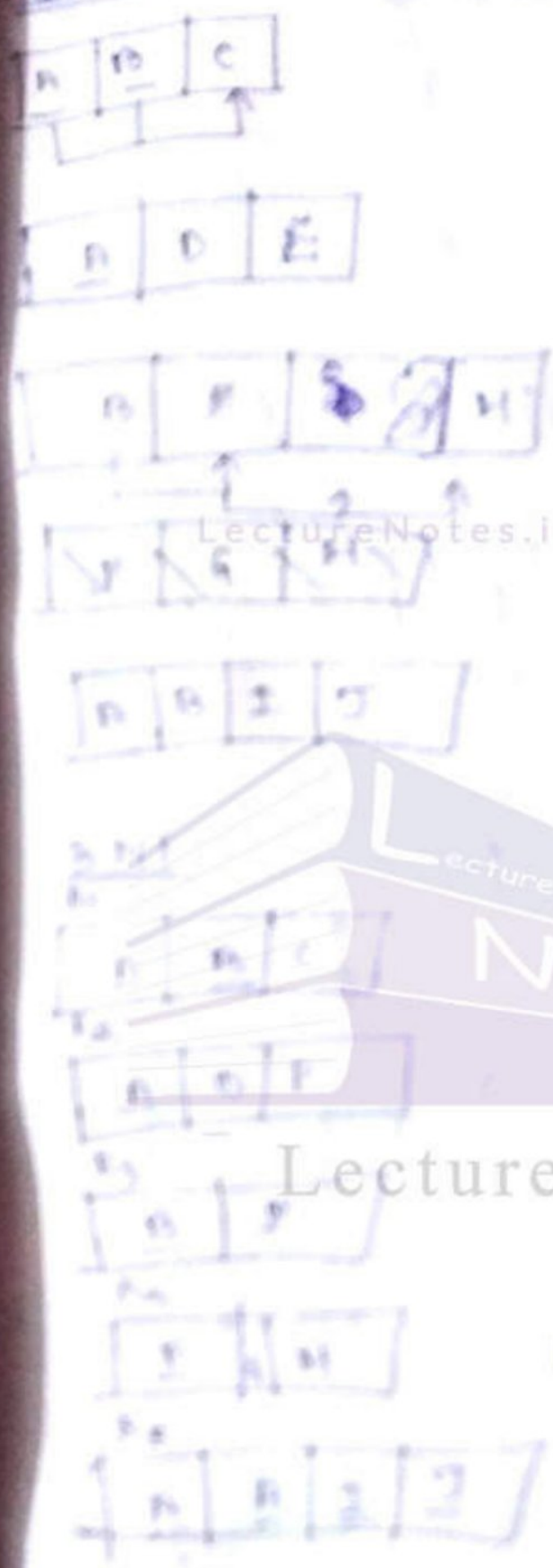
$B \to F$

$F \to GH\}$.

ABDEFCGH

$\{ABDJ\}^+$ is the key.





LectureNotes.in

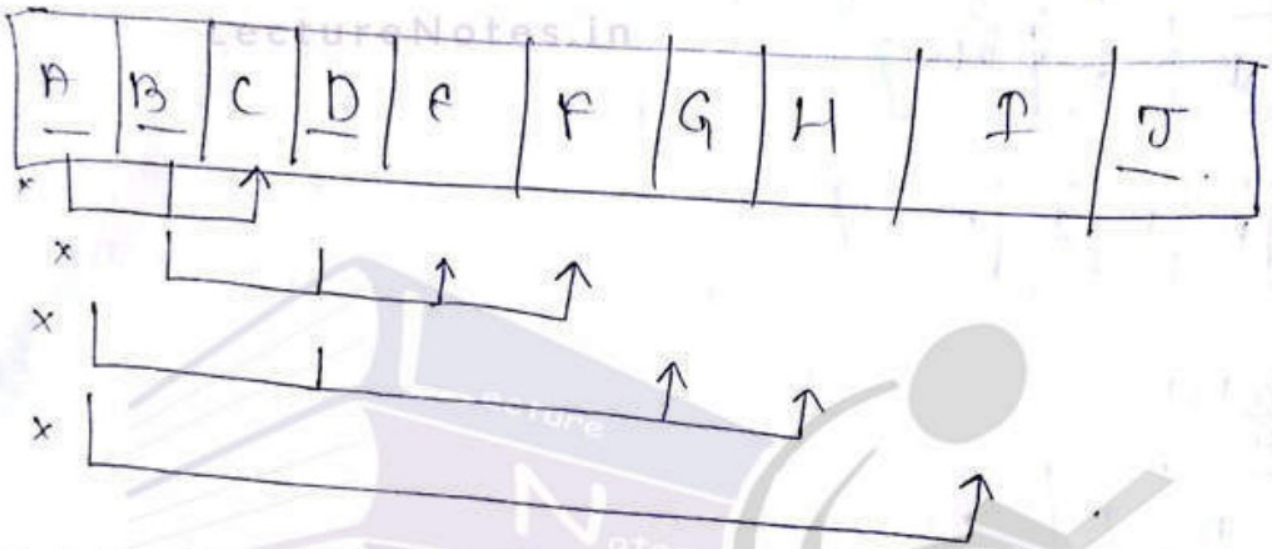
$R = \{A, B, C, D, E, F, G, H, I, J\}$

$F = \{AB \rightarrow C$
$BD \rightarrow EF$
$AD \rightarrow GH$
$A \rightarrow I\}$

ABIC

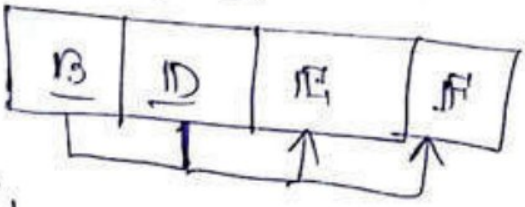
Ans:—

ABDJ is the Key.

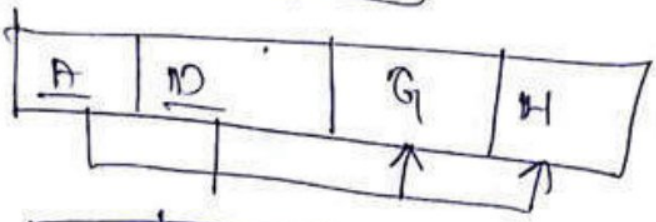


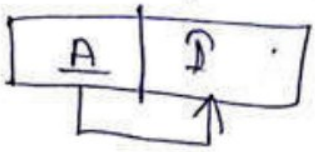
2NF & 3NF.

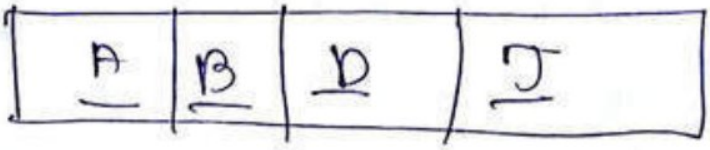
BDEF.



ADIGH







LectureNotes.in

Ex: $R = \langle A, B, C, D \rangle$

$F = \langle AB \to C$

$AB \to D$

$C \to A$

$D \to B$.

CDABCD .

ADBCD .

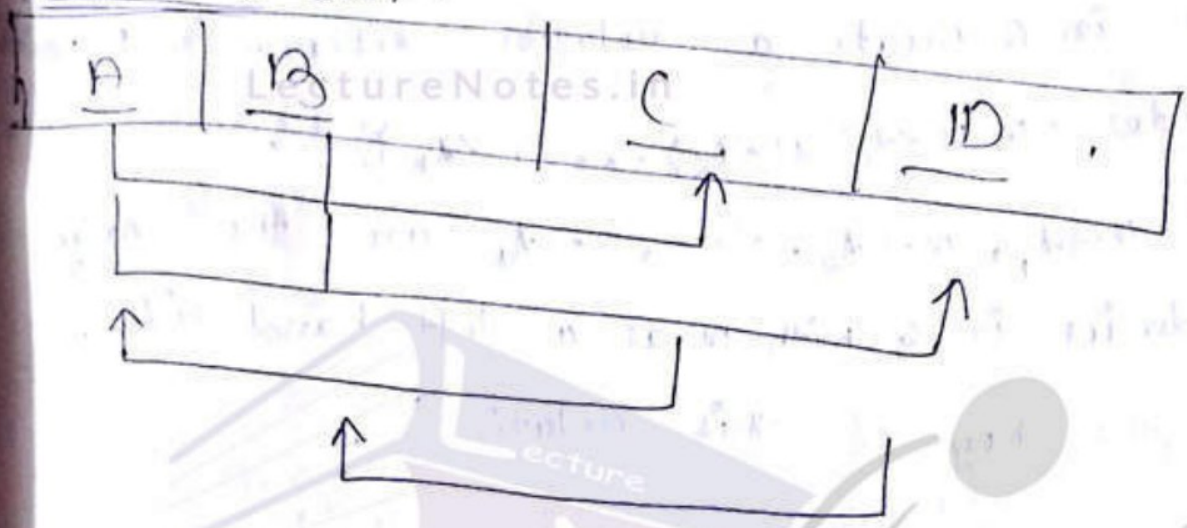
BCAD BC .

ADBC .

Ans:

AB, CD, AD, BC .

1st, 2nd & 3NF .



Ex: $R = \langle A, B, C, D \rangle$

$F = \langle A \to B$

$BC \to D$

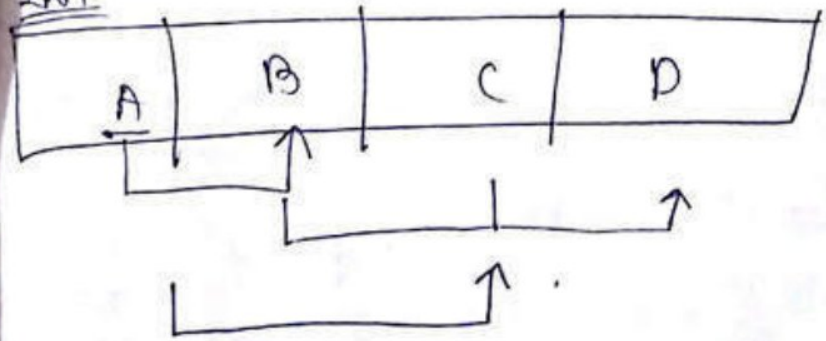
$A \to C \rangle$.

ABC .

Ans:

$\langle A^+ \rangle$ is the key .

2NF .



LectureNotes.in

16/9/2015.

Dependancy preserving and non-additive join decomposition into 3NF schema.

Step-I.

Find a minimal cover G of F.

Step-II.

For each left hand side (X) of a FD that appears in G create a relation schema in D with attributes $\{X \cup \{A_1\} \cup \{A_2\} \ldots \cup \{A_K\}\}$.

Where $X \to A_1, X \to A_2, \ldots, X \to A_K$ are the only dependencies in G with X as a left hand side. (X is the key of this relation).

Step-III.

If none of the relation schema in D contain a key of R, then create one or more relation schema in D that contains attributes that form a key of R.

Ex: $R = \{A, B, C, D\}$

$F = \{B \to C, D \to A\}$.

Ans:

~~Step-I~~ $F = \{B \to C$
$D \to A\}$.

$E = \{B \to C$
$D \to A\}$.

Let remove $B \to C$.

Atul Kumar
7717888822

$G = \{D \to A\}$

E cover G.

$\{D\}^+ = \{D, A\}$.

G cover E.

$\{B^+\} = \{B\}$.

This is not equivalent.

Remove $D \to A$.

$G = \{B \to C\}$.

E cover G

$\{B^+\} = \{B, C\}$.

G cover E.

$\{B\}^+ = \{B, C\}$.

$\{D^+\} = \{D\}$.

$E = \{B \to C, D \to A\}$.

This is not equivalent.

So, we cannot remove any of the attribute.

Step 2.

D =

R1

B	C

R2

D	A

$\{BD^+\} = \{B, D, C, A\}$.

R3

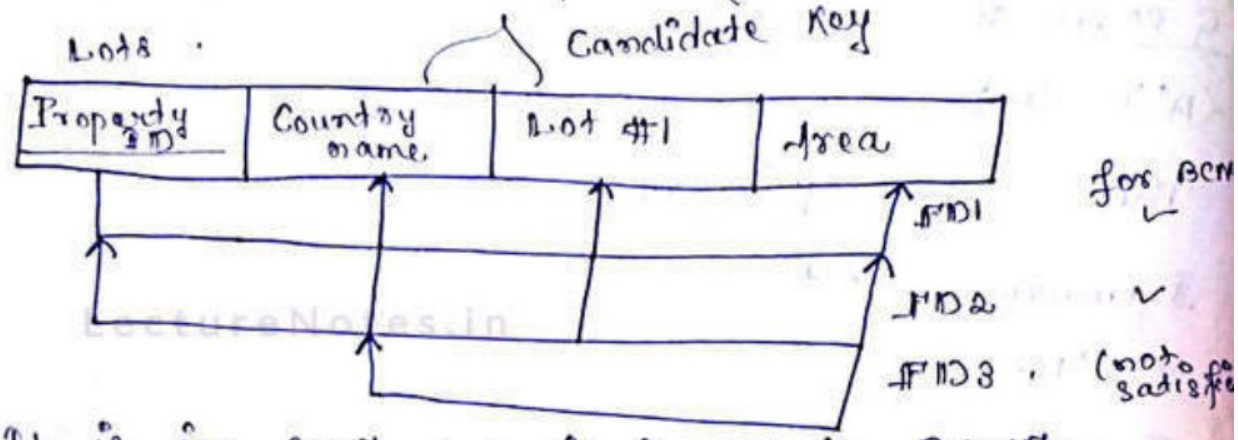
B	D

) BCNF. (Boyce Codd Normal Form). guarantees about loss less join.

- BCNF does not guarantees for dependency preservation property.

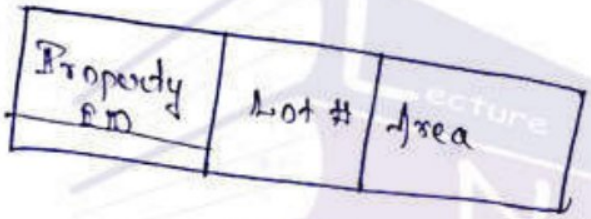
A relation schema (R) is in BCNF if whenever a non-trivial F.D $\alpha \rightarrow A$ holds in R, then α is a super key of R.

Ex:



It is in 3NF, but it is not in BCNF.

Decompose



(as country name is violating so, removed it).

Q. $R = \langle A, B, C, D \rangle$

$F = \langle B \rightarrow C, C \rightarrow A, C \rightarrow D \rangle$.

BCAD

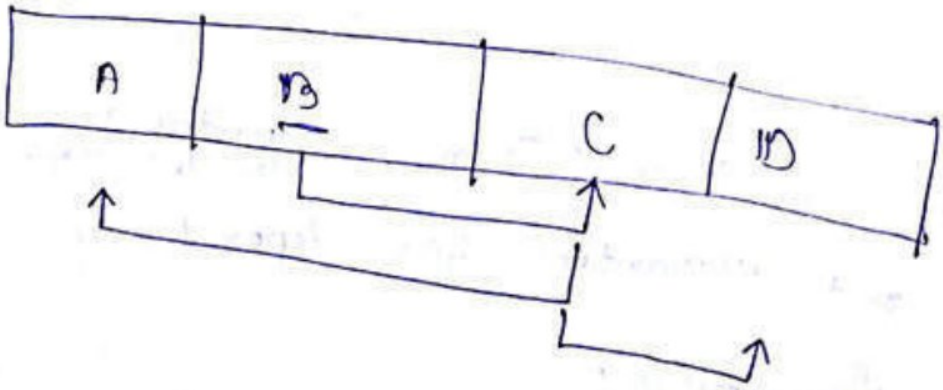
~~CAD~~

CAD

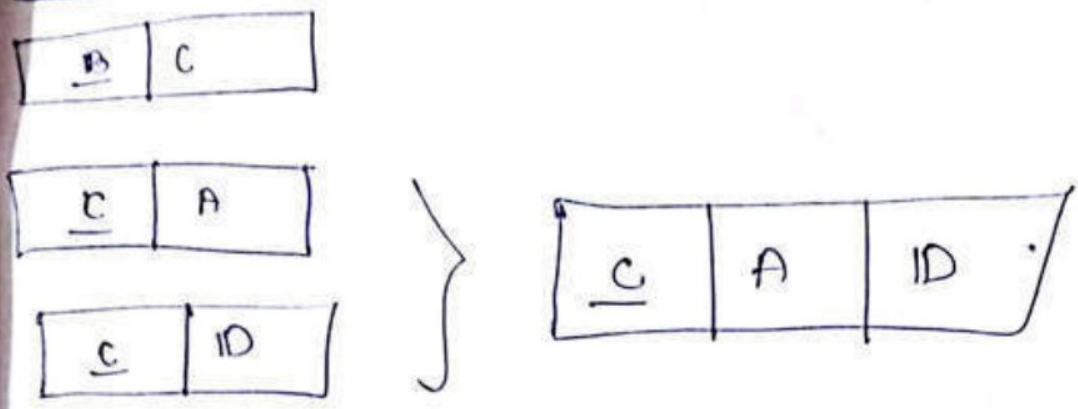
Ans:

$\langle B \rangle^+$ is the key.

2NF



3NF & BCNF.



Q. $R = \{A, B, C, D\}$

$F = \{ABC \rightarrow D$
$D \rightarrow A\}$.

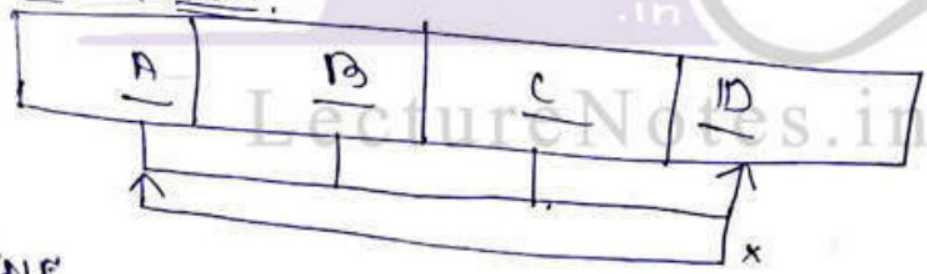
ABCDA.

BCDA.

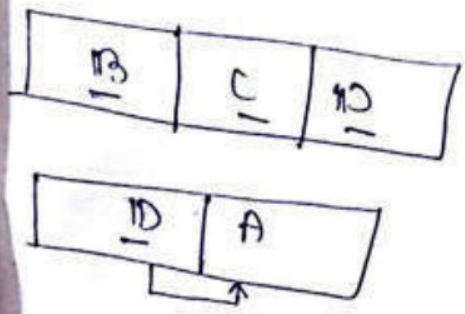
A:-

ABC, BCD.

2NF & 3NF.



BCNF



LectureNotes.in

18/9/2015.

Algorithm of relation decomposition in BCNF with non-additive join property.

Step-I:

Set $D = \{R\}$.

Step-II:

While there is a relation schema (Q) in D that is not in BCNF do

{

choose a relation schema (Q) in D that is not in BCNF.

Find FD $x \to y$ in Q that violates BCNF replace Q in D by two relation schema $(Q-y)$ and $(x \cup y)$.

