4/1/17

4/12/16

- Database is a collection of related data.
- A data is known facts that can be recorded and that knows implicit meaning.

Properties of database

- It represents some aspects of real world and changes in the real world should be reflected in the database.
- It is a logically coherent collection of data with some inherent meaning.
- It is designed for specific purpose.

DBMS - It is a collection of program that enables user to create & maintain a database. It facilitates the process of defining, constructing and manipulating the databases.

⇒ Defining - It involves specifying the datatypes, structures and constraints for the data to be stored.

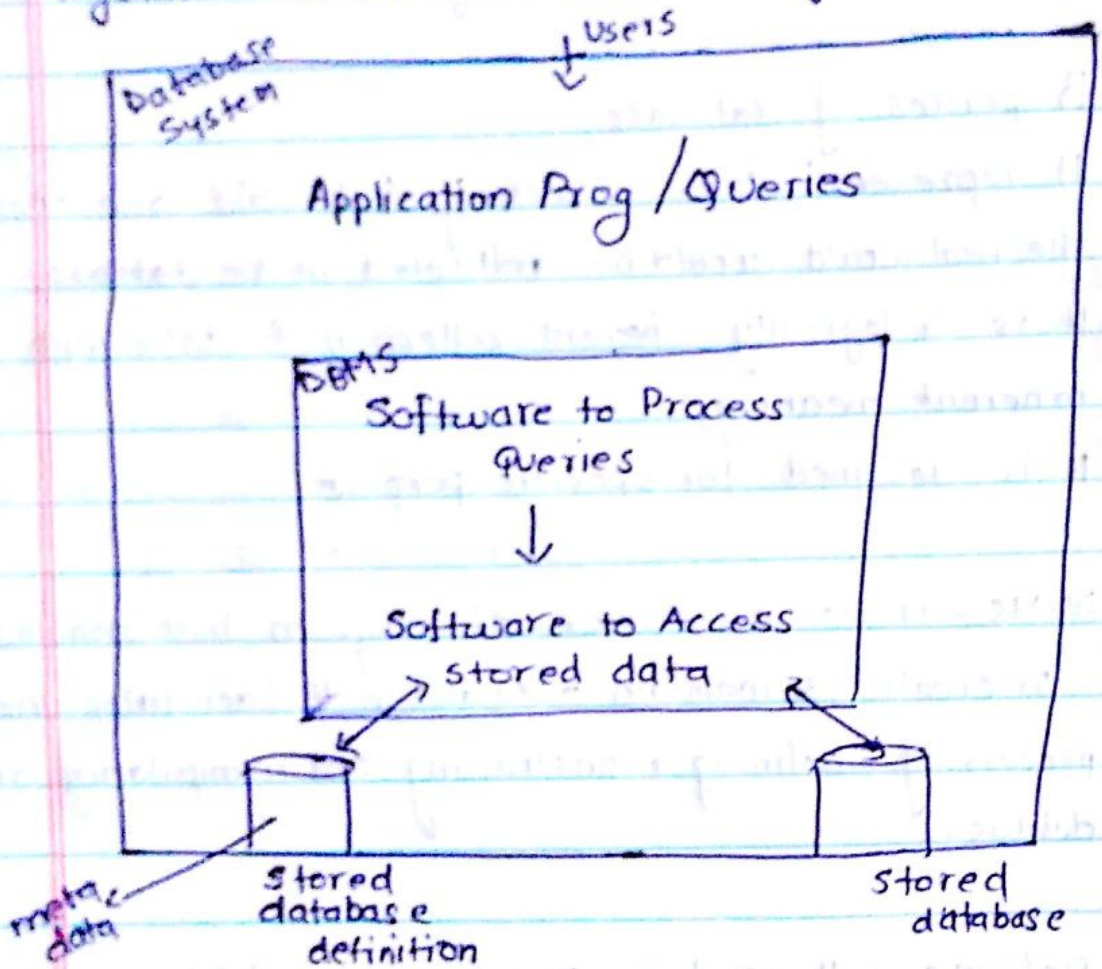
⇒ Construction - Storing the database on some storage medium.

⇒ Manipulation - Quering the database to retrieve specific data, updating the database to reflect changes & generating reports from the data.

Scanned by CamScanner

* Database System

Database & Database Management System together is called Database System.



Characteristic of database

1) Self describing nature of DBS
2) Insulation b/w prog & data and data abstraction.

Scanned by CamScanner

5 1/12

Database designer - They are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data

End users - They access the database for querying, updating and generating reports.

- Casual end users - They occasionally access the database for different info. each time. They use sophisticated database query language.

- Naive or parametric end users - They constantly query and update the database using standard query and updates called Transaction.

- Sophisticated end users - They include engineers, scientists, business analyzer who makes them familiar with the DBMS feature to meet complex requirement.

Scanned by CamScanner

- StandAlone Users - They maintain personal databases by using readymade packages that are easy to use and have graphics based interface

System Analyst and Application Programmers

They determine the requirements of end users and develop specifications for canned transactions that meet these requirements. Application programmers implements these specifications as programs, then they test, debug documents & maintain these transactions.

Scanned by CamScanner

→ Workers behind the scene

1. DBMS System designers and implementers.
They design and implement DBMS modules and interfaces as a software package.

2. Tool developers
They design & implement packages that helps to improve the performance of the database. For eg. packages for performance monitoring, stimulation etc.,.

3. Operators & maintenance personals
They are responsible for running & maintenance of hardware & software env. for database system.

→ Adv. of using DBMS approach

1. Controlling redundancy
In traditional systems, every user group maintains its own files for handling its data processing applications. In such systems, much of the data is stored twice, this redundancy of storing the data multiple times may lead to several problem like duplication of efforts, storage

Scanned by CamScanner

space is wasted, files may be inconsistent.

In database approach each logical data item is stored at one place to remove inconsistency and avoid wastage of space.

2. Restricting unauthorized access

When multiple users share a database, it maybe a chance that some users are not authorized to access all info. in the database. In addition some users maybe permitted only to retrieve data, whereas others are allow to retrieve & update. A DBMS provide a security & authentication subsystem, where the type of access operation is control

3. Providing persistent storage for program, objects & data structures

In programming languages, like c++ have complex structure where the value of the program variables are discarded, once a program terminates unless the programmer explicitly stores them in files which involves converting these structures into suitable format for storage. When the programmer needs the same data again, then he has to convert it back from file format to program

Scanned by CamScanner

structure.

Object oriented database system are compatible with programming languages and DBMS software automatically performs these conversion and the objects survives after program termination and is said to be persistent.

4. Permitting inferencing and actions using rules

Some database system provides capabilities for defining deduction rules for inferencing new info. from stored database facts.

5. Providing multiple user interface

Many types of users with diff. level of Knowledge use a database, so DBMS provide a variety of UI's like query language for casual & end users, prog. lang. interface for application programmers, menu driven interfaces for parametric users & various other interfaces for generating forms & reports.

6. Enforcing Integrity Constraint

DBMS provides capability for defining and enforcing integrity constraints like specifying a

Scanned by CamScanner

datatype for each data item, a record in one file must be related to records in other files, uniqueness of data item values. Database designers identify these constraints during database designs and some are enforced automatically by DBMS.

Practical

Select * from tab;

* → all

tab → System table

Select * from table_name;

view → result table, temporary in nature.

Eg

Select * from emp where EMPNO > 7500;

desc table-name;

desc → describe

Datatype → Number 0-9

Varchar

date (DD-MM-YY)

Scanned by CamScanner

12/1/17

Chapter-2

Database System Concepts and architecture

Data Model

A data model determines the logical ~~architecture~~ structure of a database and in which manner, data can be stored, organised and manipulated.

By structure we mean the data types, relationships and constraints that should hold on data.

- Categories

1. High level or conceptual data model

It identifies the highest level relationship b/w different entities, features of conceptual data model are

a) It includes the imp. entities and relationships among them.

b) No attribute is defined

c) No primary key is specified.

2. Logical data model

It describes the data in as much detail as possible, regardless of how they will be physically implemented in the database. Features of logical data model -

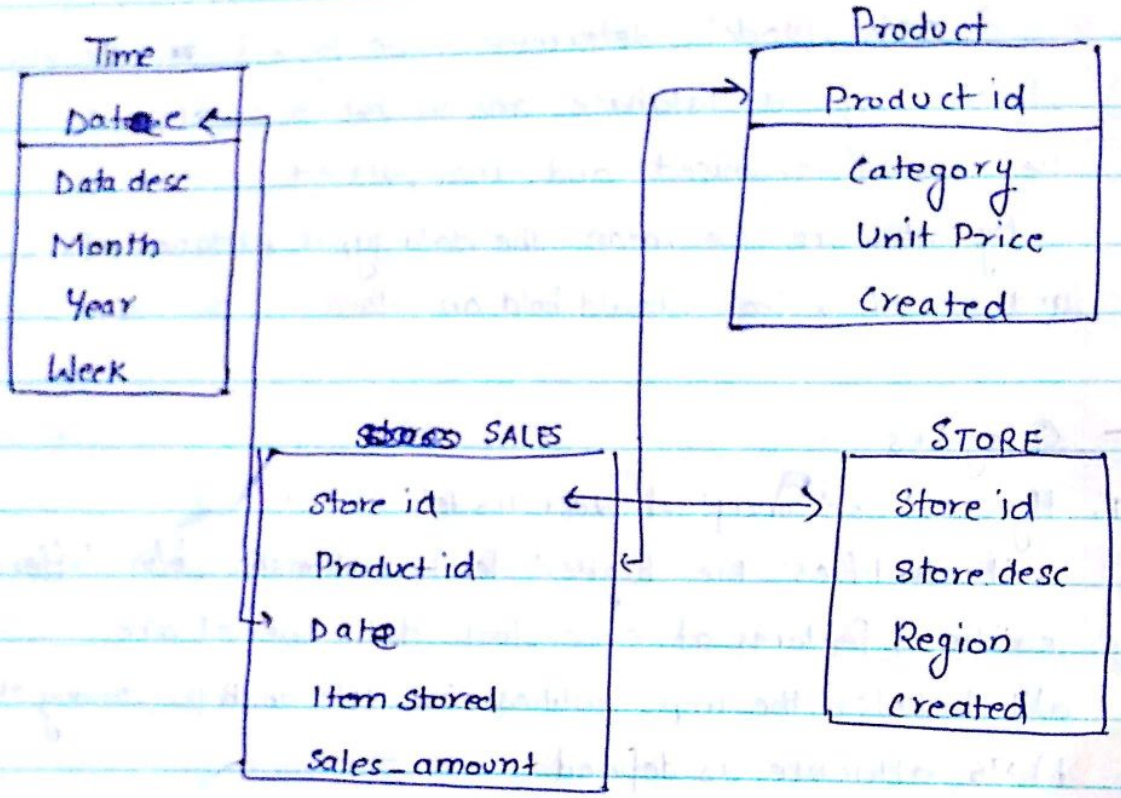
a) It includes all entities and relationships.

b) All attributes for each entity are specified.

c) Primary key for each entity is specified.

Scanned by CamScanner

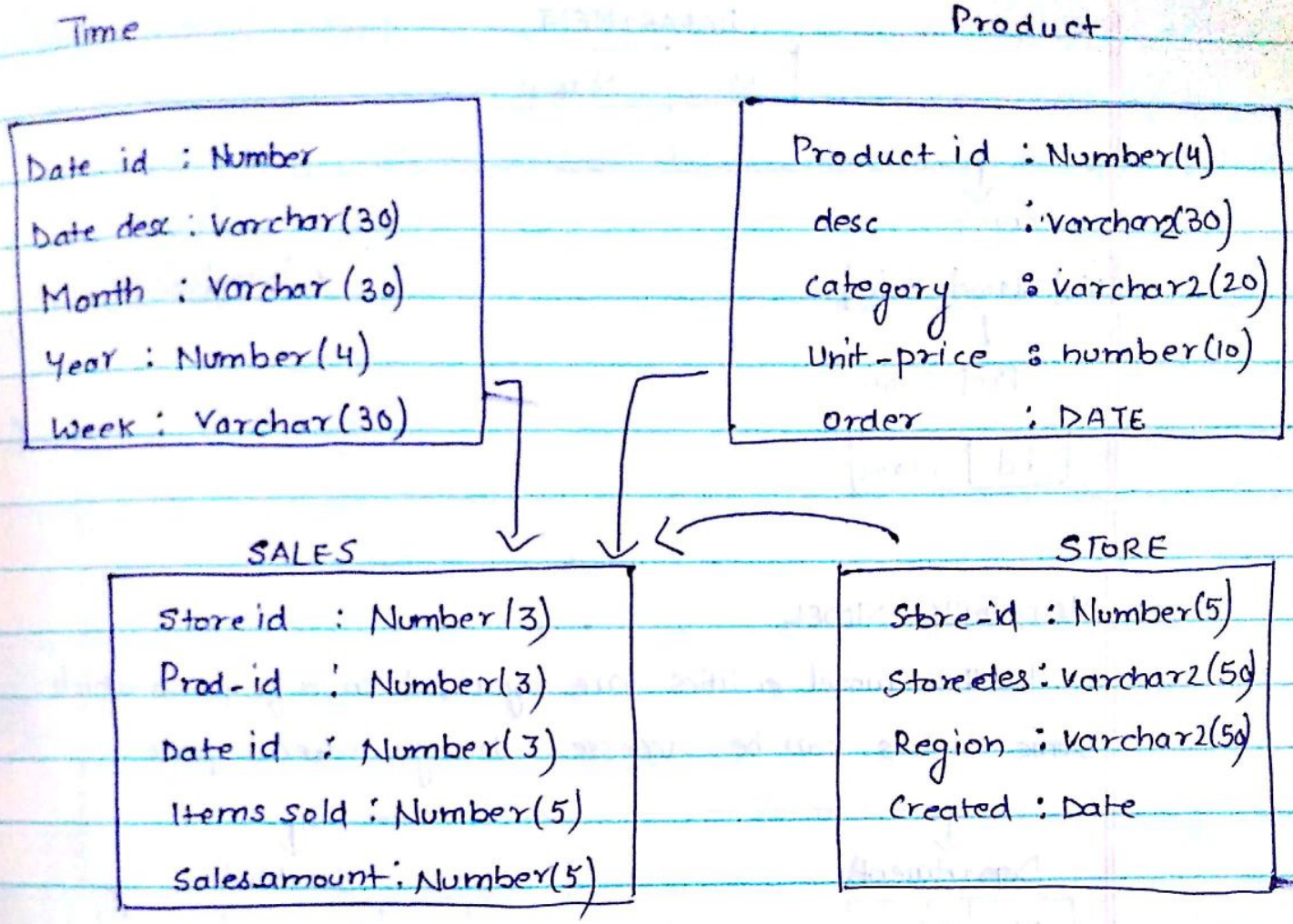
d) Foreign Key for each entity is specified.
e) Normalization occurs at this time.



3. Low level or physical data Model

It provides concepts that describes the details of how data is stored in computer. It shows all the table structures including column name, data type, constraints, primary key, foreign key and the relationship b/w the tables

Scanned by CamScanner



- DATABASE MODELS

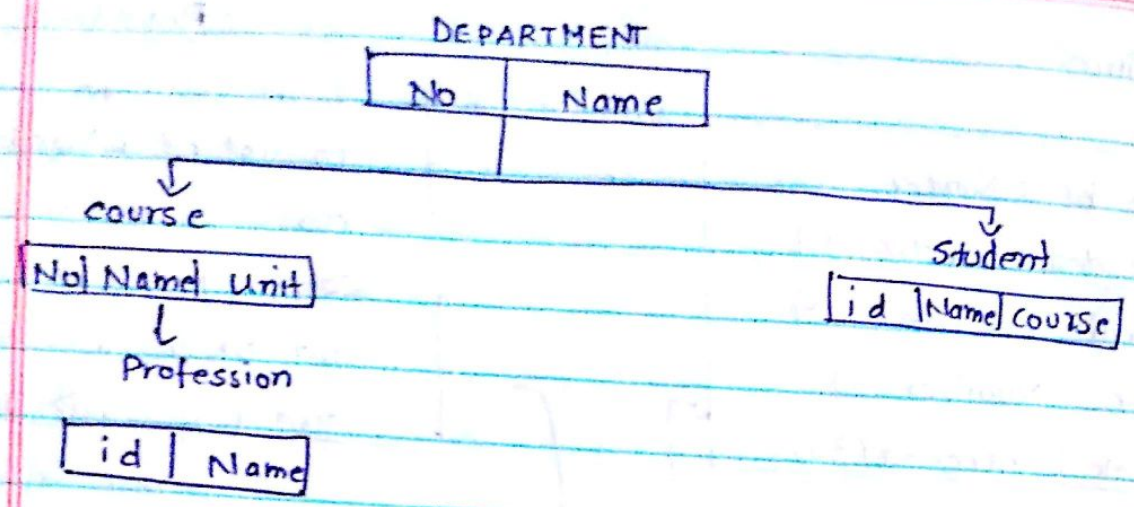
- A database model define logical design of data, the model describes the relationship b/w different parts of the data.

