सुगम स्टेसनरी सप्लायर्स एण्ड फोटोकपी सर्भिस
बालकुमारी, ललितपुर ९८४१५९९५९२
NCIT College

CHAPTER 1.

INTRODUCTION

Database :-

↳ is a repository for collection of related data and facts.

↳ a logical collection of related data that
→ describes the entities and their inter relationships.

→ is designed, built and populated for a specific reason.

Data

→ is a raw, unorganized facts that need to be processed.

→ itself is meaningless and worthless.

Information

→ when data is processed, organized, structured or presented in a given context so as to make it useful, is called information.

→ is output of data processing operation.

Scanned with CamScanner

CamScanner

Date / /

Page No.

Database Management System (DBMS):

→ is a collections of programs that enables users to perform certain action on a particular database.

→ "define" the structure of database information.

→ "populate" the database with appropriate info.

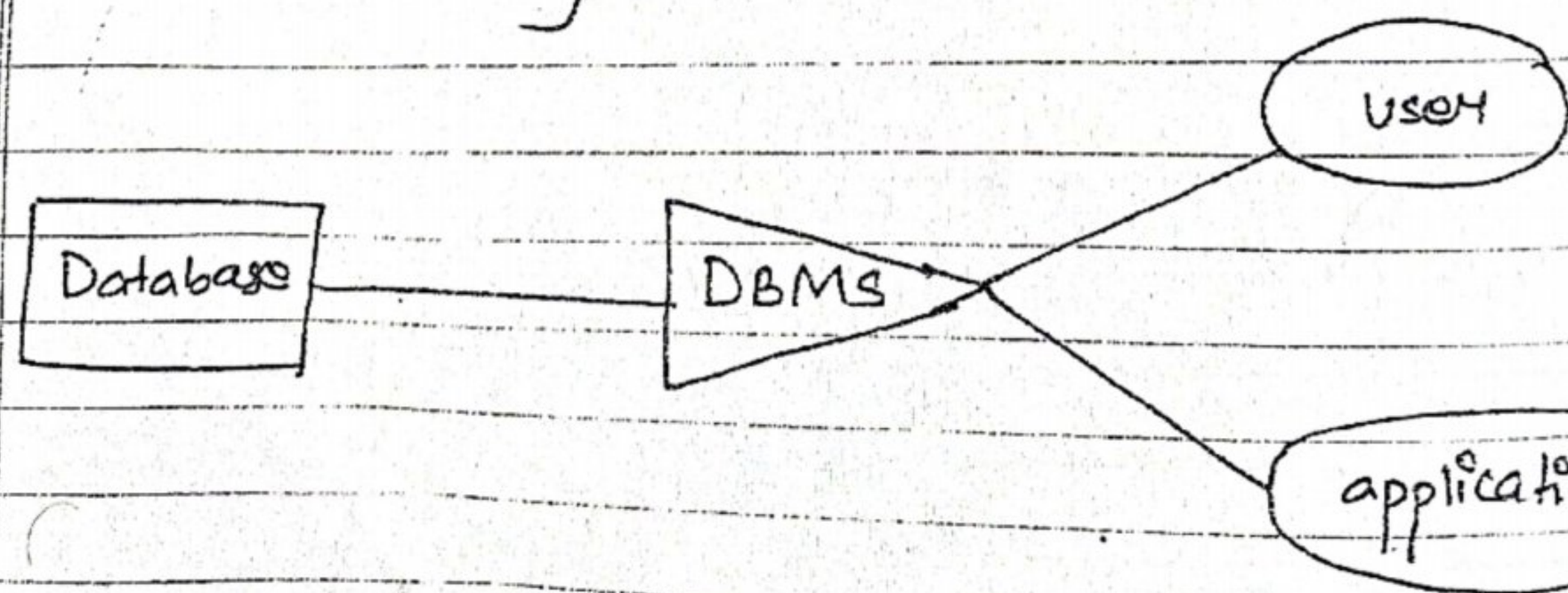
→ "Manipulate" the database.

→ "protect" the database content against accidental or deliberate corruption of contents.

→ "share" the database among multiple users

→ DBMS is a software that defines a database, stores data, supports a query language, produce reports and creates data entry screen.

→ provide an environment that is both convinient and efficient to use in retrieving and storing database information.