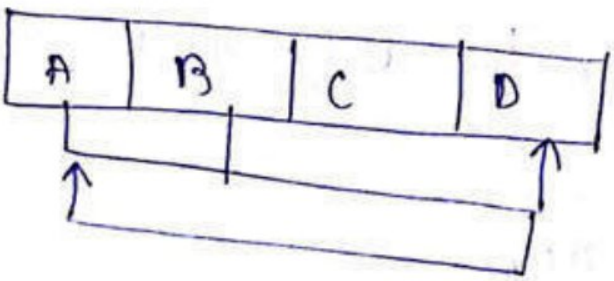
}

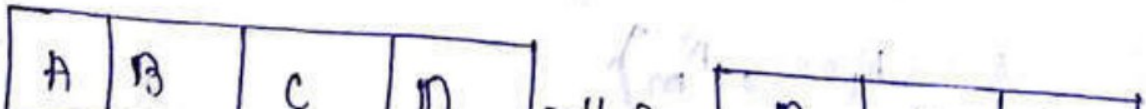
Ex: $R = \{A, B, C, D\}$

$F = \{AB \to D$

$\underset{x}{D} \to \underset{y}{A}\}$.

Ans:





X ∪ Y

= | D | A | R_2 (It is in BCNF).

Dependency Preservation Property of a decomposition.

A decomposition

$D = \{R_1, R_2, \ldots, R_n\}$

of R is dependency preserving w.r.t F if the union of the projection of F on each R_i in D is equivalent to F i.e.,

$$\left(\left(\pi_{(R_1)}(F)\right) \cup \left(\pi_{(R_2)}(F)\right) \cdots \left(\pi_{(R_n)}(F)\right)\right) = F.$$

Ex: $F = \{B \to C, C \to A, C \to D\}$

R_1

B	C

R_2

C	A	D

$\{B \to C, C \to A, C \to D\}^+ = F^+$ (Dependency preservation exist).

Loss Less Join or Non-Additive Join Lossless Join Property of a Decomposition.

A decomposition Key

of R has the lossless property w.r.t the set of dependencies F on R, if for every relation state r of R that satisfies F, the condition is

$$\left(\pi_{(R_1)}(r) * \pi_{(R_2)}(r) * \ldots * \pi_{(R_n)}(r)\right) = r.$$

LectureNotes.in

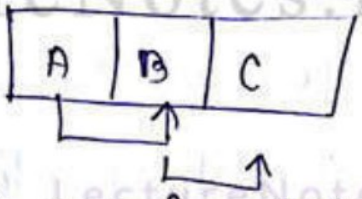
Ex: $R = \{A, B, C\}$

$F = \{A \to B, B \to C\}$.

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

Ans:

$\{A\}^+ = \{A, B, C\}$



It is in 2NF, but not in 3NF.

A	B

B	C

A	B
a_1	b_1
a_2	b_1

B	C
b_1	c_1
b_1	c_2

$R_1 * R_2 = R_1 \cdot B = R_2 \cdot B$

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_2

We are getting spurious tupple.

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

Ans:—

A	B
a_1	b_1
a_2	b_2

B	C
b_1	c_1
b_2	c_2

A	B	C
a_2	b_2	c_2
a_1	b_1	c_1

Loss Less Join for Binary Decomposition.

A decomposition (D)

$$D = \langle R_1, R_2 \rangle$$

of R has the loss less join property w.r.t. a set of functional dependency (F) on R if and only if the FD

$$((R_1 \cap R_2) \rightarrow (R_1 - R_2))$$

in F^+, or the FD

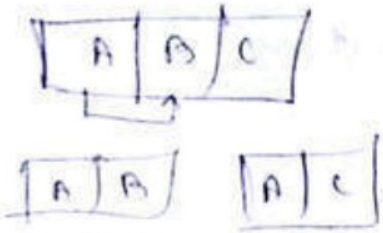
$$((R_1 \cap R_2) \rightarrow (R_2 - R_1))$$

in F^+.

Q: $R = \langle A, B, C \rangle$

$F = \langle A \to B \rangle$

Ans:—



$R_1 \cap R_2$ A $R_1 - R_2$ B

$A \to B$

~~$B \to C$~~

Different way of decomposition

$R_1 = \langle AB \rangle$ $R_2 = \langle AC \rangle$

AB BC

AB BC

$A \to B$ in F^+

$(R_1 \cap R_2)$

$(R_1 - R_2)$ $B \to C$ ×

$C \to A$ ×

22/9/2015.

Lossless Join Property.

Ensures that no spurious tuples are generated when a natural join operation is applied to the relations in the decomposition set.

Algorithm.

Input: A universal relation schema (R),

A decomposition set (D) = $\langle R_1, R_2, \ldots, R_m \rangle$ of R,

A set of functional dependencies F.

1. Create an initial matrix (S) with one row (i) for each relation R_i in D and one

column (j) for each attributes A_j in R.

2. For each row(i) representing a relation schema (R_i)

{

For each column (j) representing attribute (A_j)

{

if (relation (R_i) includes attribute A_j), then

set $S(i, j) = a_j$

}

}

3. Repeat the following loop until a complete loop execution results in no changes to S.

{

for each fd $X \to Y$ in F

{

for all rows in S that have the same symbols in the columns corresponding to attributes in X.

{

makes the symbols in each column that

4. If a row is 'made up' entirely of "a" symbols, then the decomposition has the lossless join property; otherwise it is not.

Ex: Consider the following decomposition of the relⁿ schema $R = \langle a, b, c, d, e, f, g, h, i, j \rangle$.

F.D $F = \{AB \to C, A \to DE, B \to F, F \to GH, D \to IJ\}$.

$D = \{R_1, R_2, \ldots, R_5\}$

$R_1 = \{A, B, C\}$

$R_2 = \{A, D, E\}$

$R_3 = \{B, F\}$

$R_4 = \{F, G, H\}$

$R_5 = \{D, I, J\}$

Ans:

	R_1 1	R_2 2	R_3 3	R_4 4	R_5 5
R_1 1					
R_2 2					
R_3 3					
R_4 4					
R_5 5					

LectureNotes.in

	A	B	C	D	E	F	G	H	I	J
R_1	a_1	a_2	a_3							
R_2	a_1			a_4	a_5					
R_3		a_2				a_6				
R_4						a_6	a_7	a_8		
R_5				a_4					a_9	a_{10}

$AB \rightarrow C$

check $\frac{a_1, a_2}{x}$ in other rows if present

write $\frac{a_3}{y}$ there.

	A	B	C	D	E	F	G	H	I	J
R_1	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
R_2	a_1			a_4	a_5				a_9	a_{10}
R_3		a_2				a_6	a_7	a_8		
R_4						a_6	a_7	a_8		
R_5				a_4					a_9	a_{10}

Search if any of the rows contains all the attributes, then there is no loss

Q. $R = \langle A, B, C, D, E, F, G, H, I, J \rangle$

$D = \langle R_1, R_2, R_3 \rangle$

$F \cdot D = \langle AB \to C, A \to DE, B \to F, F \to GH, D \to IJ \rangle$

$R_1 = \langle A, B, C, D, E \rangle$

$R_2 = \langle B, F, G, H \rangle$

$R_3 = \langle D, I, J \rangle$

	A	B	C	D	E	F	G	H	I	J
R_1	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10} → $AB \to C$ and $A \to DE$.
R_2		a_2				a_6	a_7	a_8		
R_3				a_4					a_9	a_{10}

Ex. $R = \langle A, \ldots, J \rangle$

$F \cdot D = \langle AB \to C, A \to DE, B \to F, F \to GH, D \to IJ \rangle$

$D = \langle R_1, \ldots, R_5 \rangle$

$R_1 = \langle A, B, C, D \rangle$

$R_2 = \langle D, E \rangle$

$R_3 = \langle B, F \rangle$

$R_4 = \langle F, G, H \rangle$

$R_5 = \langle D, I, J \rangle$

	A	B	C	D	E	F	G	H	I	J
R_1	a_1	a_2	a_3	a_4	•	a_6	a_7	a_8	a_9	a_{10}
R_2				a_4	a_5				a_9	a_{10}
R_3		a_2				a_6	a_7	a_8		
R_4						a_6	a_7	a_8		
R_5				a_4					a_9	a_{10}

LectureNotes.in

Q. $R = \{eno, ename, pno, pname, plocation, hours\}$

$D = \{R_1, R_2\}$.

$R_1 = \{ename, plocation\}$

$R_2 = \{eno, pno, hours, pname, plocation\}$.

$F = \{eno \rightarrow ename, pno \rightarrow pname, plocation, eno, pno \rightarrow hours\}$

	1 eno	2 ename	3 pno	4 pname	5 plocation	6 hours
R_1		a_2			a_5	
R_2	a_1		a_3	a_4	a_5	a_6

(loss of information).

Q. $R = \{eno, ename, pno, pname, plocation, hours\}$

$D = \{R_1, R_2, R_3\}$

$R_1 = \{eno, ename\}$.

$R_2 = \{pno, pname, plocation\}$

$R_3 = \{eno, pno, hours\}$.

$F = \{eno \rightarrow ename, pno \rightarrow pname, plocation, eno, pno \rightarrow hours\}$.

	1 eno	2 ename	3 pno	4 pname	5 plocation	6 hours
R_1	a_1	a_2				
R_2			a_3	a_4	a_5	
R_3	a_1	a_2	a_3	a_4	a_5	a_6

(no loss).

23/9/2015.

(v) 4NF (Multivalue Dependency).

$A \rightarrow B$

$A \rightarrow\rightarrow B$ (for A, B have multivalue). (A multi-determines B.

Emp-Proj $n=3$

Ename	Pname	Dname
Smith	P_1, P_2	D_1, D_2
What is	this? waste of	paper IDIOT!

This is not a relation and not in 1NF as each cell should contain single value.

So, we have to normalise it.

Emp-Proj

Ename (A)	Pname (B)	Dname (C)
Smith	P_1	D_1
Smith	P_2	D_2
Smith	P_1	D_2
Smith	P_2	D_1

$A \rightarrow\rightarrow B$

$A \rightarrow\rightarrow C$.

$\Rightarrow A \rightarrow\rightarrow B/C$.

Multivalue Dependency (MVD).

$x \rightarrow\rightarrow y$ specified on relation schema (R) where x and y are subset of R specifies

the following constraint on any relation state r of R.

If two tuples t_1 and t_2 exist in r such that

$$t_1[x] = t_2[x],$$

then two tuples t_3 and t_4 should also exist in r with following properties.

$$t_1[x] = t_2[x] = t_3[x] = t_4[x].$$

if $t_1[y] = t_3[y]$ and $t_2[y] = t_4[y]$.

$t_1[z] = t_4[z]$ and $t_2[z] = t_3[z]$.

*- A multi-value dependency $x \rightarrow\rightarrow y$ can be trivial or non-trivial.

4NF Form.

* A

Condition for trivial multi-value dependency.

y is subset of x or $(x \cup y) = R$.

4NF Form.

A relation schema (R) is in 4th NF w.r.t set of dependencies (F) if for every non-trivial multi-value dependency $x \rightarrow\rightarrow y$, x is a super key of R.

LectureNotes.in
Atul Kumar
7778888822

x	y	z
Ename	Pname	Dname
Smith	P_1	D_1
Smith	P_2	D_2
Smith	P_1	D_2
Smith	P_2	D_1

This relation is not in 4NF.

Algorithm to decompose the relation.

1. Set D = {R}.
2. While there is a relation schema (Q) in D that is not in 4NF do

{

Choose a relation schema (Q) in D that is not in 4NF.

Find a non-trivial MVD $x \to y$ in Q that violates 4NF.

Replace Q in D by two relation schema $(Q-y)$ and $(x \cup y)$

R_1

x	z
Smith	D_1
Smith	D_2

R_2

x	y
Smith	P_1
Smith	P_2

trival as $x \cup y = R$.

}

vi) 5NF Norm. (PJNF) (Project join normal form).

*- 5NF is based on join dependency.

A relation schema (R) is in 5NF with a set of F.D, multi-value dependency,

join dependency (JD) if for every non-trivial join dependency, every R_i is a super key of R.

Query By Examples. (QBE)

*- It was originally developed by M. M Zloof at IBM research laboratory.

*- QBE is based on domain calculus.

*- The queries are written in the horizontal and vertical dimension of a table.

*- Queries are formed by entering an example of a possible answer in a skeleton table.

*- The skeleton table does not have column headings.

*- The first column is used for relation name.

Ex: Find all the relation name in the database.

To print → P.

Ans:

P.	IDIOT	IDIOT	STUPID
	Khali parents	manaukia	paisa sare
	wastage of paper	Khali Karuchi	ABAHADRA !!

Ex: Find all the attributes of employee relation.

Ans:

	Ananya	Soni	
employee P.	Akanksha	Sushri	
	Arpita		

Ex: Find all the details of employee relation.

Ans:

employee	Eno	Ename	Salary
	420	Sushri	
P.		Sangita Rout	2.51/-

or

Employee	Eno	Ename	Salary
	P.a	P.b	P.c

Ex: Find all the details of employee whose salary = 1000.

Ans:

employee	Eno	Ename	Salary

Ex: Find the names of employees who manager and salary > 50,000.

Ans:

	a	b	c	d
employee	Eno	Ename	Salary	Job
P.		P.b	>50,000	Manager

Ex: Find the names of employee whose Job = manager and salary >10,000 and < 20,000

Ans:

	a	b	c	d	e
employee	Eno	Ename	Salary	Salary	Job
		P.b	>10,000	<20,000	mng

or

	a	b	c	d
employee	Eno	Ename	Salary	Job
		P.b	>10,000 <20,000	Manager

LectureNotes.in

24/9/2015.

Transactions.

It is a unit of program execution that accesses and possibly updates various data items.

*- To ensure integrity of the data, the database system maintain the following properties of transaction. That property is called ACID property. (A → Atomicity, C → Consistency, I → Isolation, D → Durability).

What is Atomicity Property.

*- Either all the operations of the transaction are reflected properly on the database or none are.

What is consistency property.

*- The execution of a transaction in isolation (i.e., no other transaction executing concurrently preserves the consistency of the database.).

What is isolation property.

*- Even though multiple transactions may execute concurrently, system guarantees that for every pair of transactions T_i and T_j, it appears to T_i that either T_j has

finished execution before T_i started or T_i started execution after T_i finished.

Thus, each transaction is unaware of other transaction executing concurrently in the system.

What is Durability Property.

* After a transaction complete successfully, the changes it has made to the database persists even if there are system failure.

Ex. of transaction.

T_i		
read(A)		A = 1000, B = 2000
A = A − 50	//1000	
write(A)	//950	
read(B)		
B = B + 50		
write(B)	//2050	Total = 3000

(Consistency is preserved).

Use of Atomicity Property.

* If the atomicity property is present, all the actions of the transaction are reflected in the database or none are.

*- The basic idea behind ensuring atomicity property is that the database system keeps track of old values of any data on which a transaction performs a write operation and if the transaction does not complete its execution, the database system restores its old values.

Use of Consistency Property.

*- By the help of consistency property, it can be verified easily that the database is consistent before an execution of the transaction, the database remains consistent after the execution of the transaction.

Use of Isolation Property.

*- If several transactions are executed concurrently, their ~~an~~ operations may interleave in some undesirable way resulting in an ~~an~~ inconsistence state.

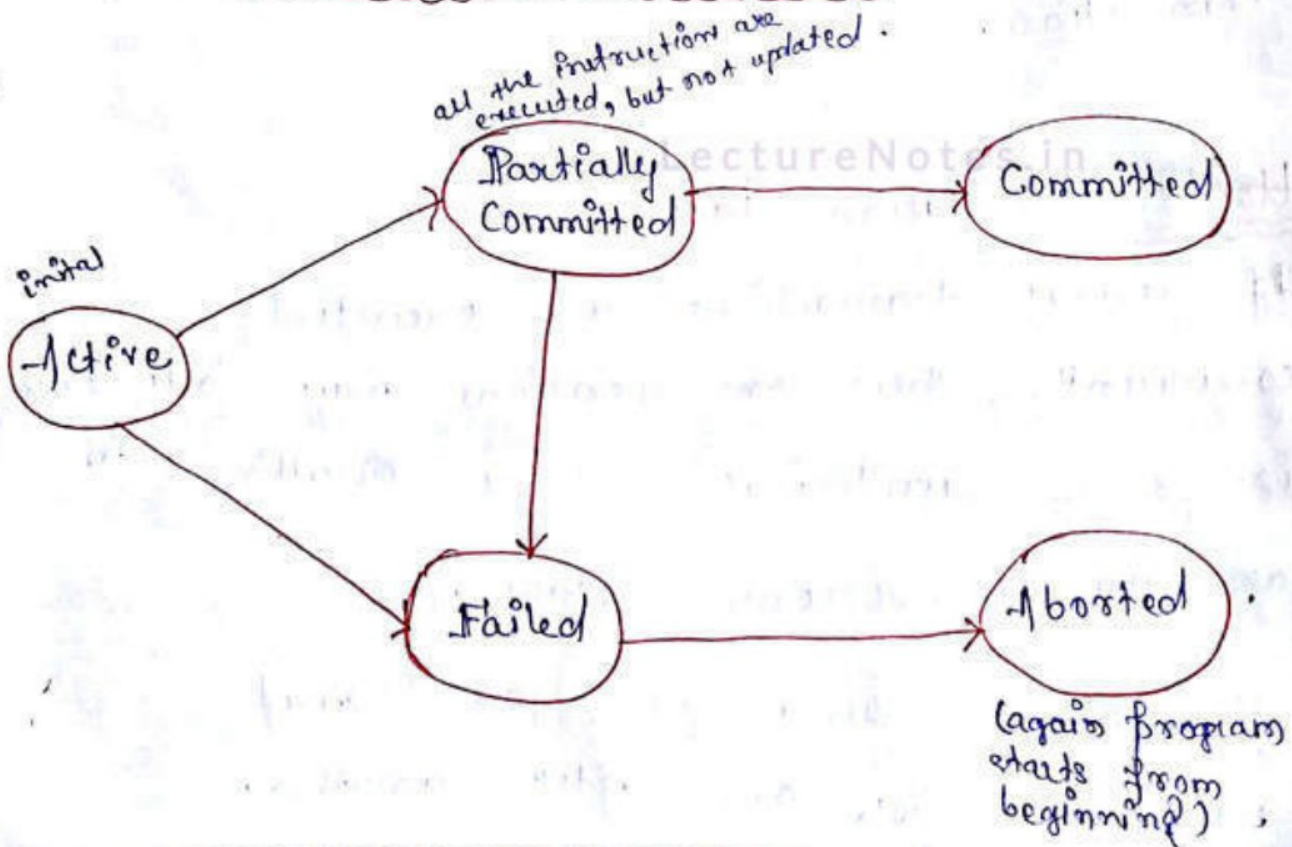
*- Solution is that do for serial execution i.e., one after another.

Use of Durability Property.

*- We can guarantee durability property by ensuring that either the updates carried out by the transaction have been written to the disk before transaction completes.

*- Information about the updates carried out by the updation and written to the disk is sufficient to enable to reconstruct the updates when the database system restarted after the failure.

States of a transaction.



TCL (Transaction control language).

roll back

commit

save point

why is this place so.... blank!! Nonsense!

25/9/2015.

Active State.

The initial state, the transaction stays in this state while it is executing.

Partially Committed State.

After the final statement has been executed.

Failed State.