1. HIERARCHIAL MODEL

In this model each entity has only one parent but can have several children

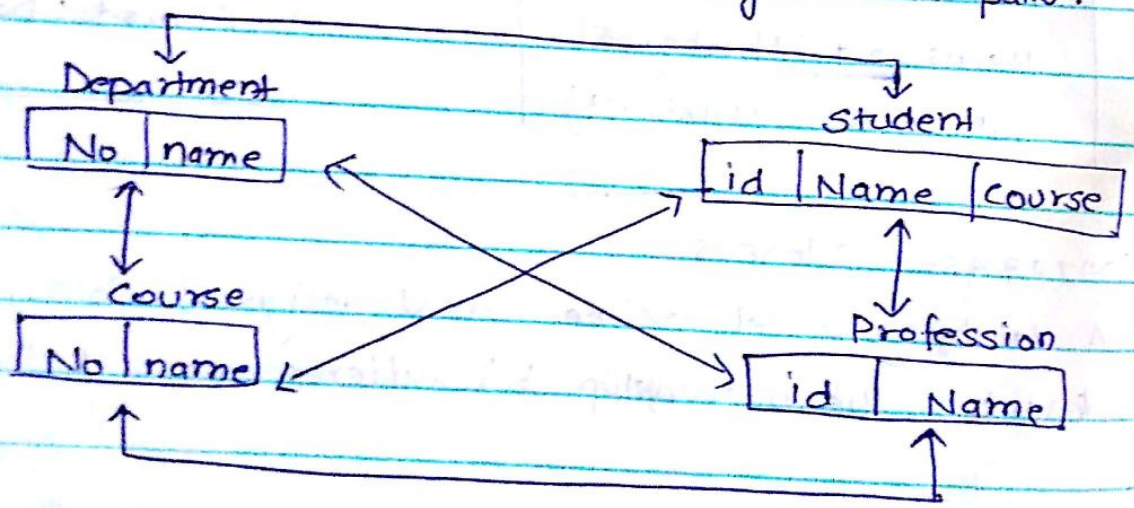
- At the top of the hierarchy, there is only one entity which is called ROOT.

Scanned by CamScanner



2. NETWORK MODEL

In this model entities are organised in a graph in which some entities can be accesses through several paths.



Scanned by CamScanner

3. RELATIONAL MODEL

In this model data is organised in 2D model called Relations. The tabels or relations are related to each other.

Department

No	Name
10	A/cs
20	C.S

Student

id	Name	Course
S1	Stu1	C_1
S2	Stu2	C_2
S3	Stu3	C_3

Course

No	DeptNo.	Prof-Id	Unit
C_1	10	1	U_1
C_2	20	2	U_2
C_3	10	3	U_3

Professor

id	Name	DeptNo	Course
1	ABC	10	C_1
2	XYZ	20	C_2
3	UVW	10	C_3

Scanned by CamScanner

- **SCHEMA**

- The description of a database is called schema. It is a skeleton structure that represents logical view of the entire database. It is specified during database design. It is also called intension.

- **DATABASE INSTANCE**

- It is a state of operation database with data at any given time. They tends to change with time.
- DBMS ensures that its every instance is in the valid state by following all the validation constraints & conditions that are being imposed. It is also called extension of schema.

- **DBMS ARCHITECTURE** (3 schema architecture)

- It separates the user applications & the physical database
- In this architecture, schema's are defined at 3 levels

1. Internal Level 2. Conceptual Level 3. External Level

- **Internal Level**

- It is the lowest level of data abstraction that deals with the physical representation of the database & is also known as physical level.

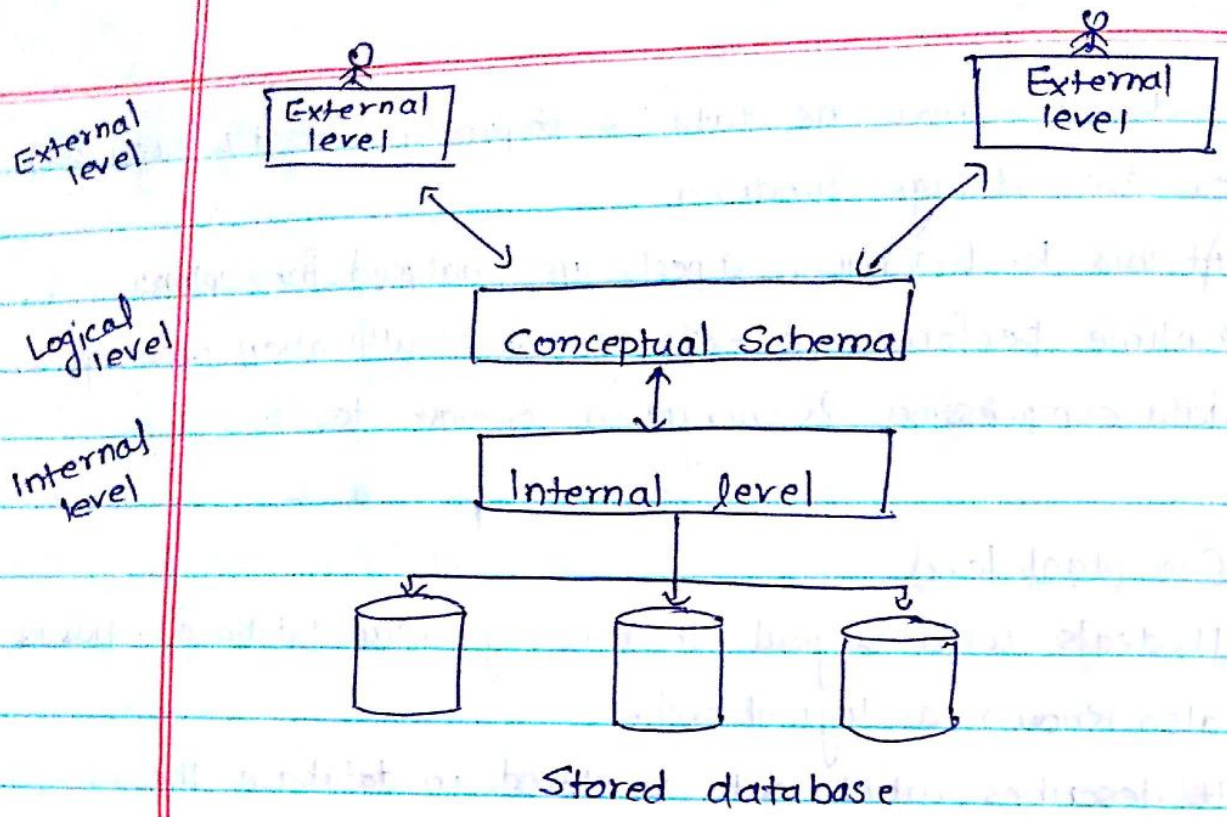
Scanned by CamScanner

- It describes how the data is physically stored & organised on the storage medium.
- At this level various aspects are analysed for optimal runtime performance - storage space utilization techniques, data compression & encryption tecnique etc.

• Conceptual level
- It deals with logical structure of entire database, thus is also known as logical level.
- It describes what data is stored in database, the relationships among data & complete views of users req. without any concern from the physical implementation.

• External level
- It is the highest level of abstraction that deals with users view as database & thus is also known as view level.
- Most of the users do not require the entire data stored in the database.
- It permits users to access data in a way i.e. customized according to their need, so that the same data can be seen by different users in different ways at the same time.

Scanned by CamScanner



Scanned by CamScanner

16/1/17

Data Independence

- It is the capacity to change the schema at one level without having to change the schema at next higher level.
- It is the idea that says generated & stored data should be kept separate from applications that use the data for computation & presentation.

- Types of data Independence

1. Logical data Independence

It is the ability to modify the logical schema or the conceptual schema without causing application program to be rewritten.
We may change conceptual schema to expand the database or to reduce the database. In both the cases external schema refers only to the remaining data should not be affected, only view definition needs to be change

2. Physical data Independence

It is the capacity to change the internal schema without having to change conceptual/external schema. These changes in the physical schema may be needed because some physical file had

Scanned by CamScanner

to be reorganized.

DBMS LANGUAGES & THE QUERIES

1. Select - It is used for data retrieval.

2. Data Manipulation Language [DML]
   a) Insert
   b) Update
   c) Delete
   d) Merge

3. Data Definition Language [DDL]
   a) Create
   b) Alter
   c) Drop
   d) Rename
   e) Truncate

4. Transaction Control
   a) COMMIT
   b) ROLLBACK
   c) SAVE POINT

Scanned by CamScanner

multi-valued attribute
composite attribute

5. Data Control Language [DCL]
a) GRANT
b) REVOKE

Practical

```
Create table <table name>
(
    col1 datatype..
    ,
    ,
    ,
);
```

eg. create table Student(
Stdid number(3) primary key,
Name varchar2(20),
Course varchar2(20),
Marks number(5),
constraint Name ForeigKey(Name) references Table-Name (——);

Constraints → 1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY

Table level constraint

Scanned by CamScanner

1. Table level constraint
2. Column level constraint (Defined at the time of table creation)

Entities - An entity is a thing in the real world with an independent existence like car, employee, person, house etc. or it maybe an object with conceptual existence like company, bank, job, course etc.

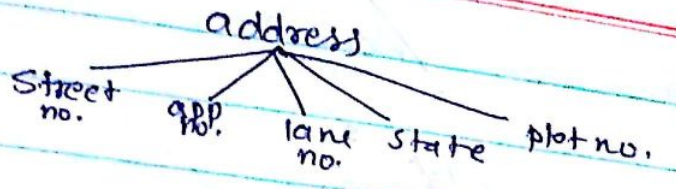
Attributes - Attributes are the properties that describe an entity.

Entity sets - It is the set of entities of the same type that share the same property.

Imp Types of attributes:-

1. Simple or Atomic - These attributes are not further divisible. eg. sex of a person, salary
2. Composite Attribute - These can be divided into small sub-part which represent more basic attributes with independent meaning eg. address

Scanned by CamScanner



3. Single-valued attribute : They can have only single value.
eg. dob, age, roll no.s etc.

4. Multi-valued attribute : They can have multiple values.
e.g. phone no., degree of stud. email etc.

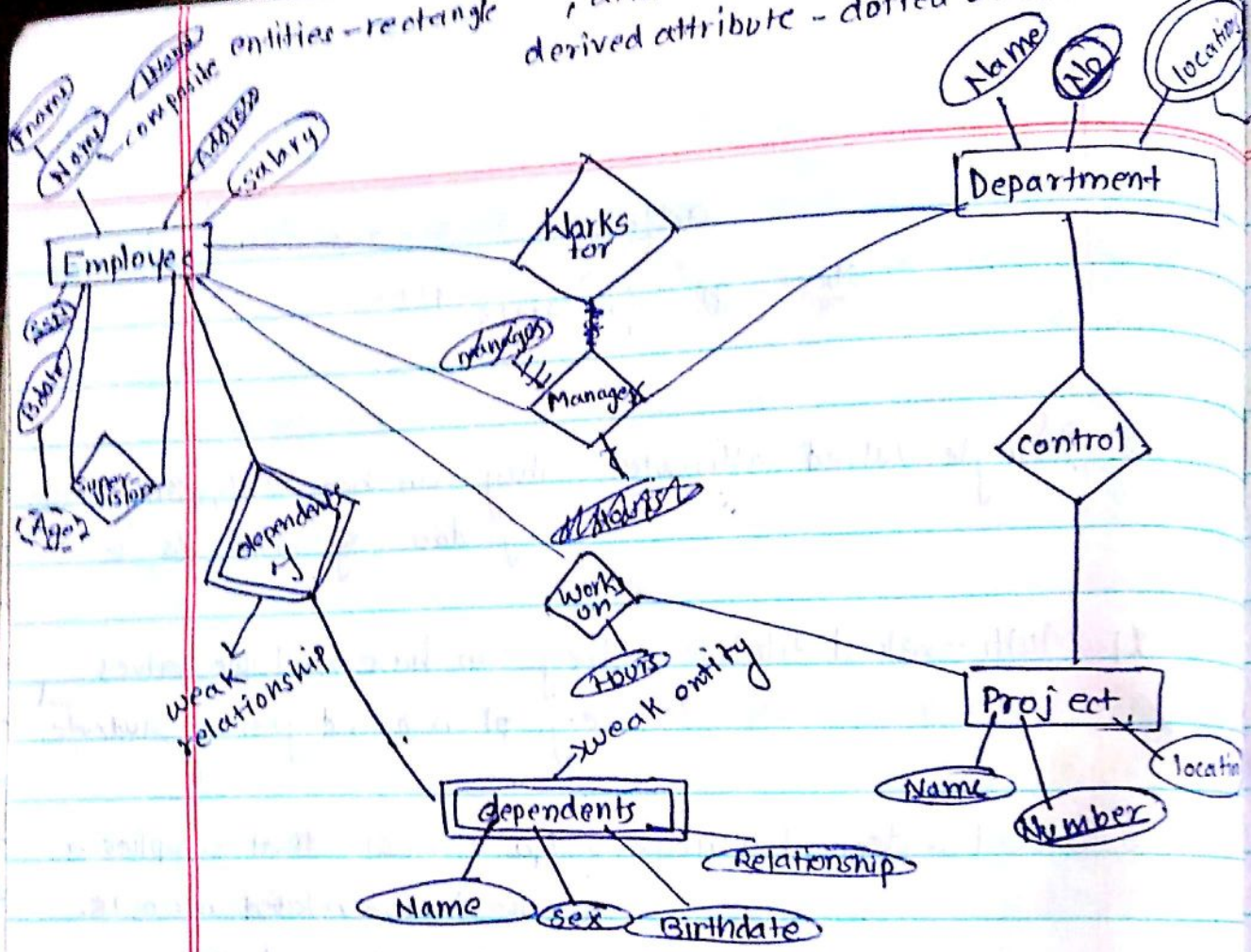
5. Stored & derived attributes : An attribute that supplies a value to the related attribute.
eg. dob. Derived attribute is a attribute whose value is derived from a stored attribute.
eg. age.

- NULL Values - When values are not known or not applicable we use null values.

- Key attributes
- Candidate Key
- ~~de~~ Dependent ~~attributes~~ entity

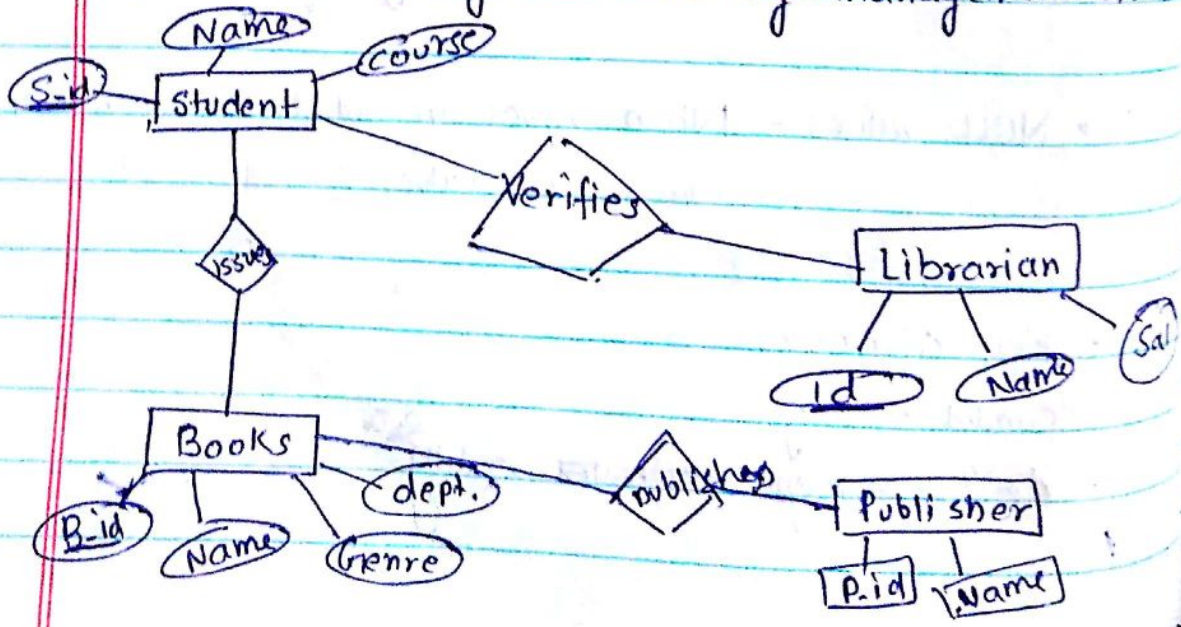
Scanned by CamScanner

, attributes - ovals
composite entities - rectangle
derived attribute - dotted ovals



Entity - Relationship Diagram

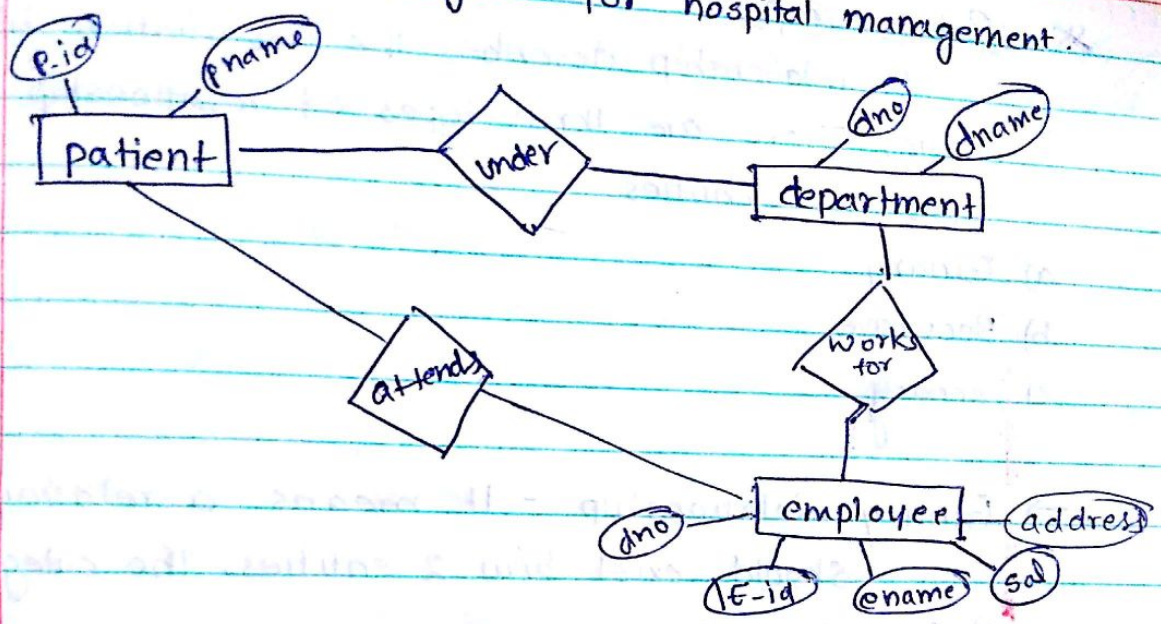
Ques. Draw an ER diagram for library management system.