Scanned with CamScanner

CamScanner

Objectives of DBMS:

→ provide for mass storage of relevant data.
→ making easy ~~excess~~ access to data for authorized user.
→ providing prompt response to user's request for data.
→ making the latest modification visible to all database users immediately.
→ eliminate redundant data. → (duplicate)
→ allow multiple users to be active at one time.
→ allows growth of database system.
→ provide data integrity (maintain consistancy).
→ protect data from physical harm and unauthorized access.
→ serving different types of users.
→ provide security with user access privilege.
→ combining inter related data to generate report.

Normalization – to eliminate redundant data.

Scanned with CamScanner

CamScanner

Date / /
Page No.

Key Terms in db

1) field :- one piece of info of an entry in
2) Record :- one full set of fields
3) Table :- complete collection of records.

fields

Roll	Name	Add	Phone
1	Ram	Ktm	9841473529
2	Sita	Pkr	9860802469

records

table

Application Areas of DBMS:

1) Universities
2) Airlines
3) Human resources
4) Education
5) Banking
6) e - commerce etc.

Disadvantages:

→ costly
→ fast changing technology
→ Requires trained manpower
→ chance of data leakage & hacking

Scanned with CamScanner

Advantages of DBMS

→ Data storing
→ Reduced data redundancy
→ ~~Comm~~ Improved data integrity
→ Increased security
→ Time saving
→ report generation.

Applications Areas

1) Universities : For student information, course details and grades. Examinations are done online today and universities and colleges maintain these all through DBMS

2) Airlines : For reservations and schedule information.

3) Human Resource Management
Big firms have many workers working under them. Human resource management department keeps records of each employee's salary, tax and work through DBMS.

4) Telecommunications :
For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards and storing information about the communication networks.

Scanned with CamScanner
CamScanner

Date / /

Page No.

5) Banking

We make thousands of transactions through banks daily and we can do this without going to the bank. So now banking has become so easy that sitting at home we can send or get money through banks.

Scanned with CamScanner

CamScanner

Data Abstraction

- ensuring easy, smooth and efficient data structures in such a way that every type of database user is able to access its desired information efficiently.
- hides details of data storage that are not needed by most database users and applications.
- different types of data abstraction
  i) internal level (physical level)
  ii) conceptual level (logical level)
  iii) external level, (view level)

i) **Internal / Physical level:**
- is the lowest level of data abstraction that is closest to physical storage.
- it describes:
  → how the data are actually stored in storage devices?
  → what will be storage technique?

ii) **Conceptual / logical level**
→ is the next higher level
→ it describes
→ what data are actually stored in db?
→ what are relationship between data entities

Scanned with CamScanner
CamScanner

Date / /

Page No.

iii) External level or view level.

- is closet to the users
- it describes:

→ what is the way of viewing information to the concerned user?