After discovering that the normal execution can no longer proceed.

Aborted State.

After the transaction has been rolled back and the database has been restored to its prior to the start of the transaction.

Committed State.

After successfully completed.

Terminate.

After the transaction either committed or aborted.

Concurrent Execution.

*- In transaction process, system, usually allow multiple transactions to run concurrently allowing multiple transactions to update data concurrently causes several complications with consistency of the data. So, the solution is to go for serial execution i.e., one after another.

Two good reason for allowing concurrency.

1. Improved throughput and resource utilisation
2. Reduced waiting time.

1. Improved throughput and resource utilisation.

2. Reduced waiting time.

Throughput:

The number of transactions executed in a given amount of time.

Ex:

T_1	T_2
read(A)	read(A)
A = A − 50	temp = A * 0.1
write(A)	A = A − temp
read(B)	write(A)
B = B + 50	read(B)
write(B)	B = B + temp
	write(B)

Serial execution.

T_1	T_2	
read(A)	// 1000	
A = A − 50	// 950	A = 1000
write(A)	// 950	B = 2000
read(B)	// 2000	
B = B + 50	// 2050	
write(B)	// 2050	
	read(A) // 950	
	temp = A * 0.1 // ~~0295~~ 095	
	A = A − temp // ~~2655~~ 855	
	write(A) // 855	
	read(B) // 2050	
	B = B + temp // 2145	
	write(B)	
	A + B // 3000	

$950 \times 0.1 = 95.0$

$2950 \times 0.1 = 295.0$

$2950 - 295 = 2655$

$950 - 95 = 855$

$2050 + 95 = 2145$

$855 + 2145 = 3000$

Serial ~~Concurrent~~ execution:

First T_2 then T_1.

1000 × 0.1 = 0100	A = 900	900	2100 + 50 = 2150
	2000 + 100 = 2100 (B)	850	2150 − 850 = 3000

Note:

If there is n transactions, then n! serial schedule can be formed.

Concurrent Execution:

T_1		T_2		
read(A)	1000			
A = A − 50	950			
write(A)	950			
		read(A)	950	
		temp = A * 0.1	95	950 × 0.1 = 095.0
		A = A − temp	855	950 − 95 = 855
		write(A)	855	
read(B)	2000			
B = B + 50	2050			
write(B)	2050			
		Read(B)	2050	
		B = B + temp	2145	2050 + 95 = 2145
		write(B)	2145	

A + B = 3000

T_1	T_2		
read (A)	//1000		
A = A − 50	//950		
	read (A)	//1000	
	temp = A * 0.1	//100	
	A = A − temp	//900	
	write (A)	//900	
	read (B)	//2000	
write (A)	//~~900~~ 950		
read (B)	//2000		
B = B + 50	//2050		
write (B)	//2050		
	B = B + temp	// ~~2150~~	
	write (B)	// ~~2150~~ 2100	
	A + B = 3050		

950
0.1

(2000+100)
2050

2100
+ 2050
100

2150
950
3100

2100
950
3050

29/9/2015

Serialisability

*- The objective of serializability is to find non-serial schedules that allow transactions to execute concurrently without interfering with one another and thereby produce a database state that to be produced by serial schedule.

Conflict serializability.
View Serializability

Conflict Serializability Equivalent -

If a schedule (S) can be transformed into a schedule (S') by a series of swaps

of non-conflicting instructions, we say S and S' are conflict equivalent.

(ii) Conflict Serializability.

* A schedule(S) is conflict serilizable schedule if it is conflict equivalent to a serial schedule.

LectureNotes.in

Which instructions are conflicting instructions.

Let us consider a schedule(S) in which there are two consecutive instructions I_i and I_j of transaction T_i and T_j.

If I_i and I_j are different data item, we can swap it.

(i) If I_i = read(Q) and I_j = read(Q), the order of I_i and I_j does not matter.

(ii) I_i = read(Q) and I_j = write(Q), the order of I_i and I_j matters.

(iii) If I_i = write(Q) and I_j = read(Q), the order matters.

LectureNotes.in

iv) I_i = write(Q) and I_j = write(Q),

order matters.

Q. $R_1(x)$, $R_3(x)$, $W_1(x)$, $R_2(x)$, $W_3(x)$.

Ans:

S

T_1	T_2	T_3
read(x)		
	swap	
write(x)		read(x)
	read(x)	
		write(x).

S'

T_1	T_2	T_3
		read(x)
read(x)		
write(x)		
	read(x)	
		write(x).

(Conflict equivalent).

This table is not conflict serializable schedule.

Ans.

T_1	T_2.
read(A)	
write(A)	
	read(A)
	write(A)
read(B)	
write(B)	
	read(B)
	write(B).

LectureNotes.in

T_1	T_2
read(A)	
write(A)	
	read(A)
read(B)	
	write(A)
write(B)	
	read(B)
	write(B)

T_1	T_2
read(A)	
write(A)	
read(B)	
	read(A)
	write(A)
write(B)	
	read(B)
	write(B)

T_1	T_2
read(A)	
write(A)	
read(B)	
	read(A)
write(B)	
	write(A)
	read(B)
	write(B)

T_1	T_2
read(A)	
write(A)	
read(B)	
write(B)	
	read(A)
	write(B)
	read(B)
	write(B)

This is a serializable schedule.

Q: $R_1(x)$, $R_3(x)$, $W_3(x)$, $W_1(x)$, $R_2(x)$.

Ans:—

T_1	T_2	T_3
read(x)		
		read(x)
		write(x)
write(x)		
	read(x)	

LectureNotes.in

T_1	T_2	T_3
		read(x)
read(x)		write(x)
write(x)	read(x)	

This is not conflict serializable schedule.

Q. $R_3(x)$, $R_2(x)$, $W_3(x)$, $R_1(x)$, $W_1(x)$.

Ans:

T_1	T_2	T_3
		read(x)
	read(x)	
		write(x)
read(x)		
write(x)		

T_1	T_2	T_3
	read(x)	
		read(x)
		write(x)
read(x)		
write(x)		

This is a conflict serializable schedule.

Q. $R_3(x)$, $R_2(x)$, $R_1(x)$, $W_3(x)$, $W_1(x)$.

Ans: T_1

T_1	T_2	T_3
		read(x)
	read(x)	
read(x)		
		write(x)
write(x)		

T_1	T_2	T_3
	read(x)	
read(x)		read(x)
write(x)		write(x)

T_1	T_2	T_3
read(x)	read(x)	
		read(x)
write(x)		write(x)

T_1	T_2	T_3
read(x)		
	read(x)	
		read(x)
		write(x)
write(x)		

This is a non-conflict serializable schedule.

LectureNotes.in

30/9/2015.

Precedence Graph.

*- A schedule(S) is given. Check whether the schedule(S) is conflict ~~set~~ serializable schedule or not.

Consider a schedule(S), we construct a directed graph called a precedence graph from schedule(S). The graph consists of a pair

$$G = (V, E)$$

where V is the set of vertices.
and E is the set of edges.

The set of vertices consists of all transactions participating in the schedule.

The set of edges consists of all edges $T_i \rightarrow T_j$ for which one of the three conditions holds:

(i) T_i executes write(Q) before T_j executes read(Q).

(ii) T_i executes read(Q) before T_j executes write(Q).

(iii) T_i executes write(Q) before T_j executes write(Q).

LectureNotes.in

Atul Kumar
7777888822

*- If an edge $T_i \rightarrow T_j$ exists in the precedence graph, then any serial schedule (S') is equivalent to S.

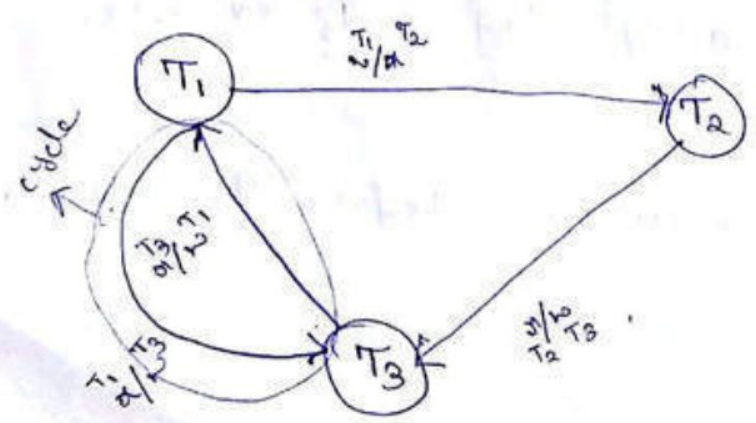
T_i must appear before T_j.

*- If the precedence graph from S has a cycle, then the schedule (S) is not conflict serializable schedule.

Q. $R_1(x)$, $R_3(x)$, $W_1(x)$, $R_2(x)$, $W_3(x)$.

Ans:

T_1	T_2	T_3
read(x)		
		read(x)
write(x)		
	read(x)	
		write(x)



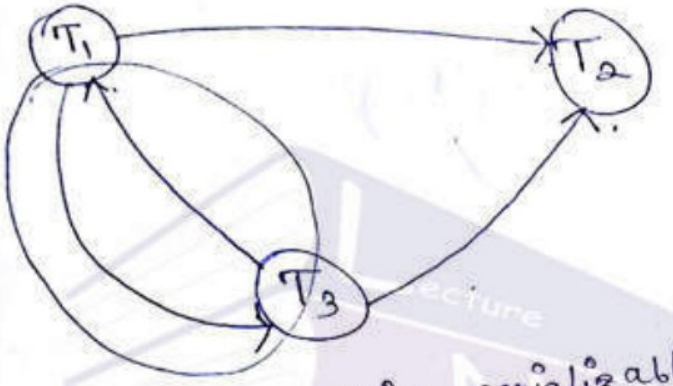
So, not a conflict serializable schedule.

Q. $R_1(x)$, $R_3(x)$, $W_3(x)$, $W_1(x)$, $R_2(x)$.

Ans:

T_1	T_2	T_3
read(x)		
		read(x) write(x)
write(x)		
	read(x)	

Soni Dartu
Ananya
Akanksha
Arpita
Sushri

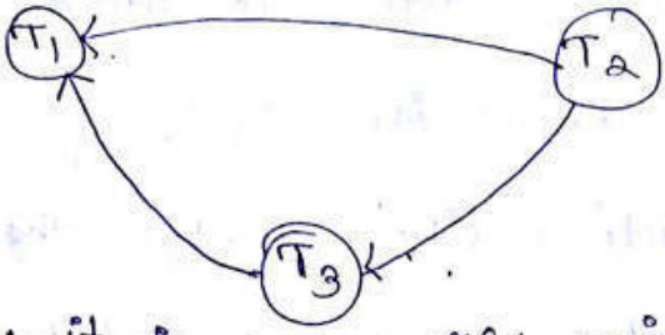


So, not a conflict serializable schedule.

Q. $R_3(x)$, $R_2(x)$, $W_3(x)$, $R_1(x)$, $W_1(x)$.

Ans:

T_1	T_2	T_3
		read(x)
	read(x)	
		write(x)
read(x) write(x)		



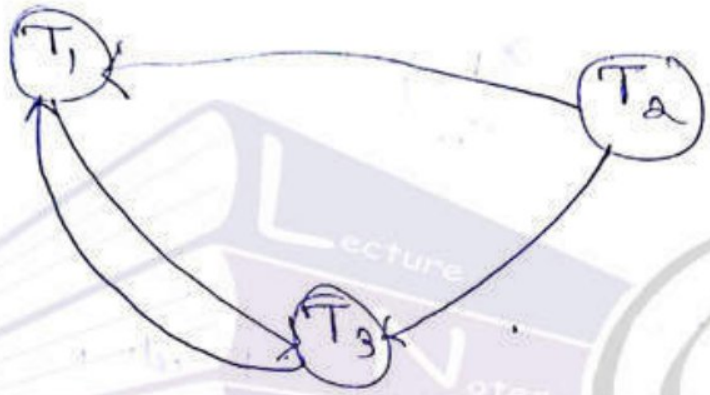
So, it is a conflict serializable schedule.

LectureNotes.in

Q: $R_3(x)$, $R_2(x)$, $R_1(x)$, $W_3(x)$, $W_1(x)$:-

Ans:-

T_1	T_2	T_3
		read(x)
	read(x)	
read(x)		
		write(x)
write(x)		



So, it is not a conflict serializable schedule.

ii) View Serializability.

Consider two schedules S and S' where same set of transactions participates in both the schedules.

The schedule S and S' are said to be view equivalent if three conditions meet:

i) For each data item (Q),
if transaction (T_i) reads the initial value of Q in schedule (S),

then transaction (T_i) must in schedule (S') also read the initial value of Q.

(ii) for each data item (Q), if transaction (T_i) executes read(Q) in schedule (S) and if that value is produced by a write(Q) operation executed by transaction (T_j), then read(Q) transaction (T_i) must in schedule (S') also read the value of Q that was produced by same write(Q) operation of transaction (T_j).

(iii) for each data item (Q) (if any), the transaction that performs the final write(Q) operation in schedule(S), must perform the final write(Q) operation in schedule (S').

Ex:

S

T_1	T_2
read(A)	
write(A)	
	read(A)
	write(A)
read(B)	
write(B)	
	read(B).

S'

T_1	T_2
read(A)	
write(A)	
read(B)	
write(B)	
	read(A)
	write(A)
	read(B)
	write(B)

Ans:

1/10/2015.

It is a view serializable schedule.

Ex:

T_3	T_4	T_6
read(Q)		
	write(Q)	
write(Q)		
		write(Q)

T_3	T_4	T_6
read(Q)		
write(Q)		
	write(Q)	
		write(Q)

Ans:

It is a view serializable schedule.
(For case - II, if it is in S, check in S').

Ex:

T_3	T_4
read(Q)	
	write(Q)
write(Q)	

T_3	T_4
read(Q)	
write(Q)	
	write(Q)

Ans:

It is not a view serializable schedule
cas last write is in T_3 in S but it is in

Recoverability.

Recoverability Schedule.

T_1	T_2
read(x)	
write(x)	
	read(x)
	write(x)
	read(y)
	write(y)
	commit
read(y)	
write(y)	

1st T_1 will commit then T_2 will commit.

Recoverable schedule is one where for each pair of transaction T_i and T_j such that T_j reads a data item previously written by T_i.

The commit operation of T_i appears before the commit operation of T_j.

LectureNotes.in

Cascadeless Schedule.

T_1	T_2	T_3
read(x)		
write(x)		
	read(x)	
	write(x)	
		read(x)
		write(x)

T_2 will start reading after T_1 commit.

A single transaction failure may leads to a series of transaction roll back. It is called cascading roll back.

A cascadless schedule is one where for each pair of transaction T_i and T_j such that T_j reads a data item previously written by T_i.

The commit operation of T_i appears before read operation of T_j.

6/10/2015.

Concurrency Control.

Different types of problems occurs due to concurrency.

1. Lost Update Problem.
2. Uncommitted dependency problem.
3. Inconsistent analysis problem.

Atul Kumar
7777888822

1. Lost Update Problem.

*- A completed update operation by one user can be overridden by another user. This is

known as lost update problem.

T_1	T_2
	read(x)
read(x)	x = x+100
x = x-10	write(x)
write(x)	

x = 100.

200.

read(x) 100 T_2.
read(x) 100 T_1
x = x+100 200 T_2
x = x-10 90 T_1
write(x) 200 T_2
write(x) 90 T_1

Serial = 190. Concurrent = 90.

2: Uncommitted Dependency Problem (Dirty Read Problem).

*- The uncommitted dependency problem occurs when one transaction is allowed to see the intermediate results before it has committed.

		T_3	T_4		
	concurrent ~~Serial~~		read(x)	~~Serial~~ concurrent	
100	200	read(x)	x = x+100	200	100
90	190	x = x-10	write(x)	200	200
90.	190	write(x).	rollback (failure occurs		200.

3: Inconsistent analysis Problem.

*- It occurs when a transaction reads several values from the database, but second transaction updates some of them during the execution of the first transaction.

* The serializability can be achieved in several ways.

* There are two main techniques :

(i) locking

(ii) time stamp based protocol.

(i) **Locking**

Locking is a procedure which is used to control concurrent access to data. When one transaction is accessing the database, a lock may deny access to other transaction to prevent incorrect results.

* There are two types of lock :

(i) shared lock. (only read) (S)

(ii) exclusive lock. (both read & write) (X).

lock is granted by concurrency control manager.

	S	X
S	T	F
X	F	F

{Compatible matrix}.

(ii) Shared Lock.

If a transaction (T_i) has obtained a shared lock (denoted by S) on data item (Q), then T_i can read, but cannot write on data item (Q).

(iii) Exclusive Lock.

If a transaction (T_i) has obtained an exclusive lock (denoted by X) on data item (Q), then T_i can both read and write Q.

*- Shared mode is compatible with shared mode, but not with exclusive mode.

*- An exclusive mode lock request has to wait until the currently held shared mode locks are released.

*- A transaction requests a shared lock on data item (Q) by executing Lock-S(Q).

*- A transaction requests an exclusive lock through Lock-X(Q).

*- A transaction can unlock a data item (Q) by unlock Q. instruction.

Ex: Serial

T_1		T_2			
lock-X(B) (T1 can read & write)		lock-S(A)		A	B
read(B)	//200	read(A)	//150	100	200
B = B-50	//150	unlock(A)			
write(B)	//150	lock-S(B)			
unlock(B)		read(B)	//150		
lock-X(A)		unlock(B)			
read(A)	//100	display(A+B)	//300		
A = A+50	//150				
write(A)	//150				
unlock(A)					

Ex: concurrent

(Not use of proper lock leads to inconsistency)

T_1	T_2	
lock-X(B)		
read(B)	//200	
B = B-50	//150	
write(B)	//150	
unlock(B)		
	lock-S(A)	
	read(A)	//100
	unlock(A)	
	lock-S(B)	
	read(B)	//150
	unlock(B)	
	display(A+B)	//250
lock-X(A)		
read(A)	//100	
A = A+50	//150	
write(A)	//150	

Incorrect output

Problems in case of locking

7/10/2015.

Deadlock:

T_3	T_4
lock-X(B)	//granted.
read(B)	
B = B-50	
write(B)	
	lock-S(A) (granted)
	read(A)
	lock-S(B) //not granted.
lock-X(A) (not granted)	

(Deadlock situation).

Starvation. (if the locks are not released).

T_1	T_2	T_3	T_4
	lock-S(A)	//granted	
lock-X(A)	//not granted so it have to wait.		
		lock-S(A)	//granted.

How to avoid starvation:

*- When a transaction (T_i) request a lot on data item (Q) in particular mode (M), the concurrency control manager will grant the lock when there is no other transaction holding a lock

on Q in a mode that conflicts with M.

*- There is no other transaction that is waiting for a lock on Q and that made its lock replaced before transaction (T_i).

Two-Phase Locking Protocol.

growing → acquire lock
shrinking → release lock.

According to the rules of this protocol, every transaction can be divided into two phases:

- growing phase in which it acquires all the locks it needed, but cannot release any locks and then, a shrinking phase in which it releases its locks, but cannot acquire any new lock.

Ex.

T_1	T_2		
requesting (wait) not granted. → lock-x(A)	lock-x(A)	// granted	A = 100
	read(A)	//100	
	A = A+100	//200	
	write(A)	//200	
	unlock(A)		
read(A)	//200		
A = A-10	//190		
write(A)	//190		
unlock(A)			

(lost update problem)

Strictly Two-Phase Locking Protocol

This protocol requires not only that locking be in two-phase, but also that all exclusive mode locks taken by a transaction will be held until transaction committs.