Scanned by CamScanner

multi-valued

Ques. Create an ER diagram for hospital management.



Scanned by CamScanner

30/1/17

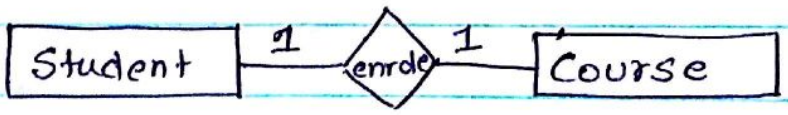
* Relationships

A relationship describes the association b/w two entities. There are three types of relationship that exists b/w entities

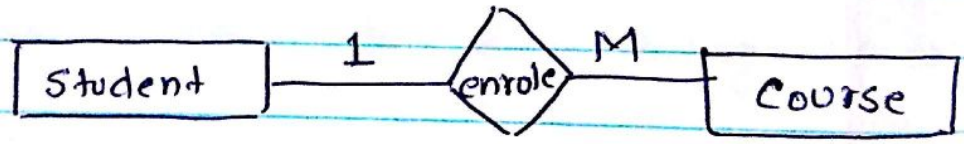
a) Binary
b) Recursive
c) Ternary

→ Binary Relationship - It means a relationship should exist b/w 2 entities. The categories are

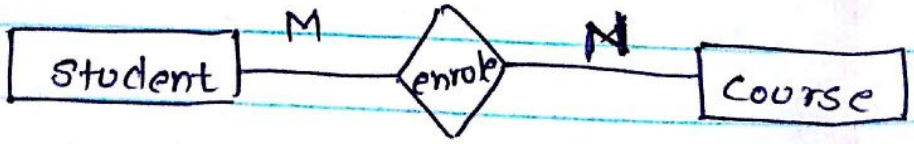
1) One-to-one



2) One-to-many



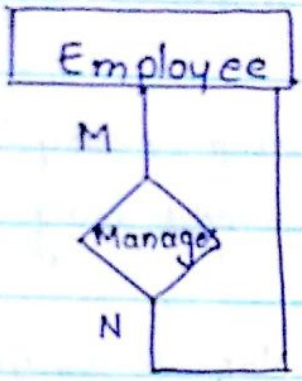
3) Many-to-Many



Scanned by CamScanner

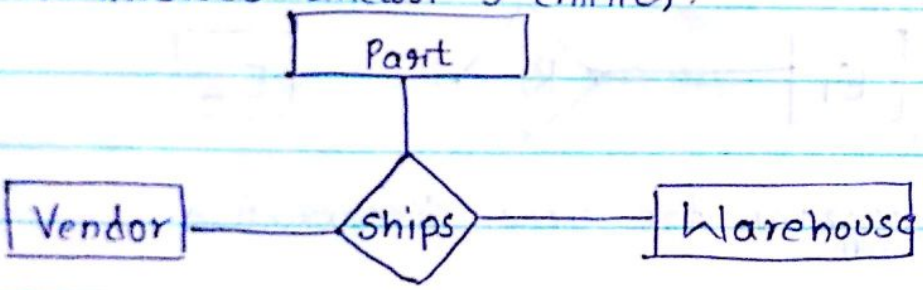
→ Recursive Relationship

When an entity is related with itself, it is known as recursive relationship.

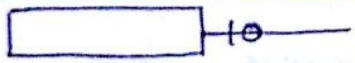


→ Ternary Relationship

It involves atleast 3 entities.



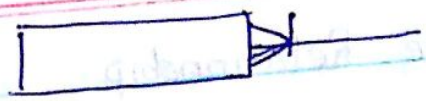
Symbols (Used in ER diagram)

Optional one 

Mandatory one

Optional many

Scanned by CamScanner

Mandatory many 

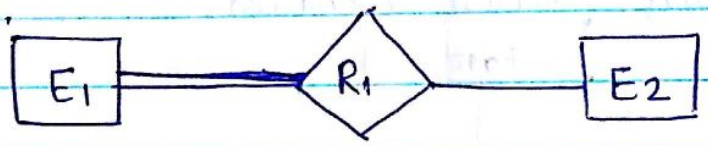
2/2/17

Participation Constraint

- Total participation - Each entity is involved in the relationship. It is represented using double line.

- Partial participation - Not all entities are involved in the relationship. It is represented using single line.

eg.



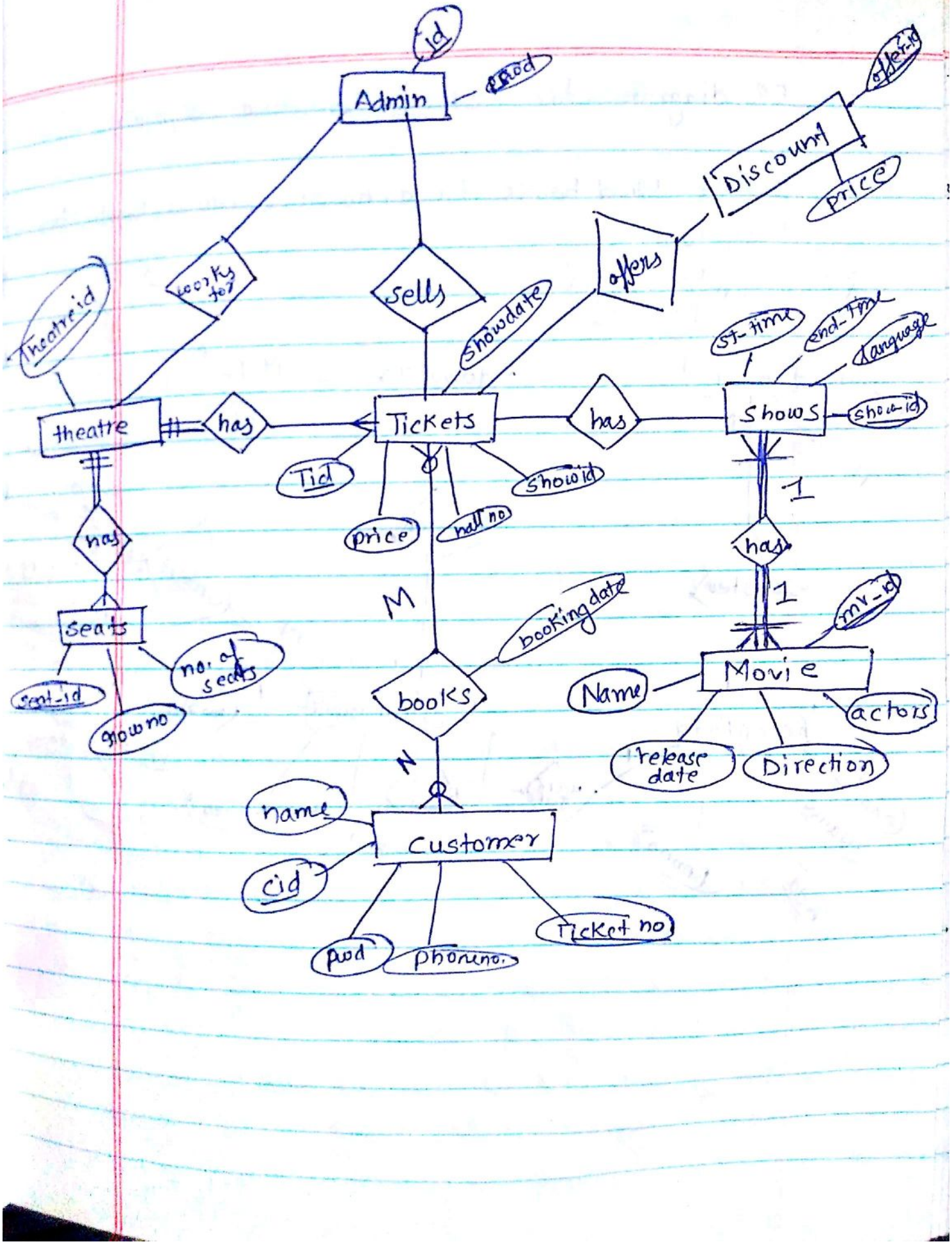
→ ER diagram for Ticket Reservation System

~~Customer~~ ~~Tickets~~

Customer, tickets, show, movie, admin, seats, discount

~~Admin~~

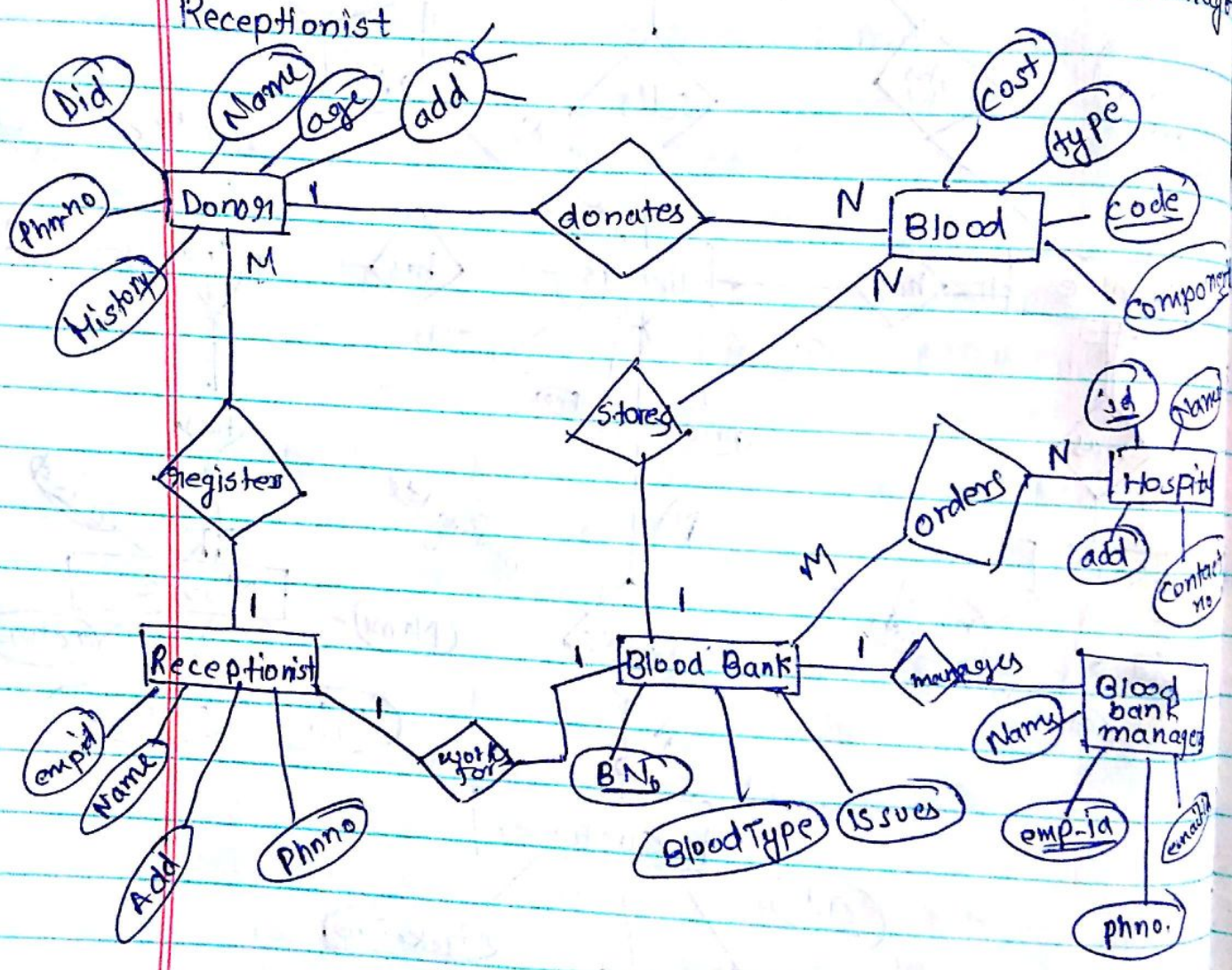
Scanned by CamScanner



Scanned by CamScanner

→ ER diagram for Blood Donation System.

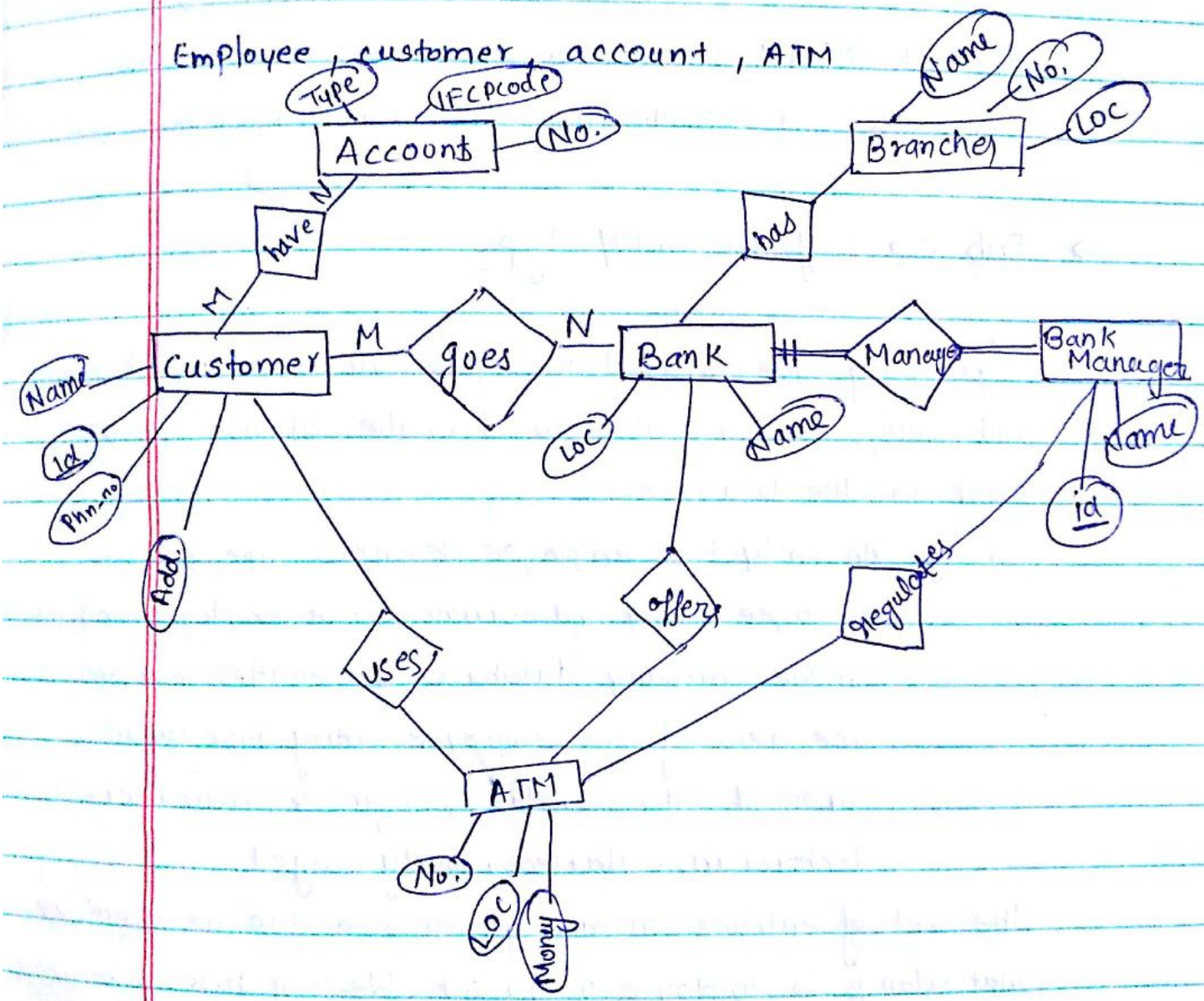
Donors, Blood banks, blood, hospitals, Blood Bank Manager, Receptionist



Scanned by CamScanner

→ ER diagram for Banking System

Employee, customer, account, ATM



Scanned by CamScanner

9/02/17

Enhance Entity Relationship & Object Modeling:-

In EER Model, we include the concepts of sub-class, super class and specialization and generalization.