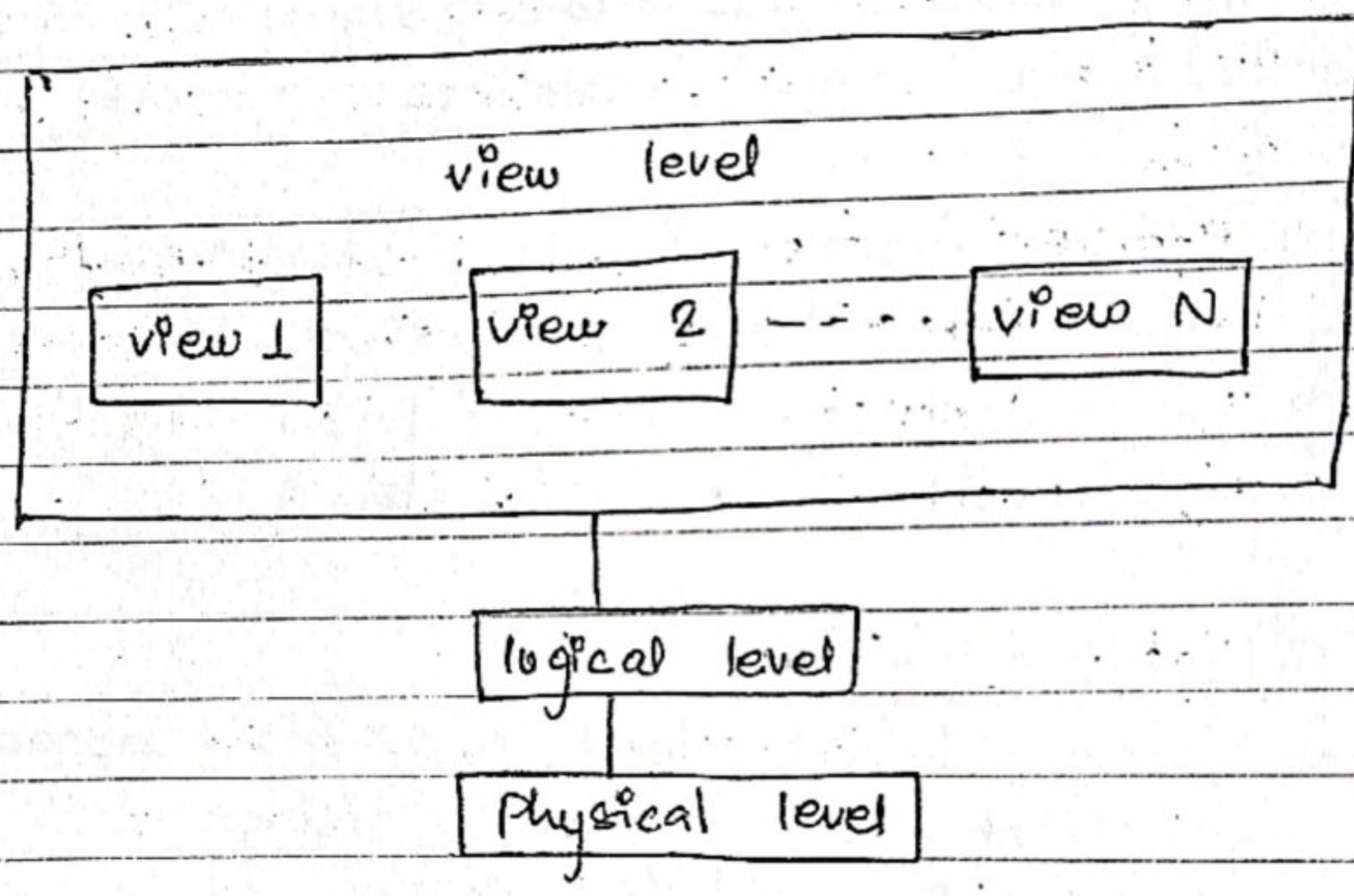


fig: three levels of data abstraction

Schema and Instances:

Schema

- is the overall design of database

Instances

- the collection of information stored in db at particular moment.

Scanned with CamScanner

Physical Schema:
- is database design at ~~partic~~ physical level of abstraction.

logical schema:
- is the database design at logical level of abstraction.

Sub Schemas
- Schemas at view level that describes different view of database.

Data Independence
- the ability to modify a schema definition in one level without affecting schema definition in next high level is called data independence.

- each higher level of data architecture is immune to changes of next lower level of architecture.

- we can change the structure of a db without affecting data required by users and programs.

Scanned with CamScanner
CamScanner

Date / /
Page No.

Physical Data Independence:

→ refers to ability to modify schema followed at physical level without affecting schema followed by conceptual level.

Logical Data Independence

→ refers to ability to modify conceptual schema without causing any cha in schemas at view level.

→ it is more difficult to achieve than physical data independence because application programs are dependent on logical structure of data that user access.

Concepts of DDL, DML, DCL

DDL (Data Definition Language)

→ used to define database structure or schem

e.g.

CREATE ⇒ to create objects in db
ALTER ⇒ to alter structure of db
DROP ⇒ to delete objects from db
TRUNCATE ⇒ to remove all records from table, including allocated spaces.

Scanned with CamScanner

DML (Data Manipulation language):

→ used for managing data within schema object.

e.g.

DRL (DQL) ⇐ SELECT ⇒ to retrive data from db
(Data Retrived language)

INSERT ⇒ to insert data into table
UPDATE ⇒ to update within table
DELETE ⇒ to delete records from table

DCL (Data Control language)

→ used to control db

e.g:

GRANT ⇒ to give user's access privilegs to db.
REVOKE ⇒ to withdraw access privileges.