T_1	T_2	T_3
lock-X(A) read(A) write(A) commit unlock(A)		
	lock-X(A) read(A) write(A) commit unlock-(A)	
		lock-X(A) read(A) write(A) commit unlock-(A)

Rigorous Two-Phase Locking Protocol.

In case of rigrous two-phase locking protocol it requires that all lock will be held until the transaction commits.

How to handle the deadlock problem.

*- There are two principal methods for handling deadlock problem :

(i) deadlock prevention.

*- This protocol to ensure that the system will never enter a deadlock state

(ii) deadlock detection and recovery.

*- We allow the system to enter a deadlock state and then, try to recover by using deadlock detection and recovery.

Deadlock Prevention Methods.

1. Each transaction locks all its data items before it begins execution. Either all are locked in one step or not all locked.

Disadvantages.

1. It is often hard to predict before transaction begins what data item need to be locked.

2. Data item utilisation may be very low.

8/10/2015.

2. The second approach to for preventing deadlock is to use pre-emption and roll back transaction. To control the pre-emption, we assign a unique time stamp to each transaction. The system uses this time stamp only to decide whether a transaction should wait or roll back.

i) Wait - die. (Non-preemption).

*- When transaction (T_i) requests data item currently held by T_j, T_i is allowed to wait only if it has a time stamp smaller than T_j. (T_i is older than T_j). Otherwise, T_i roll back. (dies).

T_{22}	T_{23}	T_{24}
	lock - X(A)	(hold the resource).
lock - X(A)	(request).	

Time stamp (T_{22}) = 5
T_{23} = 6
T_{24} = 7.

T_{22} waits as its time stamp is less than T_{23}.

ii) Wound - Wait. (pre-emption technique).

When transaction (T_i) requests a data item currently held by T_j, T_i is allowed to wait only if it has a time stamp larger

than T_j. (T_i is younger than T_j).
Otherwise, T_j is rolled back. T_j is wounded by T_i.

T_{22}	T_{23}	T_{24}
	lock-x(A)	//hold
		lock-S(A) //wait as its time > T_{23}.

3. Time Based Scheme (Lock Time Out).

*- In this approach, the transaction that has requested a lot waits for atmost a specified amount of time. If the lock has not been granted within that time, the transaction is said to be time out. and it rolls itself back and restart.

Disadvantages.

1. It is hard to decide how far a transaction wait before time is out.

2. Too short wait results a transaction roll back even when there is no deadlock leading to wastage of resources.

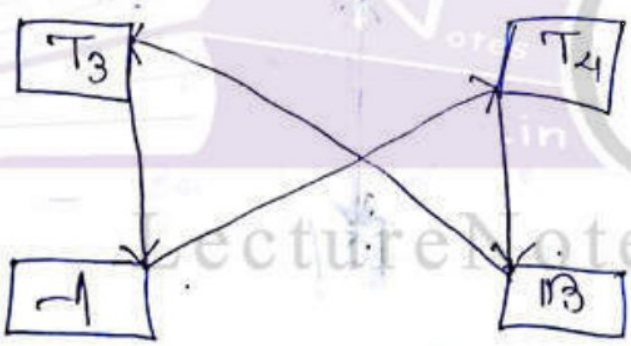
3. Starvation is also possible.

Deadlock Detection and Recovery

↑ by resource allocation and wait for graph.

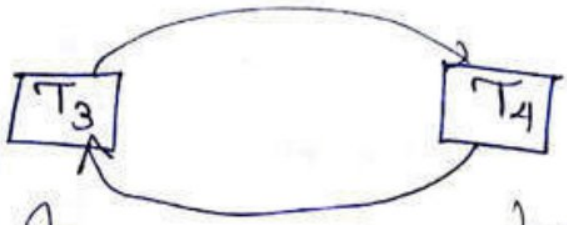
T_3	T_4
H — lock-X(B)	
	lock-S(A) — H
	lock-S(B) — R
R — lock-X(A)	

Draw the resource allocation graph and wait for graph, and check deadlock.



Resource Allocation Graph

If there is a path from T_3 to T_4, remove the resource.

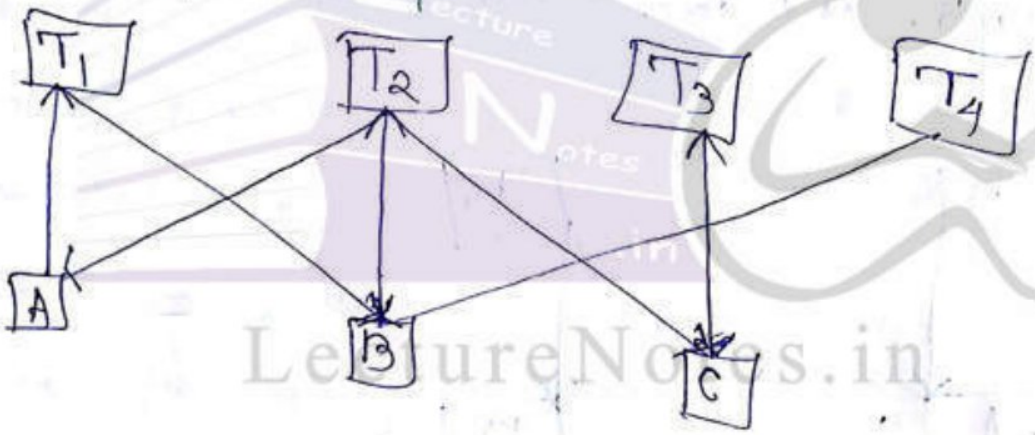


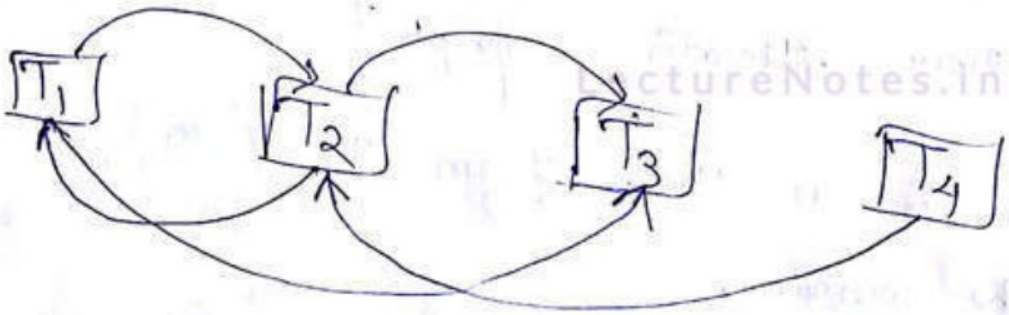
Wait for graph

As this graph contains a cycle

Q.

T_1	T_2	T_3	T_4
lock-S(A)			
read(A)			
	lock-X(B)		
	write(B)		
lock-S(B)			
		lock-S(C)	
		read(C)	
	lock-X(C)		
			lock-X(B)
	lock-X(A)		





LectureNotes.in

9/10/2015.

Deadlock Recovery.

1. How long the transaction has been running (it may be better to abort a transaction that has just started rather than one that has been running for some time).

2. How many data items have been updated by the transaction (it will be better to avoid a transaction that has made little change to the database rather than one that has made significant changes to the database).

3. How many data items the transaction is still to be updated (it will be better to abort a transaction that has many changes still to make to the database rather than one that has few changes to make).

How far to roll back a transaction.

1. Total Roll back.

* Abort the transaction and then, restart it.

2. Partial Roll back.

* It may be possible to resolve the

Atul Kumar 7777888822

deadlock by rolling back by rolling back only part of the transaction.

Avoiding Starvation.

*- Starvation occurs when the same transaction is always choosen as the victim and the transaction can be never completed.

*- The DBMS can avoid the starvation by storing a count of the no. of times a transaction has made selected as a victim and using a different selection criteria once the count reaches some proper bound.

Time Stamp Based Protocol.

A unique identifier created by the DBMS that indicates the relative starting time of transaction.

*- If a transaction have assigned time stamp (T_s (T_e)) and new transaction (T_j)

$T_s(T_i) < T_s(T_j)$.

*- There are two methods for implementing the scheme :

(i) use the value of the system clock as the time stamp, i.e., a transaction's time stamp is equal to the value of the clock when the transaction enters the system.

(ii) use a logical counter that is incremented after a new time stamp has been assigned that is a transaction time stamp is equal to the value of the counter when the transaction enters the system.

*- To implement this scheme, we associate with each data item (Q) two time stamp values :

i) read(Q)-timestamp(Q)

ii) write(Q)-timestamp(Q).

i) W-timestamp(Q).

*- It denotes the largest time stamp of any transaction that executed write-(Q) successfully.

iii) r-timestamp(Q):

* It denotes the largest time stamp of any transaction that executed read(Q) successfully.

Time Stamp Ordering Protocol:

1. Suppose the transaction (T_i) issues read(Q),

(i) If $TS(T_i) < w\text{-timestamp}(Q)$,

then T_i needs to read the value of Q that was already overwritten.

Hence, read operation is rejected and T_i is rolled back.

(ii) If $TS(T_i) \geq w\text{-timestamp}(Q)$,

then the read operation is executed and r-timestamp(Q) is set to the maximum of r-timestamp(Q) and $TS(T_i)$.

2.

Suppose the transaction (T_i) issues write(Q).

(i) If $(T_i) < R\text{-timestamp}(Q)$,

then the value of Q that T_i is producing was needed previously and the system assumed that the value would never be produced.

Hence, the system rejects the write operation and rolls T_i back.

(ii) If $TS(T_i) <$ w-timestamp(Q), then T_i attempting (old) to write an absolute value of Q.

Hence, the system rejects the write operation and rolls T_i back.

(iii) Otherwise, the system executes the write operation and sets the write-timestamp(Q) to $TS(T_i)$.

*— If transaction (T_i) is rolled back by concurrency controlled scheme as result of either read or write operation, the system assign a new time stamp and restart it.

Ex

T_1	T_2
read(A)	
	read(A)
write(A)	
read(B)	
write(B)	
	read(B)
	write(B)

max(0,5)

T_1 — 5
T_2 — 6.

initially:
r-timestamp(A) = 0
w-timestamp(A) = 0
r-timestamp(B) = 0
w-timestamp(B) = 0

T_2: 0 (x)

read(A) 5 (0 (x)

write(A) 5 ≥ 0

max(5,6)

r-timestamp(A) = 6

then, r-timestamp(A) = 5.

T_1 — 7
T_2 — 6

13/10/2015.

Query Processing and Optimization

→ size of the intermediate result shud be less.

Clauses : select, from, where, group by having.

$$\pi_{ename}\left(\sigma_{deptno > 30}(emp)\right)$$

Select all salary values > 5000;

Select sal from emp where sal > 5000;

$$\pi_{sal}\left(\sigma_{sal > 5000}(emp)\right)$$

or $$\sigma_{sal > 5000}\left(\pi_{sal}(emp)\right)$$

The process which checks that the keywords are correct or not, syntax is correct or not, validation.

scanner / scanning

validator

↓ parsing (parser).

↓ translated

Relational Algebra.

Evaluation of SQL Statement.

*- The query is evaluated in a different order:

(i) the tables in the from clause are combined using Cartesian products,

(ii) The where predicate is then applied.

(iii) The resulting tuples are grouped according to the group by clause.

(iv) The having predicate is applied to each group, possibly eliminating some groups.

(v) The aggregates are applied to each remaining group.

The select clause is performed last.

Overview of Query Processing.

1. Parsing and translation.
2. Optimization
3. Evaluation
4. Execution.

Query Processing:

A 3-step process that transforms a high-level query (of relational calculus/SQL) into an equivalent and more efficient lower-level query (of relational algebra).

1. Parsing and translation.

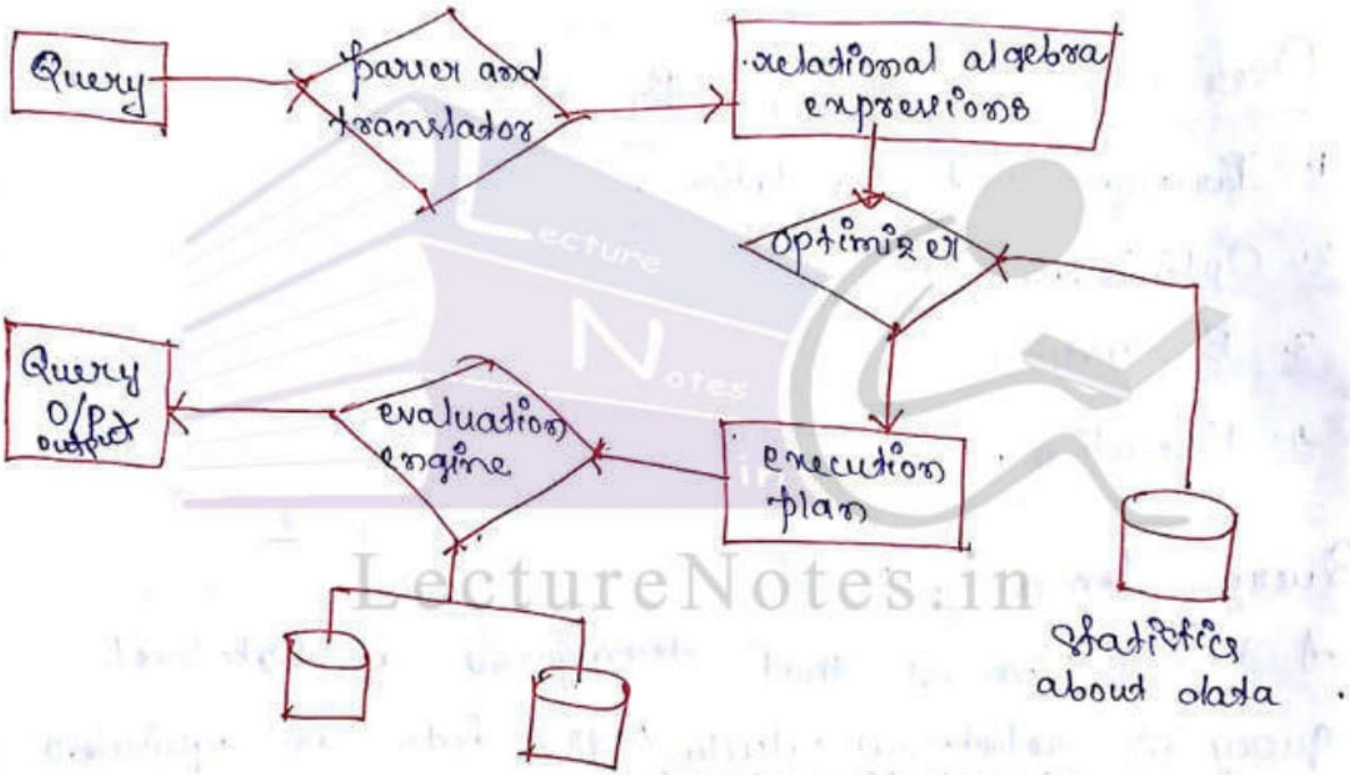
* Check syntax and verify relations.
* Translate the query into an equivalent relational algebra expression.

2. Optimization.

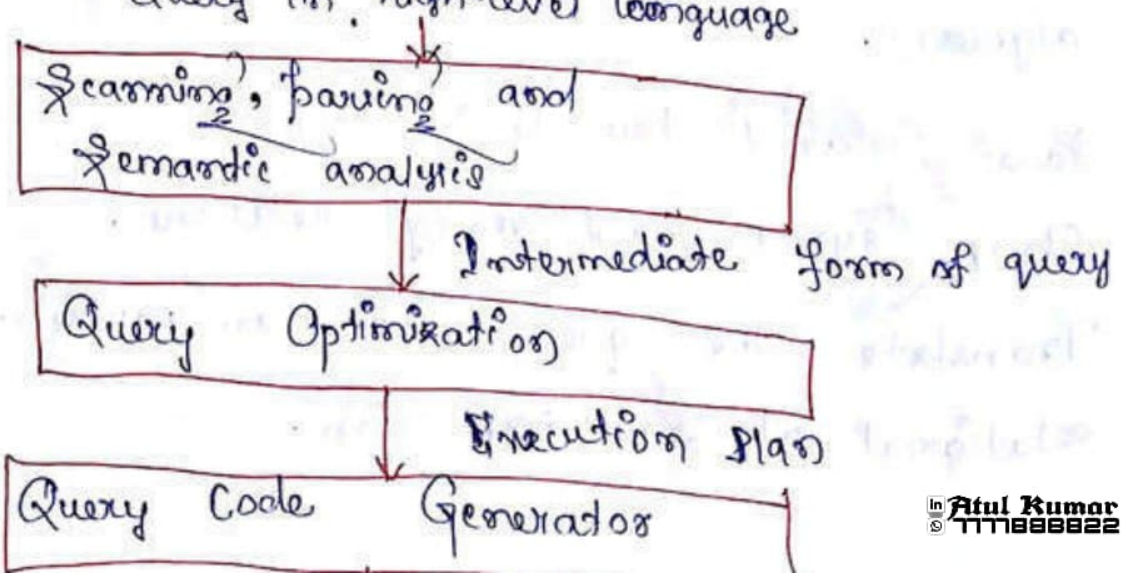
*- Generate an optimal evaluation plan (with lowest cost) for the query plan.

3. Evaluation.

The query execution engine takes an (optimal) evaluation plan, executes that plan and returns the answers to the query.

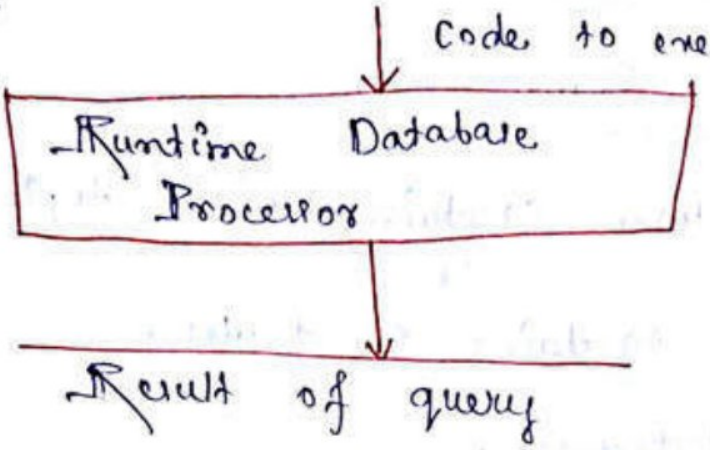


-x-



LectureNotes.in

Atul Kumar 7778888822



14/10/2015.

Cost : No. of disc accesses.

Consider a database with two relations :

Staff (staffno, name, position, branch no).

Branch (branch no, name, city).

Q. Find the details of all managers who work at London branch.

select * from Staff s ~~where~~ and Branch b where b.city = 'London' and s.branchno = b.branchno and s.position = 'manager';

c_3 c_1 c_2

$\sigma_{staff.branchno = Branch.branchno}$ (Staff

(i) $\sigma_{c_1 \wedge c_2 \wedge c_3}$ (Staff $\times$ Branch).

(ii) $\sigma_{c_2 \wedge c_3}$ (Staff $\bowtie_{c_1}$ Branch).