→ **Sub-Class of an Entity-type**

An entity type is used to represent both type of entity and entity set or collection of entities of that type that exist in the database.

For eg.- An entity type employee describes type of each employee entity and current set of employee entities in the company database. The entities that are members of the employee entity type can be grouped into secretary, engineer, manager, technician, salaried, hourly waged.

The set of entities in each group is a sub-set of entities that belongs to employee entity set. We call these sub-groups a sub-class of employee entity type. An entity can be a member of sub-class & super-class. It can be included in any no. of sub-classes.

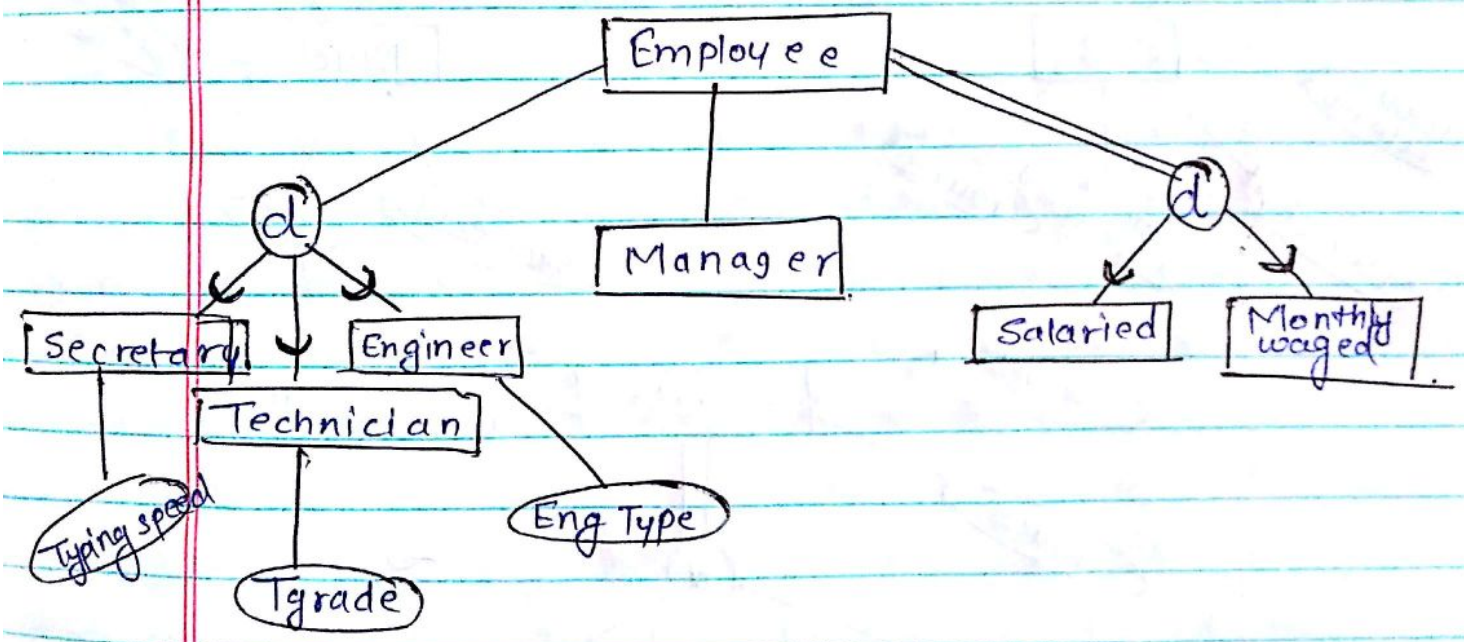
For eg.:- A salaried employee who is also an engineer ~~be~~ belongs to the 2 sub-classes engineer and salaried employee.

Scanned by CamScanner

→ Specialization & Generalization

Specialization

Specialization is the process of defining a set of sub-classes of an entity type. This entity type is called a superclass of specialization. The set of sub-classes that form a specialization is defined on the basis of some distinguishing characteristics of the entities in the superclass.



Ques. What are reasons for including superclass, sub-class, relationships or specialization in a data model?

Ans.- Certain attributes may apply to some but not all entities of the superclass or sub-class is defined in order to group the entities to which they apply.

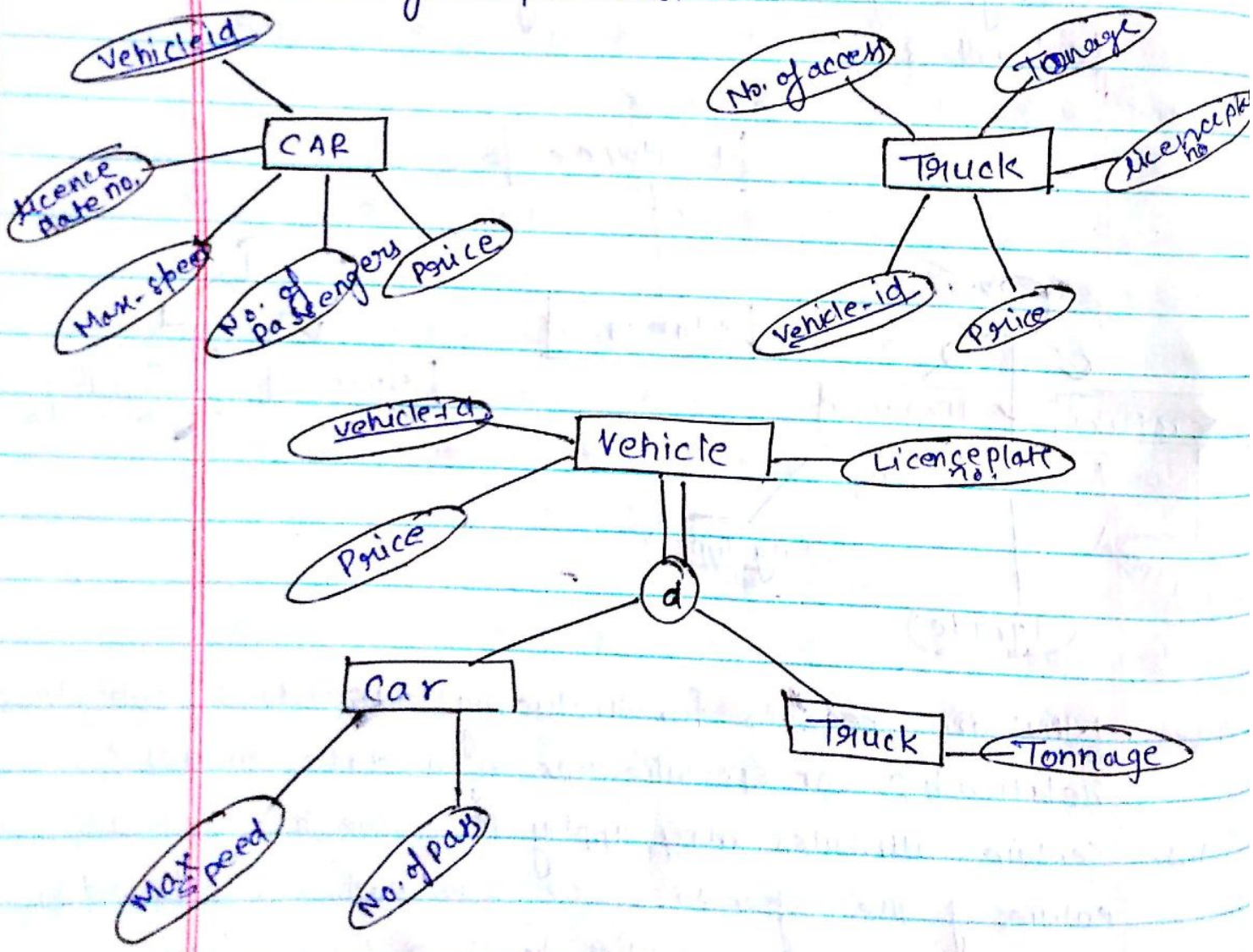
- Some relationship types may be participated in only

Scanned by CamScanner

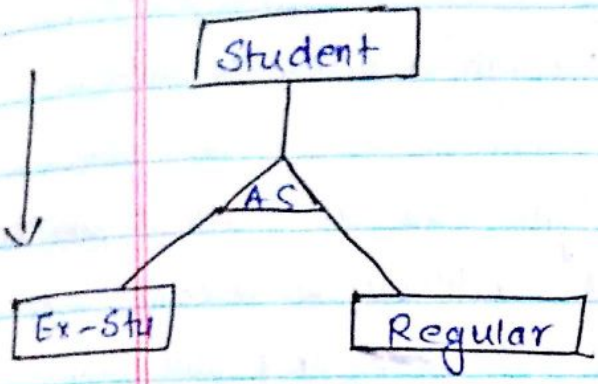
by entities that are members of the sub-class.

Generalization

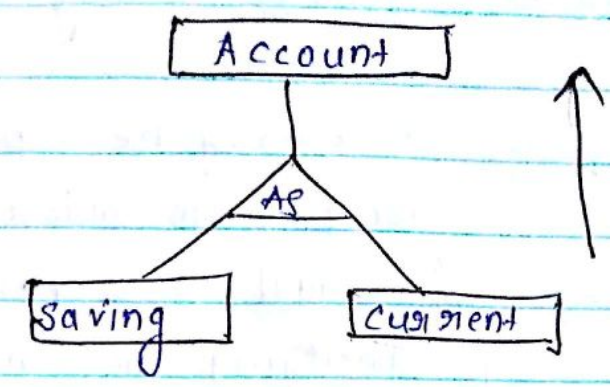
This is the reverse process of abstraction in which we supress the differences among several entity types, identify there common features and generalize them into a single super-class.



Scanned by CamScanner



Specialization

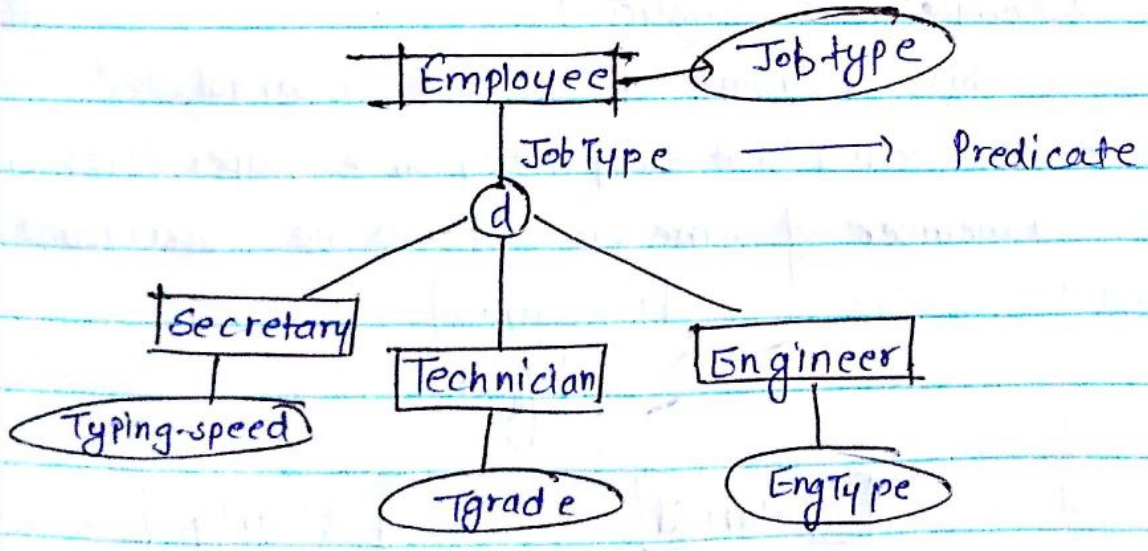


Generalization

→ Constraint on Specialization & Generalization

(d) <u>Disjoint Constraint</u> → The sub-classes must have disjoint sets of entities.

(o) <u>Overlap constraint</u> → The sub-classes may have overlapping set of entities.



Scanned by CamScanner

→ **Predicate Defining Sub-class**

The sub-class is defined through a predicate on the attributes of the super-class.

For eg. If the employee entity type has an attribute job-type is equal to secretary. This condition is a constraint specifying that members of the secretary sub-class must satisfy the predicate.

→ we can specify the condition of member-ship in the secretary sub-class by the predicate job-type is equal to secretary.

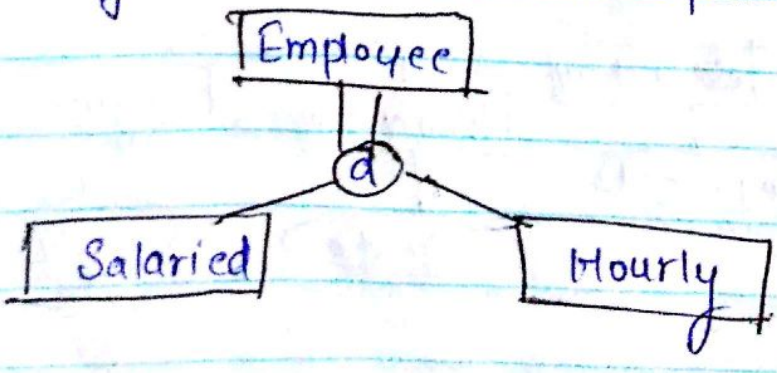
→ **Attribute Defining Sub-classes**

If all the sub-classes in a specialization have a membership cond^n on the same attribute of the super class the specialization itself is called attribute defining specialization & attribute is called defining attribute.

→ **Completeness Constraint**

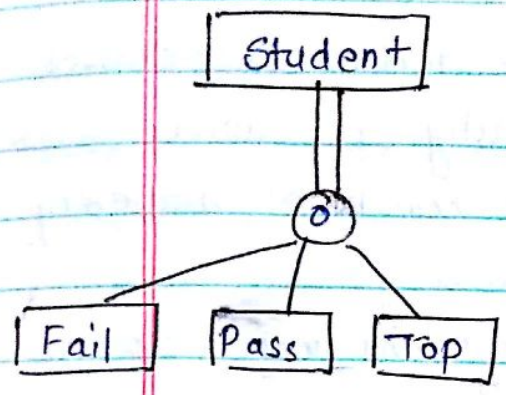
This constraint can be total or partial.