TCL (Transaction control language)

- used to manage changes made by DML statements e.g.

COMMIT ⇒ save work
ROLLBACK ⇒ restore db to original since commit

Scanned with CamScanner
CamScanner

2073/04/30

Date / /

Page No. 7

CHAPTER 2: Data Models

↳ Data model is a collection of concep tools for describing data, data relations semantics etc.

↳ Data model is simple representation of comp real world data structure.

↳ it is a communication tool to facilitate interaction among designer, application programmer and end user.

↳ it define how data is connected to each other and how it will be processed and stored inside the system.

3 levels of data modeling

1) Conceptual data model:
- identifies highest level relationship betw een the different entities.
- It include important entities and their relat onship.
- No attribute and primary key is specifie

2) Logical Data Model
↳ describes the data in as much detail as possible without saying much about their physical implementation.
↳ it includes all entities and relationship among them.
↳ all attributes for each entity are specified.

Scanned with CamScanner

CamScanner

Date / /
Page No.

↳ primary key and foreign key are also specified.

3) Physical Data Model

→ represents how the model will be built in db.

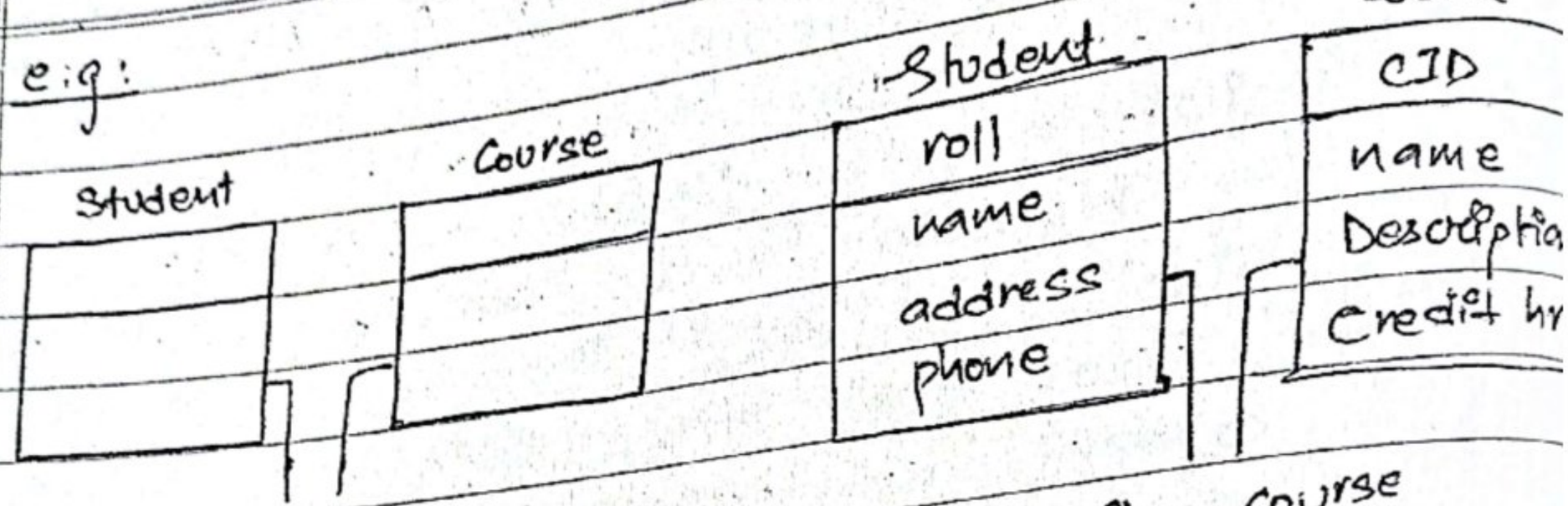
→ shows all table structure including column name, data type, constraints, primary key, foreign key and relationship between tables.