(iii) $(\sigma_{c_2}(\text{Staff})) \bowtie_{c_1} (\sigma_{c_3}(\text{Branch}))$.

Statistics about IData.

Let staff relation contains 1000 tuples

Branch relations contains 50 tuples.

There are 50 managers.

Let London city has 5 branches.

Assumptions:

1. Results of intermediate operations are stored back on to the disk.
2. Tuples are accessed one at a time from the disk.

Case-I:

$$\underset{\text{staff}}{1000} + \underset{\text{branch}}{50} + \underset{\substack{\text{join}\\ \text{to store}\\ \text{in disk}\\ \text{(intermediate storage)}}}{50{,}000} + \underset{\text{select}}{50{,}000} = 10{,}1050$$

$$\begin{array}{r} 50000 \\ 50000 \\ 1000 \\ 50 \\ \hline 101050 \end{array}$$

The 1st query calculates the cartesian product of staff and branch relation which requires 1000 + 50 no. of disc accesses to read the relations and creates a relation with (1000 × 50) no. of tuples.

We then, have to read each of these

tuples again to test them against the selection predicate which requires another (1000×50) disc accesses.

Therefore, the total no. of disc accesses

$= 1000 + 50 + 50,000 + 50,000$.

$= 101050$.

Case-II.

$1000 + 50 + 1000 + 1000 = 3050$.

Case-III.

restrictive selection. (those selection which are confined to one relation).

$1000 + 50 + 1000 = 2050$.

σ_{c_2} (staff) $\bowtie_{c_1}$ σ_{c_3} (branch)

$\downarrow$ 1000 $\qquad$ $\downarrow$ 50

5

50

$\bowtie_{c_1}$

50 → $\qquad$ 50 $\qquad$ ← 5

$1000 + 50 + 50 + 5 + 50 + 5 = 1160$.

$$\begin{array}{r} 1000 \\ 50 \\ 50 \\ 5 \\ 50 \\ \hline 1155 \end{array}$$

One of the fundamental strategies in query processing is to perform the unary selection and projection operation as early as possible.

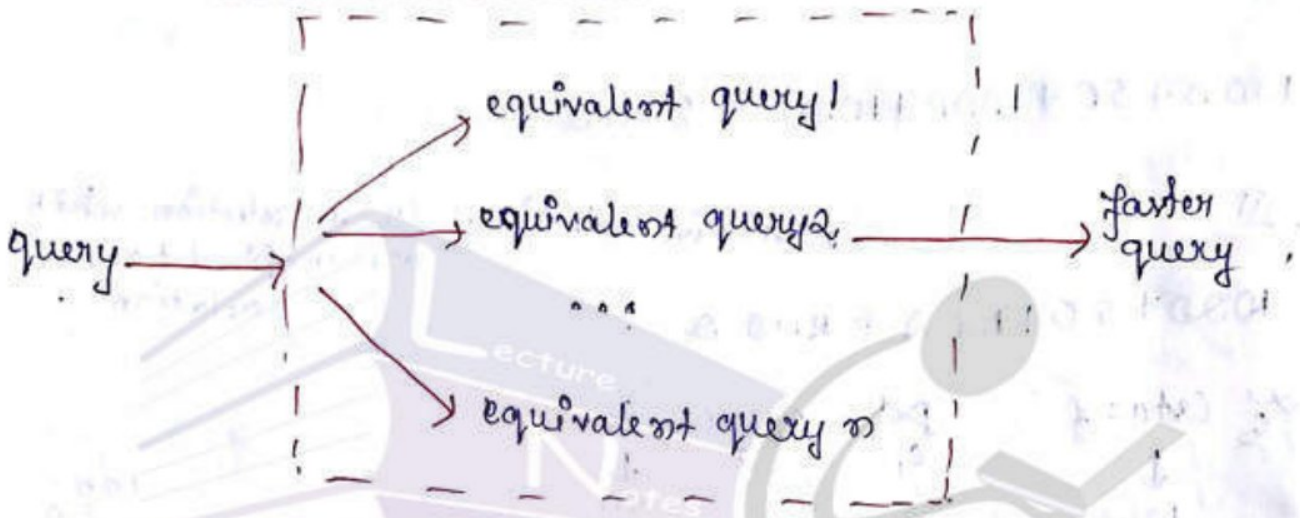
15/10/2015.

When we have various relational algebra from a sql query, we go for optimization.

- Relational Algebra → Query tree. (less no. of intermediate results)
- TRC & DRC → Query Graph.

Query Optimization.

Aim: Transform query into faster, equivalent query.



(i) Heuristic (logical) optimization.

* Query tree (relational algebra) optimization
  or
* Query graph optimization.

(ii) Cost-based (physical) optimization.

Heuristic Approach to Query Optimization.

A rule which is formulated and tested to be efficient.
→ not every time, most of the time.

It has no mathematical proof associated with it.

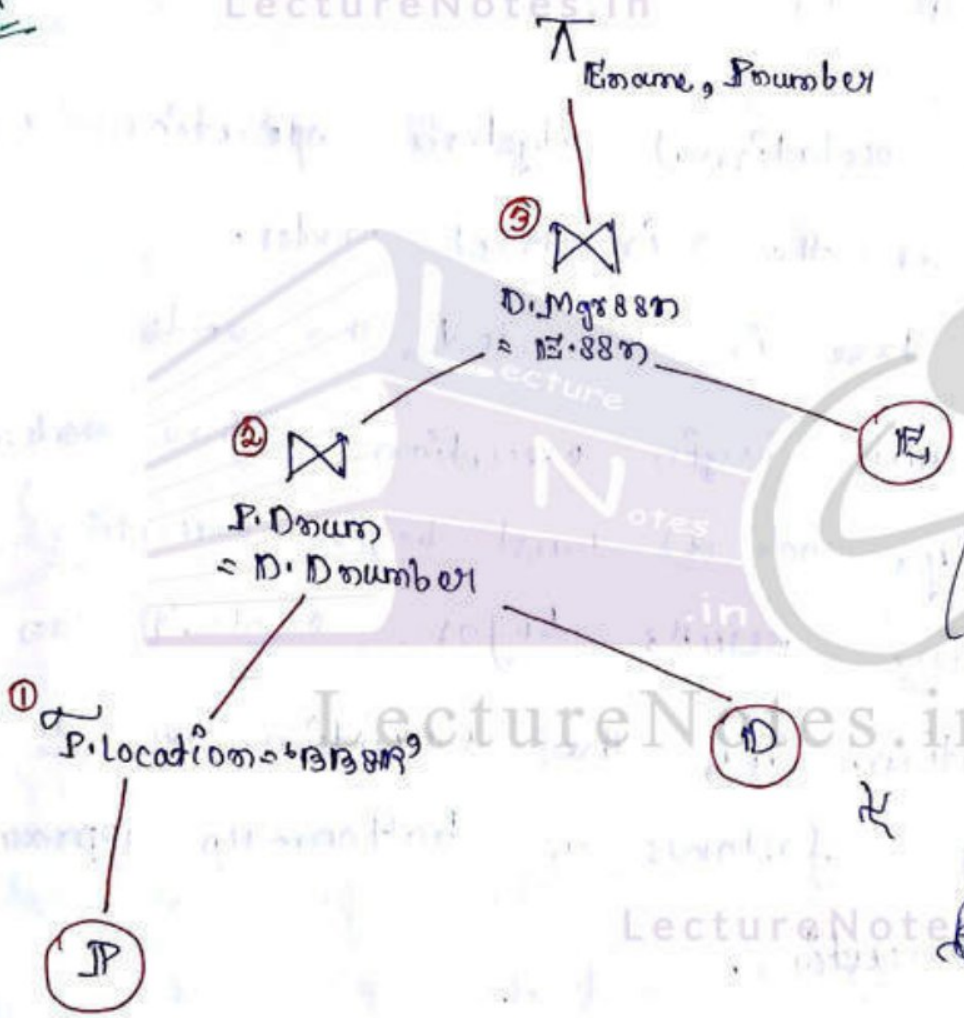
→ A heuristic is a rule that has been formulated, tested and found efficient.

→ A heuristic approach of query optimization is based on query trees.

Query Tree.

leaf → relⁿ name, internal → relⁿ operators, root → result.

Ex:



→ A query tree is a tree data structure that corresponds to a relational algebra expression.

It represents the input relations of the query as leaf nodes of the tree and

represents the relational algebra expression operations as internal nodes. The result is at the root node.

*- In the given query tree, three relations Project, Department and Employee are represented as P, D and E respectively.

*- While the relational algebra operations are represented as the internal nodes. When this tree is executed, the node marked ① must begin execution before node ② and similarly, node ② must begin executing and producing results before node ③ can start execution. i.e., the execution of a query tree follows a bottom-up approach for its execution.

*- Like a sql statement is each having many equivalent relational algebra expressions. Similarly, there can be several query trees which are equivalent to each other.

16/10/2015

Transformation Rule / Rule of Equivalence

1. Cascade of σ : $\sigma_{c_1 \wedge c_2 \wedge \dots \wedge c_n}(R) = \sigma_{c_1}(\sigma_{c_2}(\dots(\sigma_{c_n}(R))\dots))$.

2. Commutativity of σ : $\sigma_{c_1}(\sigma_{c_2}(R)) = \sigma_{c_2}(\sigma_{c_1}(R))$.

3. Commutating σ with π : $\pi_L(\sigma_c(R)) = \sigma_c(\pi_L(R))$.
   Only if c involves solely attributes in L.

4. Commutating σ with $\bowtie$: $\sigma_c(R \bowtie S) = \sigma_c(R) \bowtie S$.
   Only if c involves solely attributes in R.

5. Commutating σ with set operations :
   $\sigma_c(R \theta S) = \sigma_c(R) \theta \sigma_c(S)$.
   where θ is one of $\cup, \cap, -$.

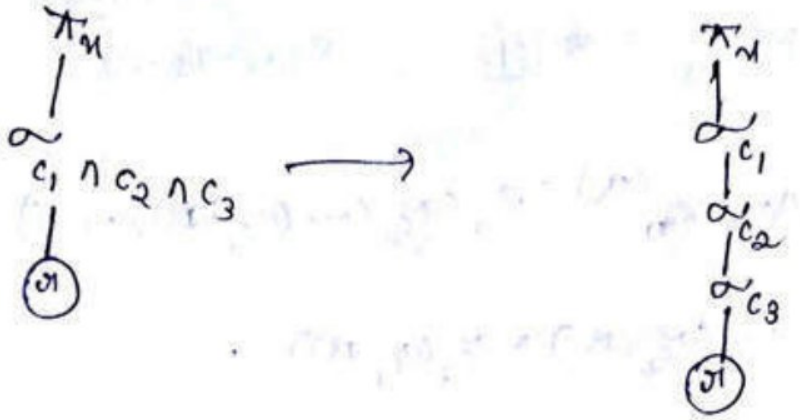
6. Commutativity of $\cup, \cap$, and $\bowtie$:
   $R \theta S = S \theta R$.
   where θ is one of $\cup, \cap, \bowtie$.

7. Associativity of $\bowtie, \cup, \cap$:
   $(R \theta S) \theta T = R \theta (S \theta T)$.

Ex 3. $\sigma_{sal > 5000}(\pi_{name, sal}(emp)) = \pi_{name, sal}(\sigma_{sal > 5000}(emp))$.

$\sigma_{sal > 5000}(\pi_{name}(emp))$ ×.

LectureNotes.in



Heuristic Transformation Algorithm.

Transform a query represented in relational algebra to an equivalent one (generates the same result).

Step I:

Decompose σ operation;

Step-II:

Move σ as far down the query tree as possible.

Step-III:

Rearrange leaf nodes to apply the most restrictive (σ) operations first.

Step-IV:

Form joins from $\times$ and subsequent σ operations.

Step-V:

Decompose π and move down the query tree as far as possible.

LectureNotes.in

Step VI :

Identify candidates for combined operations.

Ex: Consider the following database :

employee (SSN, Name, Bdate, address).

works_on (ESSN, Pno).

Project (Pnumber, Pname).

: Find the names of all employees born after 1976 who work on a project named "xyz".

Ans:

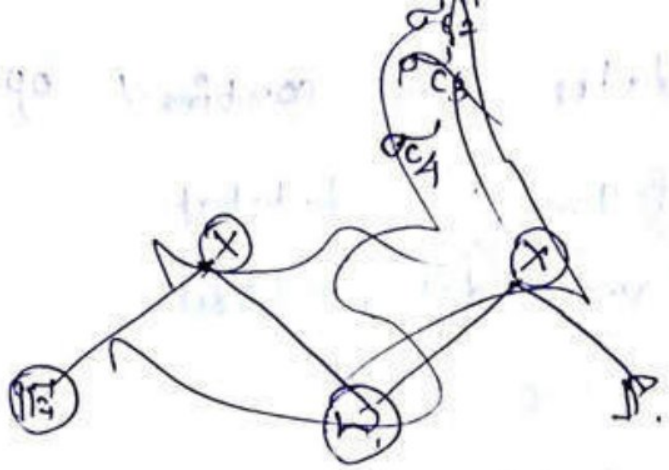
select name from employee e, works_on w, Project P where $\underbrace{P.Pname = \text{"xyz"}}_{C_1}$ and $\underbrace{P.Pnumber = w.Pno}_{C_2}$ and $\underbrace{w.ESSN = e.SSN}_{C_3}$ and $\underbrace{e.Bdate \geq \text{'31-DEC-1976'}}_{C_4}$;

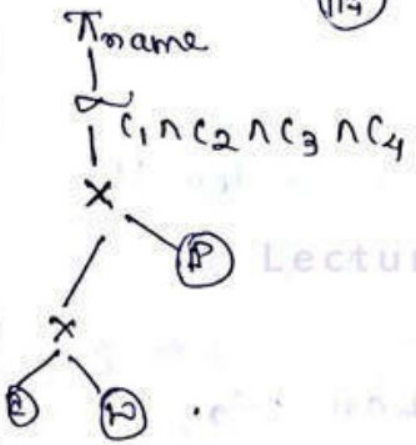
Relational Algebra :

$\sigma_{C_1 \cap C_2 \cap C_3 \cap C_4}$ (employee × works).

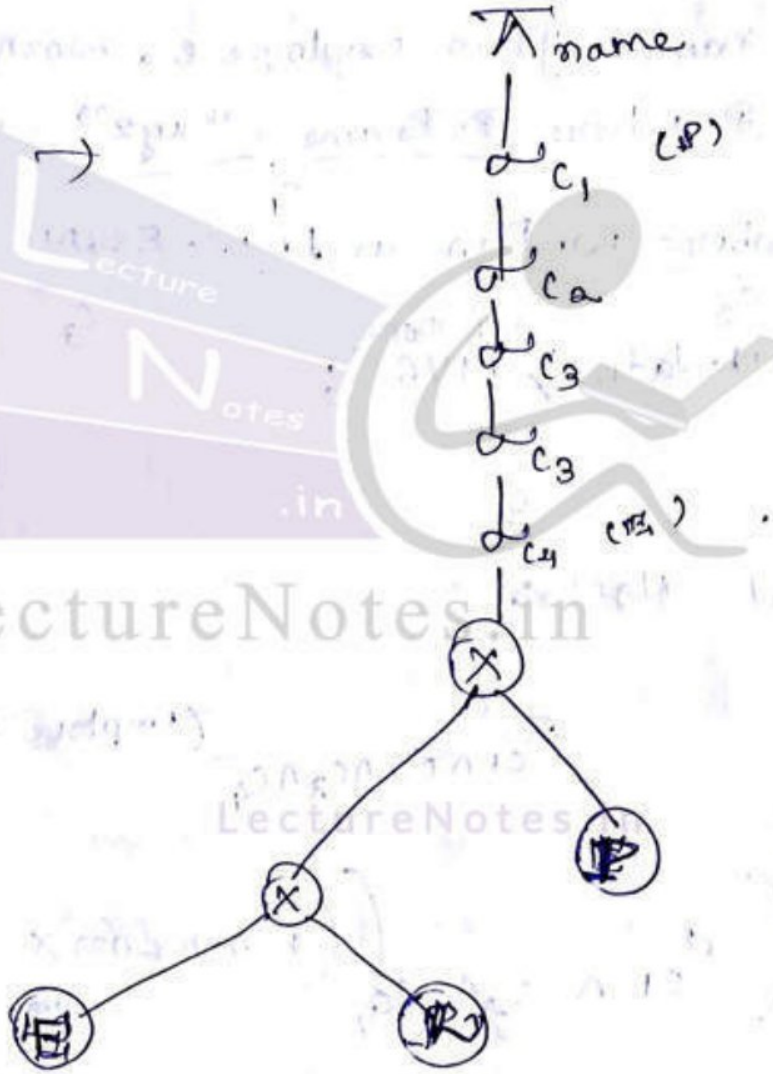
$$\pi_{name}\left(\sigma_{C_1 \cap C_2 \cap C_3 \cap C_4}\left((\text{works_on} \times \text{Project}) \times \text{employee}\right)\right)$$

or, $\pi_{name}(\sigma_{C_1 \cap C_2 \cap C_3 C_4}(E \times W \times P))$.





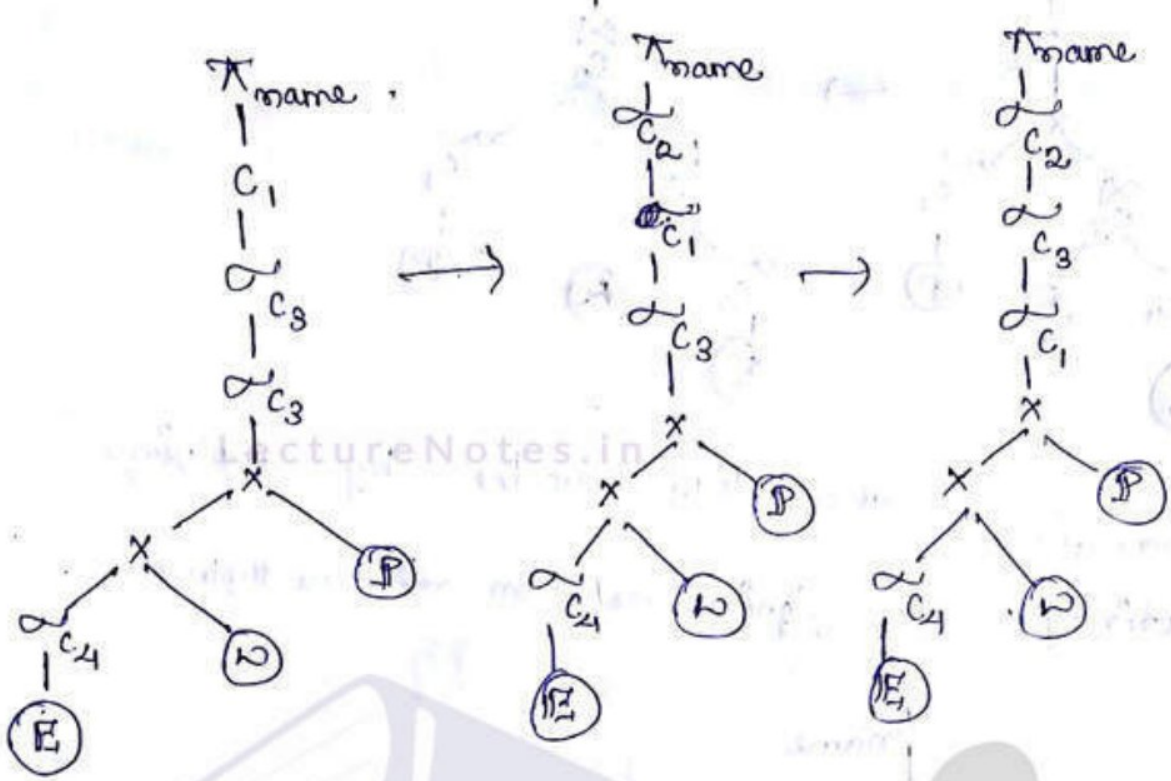
After rule-1 →

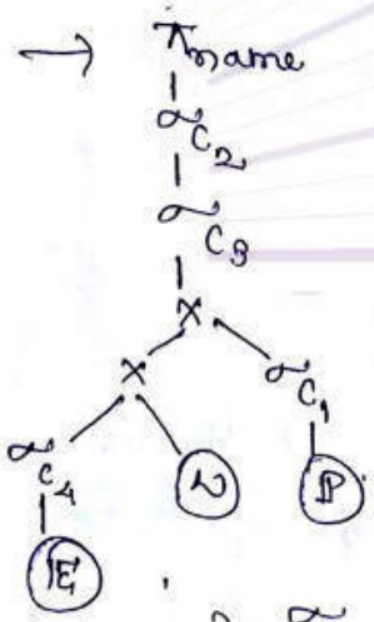


LectureNotes.in

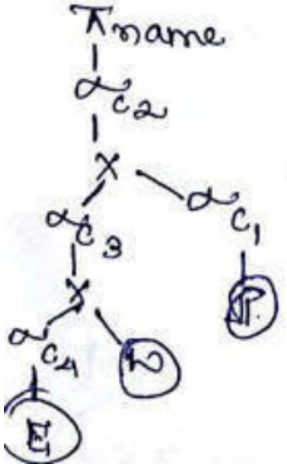
Step-3:

Moving C_1 down.