- It specifies that every entity in the super-class must be a member of some sub-class in the specialization. (Total)

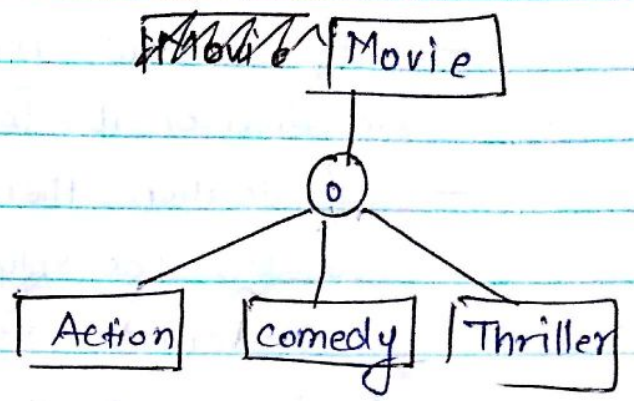


Scanned by CamScanner

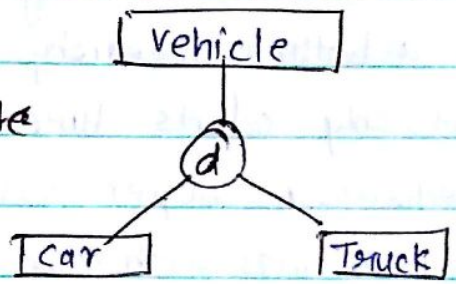
Overlapping Total



Overlapping Partial



disjoint partial



→ Insertion & Deletion Rule in Specialization

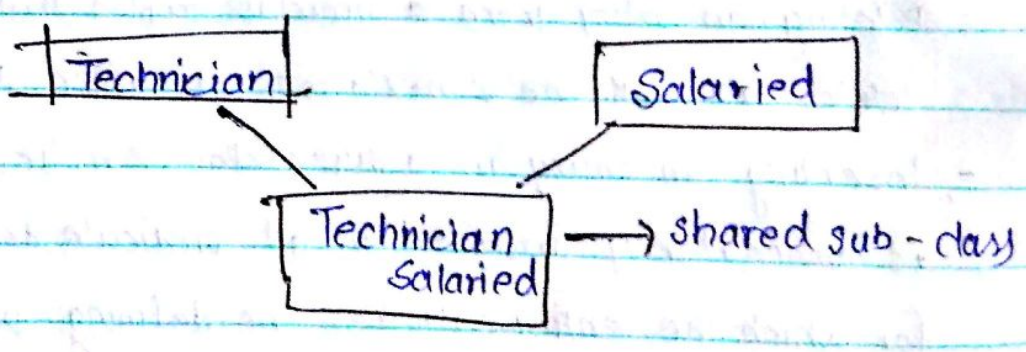
- Deleting an entity from a superclass implies that it is automatically deleted from all the sub-classes to which it belongs.
- Inserting an entity in a super-class implies that the entity is mandatorily inserted in all predicate defined sub-classes for which the entity satisfies the defining predicate.
- Inserting an entity in a super class of total specialization implies that the entity is mandatorily inserted in atleast one of the sub-class of specialization

Scanned by CamScanner

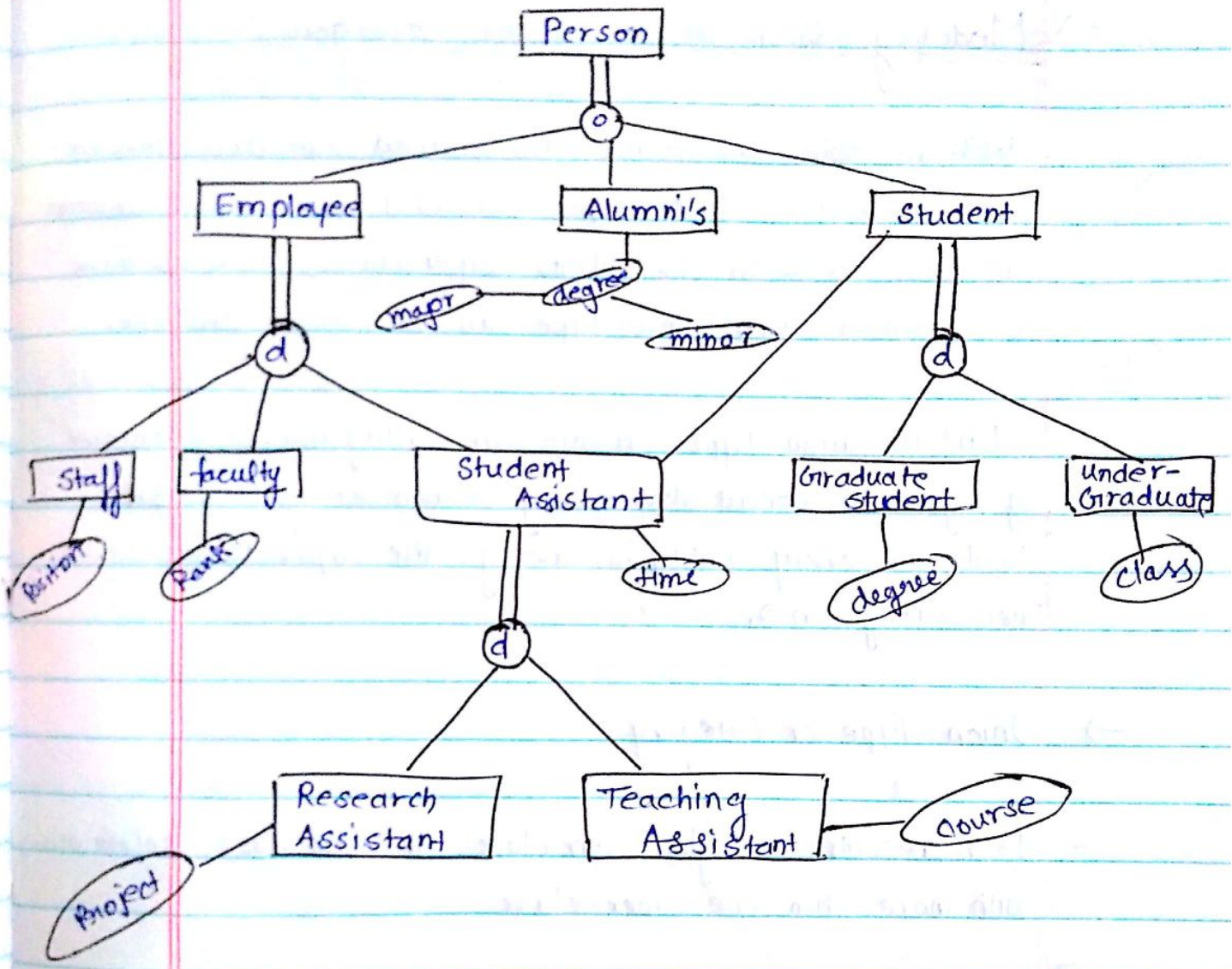
Specialization, Generalization, Hierarchy and Lattices

- A sub-class may itself have further sub-classes specified on it forming a hierarchy or a lattice.
- Specialization Hierarchy have a constraint that every sub-class has only one super class.
- Specialization Lattice has a constraint that a sub-class can be a sub-class of more than one super-class.
- In a Lattice hierarchy, a sub-class inherits attributes not only of its direct superclass but also of all its predecessor superclasses.
- A sub-class with more than one super-class is called a shared sub-class.

eg.



Scanned by CamScanner



Scanned by CamScanner

13/2/17

→ **Modeling Union Types Using Categories**

With multiple inheritance the shared sub-class inherits all the attributes of its super-classes. If we want a shared sub-class to be in one of its super-classes then we have to use union type. Union Type can be partial and total.

Partial union Type means an entity maybe a member of Type A without also having a member of type B it total then every entity is one of the superclasses would necessarily in D.

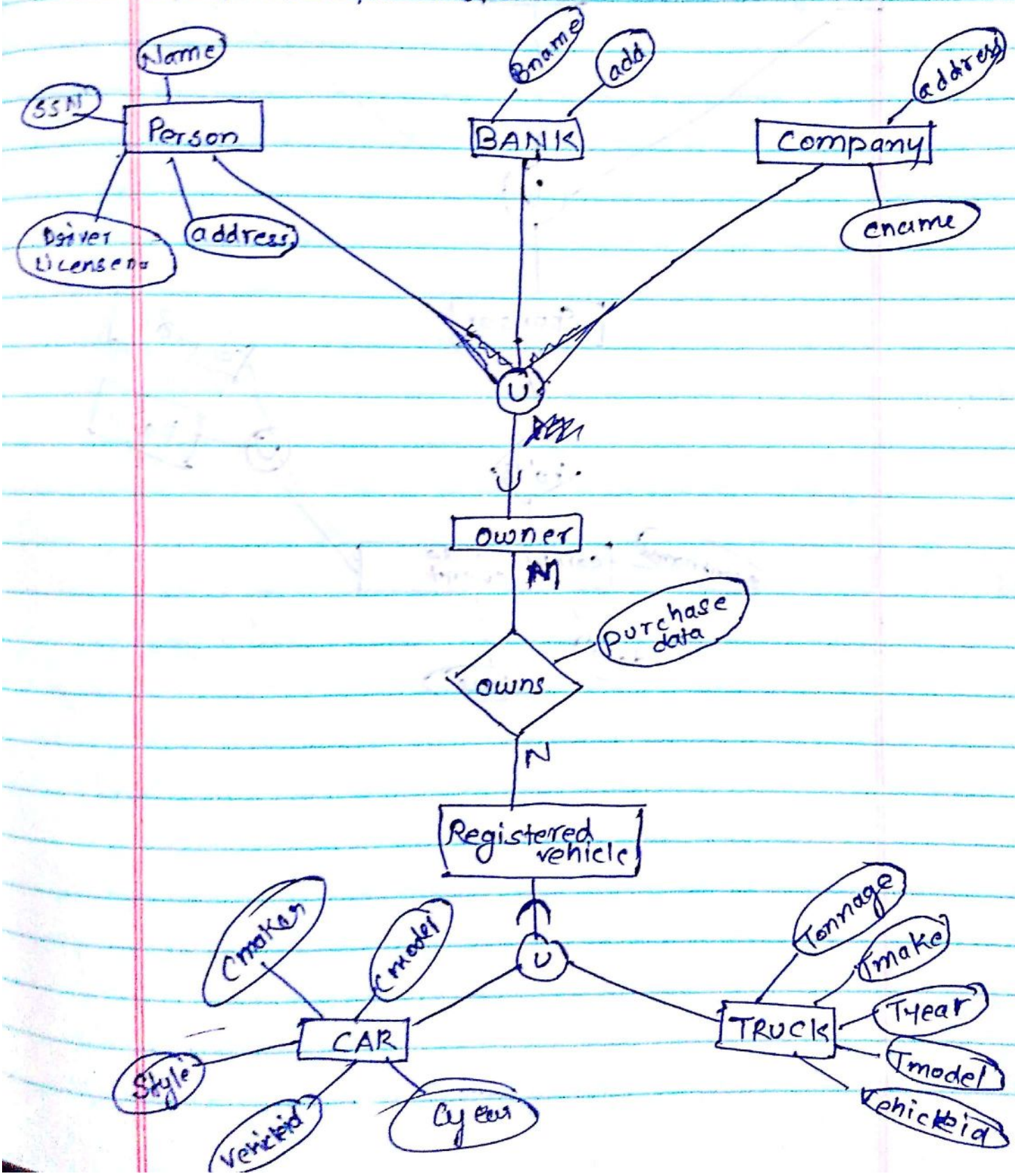
→ Union Type or Category

- It represents a single super class or sub-class relationship with more than one super-class.

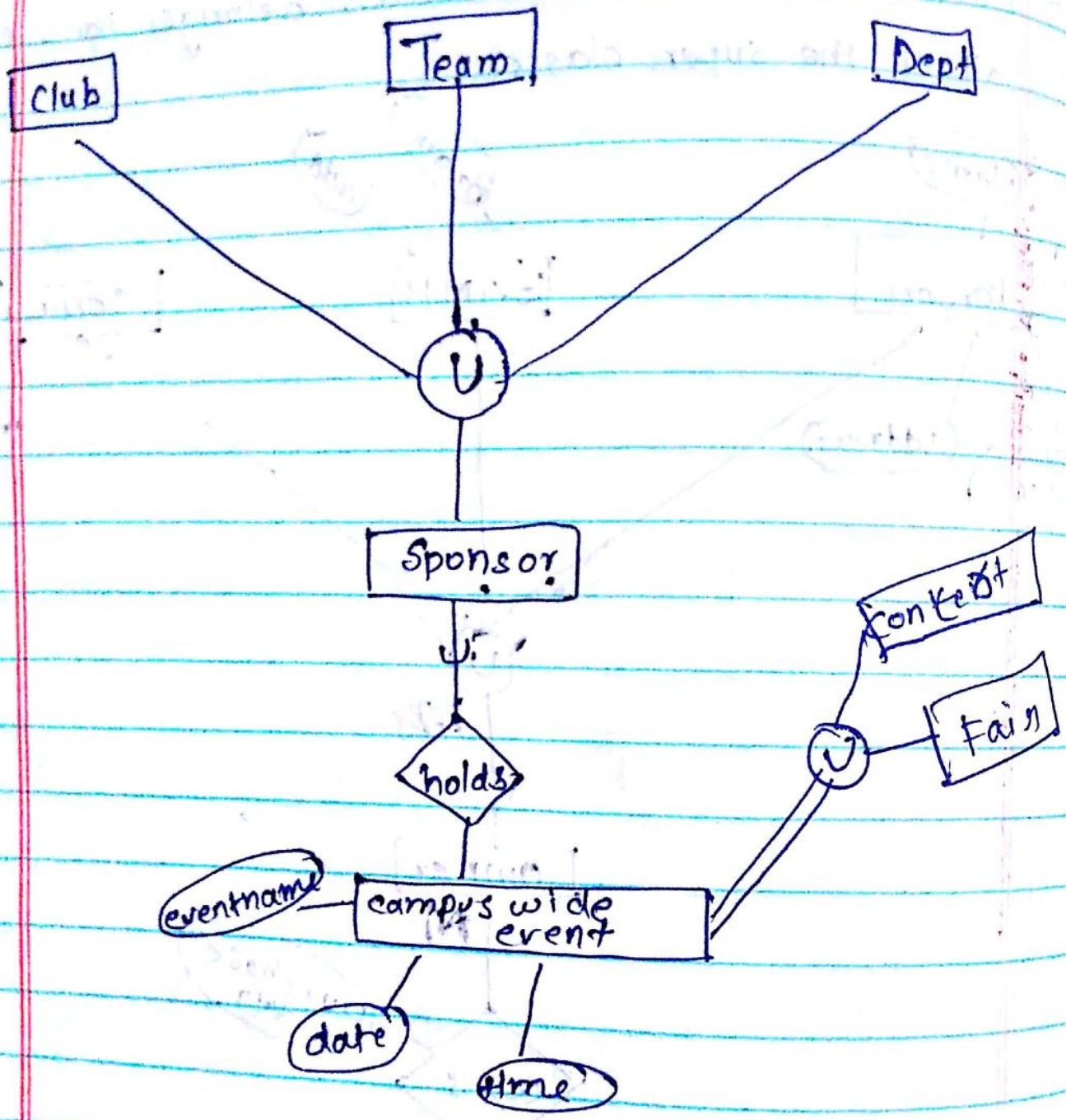
- Sub-class represents a collection of objects that is a sub-set of union of distinct entity types

Scanned by CamScanner

- Each instance of a sub-class belongs to one, not all of the super-classes.



Scanned by CamScanner



Scanned by CamScanner

16/2/17

Ch-7

~~Relational Algebra~~

The Relational Data Model, Data Constraints & Relational Algebra

Relational Model Concepts

Domain- The datatype describing the types of values that can appear in each column, is called a domain.

A domain D is a set of atomic values, by atomic mean that each value in the domain is indivisible.

A domain is given a name, datatype and the format.

Relation Schema (R) - A relation schema R is denoted by

$$\underset{\substack{\downarrow \\ \text{Relation name}}}{R}(\underbrace{A_1, A_2, A_3, \ldots, A_n}_{\substack{\downarrow \\ \text{set of attributes}}})$$

degree = 4

EMPLOYEE (SSN, NAME, AGE, SALARY)

Each attribute A_i, is the name of the role played by some domain D in the relation schema R.

Dom (A_i)

Scanned by CamScanner

Relation state(r) - A relation state r of the relation Schema $R(A_1, A_2, A_3, \ldots, A_n)$.

$r(R)$ is a set of n tuples $t_1, t_2, t_3, \ldots t_m$ where each tuple is a set of values.

$t = \langle v_1, v_2, v_3, \ldots, v_n \rangle$

Each v_i is an element of $dom(A_i)$

	A_1	A_2	A_3	A_4	.....	A_n
t_1	V_1	V_2	V_3	V_4		
t_2						
t_3						
⋮						
t_n						

$t_1[A_1]$

⇒ Characteristics of Relation

1) Ordering of tuples in a relation - A relation is defined as a set of tuples. Elements of a set have no order among them. When we display a relation as a table the rows are displayed in a certain order.

2) Ordering of values within a Tuple - In a relation schema definition order of elements is important. However at a logical level order of attributes and their values are not really important.

Scanned by CamScanner

3. Interpretation of a relation - We should take care of the predicate on which the relation is defined and based on that predicate, the subset has those values. If the condition satisfies then only that tuple becomes the member of relation.

Relational Model Notations

Relational constraints and relational database schema.

1. domain - constraints - Specify that the value of each attribute A must be an atomic value from the domain A.
2. Key constraints and constraints on null - Key constraints specify that a certain minimal subset of the fields of a relation is a unique identifier for a Tuple.

A Key K of a relation schema R is a super Key of R, with additional property that removing any attribute A from K leaves a set of attributes 'K' that is not a super Key of R.