Feature	Conceptual	logical	physical
1) Entity name	✓	✓	
2) Entity Relationship	✓	✓	
3) Attributes	–	✓	
4) primary keys	–	✓	✓
5) Foreign keys	–	✓	✓
6) Table name	–	–	✓
7) Column name	–	–	✓
8) Column datatypes	–	–	✓

Scanned with CamScanner

CamScanner

e.g:



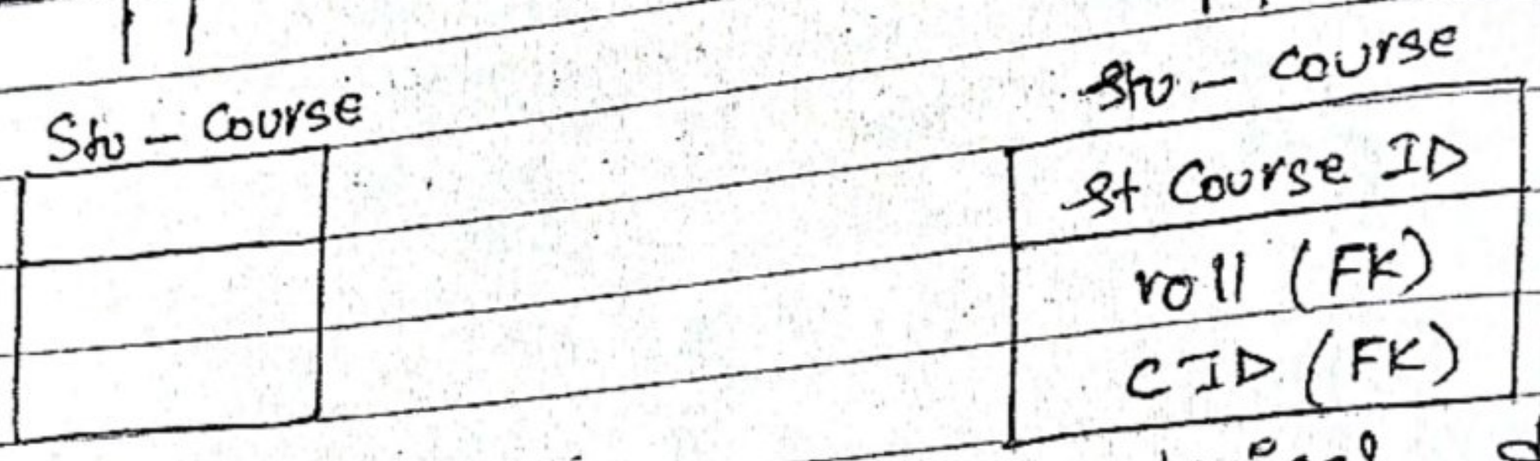


fig: 1) Conceptual Data Model

fig: 2) logical data model

tbl – Student
roll : integer pk
name : varchar (25)
address : varchar (25)
phone : varchar (20)

tbl – Course
CID : integer pk
name : varchar (20)
Description : varchar (
credit hr : int

tbl – Stu Course
St Course ID : integer
roll : integer FK
CID : integer FK

fig: 3) physical data model

Scanned with CamScanner

Date / /

Page No.

E-R (Entity Relationship) Model: (ERM)

→ defines the conceptual view of database.
→ works around real world objects called entities and relationship among them.

Entity

→ a real world thing that is easily identifiable and distinguishable.

Entity Set

→ a collection of similar types of entities.
→ entity sets do not need to be disjoint.
→ e.g. entity set of employee and entity set of customer may have members in common.

Attributes

→ are the properties of the entities.
→ Simple attribute:
- which can't be divided further (e.g. phone no)

→ Composite attribute:
- made of more than one simple attribute (e.g. name (∵ name ⇒ first name, lastname)

Scanned with CamScanner

CamScanner

→ Derived attribute
- do not exist in physical database but values are derived from other attribute db (e.g. age)

→ Single valued attribute :
e.g: citizenship no.

→ Multivalued attribute :
e.g: phone no, email_address.