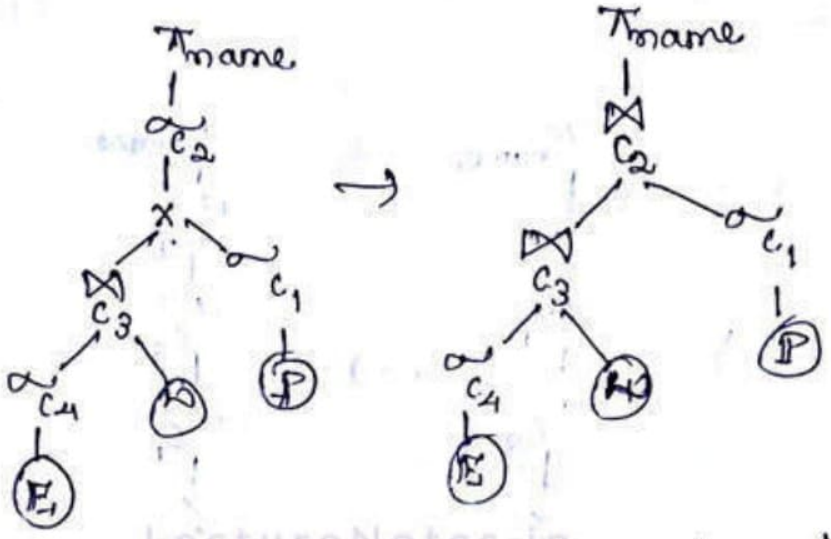


Moving σ_{C_3} down.

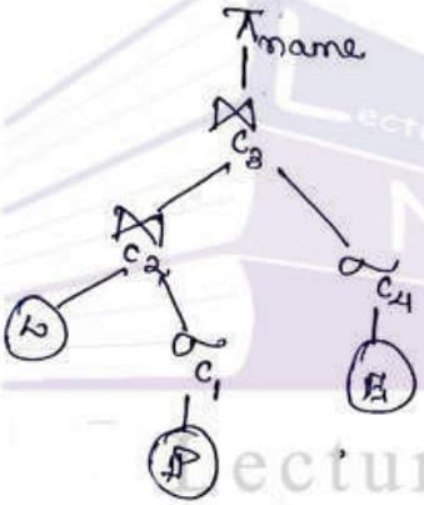


LectureNotes.in

As $\times$ and σ is $\bowtie$.



Rearranging the leaf nodes by applying associativity. (join relation with less tuples ①).



20/10/2015

Q. Find the customer name whose branch city = BBSR and balance > 1000.

Ans:

Branch (bname, bcity, assets).

Account (acno, bname, balance).

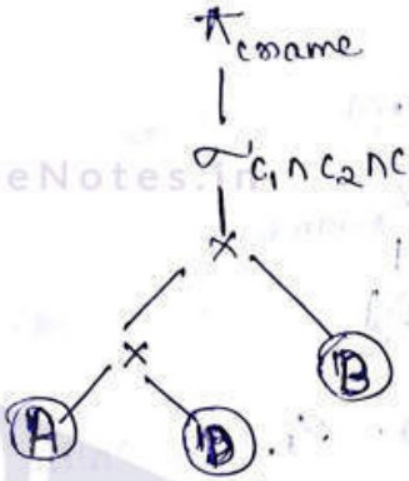
Depositor (cname, acno).

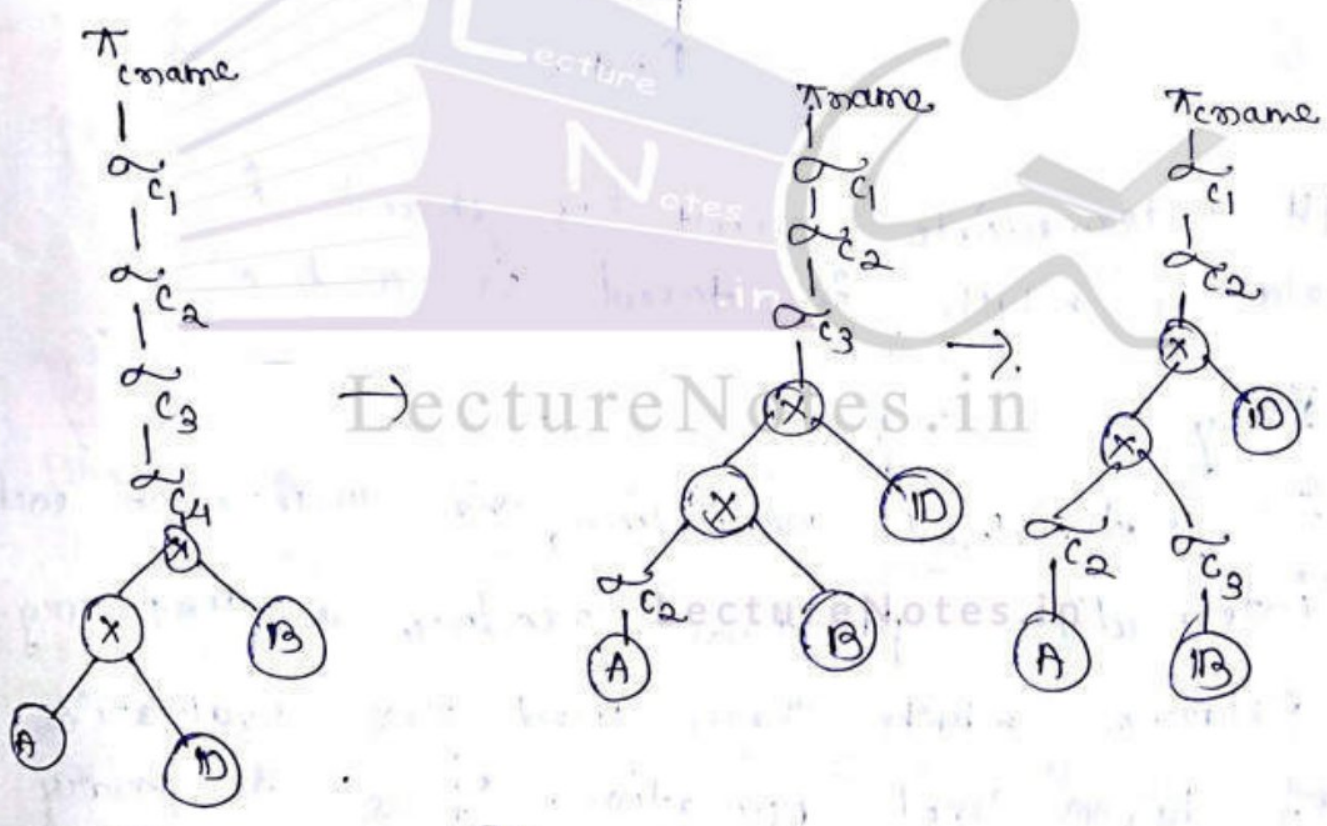
LectureNotes.in

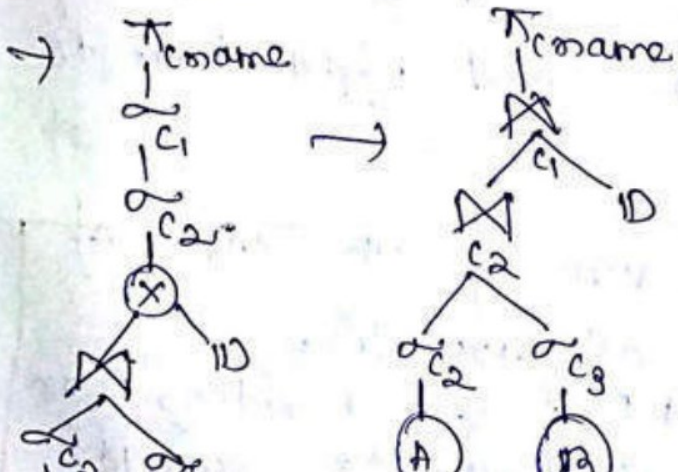
Atul Kumar
7777888822

select cname from Depositor d, Branch b, Account a
where b.bcity = 'BBSR' and a.balance > 1000 and
a.acno = d.acno c_1 and b.bname = a.bname; c_2
c_3 c_4

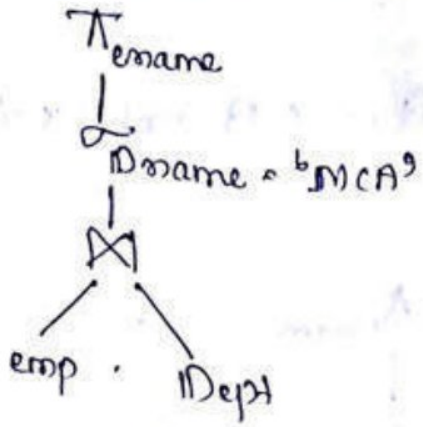
$$\pi_{cname}\left(\sigma_{c_1 \wedge c_2 \wedge c_3 \wedge c_4}\left((\text{Depositor} \times \text{Branch}) \times \text{Account}\right)\right)$$



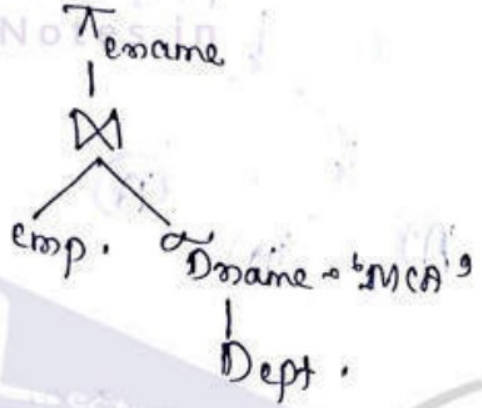




Difference between materialisation and pipelining



Optimised query :



All intermediate result is stored in a relation which is stored in a disk.

In materialised evaluation, the results of each intermediate operation produce a temporary relation which then used for evaluation of higher level operation. These temporary relation vary in size and might have to store in disk.

Cost of evaluation = sum of the cost of all operation + cost of writing or reading the result

of intermediate results to disk.

We can improve query evaluation by reducing a number of temporary relations that are produced. This reduction can be achieved by combining some several relational operations into pipeline of operations in which results of one operation are passed along to the next operation in the pipeline.

- Pipelines can be executed in either of the two types :
  (i) demand driven
  (ii) producer driven.

i) Demand driven.

- The system makes repeated requests for tuples from the operation at the top of the pipeline.
- Each time that an operation receives a request for tuples, it computes the tuples to be returned and returns that tuple.

iii) Producer Driven :

*- Operations do not wait for requests to produce tuples. But, instead generate the tuples eagerly.

*- Each operation at the bottom of the pipeline generate output tuples and put them in its output buffer until the buffer is full.

*- An operation at any level of a pipeline generate output tuples when it get input tuples from lower down in the pipeline until its outer buffer is full.

*- Once the operation uses tuples from pipeline ~~for~~ input, it removes the tuple from its input buffer.

*- In either case, once the output buffer is full, the operation waits until its parent removes tuple from buffer so that buffer has space

LectureNotes.in

for tuples.

29/10/2015.

Database Recovery.

software failure
hardware failure
deadlock
starvation

* A computer system like any other device is subjected to failure from a variety causes like disk crash, power failure, software error and a fire in the machine room.

* In any failure, information may be lost. Therefore, database system must take actions in advance to ensure atomicity and durability property of transactions.

Different types of failure

i) Transaction Failure.

ii) Logical Error. (because of bad input, overflow condition, resource limit exceeded).

iii) System Error. (because of deadlock).

iv) Sqrt

v) System Crash. (hardware).

vi) Disk Crash Failure.

A disk block loses its content as a result of either head crash or failure during data transfer operation.

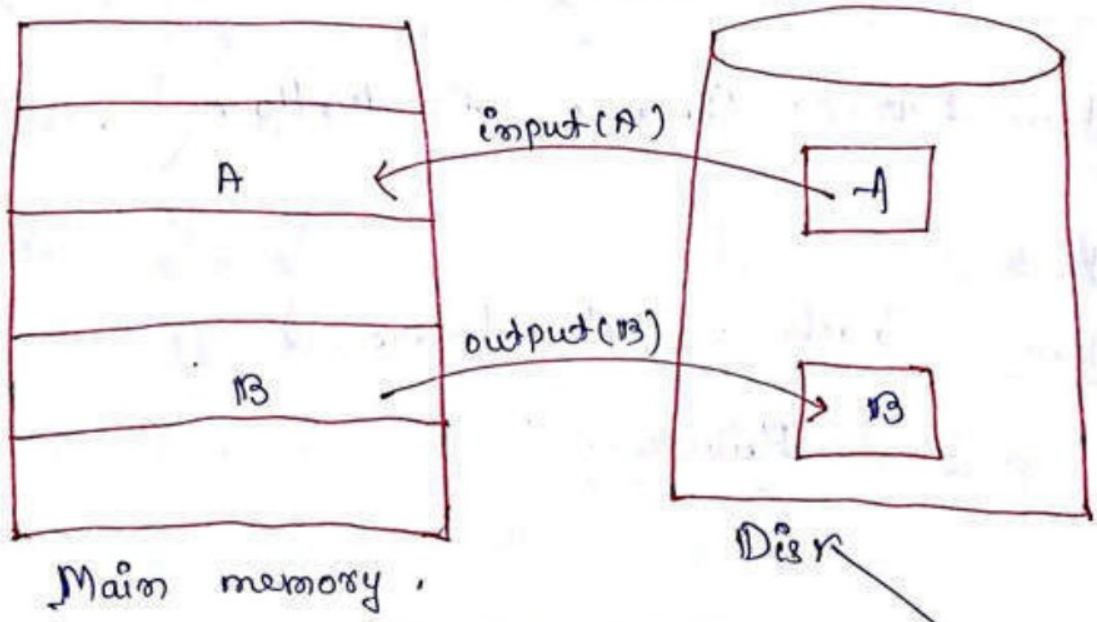
Storage Structure.

1. Volatile Storage. (main memory)
2. Non-Volatile Storage. (Disk)
3. Stable Storage (failure, data are there).

Data Access.

- The database system resides permanently on non-volatile storage and partitioned into fixed length of blocks.

Blocks are the units of data transfer to and from disk may contain several data items.



- The blocks residing on the disk are referred to as physical blocks.

The blocks residing temporarily in the main memory are referred to as buffer block.

- The area of memory where blocks reside temporarily is known as disc buffer.

LectureNotes.in

29/10/

· Block Movement between disk and Main memory.

*- Each transaction (T_i) has a private work area in which copies of all data item accessed and updated by T_i are kept.

*- Each data item (x) kept in the ~~data~~ work area of transaction (T_i) is denoted by x_i.

Read(x).

It assigns the value of data item (x) to the local variable (x_i). It executes this operation as follows:

(i) if block (B_x) on which x resides is not in the main memory, then issues input (B_x).

(ii) it assigns to x_i the value of x from the buffer block.

Write(x).

It assigns the value of local variable (x_i), the data item (x) in the buffer block. It executes this operation as follows:

(i) if block (B_x) on which x resides is not in the ~~memor~~ main memory, it issues input (B_x).

(iii) it assigns the value of V_i in buffer block (B_x).

Recovery and Atomicity :

read(A)			re-executing
A = A − 50	A = 1000		
write(A)	B = 2000		
read(B)			
B = B + 50			
write(B)	3000	2950	2950
A + B			

* Two techniques are used to achieve atomicity property :

(i) log based recovery. log file is in stable storage.

(ii) shadow paging.

30/10/2015

(i) Log Base Recovery.

The log is a sequence of records recording all the update activities in the database.

* An update log record describes a single database write. It contains a

(i) transaction identifier.
(ii) data item identifier.
(iii) old value identifier.
(iv) new value identifier.

(i) Transaction identifier.
It is the unique identifier of the transaction that perform write operation.

(ii) Data item identifier.
It is the unique identifier of the data item written

(iii) Old Value
It is the value of the data item prior to the write operation.

(iv) New Value.
It is the value of data item will have after the write operation.

Various Types of Log Records.

$\langle T_i$ start$\rangle$.

$\langle T_i, x_j, V_1, V_2 \rangle$.

$\langle T_i$, commit$\rangle$

LectureNotes.in

$\langle T_i$ start$\rangle$

Transaction (T_i) has started.

$\langle T_i, x_j, V_1, V_2 \rangle$

Transaction (T_i) has performed a write operation to the data item (x_j). V_1 is the previous value of x_j before modification. V_2 is the modified value of x_j after modification.

$\langle T_i$ commit$\rangle$

Transaction (T_i) has committed.

$\langle T_i$ aborted$\rangle$

Transaction (T_i) has aborted.

Two Techniques Under Log Based.

1. Deferred database modification.
   ↓
   late database modification

- It ensures transaction atomicity by recording all the database modification in the log, but deferring the execution of all write operations of a transaction, until the transaction partially commit.

*- If the system crashes before the transaction completes its execution or if the transaction aborts, then information on the log is simply ignored.

	T_0	T_1	log record
	read(A)	read(C)	$\langle T_0$ start$\rangle$
A = 1000	A = A − 50	C = C − 100	$\langle T_0, A, 950\rangle$
B = 2000	write(A)	write(C)	$\langle T_0, B, 2050\rangle$
C = 700	read(B)		
	B = B + 50		$\langle T_0$ commit$\rangle$
	write(B)		$\langle T_1$ start$\rangle$
			$\langle T_1, C, 600\rangle$
			$\langle T_1$ commit$\rangle$

redo(T_i) operation to maintain atomicity.