⇒ Entity Integrity, Referential Integrity & Foreign Key Constraints

Entity Integrity Constraint

It states that no primary Key value can be null, because it is used to identify tuples in a relation and having nulls we can't identify some tuples.

Scanned by CamScanner

next page missing

= Referential Integrity Constraint

It is specified b/w 2 relations and is used to maintain consistency among tuples of the relation. It states that a tuple in one relation that refers to to another relation must refer to an existing tuple in the relation.

Referential Integrity Constraint is specified using foreign Key. A set of attributes FK in the relation schema R_1 is a foreign Key of R_1, that references relation R_2 if it satisfies the following 2 rules:-

1) The attributes in FK have the same domain as the primary Key attribute of R_2

2) A value of FK in a tuple t_1 of the current state $r_1(R_1)$ either occurs as a value of FK for some tuple t_2, in the current set $r_2(R_2)$ or is NULL.

$$t_1[FK] = t_2[FK]$$

and, we say that tuple t_1 references or refers to tuple t_2. R_1 is called <u>Referencing Relation</u> and R_2 is called <u>Referenced Relation</u>.

Scanned by CamScanner

→ Delete Operation

It can violate Referential Integrity if tuple being deleted is referenced by foreign key by other tuple in the database. Options :-

1) Either to reject the deletion.
2) To attempt to cascade the deletion by deleting tuples that references the tuple that is being deleted.
3) To modify attribute referencing value that causes the violation.

→ Update Operation

This operation is used to change the values of one or more attributes in a tuple of some relation R. Updating an attribute that is neither a PK or FK cause no problems, the DBMS only checks for the correct datatype and format, but modifying a PK or FK may violate entity integrity, referential integrity or Key constraints.

→ Relational Algebra

The fundamental operations of relational algebra are SELECT, PROJECT, UNION, SET DIFFERENCE, CARTESIAN PRODUCT, RENAME.

Scanned by CamScanner

1. SELECT

It is used to select a subset of ~~tuples~~ tuples from a tuples that satisfies a select condition $\sigma_P(R)$

$\sigma_{(condition)}(R)$

- Select the employees from the employee relation where department is 4.

$\sigma_{DNO=4}(EMPLOYEE)$

- Select the employees from the employee relation where salary is greater than 3000.

$\sigma_{salary>3000}(EMPLOYEE)$

- Select employees who either work in dept no. 4 and earn 20000 per year or work in dept no. 5 and earn 30000 per year.

$\sigma_{(DNO=4 \text{ AND } SAL=20000) \text{ OR } (DNO=5 \text{ AND } SAL=30000)}(EMPLOYEE)$

Scanned by CamScanner

28/2/17

Character Functions

1. CONCAT ("Hello", 'World').

   Select CONCAT (FNAME, LNAME) from emp;

2. SUBSTR

   Select SUBSTR('HelloWorld', 1, 2) from dual;

   -1

   starting posn

   no. of characters

3. LENGTH

   Select LENGTH (fname) from emp;

4. INSTR

   Select INSTR('HelloWorld', 'W') from dual;

5. LPAD

   select LPAD (Salary, 10, '*') from emp;

6. RPAD

7. TRIM

   Select TRIM ('H' from 'HelloWorld') from dual;

8. ROUND

   Select ROUND (45.926, 2) from dual;

   45.93

Scanned by CamScanner

9. MOD (Salary, 12)

10. TRUNC (45.926, 2)

Scanned by CamScanner

2/08/17

2 Projection ($\Pi_{A1, A2, A3}$)

$\Pi_{FNAME, LNAME, SALARY}$

→ Retrieve the first name, last name & salary of all employees who work in dept no 5.

$temp \leftarrow (\sigma_{Deptid=5} (Emp))$ //rename operation

$\Pi_{FNAME, LNAME, SAL} (temp)$

→ Set Operations

Set Operations that are performed on 2 sets are called binary operations. There are 3 set operations

1) Union
2) Intersection
3) Set Difference

~~# Union Operation~~

Student

FN	LN
Susane	Yao
Ramesh	Shah
Ammy	Ford
Jimmy	Gillbert

Instructors

FName	LName
John	Smith
Ricardo	Brownie
Susane	Yao
Francis	John
Ramesh	Shah

Scanned by CamScanner

Student ∪ Instructor

FN	LN
Susane	Yao
Ramesh	Shan
Amy	Ford
Jimmy	Gillbert
John	Smith
Ricardo	Brownie
Francis	John

Student ∩ Instructor

FN	LN
Susane	Yao
Ramesh	Shah

Student − Instructor

FN	LN
Amy	Ford
Jimmy	Gillbert

Scanned by CamScanner

Cartesian Product or Cross Product

* Wastes Memory

Student X Instructor

FN	LN	FName	LName

JOIN OPERATION

It is used to combine related tuples from 2 relations into a single tuple.

In JOIN operation, only the combinations of tuples satisfying the join cond^n appear in the result.

A general JOIN operation is represented as $R \bowtie_{(\text{Join cond}^n)} S$

The general form of JOIN cond^n is
<condition> AND <condition> OR <condition>

Each cond^n is of the form A_i θ B_j

A_i → Attribute

θ → $<, >, \leq, \geq, =$

JOIN operation does not necessarily preserve all information

The JOIN which involves equality, comparison in the JOIN cond^n is called equi-join

Scanned by CamScanner

→ Retrieve the name & address of all employee who work for the research department.

Name	address	Dept-id
Priya	Delhi	5
Moksh	Noid	2
Sachin	Gurgon	3
Rituvee	Punjab	1
Veeru	Haryana	5

Did	Dname
5	research
1	—
2	—
3	—
4	—

$$\&dept \leftarrow \sigma_{Dname='Research'}(Dept)$$

$$Temp \leftarrow Emp \bowtie_{DeptnoR} (Dept)$$

$$\pi_{Name, Address}(temp)$$

Ques. Boats

BID	BName	Color
101	Interlution	Blue
102	"	Red
103	Clipper	Green
104	Marine	Red

Scanned by CamScanner

Sailor

Sid	Sname	Rating	Age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horato	7	35
71	Zorba	10	16
74	XYZ	9	35
85	Art	3	25
95	Bob	3	63

Reserves

SID	BID	DAY
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/07/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Scanned by CamScanner

a) Find the names of sailors who has reserved the boat 103.

b) Find the colours of boats reserved by Lubber.

c) Find the names of sailors who have reserved atleast 1 boat.

d) Find the names of the sailors who have reserved a red or green boat.

e) Find the SIDs of sailors, with the age over 20 who have not reserved a red boat.

f) Find the names of sailors who have reserved all boat.

g) Find the names of sailors who have reserved all boats called Interlaken.

A

SNo	PNo
S_1	P_1
S_1	P_2
S_1	P_3
S_1	P_4
S_2	P_1
S_2	P_2
S_3	P_2
S_4	P_2
S_4	P_4

B_1

P. No
P_2

B_2

PNo
P_2
P_4

B_3

PNo
P_1
P_2
P_3

A/B_1

SNo
S_1
S_2
S_3
S_4

A/B_2

SNo
S_1
S_4

A/B_3

SNo
S_1

Scanned by CamScanner

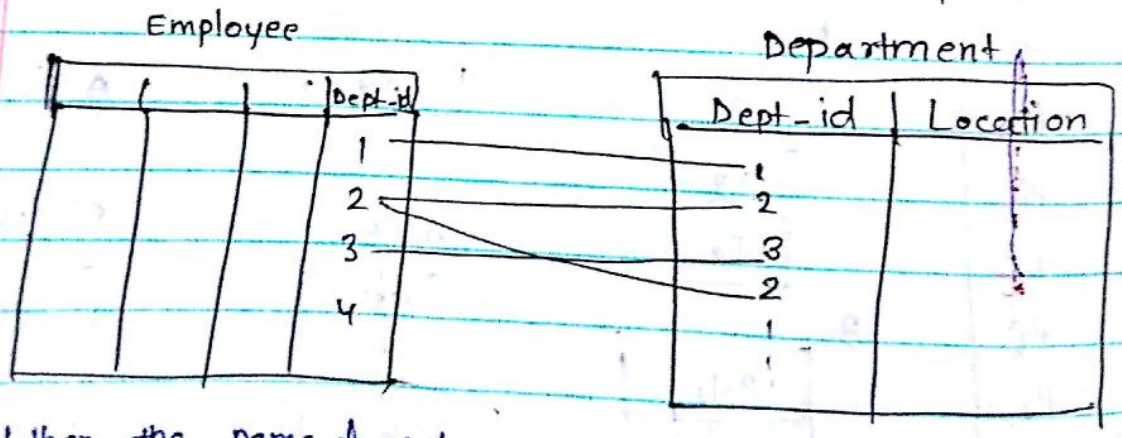
* Inner join & Equi-join is same

6/3/17

Transjection

→ Equi / Inner join

Select e.empid , e.lastname , e.dept-id , d.dept-id , d.location from employee e , department d where e.dept-id = d.dept-id;



- When the name of columns are same in 2 tables, then it becomes mandatory to use the table alliases.
- We can give multiple cond^{n}s when we are using JOIN cond^n.

Scanned by CamScanner

→ Non-Equi Join (Between, <=, >=)

Joining those table which does not have a common column.

Employee

Lname	Salary
Lakshay	50,000
Amit	24,000
Rajat	3000
Shubham	15000
Shobhit	20,000

Job_Grade

Grade	Low_Sal	Highest_Sal
A	10,000	20,000
B	21,000	30,000
C	31,000	40000
D	41,000	50,000
E	51,000	60,000
F	61,000	70,000

select e.lastname, e.salary, J.Grade, J.low_sal, J.highest_sal from Employee e, Job_Grade J where e.salary between J.low_sal and J.Highest_sal;

→ Outer Join (+)

Employees

Lname	Salary	Dept_id
Lakshay	50,000	90
Amit	24,000	10
Rajat	30,000	110
Shubham	13,000	60
Shobhit	29,000	20
Rohit	5,000	30
Sunit	10,000	30

Department

Dname	Dept_id
Admin	10
Marketing	20
Shipping	30
IT	60
Sales	80
Execution	90
ACC	110

Scanned by CamScanner

When the values are not matching and when we need the data, so we use the outer join.

Full outer JOIN both side (+) = (+)

- select e.lname, e.sal, e.dept, d.dname from employee e, department d where e.deptid(+) = d.deptid;

① (+) =

where table.column(+) = table2.column

② = (+)

where table.column = table2.column(+)

- The missing rows can be returned if an outer Join operator is used in the JOIN condⁿ. The outer join operator is a plus sign and enclosed in paranthesis and it is placed on the side that is deficient in information.

Ename	sal	dep	dname	dept id
–	–	–	Contract	190

(+) it is used where there is a deficiency.

* We can not use the OUTER JOIN when we are using in operator

Note: OR operator will not work with the OUTER JOIN operator.

Scanned by CamScanner

→ SELF JOIN (When 2 copy of the same table is created)

Employees (Worker)

EID	ENAME	MGRID
100	King	
101	Kochher	100
102	Haam	100
103	Hurnold	102
104	Einstein	103
107	Lorentz	103

Employees (MANAGER)

EID	LNAME
100	King
101	Kochher
102	Haam
103	Hurnold
104	Einstein
107	Lorentz
124	Morris

```
Select worker.LName , worker.EID || 'works for' || manager.LNAME
from Employees worker , Employees Manager where
  worker.MGRID = Manager.EID ;
```

→ NATURAL JOIN

Natural join is almost same to equi-join but with a one difference i.e. the 2 column name should be equal. There is no need of where because the name of 2 columns is same.

Scanned by CamScanner

- NVL Function
  To include the NULL values in group functions.

- Select AVG(NVL(comm, 0))
  from employees;

  NVL → Null Values

- Group By Function
  Select dept-id, AVG(salary) from employees
  group by dept-id;

We specify all the required column names ~~except~~ that are to be Select in a GROUP condⁿ except the group function.

We using HAVING clause here.

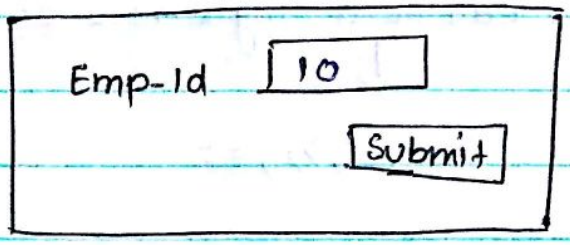
- Select from GROUP by having.

Scanned by CamScanner

- Substitution Variable (&)

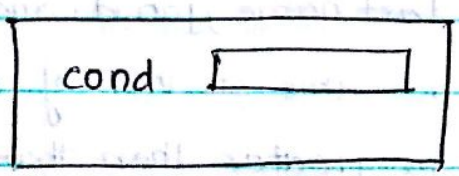
  The values are given at a run time.

```
select lname, salary, emp-id
from employees
where emp-id = &emp;
```



Also,

```
select lname, salary, emp-id
from employees
where & cond or order by & cond.
```



Scanned by CamScanner

7/03/17

Sub- Queries

Select select list
from table name
where expr operation (select select List from table name);

↓

can be a single row or multiple row operator

↓ ↓

>, <, ≤, ≥, == IN, ANY

where
from
having
} used the sub query

Q. Display the last name, job id. and salary of employee whose job id is same as that of employee 141 and whose salary is greater than that of employee 143.

```
Select lastname, job-id, salary from Employees
where job-id = (select job-id from employees
                where emp-id = 141 and
                salary > (select salary from employees
                          where emp.id = 143));
```

Scanned by CamScanner

Q. Display the employees lastname, job-id, salary of all employees where salary is equal to min^m salary.

```
select lastname, job-id, salary
from Employee
where salary = (select min(salary) from Employee);
```

Q. Display dept-id, min^m salary of all departments that have a min^m salary greater than that of department 50:

```
select dept-id, min(salary) from Employee
group by dept-id
having min(salary) > (select min(salary) from employees
                      where dept-id = 50);
```

Q. Find the job with the lowest average salary.

```
Select job, min(Avg(salary)) from Employee;
```

Scanned by CamScanner

Update operations and Dealing with constraint Violations

3 basic Update Operations in a relation are

1) Insert
2) Delete
3) Modify

→ Insert Operation

It provides a list of attribute, tuple t that is to be inserted into a relation R. It can violate any of the 4 types of constraints

- Domain constraints can be violated, if any attribute value given, does not ~~corresp~~ appear in the corresponding domain.
- Key constraints can be violated if a Key value in the new tuple t, already exists in another tuple in r(R).
- Entity Integrity constraint can be violated if primary Key is inserted null.
- Referential Integrity constraint can be violated if value of foreign Key does not exist in reference relation.

Scanned by CamScanner

20/3/17

Transaction Processing

Multiprogramming OS executes some commands from one process then suspends that process and executes some commands from next process and so on.

A process is resumed at the point where it is suspended whenever it gets its turn to use CPU again.

Hence, concurrent execution of process is interleaved.

Transactions, Read & Write Operation

A transaction is an executing program that forms a large unit of database processing. It includes one or more database access operations - insertion, deletion, modification or retrieval.

A transaction boundaries are specified by 'begin transaction' and 'end transaction'.

Read only transaction are those transactions in which a database state is not updated but it performs only retrieval.

The basic database access operations that a transaction can include are -

i) read_item (x) - Reads the database item named X into a program variable.

Scanned by CamScanner

ii) write-item(x) - It writes the value of program variable X into the database item.

The DBMS generally maintain a no. of buffers in main memory that holds database disc blocks containing the database items being processed.
A transaction includes read-item and write-item operations to access and update the database.

(A) T_1	(B) T_2
read-item(x)	read-item(x);
x := x - N	x := x + M;
write-item(x);	write-item(x);
read-item(y);	
y: y + N;	
write-item(Y);	

Several problems can occur when concurrent transaction execute in an uncontrolled manner.
Eg: Consider an airline reservation DB in which record is stored for each airline flight. Each record includes the no. of reserve seats on that flight as a name data item.

Scanned by CamScanner

In the above figure A, the transaction T_1 that transfers N reservation from one flight whose no. of reserved seats is stored in the database named Y.

Fig. B shows that, it reserves M seats on the first flight X referenced in transaction T_1.

A database access program has the flight no.s, their dates and no. of seats to be booked as parameter.

The same program can be used to execute many transactions each with different flights and no. of seats to be booked.

In the above figure T_1 and T_2 are executions of the programs that refers to specific flights whose no. of seats are stored in data items X and Y.

Scanned by CamScanner

- Problems encountered when transactions can run concurrently.

1) Lost - Update Problem

This problem occurs when 2 transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.

T_1	T_2
read-item(x);	
x := x - N;	
	read-item(x)
	x := x + M;
write-item(x);	
~~write~~ read-item(y);	
	write-item(x);
y := y + N;	
write-item(y);	

Scanned by CamScanner

2. Temporary Update Problem

This problem occurs when one transaction update a database item & then the transaction fails for some reason. The (updated item) is accessed by anothe transactions before it is changed back to its original values.