E-R (Entity-Relationship) ~~attributemodel:~~
: Domain of attribute is set of permitted values.
formally, an attribute is a function which maps an entity set into a domain.
every entity is described by a set of (attribute, datavalue) pairs.
e.g: student entity
{ (Roll, 1) ; (name, Ram) }

Relationship
→ association among entities.
→ Relationship set is a set of relationship of same type.
→ formally, it is a mathematical relation on $n \geq 2$ entity sets.
→ if $E_1, E_2, \ldots E_n$ are entity sets then relationship set R is subset of

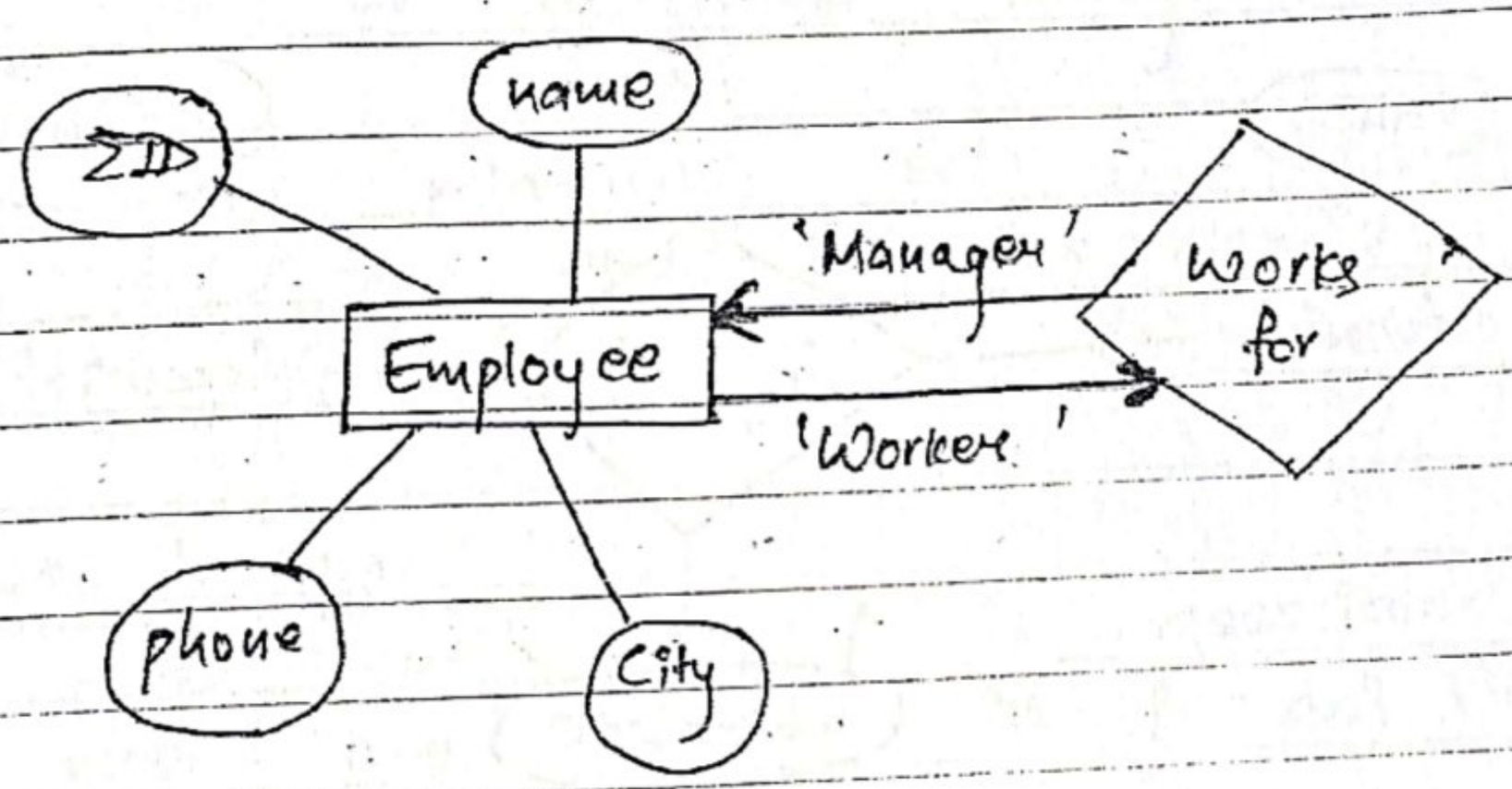
Scanned with CamScanner

Date / /
Page No.

$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots e_n \in E_n\}$

where $(e_1, e_2, \ldots e_n)$ is relationship.

↳ The role of an entity is the function it plays in a relationship.