It sets the value of all data items updated by transaction (T_i) to the new value.

*- After a failure, the recovery sub-system consult the log to determine which transaction need to be redone.

*- Transaction (T_i) need to be re-done if and if the log record contains both T_i start and T_i commit.

* If the system crashes after transaction completes its execution, the recovery scheme uses the information in the log to restore the system to a ~~few~~ previous consistent state after transaction had completed.

<T0 start>	<T0 start>	<T0 start>
<T0, A, 950>	<T0, A, 950>	<T0, A, 950>
<T0, B, 2050>	<T0, B, 2050>	<T0, B, 2050>
(no commit so no redo operation)	<T0 commit> (redo)	<T0 commit> redo
	<T1 start>	<T1 start>
	<T1, C, 600>	<T1, C, 600>
	failure } no redo	<T1 commit> redo
		failure

2. Immediate Database Update Modification.

* It allows database modification to be output to the database while the transaction is still in the active state.

Handling the Failure.

- The recovery scheme uses two procedures:
  (i) undo (T_i)
  (ii) redo (T_i).

ii) undo(T_i)

It restores the value of data items updated by transaction (T_i) to the old value.

iii) redo(T_i).

It sets the value of all data items updated by transaction (T_i) to the new values.

*- Transaction(T_i) needs to be undo if the log record contains T_i start but, no T_i commit.

*- Transaction(T_i) needs to be redo if log record contains both T_i start and T_i commit.

- savepoint	savepoint	savepoint
<T_0 start>	<T_0 start>	<T_0 start>
<T_0, A, 1000, 950>	<T_0, A, 1000, 950>	<T_0, A, 1000, 950>
<T_0, B, 2000, 2050>	<T_0, B, 2000, 2050>	<T_0, B, 2000, 2050>
- failure	<T_0 commit> redo	<T_0 commit> redo
T_0 - undone	savepoint	savepoint
	<T_1 start>	<T_1 start>
	<T_1, C, 700, 600>	<T_1, C, 700, 600>
	undo	<T_1 commit>
		redo.

checkpoint is used to check from bottom

Check point.

* When a system failure occurs, we must consult the log to determine those transaction that need to be redone and need to be undone.

* There are two major difficulties with this approach:

i) such process is time consuming.

ii) most of the transactions that according to the algorithm need to be redone have already written their updates into the database.

LectureNotes.in

How to apply checkpoint.

After a failure has occurred, the recovery scheme examines the log to determine the most recent transaction. Transaction (T_i) that started executing the most recent checkpoint.

i) It can find such a transaction by searching the log backward from

the end of the log untill it finds the first checkpoint.

Then, it continues to search backward. It finds the next T_i start record.

Once the system has identified the transaction (T_i), redo and undo operation need to be applied to only transaction (T_i) and all transactions T_j that started executing after transaction (T_i).

Let us denote these transaction by the set(T). For immediate modification technique, the recovery operations are:

For all transactions (T_k) in $T = \{T_1, T_2, T_3, \ldots\}$ that have no $\langle T_k$ commit$\rangle$ record in the log execute undo (T_k).

For all transaction (T_k) in T, such that the record $\langle T_k$ commit$\rangle$ appears in the log execute redo (T_k).

Month. (5'4) shadow → main memory
current → stable storage.

(iii) Shadow Paging

The database is partitioned into a number of fixed length blocks which are referred to as pages.

Assume that there are n pages.

These pages do not need to be stored in particular table. The page table has n entries for each database page. Each entry contains a pointer to a page on disk.

The shadow paging technique is to maintain two page table during the life of the transaction:

i) Current page table.
ii) Shadow page table

*- When transaction starts both page tables are identical.

*- The shadow page table is never changed over the duration of transaction.

*- The current page table may be changed when a transaction performs

Suppose the transaction (T_i) performs a write(x) operation that x resides on the ith page. The system executes the write operation as follows:

1. if the ith page (i.e., the page on which x resides) is not already in main memory, the system issues input(x).

2. If this is the write first performed on the ith page by this transaction, then the system modifies the current page table as follows:

   (i) it finds the unused page on disk.

   (ii) it deletes the page in step 2(i), from the list of free page frame. It copies the content of the ith page to page found in step-2(i).

   (iii) it modifies the current page table so that, the ith entry points to the page found in step-2(i).

3. It assigns the value of x_i to x in the buffer page.

Atul Kumar
7777888822

The shadow page approach to recovery is to store the shadow page table in non-volatile storage so that, the state of the database prior to the execution of the transaction can be recovered in the event of a clash or transaction abort.

6/11/2015.

dirty page → pages that are not updated in the disk.

Another recovery technique - ARIES.

*- When the recovery manager is invoked after a crash, restart proceeds in three phases:

i) analysis phase.
ii) redo phase
iii) undo phase

i) Analysis Phase.

*- In case of analysis phase, it identifies the dirty pages in the buffer pool (i.e., changes that have not been written to the disk) and active transactions at the time of the crash.

ii) Redo Phase.

- Repeat all actions starting from the log in the log file and

restores the database state to what it was at the time of the crash.

Undo Phase.

*- Undo the actions of transactions that did not commit so, that the database reflects only the actions of committed transactions.

LSN → Log sequence number.

Ullman
Raghuram
Krishnan.

10 Update T_1 write P_5 → dirty page → redo → undo

20 Update T_2 write P_4 → ~~dirty page for T_2~~ → not dirty page as commit is there.

30 T_2 commit.

40 T_2 end.

50 Update T_3 write P_1. → dirty → redo → undo

60 Update T_3 write P_3. → dirty → redo → undo.

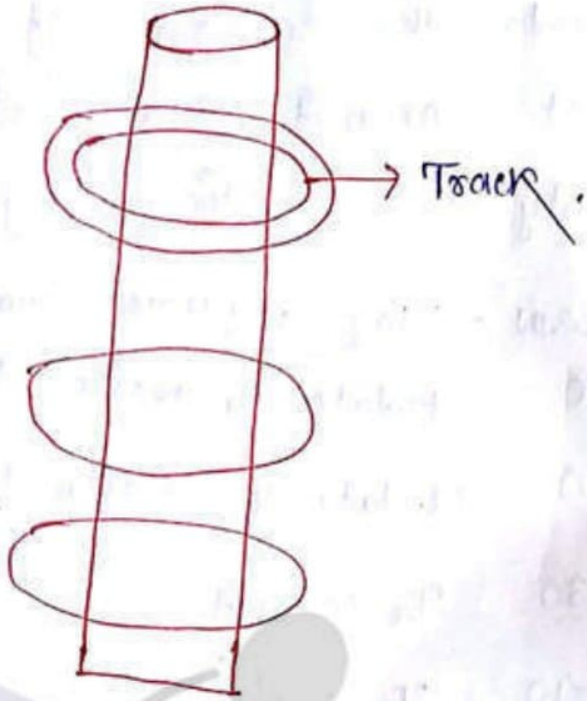
Crash Restarted.

LectureNotes.in
Atul Kumar
7777888822
LectureNotes.in

File Organization and Index.

*- Databases are stored physically as files or records which are typically stored on magnetic disk.

*- The division of a track into equal sized disk blocks (pages) is set by the operating system during disk formatting.



*- The typical block size is 512 to 4096 bytes.

*- The records of a file (file → record → fields) must be allocated to disk block because a block is the unit of data transfer between disk and main memory.

Atul Kumar
7777888822

Indexing Structure to the files.

Indexes.

Indexes are used to speed up the retrieval of records in response to the certain search condition.

*- To find a record or records in the file, certain selection criteria is done on indexing field.

*- Index field points to one or more blocks in the file where required records are located.

Different types of Index.

1. Single level Index :- for a file with a given record structure consists of several fields. Index structure is usually defined on a single field of a file called indexing file. The index typically stores each value of the indexing field along with the list of pointers to all disk blocks that contain records with field value.

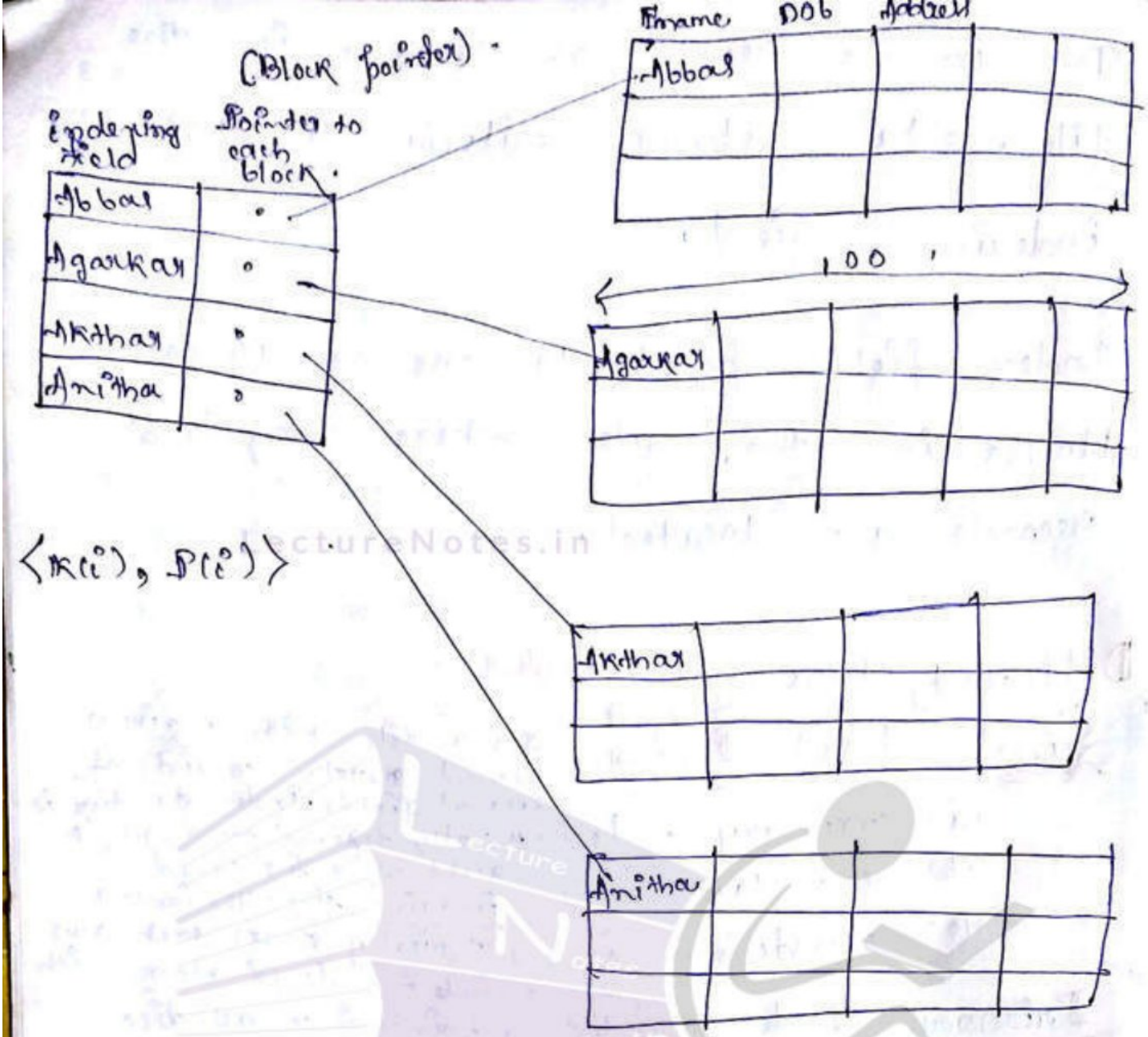
(a) primary index
(b) secondary index
(c) clustering index

(a) Primary Index :- same

A primary index is an ordered file whose records are of fixed length with two fields.

*- First field is the same data type as ordered key field called primary key of the data file.

*- Second field is the pointer to the disk block.



$\langle K(i), P(i) \rangle$

* Each index entry has the value of the primary key value for the 1st record in a block and a pointer to that block as its two field values.

* Two field values of index entries are $\langle K(i), P(i) \rangle$

* The total number of entries in the index is the same as number of disc blocks in the ordered data file.

*- The first record in each block of the data file is called anchor record of the block or its simply called block anchor.

*- Indexes can be categorised as

i) dense

ii) sparse.

i) Dense Index.

A dense index has an index entry for every search key value in the data file.

ii) Sparse Index / Non-Dense Index.

It has an index entries for only some of the search values.

*- Binary search is applied $T(n) = O(\lg n)$. 9/11/2015.

Problem.

Find out number of block access required using primary access.

Ans:

No. of records (r) = 30,000.

Block size (B) = 1024 bytes.

Record size (R) = 100 bytes.

Atul Kumar
7777888822

Blocking factor of r (Bfr)

$$Bfr = \left\lfloor \frac{1024}{100} \right\rfloor = 10.$$

No. of blocks required

$$b = \frac{30,000}{10} = 3,000.$$

Searching time.

$\log_2 3000 = 12$ blocks.

using indexing.

$B = 1024$ $R = 15$ bytes,

Ans:

No. of records for each block (bfr)

$$= \left\lfloor \frac{1024}{15} \right\rfloor = 68.$$

$\frac{r}{68}$

No. of blocks required

$$b_i = \left\lceil \frac{3000}{68} \right\rceil = 45 \text{ blocks},$$

Searching time $= \log_2 b_i = \log_2 45 = 6$ block access.

Total no. of block access is required

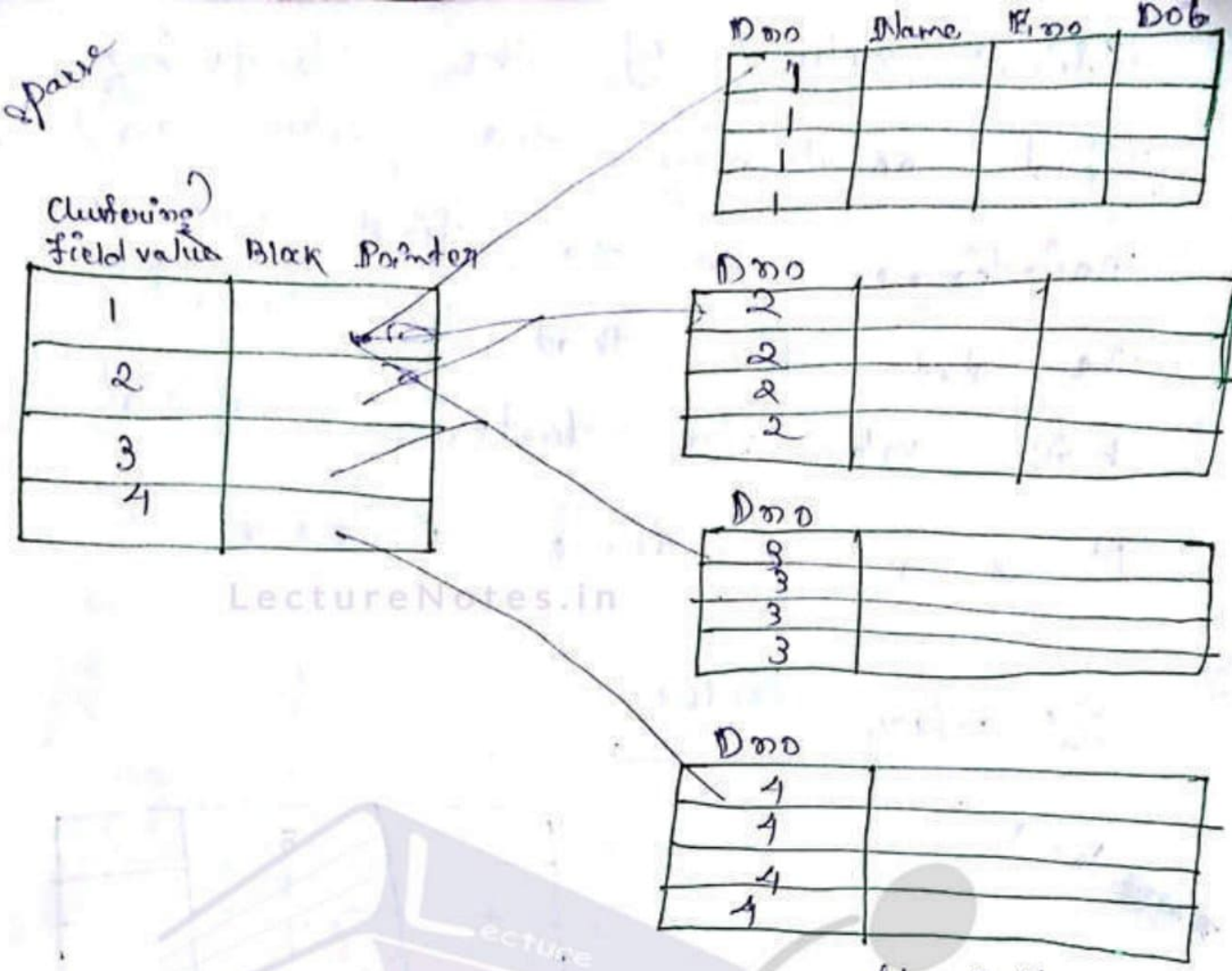
$= 6+1$

$= 7$ block accesses.

(b) Cluster Ordering.

$$\begin{array}{r} 68 \\ \times\ 45 \\ \hline 340 \\ 272 \\ \hline 3060 \end{array}$$

sparse



*- If records of a file are physically ordered on a non-key field which does not have a distinct value for each record, that field is called clustering field. It is also an ordered file with two fields.

*- The first field is of same type as clustering field of the data file and the second field is the block pointer.