T_1	T_2
read-item (x);	
x : x - N;	
write-item(x);	
	read-item(x);
	x := x + M;
	write-item(x);
read-item (y);	

Time ↓

3. Incorrect Summary Problem

If one transaction is calculating and aggregate summar funcⁿ ~~the~~ no. of records while other transactions are updating some of these record, then aggregate funcⁿ may calc. some value before they are updated & others after they are updated.

Scanned by CamScanner

T_1	T_3
	sum := 0;
	read-item(A);
	sum := sum + A;
read-item(x);	
x := x - N;	
write-item(x);	
	read-item(x);
	sum := sum + x;
	read-item(x);
	sum := sum + Y;
read-item(Y);	
Y := Y + N;	
write-item(Y);	

T_3 reads X after N is subtracted & reads Y before N is added

wrong summary is the result

4. Un-Repeatable Read

In this a transaction T reads an item twice and the item is changed by another Transaction T' b/w the two reads. Hence, T receives different values for two reads of same item.

Scanned by CamScanner

21/3/17

Need for Recovery

Whenever a transaction is submitted to DBMS for execution, the system is responsible for making sure

1) All the operations in the transaction are completed sucessfully and their effect is recorded permanently in the database.
2) The transaction has no effect on the database or on any other transaction if it fails.

→ Types of failures

1. System Crash - A hardware, software or network error occurs in the computer system during transaction execution.

2. A Transaction or System Error - Some operations in the transaction may cause it to fail. For eg. Divide by Zero, Integer overflow, Erronious parameter value or any logical programming error.

3. Exception Cond^n detected by the transaction - During transaction execution, certain cond^n force cancellation of Transaction.

Scanned by CamScanner

For eg. data not found, insufficient a/c balance etc.

4. Disk failures - Some disk blocks may loose their data because of read-write malfunch or read-write head crash.

5. Catastrophic Problems - Physical problems

→ Transaction and System Concept

A transaction is an atomic unit of work i.e., either completed in its entirety or not done at all.

For recovery purposes, the system needs to keep tracks of when the transaction start, terminates, commit or abort.

Recovery manager keeps track of the following operations

1) BEGIN TRANSACTION - This marks the beginning of transaction execution.

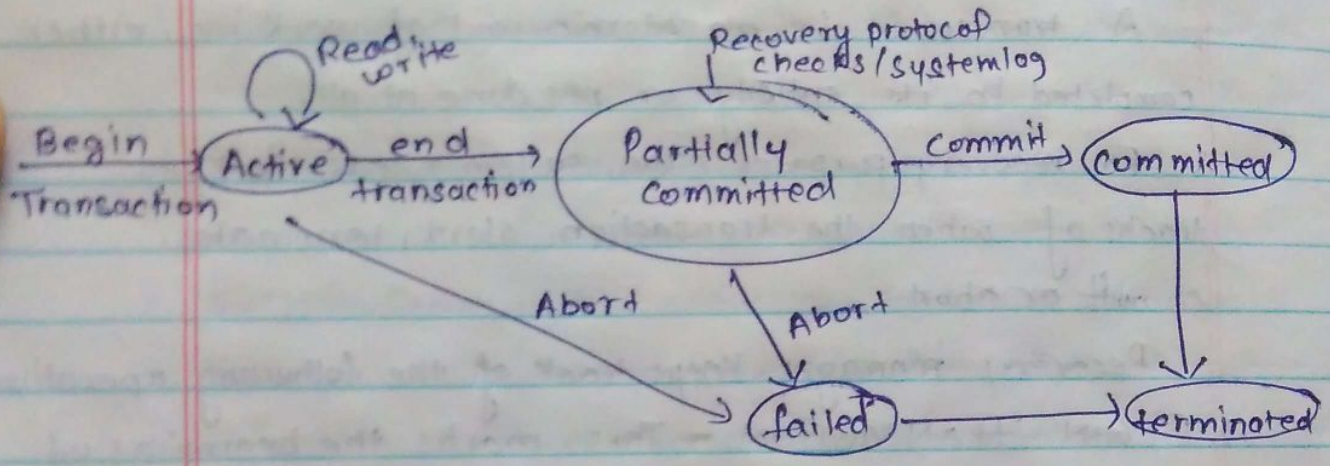
2) Read or Write Operation - This specifies the reading & writing operations on a particular item in the database.

3. END Transaction - It specified the read & write operations have ended and marks end of transaction execution. At this step, it is necessary to check whether changes made to the database are permanent (COMMITTED) or whether transaction is aborted.

Scanned by CamScanner

4) COMMIT Transaction – This signals a sucessful ends of the transaction, so that any changes executed by the transaction can be safely committed to the database and will not be undone.

5) ROLLBACK or Abort – This signals that the transaction has ended unsucessfully so that any changes or effects made by transaction must be undone.



→ System Logs

To recover from failure, the system maintains a log to keep track of all the transaction operation that affects the values of database item.

Entries in the log record includes

T:=

Unique transaction id that is generated automatically by the system,

Scanned by CamScanner

- [Start Transaction, T] - It indicates that transaction T has started execution.

- [write-item, T, x, old-value, new-value] - It indicates that transaction T has changed the value of database item X from old-value to the new-value.

- [read-item, T, x] - It indicates that the transaction T has read the value of DB item X.

- [Commit, T] - It indicates that transaction T has completed sucessfully and affirms that its effect can be committed and recorded permanently in the database

- [abort, T] - It indicates that transaction T is stopped forcefully because of its unsuccessful execution.

If System failure occurs we search back in the log for all transactions T, that have written Start Transaction record into the log but have not written COMMIT. Those transactions may have to be rolled back to undo their effect.

Scanned by CamScanner

→ ACID Properties (Properties of Transaction)

1. Atomicity - It says that a transaction is either performed in its entirety or not ~~pro~~ performed at all.

2. Consistency Preservation - It ensures that any transaction will bring the database from one valid state to another. Any data returned to the database must be valid a/c to all defined rules, constraints, cascades, triggers etc.

3. Isolation - A transaction is executed as it is being executed in isolation from other transaction i.e., the effect of an incomplete transaction is not visible to others transaction.

4. Durability - The changes ~~that are~~ applied to the database by a committed transaction must persist in the database even in the event of power loss, crashes or error.

Scanned by CamScanner

Relational Database Design & Functional Dependencies & Normalization

- Measures of Qualities for Relational Schema Design

1) Semantics of relational attributes

The semantics specifies how the attribute valuesin a tuple relate to one another. We design a schema so that it is easy to explain its meaning. We do not combine attribute from multiple entity types & relationship type into a single relation.

2) Reducing the redundant value

Our goal of schema design is to minimize the storage space used by the relation.

Grouping the attributes into a relational schema has a significant effect on storage space.

Employee -

Ename	Eno	DOB	Add	Dnumber

Department -

DName	Dno	Dlocation

Emp-dept

Ename	Eno	DOB	Add	DNo	Dname	Dlocation

Scanned by CamScanner

In Employee department table, the attribute value pertaining to a particular department i.e dno, dname, dlocation are repeated for every employee who works for that department. In contrast, department info appears only once in the dept relation.

Another ~~employe~~ problem arises in such relation is the problem of update anomaly - ~~up~~ Insert/Delete/Modify

→ Insertion Anomalies

These can be differentiated into 2 types

1. To insert a new Employee into Emp. dept table, we must include either the attribute for the department, that employee works for or keep it as null.
2. It is difficult to insert a new dept. that has no employees in it.

→ Deletion Anomalies

It may not be possible to delete certain info without some other unrelated info. If we delete an employee from emp-dept table, then the info. concerning the dept is also lost.

Scanned by CamScanner

24/3/17

→ Modification Anomaly

If one copy of repeated data is updated, inconsistency is created unless all copies are similarly updated. We should design the database relation schemas so that no insertion, deletion or modification anomalies are present in the relation.

3) Reducing the NULL value

In some schema designs, we group ~~may~~ many attributes into one relation & if many of the attributes do not apply to all tuples then we end up with many NULLs. This may waste storage space, creates difficulty in understanding semantics & performing join operation.

- This allows the possibility of generating spurious tuples.

Emp-locs

Ename	Plocation

Emp-proj

Eno	Pno	Hours	Pname	Plocation

If we apply natural join on above 2 tables, the result produces many tuples.

Scanned by CamScanner

Emp-locs

Ename	Plocation
MK	Bangalore
MK	New Delhi
RN	Hyderabad
KG	Bangalore
KG	N.D
KC	N.D
KC	Hyderabad
KC	Chennai

Emp-proj

Eno	Pno	Hours	Pname	Plocation
123	1	32	X	Banglore
123	2	7	Y	N.D
666	3	40	Z	Hyderabad
953	1	20	X	Bangalore
953	2	20	V	N.D

Eno	Pno	Hours	Pname	Plocation	Ename
123	1	32	X	Bangalore	MK
123	1	32	X	Bangalore	KG
123	2	7	Y	N.D	MG
123	2	7	Y	N.D	KG
123	2	7	Y	N.D	KC
666	3				
666	3				
953	1				
953	1				
953	2				
953	2				
953	2				

Scanned by CamScanner

In the above table, there are many rows that represent wrong info. This is because Plocation is the attribute that relates the two tables which is neither primary key nor foreign key.

→ Functional Dependencies

It is a constrain b/w 2 sets of attributes from the database.

let R be a relational schema, $R = \{A_1, A_2, A_3, \ldots, A_n\}$

Let, X and Y be non-empty sets of attributes in R, we say that an instance of $r(R)$ satisfies the functional dependencies $X \rightarrow Y$, if the following constraints holds for every pair of tuples t_1, t_2 in r.

If $t_1[X] = t_2[X]$

then $t_1[Y] = t_2[Y]$

This means that the values of Y component of a tuple in r depends on or are determined by the value of X component.

— We say that there is a functional dependency from X to Y or Y is fonctionally dependent on X.

Two points have to be taken into consideration

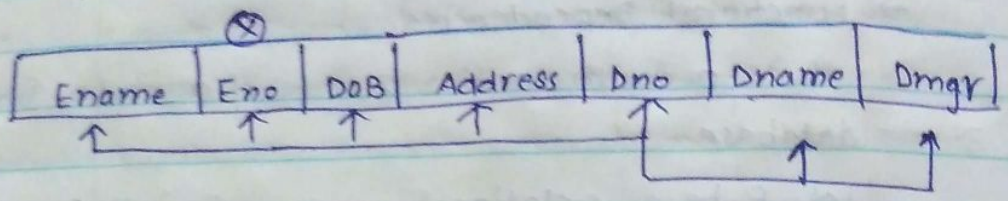
1. If a constraint on R states that there cannot be more than one tuple with a given X value in $r(R)$ i.e. X is a candidate key of R.

Scanned by CamScanner

2. If $X \rightarrow Y$ in R, this does not say whether $Y \rightarrow X$ is also true.

Each functional dependency is displayed by a horizontal line.

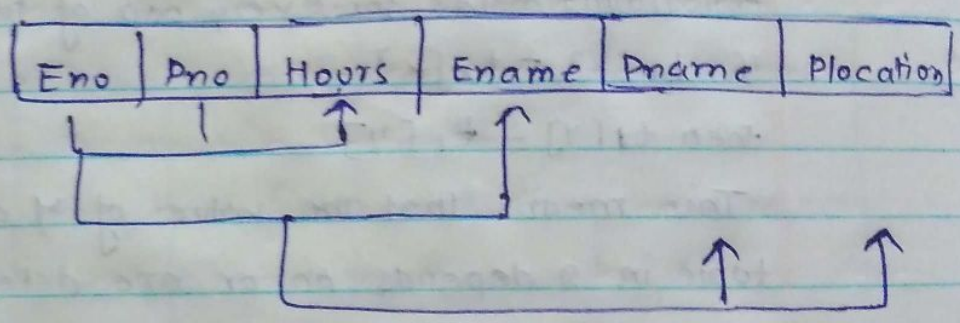
Emp-dep



ENo → Ename, DOB, Add, DNo

DNo → Dname, DMgr

Emp-proj



⇒ <u>Inference rule for functional dependencies</u>

Some dependencies are given in the set however some other functional dependencies hold on relation instances that can be inferred or deduced on FDs in F.

Scanned by CamScanner

~~Closure~~

- Closure - It includes all possible dependencies that can be inferred from the given set F & is denoted by F^+. The Armstrong axioms can be applied repeatedly to infer all FDs implied by a set F.

i) Reflexivity : If $X \supseteq Y$, then $X \to Y$, it states that a set of attribute always determine itself of any of its subset.

ii) Augmentation : If $X \to Y$, then $XZ \to YZ$, it says that adding the same set of component on both side the resulting will also implies the same.

iii) Transitive : If $X \to Y$, $Y \to Z$, then $X \to Z$

iv) Decomposition (Projector) : If $X \to YZ$, then
$X \to Y$ &
$X \to Z$

v) Union / Additive : $X \to Y$, $X \to Z$, then $X \to YZ$

vi) Pseudotransitive : $X \to Y$ & $WY \to Z$ then $WX \to Z$

Normalisation

Proposed by Dr EF Codd. It takes a relational schema to a series of test to certify whether it satisfies certain normal form. It is a top down process of decomposing a relational schema as necessarry. Initially, 3 normal forms were proposed :- 1NF, 2NF, 3NF. All these normal form are based on functional

Scanned by CamScanner

dependencies among attributes of a relation. Normalisation of data is a process of analysing the given schema to achieve the desirable properties :-

1) Minimising Redundancy.
2) Minimising insertion, deletion, modification anomaly.

Schemas that do not mean normal form test are decomposed into smaller relation schema.

Normalisation process through decomposition must confirm the two additional properties

1) The lossless join / Non-additive join property.
It gaurantees that the spurious tuples problem doesnot occur wrt relational schema created after decomposition.

2) Dependency Preservation Property
It ensures that each functional dependency is represented in some relational schema resulting after decomposition.

First Normal Form (1NF)

It disallows multi-value & composite attributes. It states that the domain of an attribute must include only atomic or simple values.

Scanned by CamScanner

Department

Dname	Dno	DMgr	Dlocation
Research	5	101	N.D, Bang, Chennai
Admin	4	15	Hyderabad
A/c	1	126	Pune, Noida

There are 3 main techniques to achieve first normal form -

1) Remove the attribute Dlocation that violates 1NF and place it in a separate relation say dept. loc. along with the DNo. The primary key of new relation will be combination of DNo & Dlocation.

Dept loc

Dno	Dlocation
5	N.D
5	Bangalore
5	chennai
4	Hyderabad
1	Pune
1	Noida

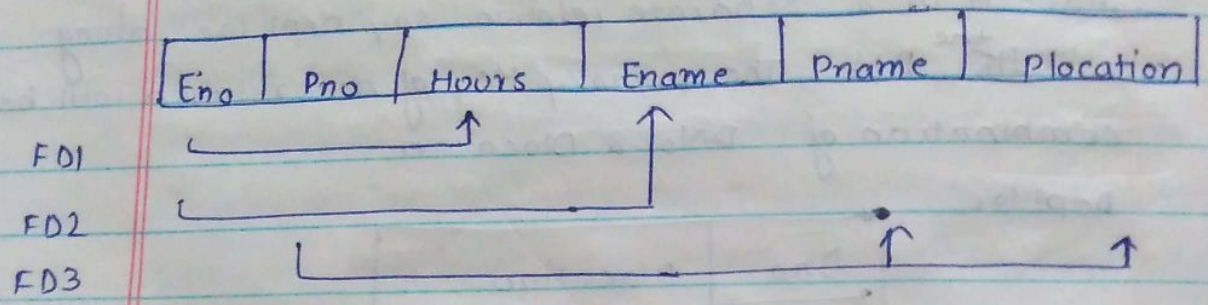
Scanned by CamScanner

30/3/17

Second Normal Form :

It is based on full functional dependency. ~~$X \to Y$~~ X

~~is~~ $X \to Y$ is a full f.d if removal of any attribute A from X, means that dependency does not hold any more i.e., for any attribute $A \in X$, $X - \{A\}$ does not functionally determine X. F.D $X \to Y$ is partial if $X\{n\}$ is done and dependency still hold. A relation schema R is in 2NF, if every non-prime attribute in R is fully functionally dependent on the primary key R



Emp-Pnoi

Eno	Pno	Hours

Emp

Eno	Ename

Project

Pno	Pname	Ploc

3rd Normal Form

It is based on the concept of transitive dependency. A F.D $X \to Y$ in a relation schema R is a transitive dependency if there is a set of attributes Z i.e., neither a candidate key nor a subset of any key of R and both $X \to Y$ and $Y \to Z$ holds.