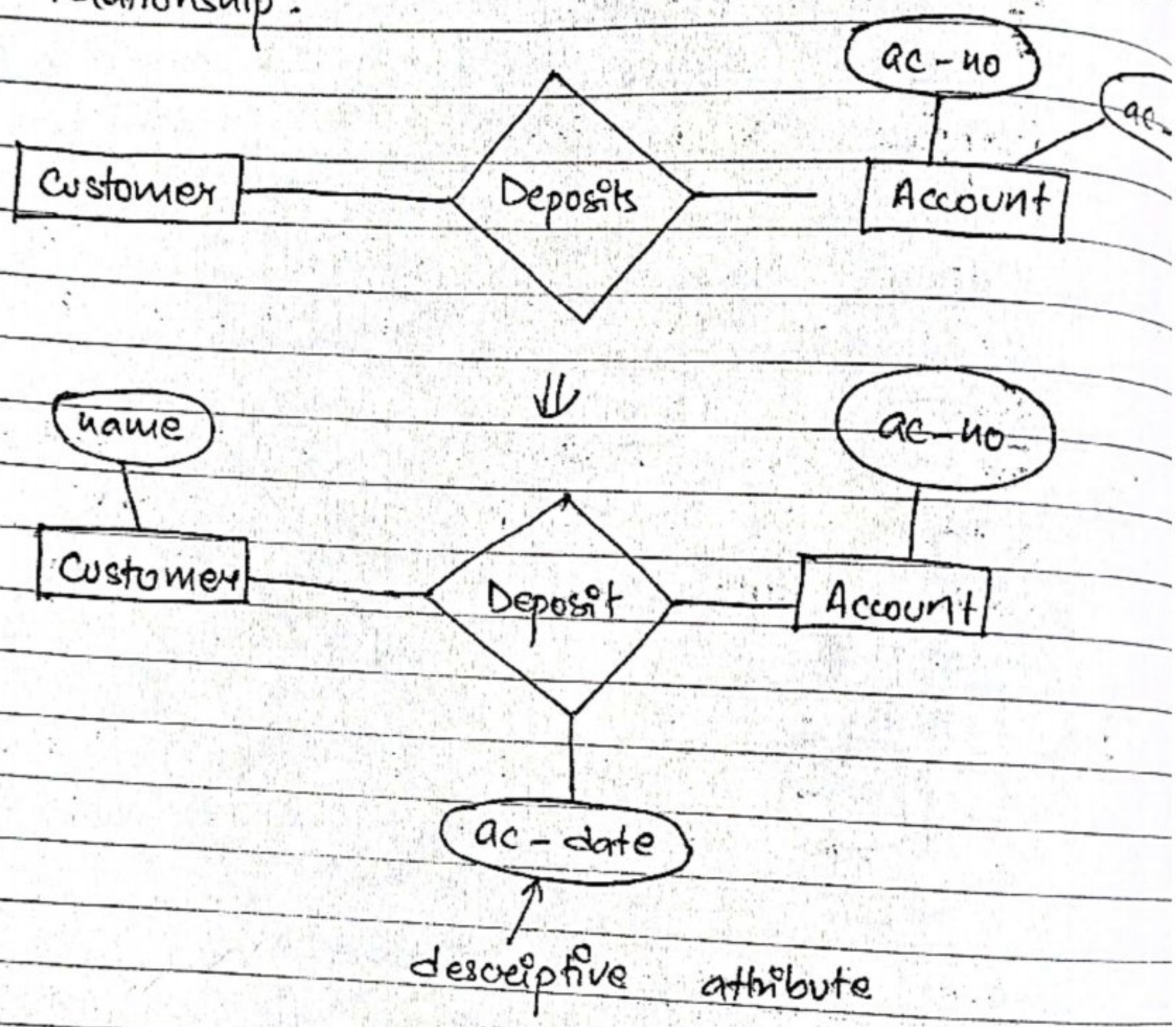


- here, 'manager' & 'worker' are roles of entity employee in 'works for' relationship.

- roles are optional and clarify semantic of a relationship.

Scanned with CamScanner

Descriptive attributes are the attributes of relationship.



Degree of relationship

- It refers no. of entity sets that participa in a relationship.
- Unary relationship is of degree 1 that involves only one entity set.
- Binary relationship is of degree 2.
- Ternary relationship is of degree 3.
- Relationship in db are of ten binary.
- It is possible to replace a non binary.