*- There is an entry for each

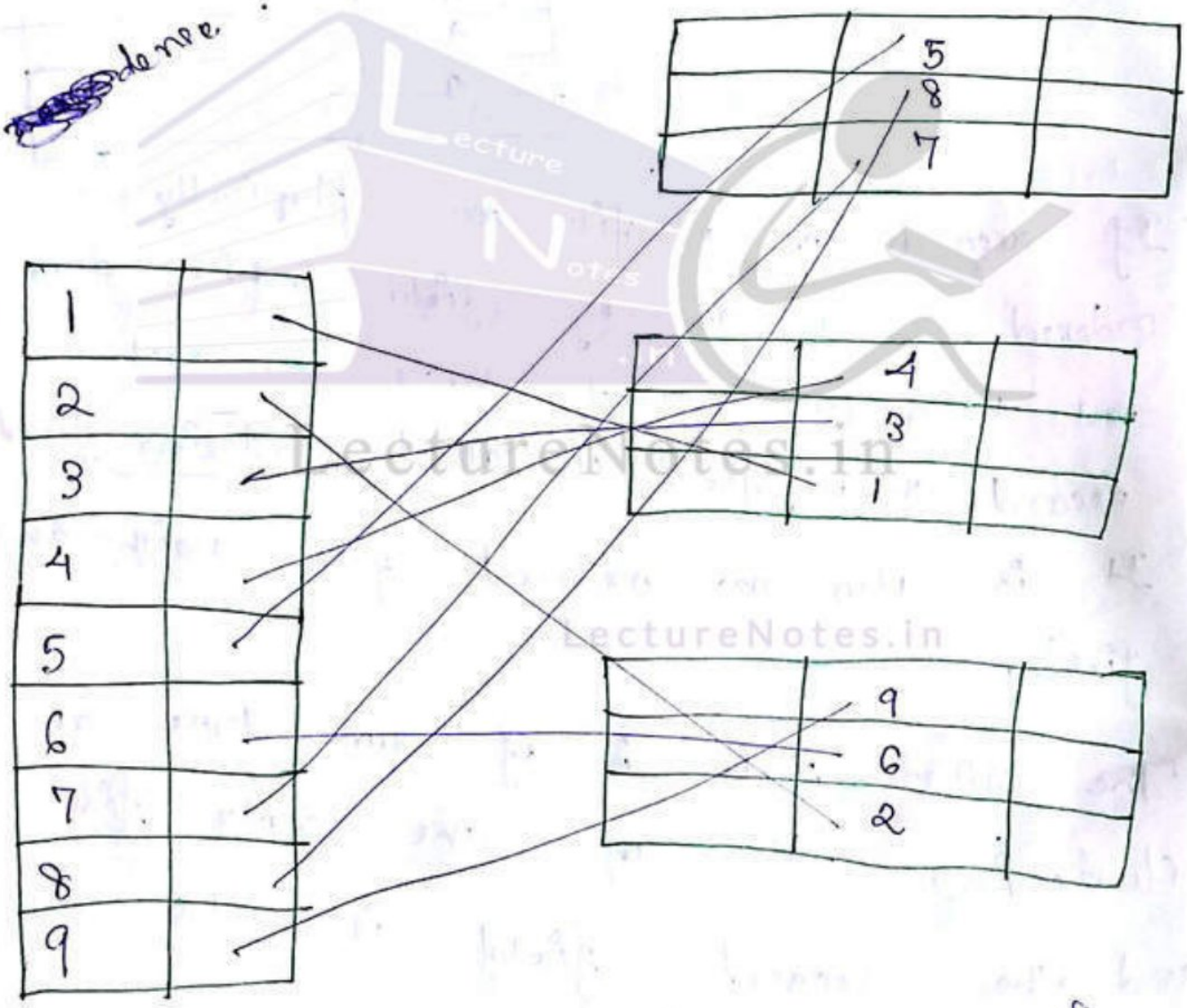
Atul Kumar
7777888822

distinct value of the clustering field containing the value and pointers to the first block in the data file that has record with value for clustering field.

It is a non-dense index.

Secondary Index.

dense



* It provides a secondary means of accessing a file for which some

*- The secondary index may be on a field which is a candidate key and has a unique value in every record.

*- The secondary index is an ordered file with 2 fields:
the 1st field is of same data type as non-ordering field of the data file i.e., an indexing field.
2nd field is either a block pointer (using candidate key) or record pointer.
Such an index is called dense index.
(linear search is applied as not ordered)

Q. r = 30,000
R = 100 bytes
B = 1024 bytes.

Ans:
No. of records in each block

$$bfr = \left\lfloor \frac{1024}{100} \right\rfloor = 10$$

No. of blocks required

$$b = \frac{30,000}{10} = 30000$$

If we are applying linear search,

average = $3000/2 = 1500$ block accesses are required.

Applying secondary index.

Block size = 1024 bytes.

Record size(r) = 15 bytes.

$bfr = \left\lfloor \frac{1024}{15} \right\rfloor = 68$ entries per block.

No. of blocks required (b_i)

$= \left\lceil \frac{30,000}{68} \right\rceil = 442$ blocks.

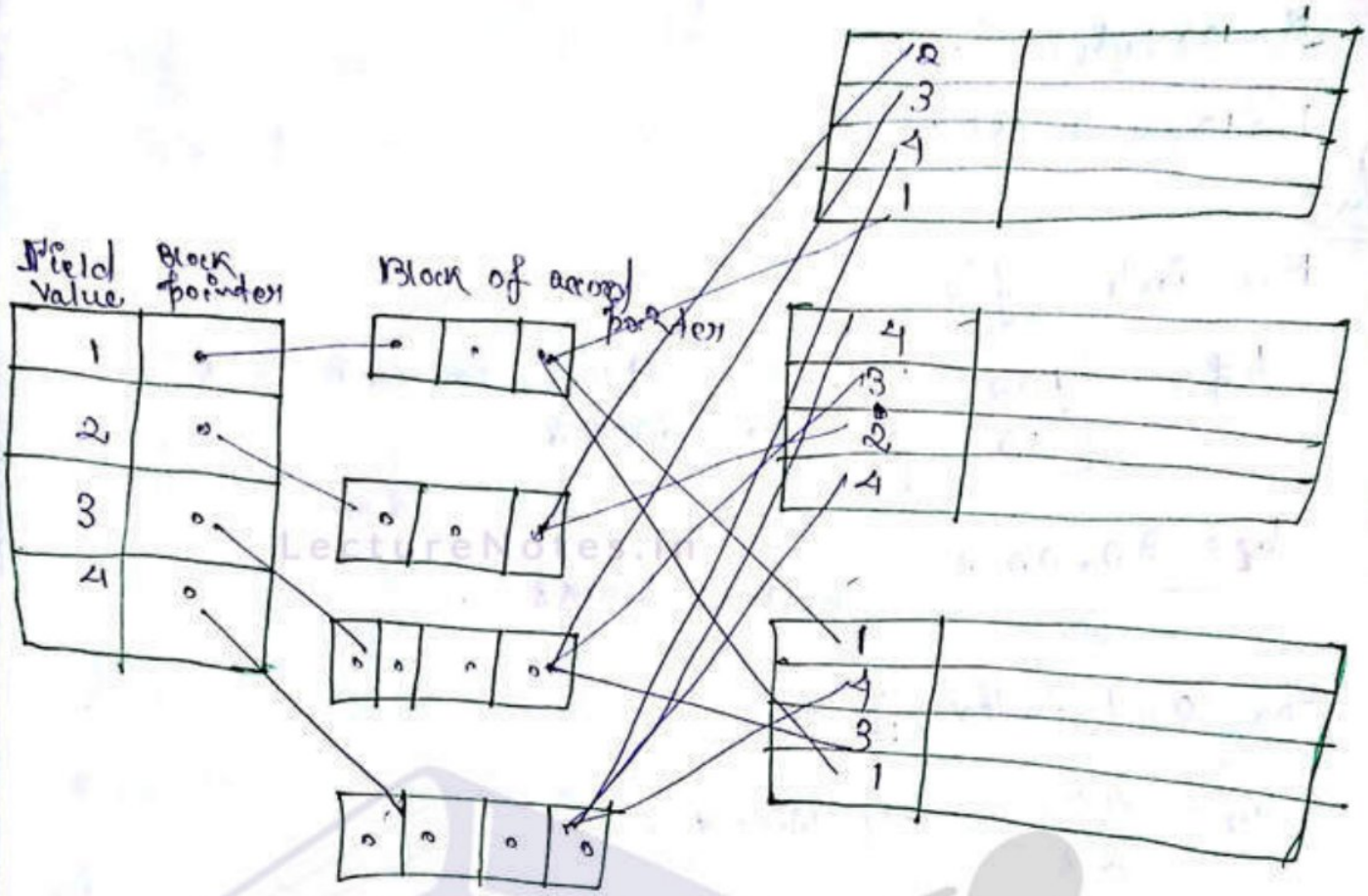
Applying binary search

$\log_2 b_i = \log_2 442 = 9$ blocks

Total no. of block accesses

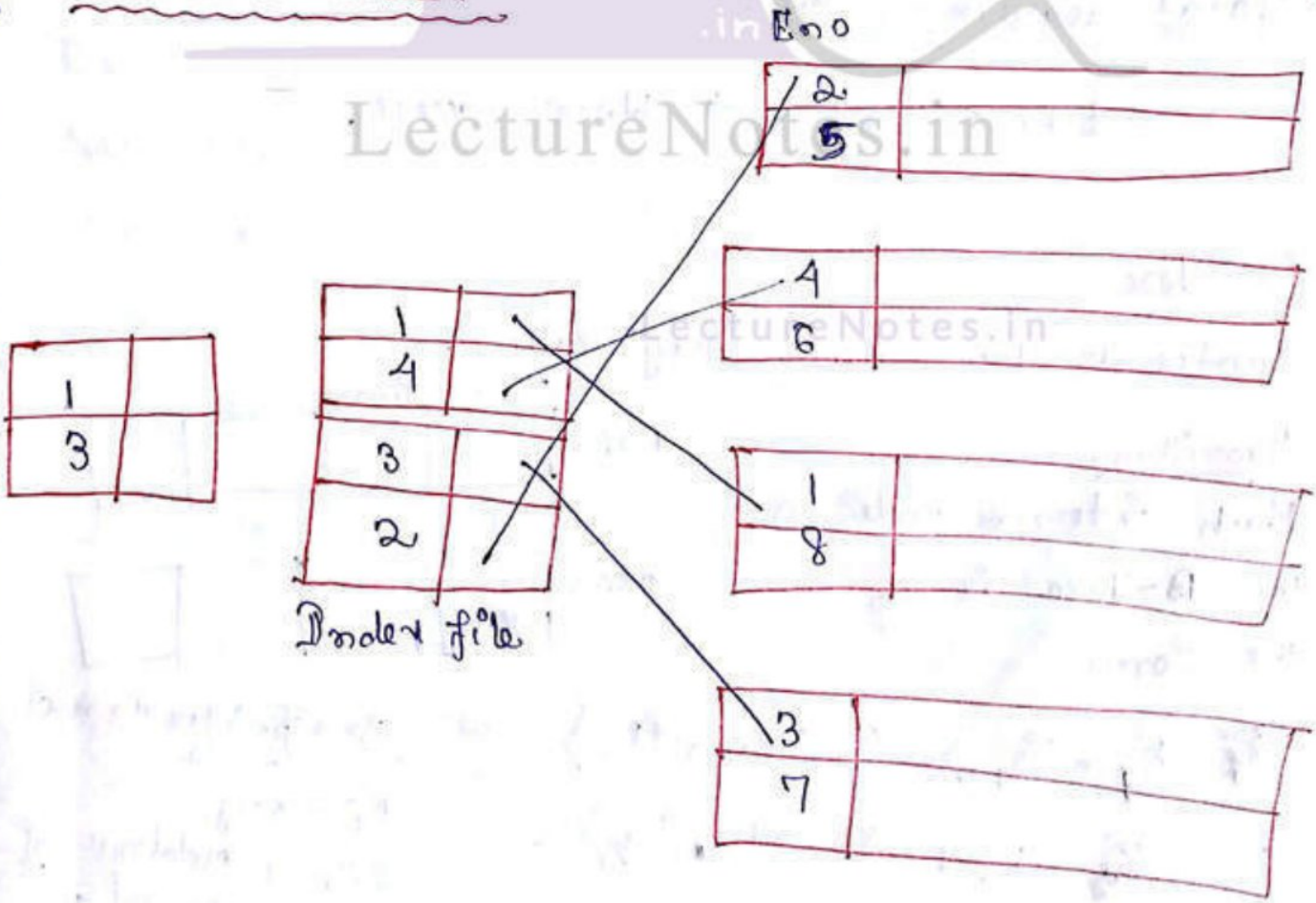
$= 9+1 = 10$ block accesses.

Using non-key.



10/11/2015.

2. Multilevel Index.



Q: $r = 30,000$

$R = 100$ bytes

$B = 1024$ bytes.

A:

For index file

$bfr = \left\lfloor \frac{1024}{15} \right\rfloor = 68$ entries in each block.

$b_1 = \frac{30,000}{68} = 442$ blocks.

For 2nd index file

$b_2 = \frac{442}{68} = 7$ blocks.

$b_3 = \lceil 7/68 \rceil = 1$ block.

Total searching time

$= t+1 = 3+1 = 4$ block access.

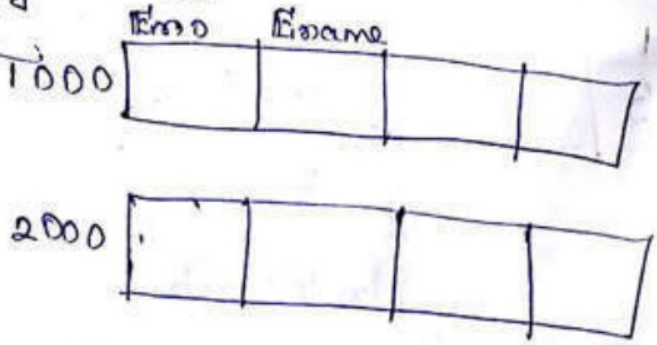
B-Tree.

344 345

Creates B-Tree using Key values.

Algorithm.

1. Each internal node in the B-Tree is of the form



$P_1 \langle K_1, Pr_1 \rangle, P_2 \langle K_2, Pr_2 \rangle$

$\ldots \langle K_{q-1}, Pr_{q-1} \rangle, P_q \rangle$.

$P_i \rightarrow$ pointer to next node

$K_i \rightarrow$ key

$Pr_i \rightarrow$ address of

where P_i is a tree pointer. A pointer to the anchor node in the B-tree.

Each Pr_i is a data pointer.

A pointer to the record whose search key field value is equal to K_i.

2. Within each node

$$K_1 < K_2 < \dots < K_{q-1}$$

3. For all search key field values (x) in the sub-tree pointed at by P_i, we have

$$K_{i-1} < x < K_i.$$

4. Each node have atmost P tree pointers.

5. Each node except the root and leaf nodes has atleast $\lceil P/2 \rceil$ tree pointers.
The root node has atleast two tree pointers unless it is the only node in the tree.

6. A node with q tree pointers

$$q \leq p$$

has $(q-1)$ search key field values.

7. All the leaf nodes are at the same level. They have same structure as

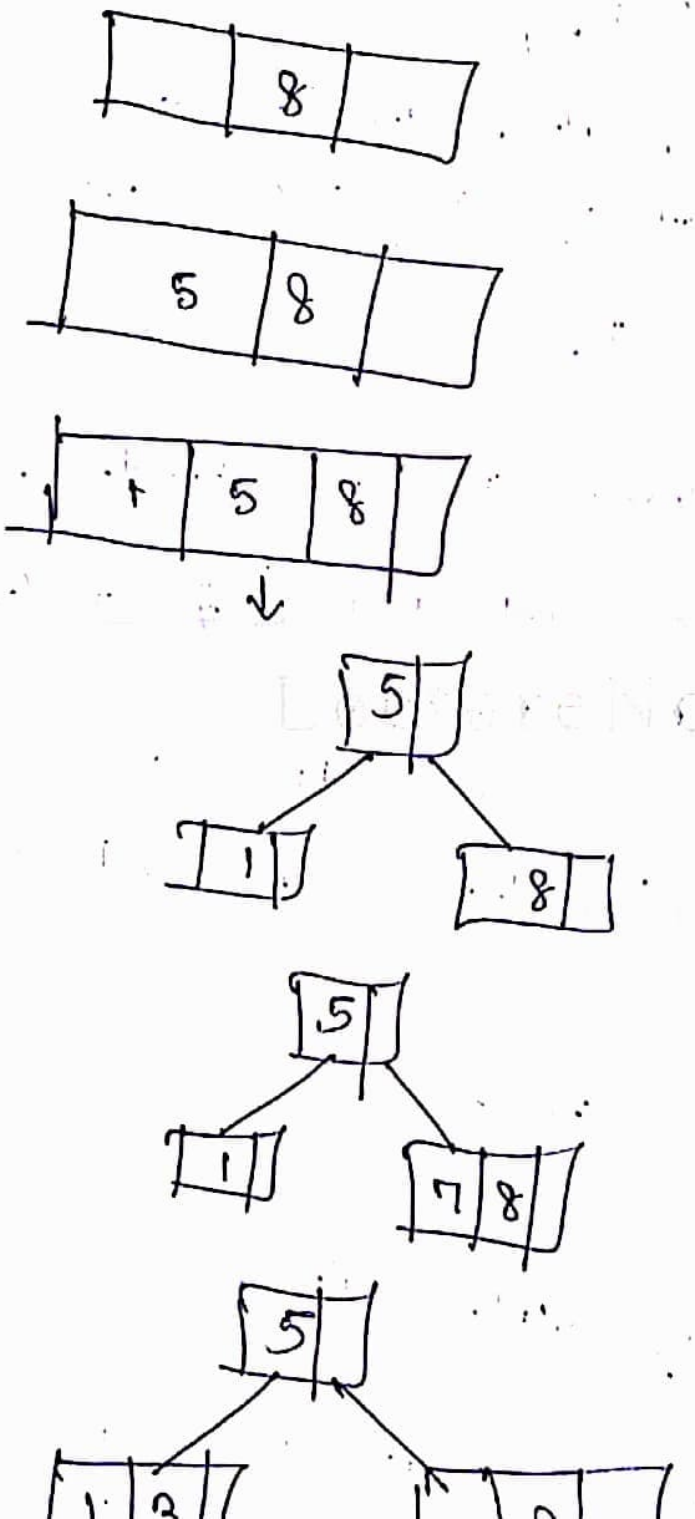
internal nodes except that all of their tree pointers (P_i) are null.

Ex.

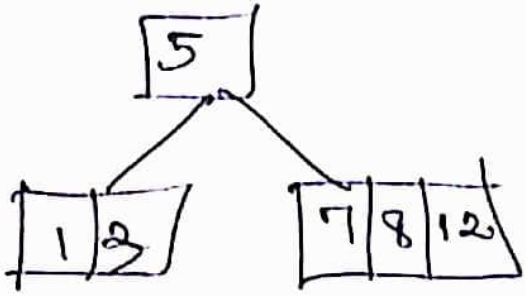
8, 5, 1, 7, 3, 12, 9, 6.

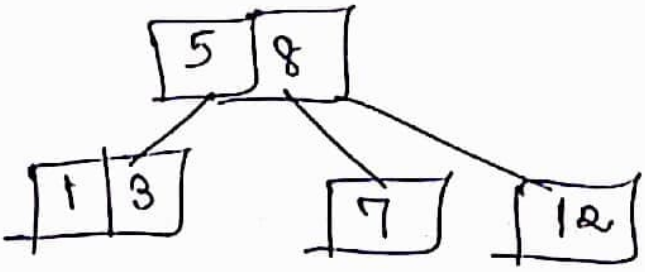
Order = 3 (P).

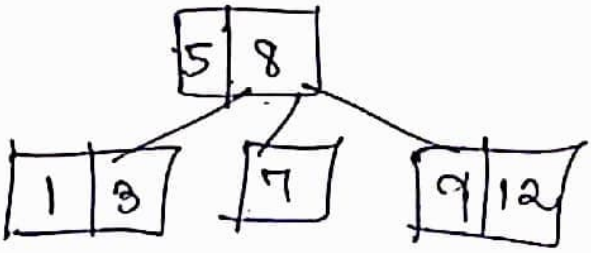
So, each field will contain 2 key values.

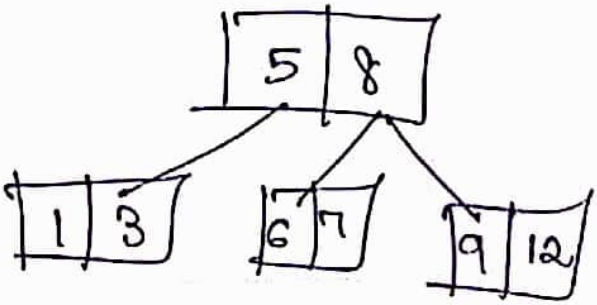


Atul Kumar
7777888822









Lecturenotes.in

Atul Kumar
7777888822