Scanned by CamScanner

Emp-dept

Ename	Eno	DOB	Add	Dno	Dname	DMgr

X FD1 FD2

Eno → Ename, DOB, Add, DNo $X \to Y$

Dno → Dname, DMgr $Y \to Z$

$X \to Z$, this should not be done

We separate then

Eno	Ename	DOB	Add	Dno

DNo	DName	DMgr

Boyce Codd Normal Form

Every relation in BCNF is also in 3NF. For a relation to be in BCNF, following condns must be satisfied

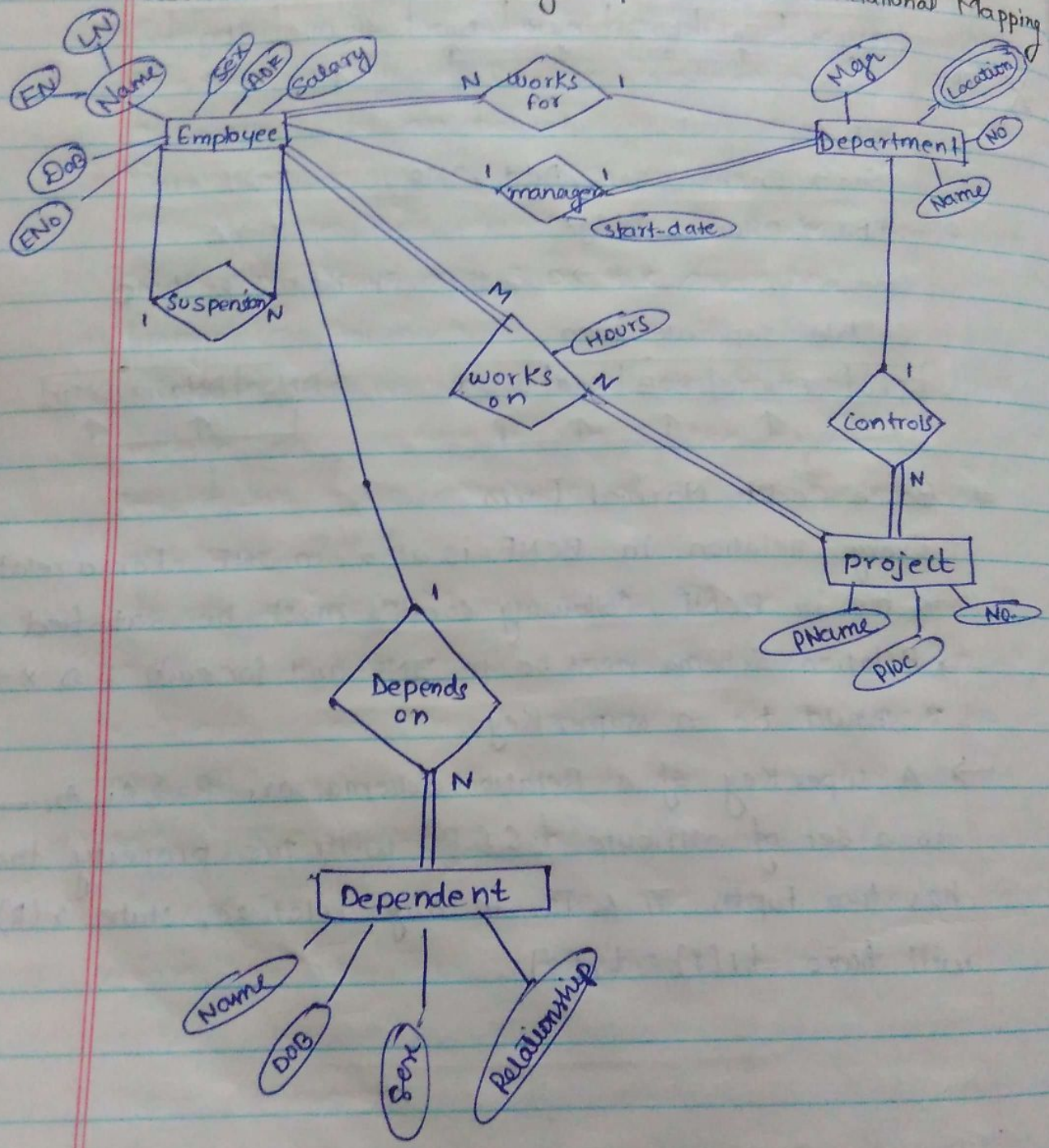
→ Relation Schema must be in 3NF and for each F.D $X \to Y$, X should be a super key.

→ A super key of a Relation Schema are $R = \{A_1, A_2, \ldots, A_n\}$ is a set of attribute $(S \subseteq R)$ with the property that has two tuples T_1 & T_2 in any relation, state $r(R)$ will have $t_1[S] = t_2[S]$.

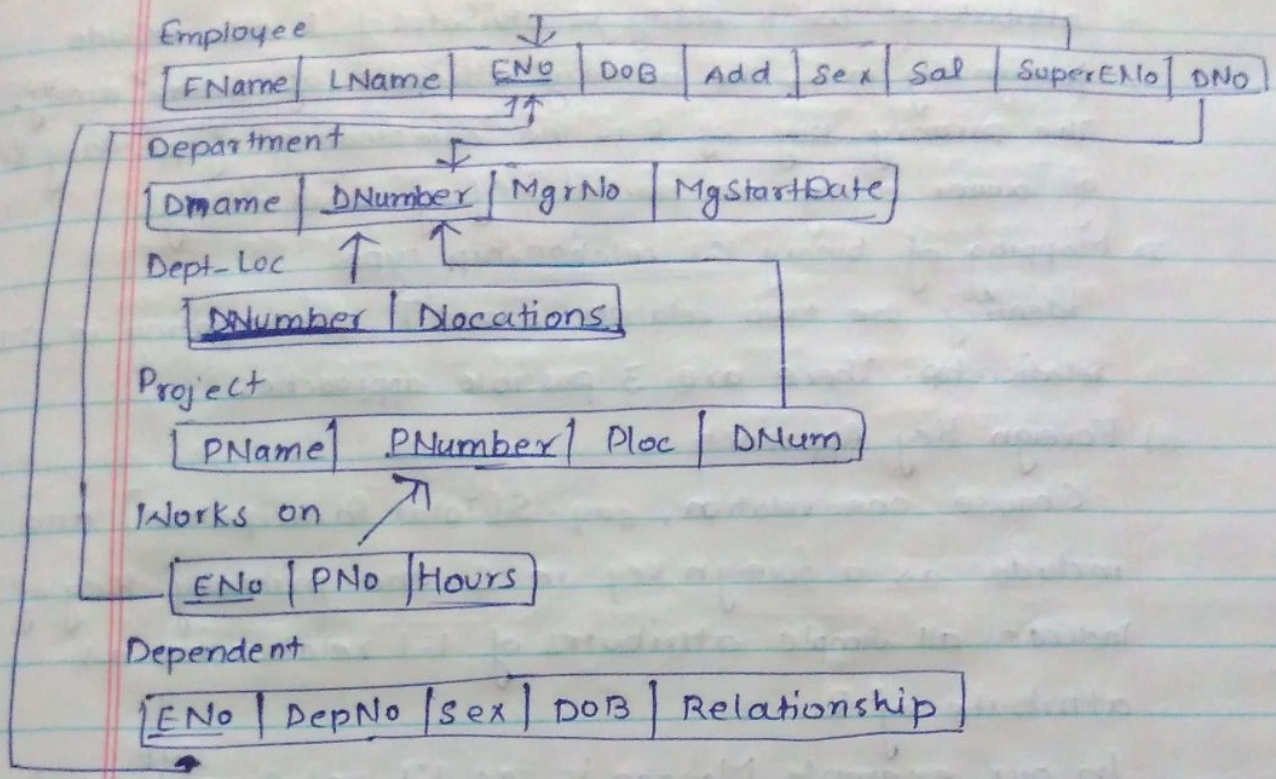
Scanned by CamScanner

30/3/17

Relational Database Design by ER & EER to Relational Mapping



Scanned by CamScanner



→ **MAPPING STEPS**

1. **Mapping of regular entity type**

For each strong entity in the ER create a relation R that includes all simple attributes of an entity. Include only the lowest component attributes of composite attribute. Create primary key from key attribute.

2. **Mapping of weak entity types**

For each weak entity type in the ER schema with owner entity create a relation R and include all simple

attributes of weak entity as attributes of R also include foreign key in R. the primary attribute of owner entity. The primary key of R is the combination of primary key of owner entity an partial key of weak entity.

3) Mapping of binary 1:1 relationship types.

Identify the two relation S and T, participating in 1:1 relationship. There are 3 possible approaches

a) Foreign Key Approach

Choose one relation, say S [Total Participation] and include as a foreign key in S the primary key of T. Include all simple attributes of 1:1 relationship as attribute of R.

In our example, Manage is a 1:1 relationship type. Dept. acts as a role of S because it has total participation i.e., every dept has a manager. We include primary key of Employee as foreign key in dept..

b) Merge Relationship Option

Merge the two entity type and the relationship into a single relation. This is appropriate when both participation is total.

c) Cross Reference

Set up a relation for cross reference the primary key of two relation S and T. This is used in M:N relationship & the resulting relation is called

Scanned by CamScanner

Relationship Relation on lookup table.

4) Mapping of binary 1:N relationship type
Identify the relation S on the N side of 1:N relationship. Include as FK in ~~tr~~ S the primary key of the relation T. Include any simple attribute of 1:N relationship in S.

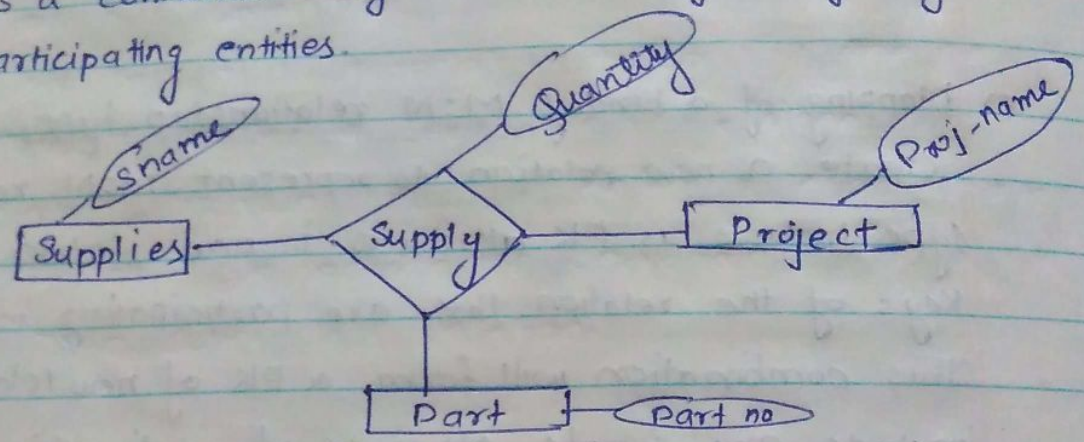
5) Mapping of a binary M:N relationship type
Create a new relation to represent M:N relationship type. Include as FK attributes in new relation the primary keys of the relation that are participating in M:N type. This combination will form a PK of new relation. Also include any simple attribute of relationship into a new relation.

6) Mapping of Multi-valued Attribute
For each multi-valued attribute, create a new relation & include the primary key of original relation along with the multi-valued attribute. The PK and new relation is a combination of multi-valued attribute & PK of original relation.

Scanned by CamScanner

7) Mapping of a Ternary Relationship type

For each ternary relationship type where $N \geq 2$, create a new relation S to represent schema. Include as FK attribute in S, the primary key of the relation that participate in ternary type. Also include simple attributes of ternary relationship type. The PK of S is a combination of all the foreign keys of participating entities.



Supply

Sname	Proj-name	Partno	Quantity

8) EER-schema option for mapping Specialization or Generalization.

Convert each specialization with m sub-classes $\{S_1, S_2, \ldots, S_m\}$ and (generalize) super-class where the attributes of superclass are $\{K, a_1, a_2, \ldots, a_n\}$ where K is PK into the relation schema using one of the 4 option.

Scanned by CamScanner

a) Multiple relation super-class & the sub-class
Create a relation L for super-class C with attributes $\{K, a_1, a_2, \ldots a_n\}$ and PK of L as K.
Create relation L_i for each sub-class S_i where $i \leftarrow 1$ to n with the attributes ~~with one~~ $K \cup$ (Attributes of S_i) and ~~make the~~ PK of L_i as K. (This works for all specialization (total or partial)(disjoint or overlap).

b) Multiple relation - subclass relation only
Create a relation L_i for each sub-class S_i with attribute of L_i as attributes of $S_i \cup \{K, a_1, a_2, a_3, \ldots a_n\}$ and make the PK of L_i as K. This operation works for specialization whose sub-classes are total i.e., every entity in the super-class ~~before~~ must belong to atleast one of the sub-class.

c) Single relation with one type of attribute
Create a single relation L with attribute $\{K, a_1, a_2, a_3, \ldots a_n\} \cup \{$Attributes of $S_1\} \cup \{$Attributes of $S_3\} \ldots \{$Attributes of $S_n\} \cup t$ and make PK of L as K. 't' is a ~~discriminating~~ discriminating factor. This works for specialization whose sub-classes are disjoint.

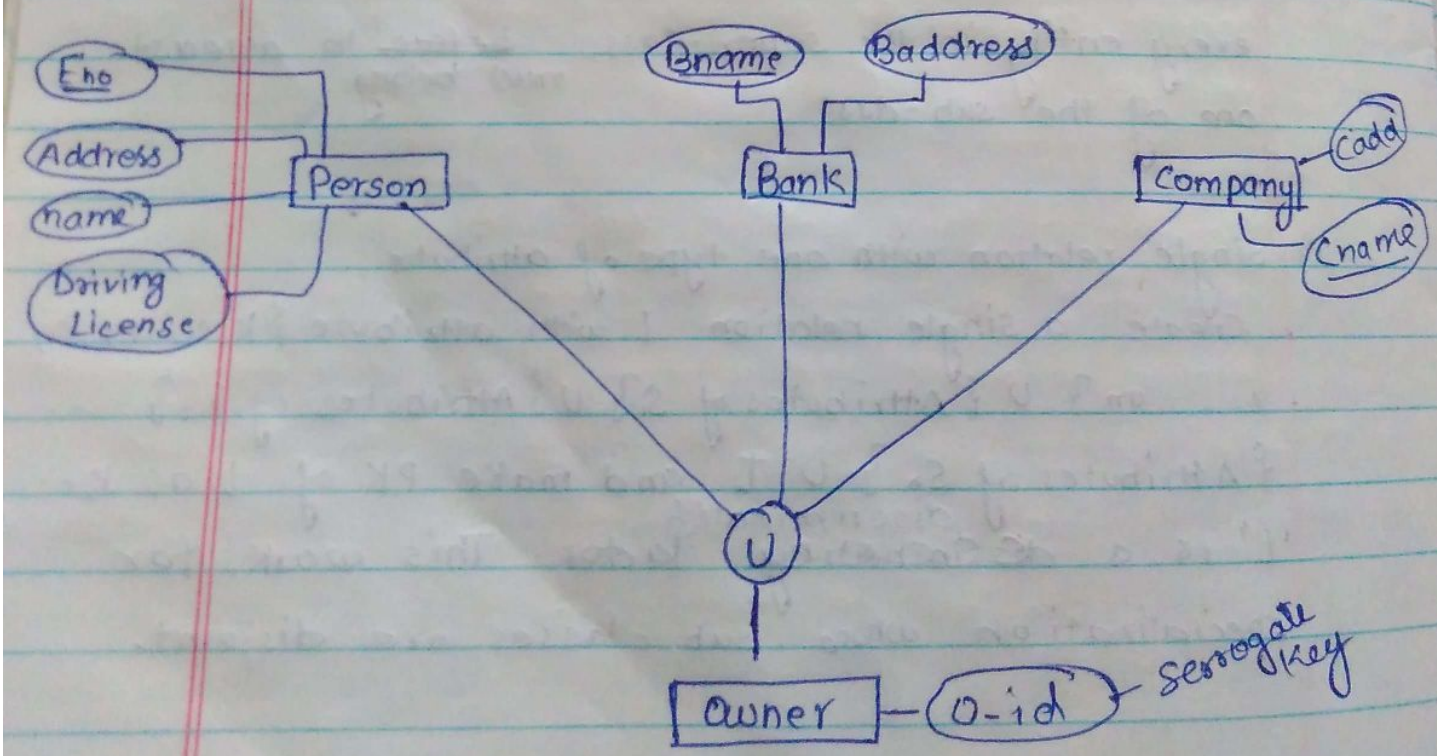
Scanned by CamScanner

d) Single relation with multiple type attribute

Create a single relation schema L with attribute $\{K, a_1, a_2, \ldots, a_n\} \cup \{\text{attribute of all sub-classes}\} \cup \{t_1, t_2, \ldots, t_n\}$ and ~~have~~ make the PK as PK of super-class.

a) Mapping of Union Type

For this, it is mandatory to specify the new key attribute called a serrogate key, When creating a relation because keys of defining classes are different. We include the serrogate key as a FK in each super-class relation.



Scanned by CamScanner