Scanned with CamScanner
CamScanner

Date / /
Page No.

(n_ary, n > 2) relationship set by number of distinct binary relationship set.

Mapping Cardinalities

↳ express the no. of entities to which another entity can be associated through a relationship set.

↳ For binary relationship set R between entity set A and B, the mapping cardinality must be one of following

1) one to one:
- an entity in A is associated with at most one entity in B and vice versa.

2) one to many:
→ an entity in A can be associated with more than one entities of entity set B but from entity set B one entity is associated with at most one entity in A.

3) Many to one
↳ More than one entities from entity set A can be associated with at most one entity of entity set B but one entity from entity set B can be associated with more than one entity from entity set A.

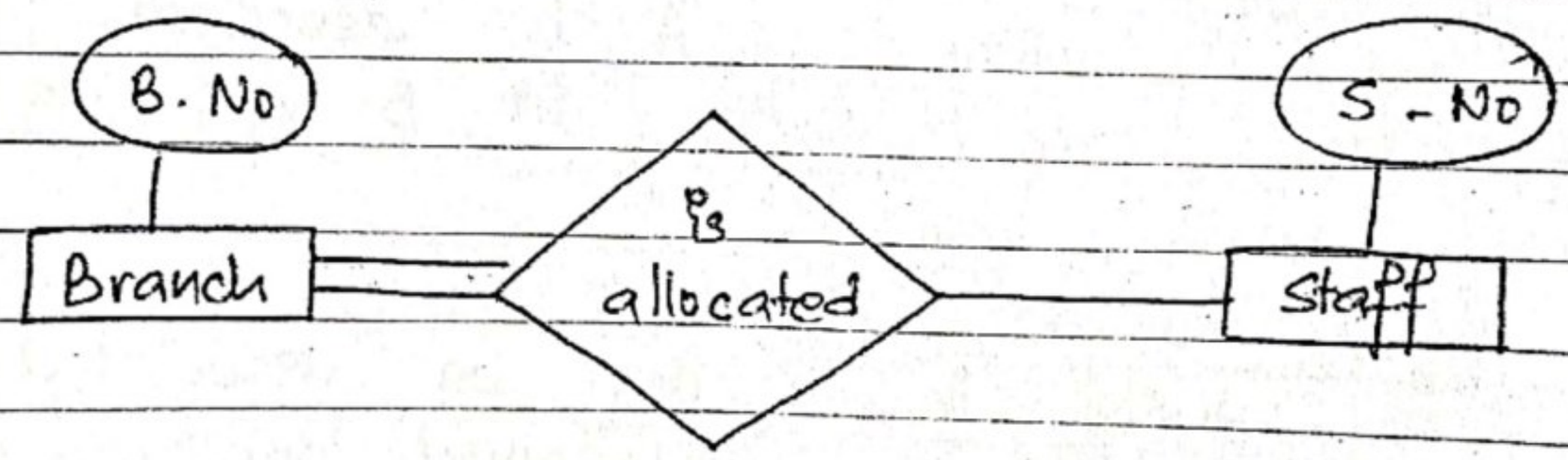
Scanned with CamScanner

4) Many to many
— one entity from A can be associated w
more than one entity from B and vice ve

Participation Constraints

↳ total participation
— if every entity in entity set E partici
tes in at least one relationship in R

↳ Partial participation
— if only some entities in E participate i
relationship in R.



Here,

↳ every branch office is allocated member
of staff i.e. total participation of entity
'branch'.

↳ A member of staff need not work a
branch office i.e. partial participation of
'staff'.

Scanned with CamScanner

Date / /
Page No.

Keys

↳ one or more columns in db table that is used to sort and for identify rows in a table is called key.

↳ a key is a single or combination of fields in a table.

↳ is used to fetch or retrieve records/ data from table according to condition.

↳ different keys are:

1) Superkey

↳ is a set of one or more attributes whose value uniquely determine each entity.

2) Candidate key

↳ is a minimal superkey.

3) Primary key

↳ is a candidate key chosen as principal means of identifying entities within an entity.

4) Foreign key

↳ is a field in one table that uniquely identifies a row of another table.

↳ it is used to establish and enforce a link between two tables.

Scanned with CamScanner

CamScanner

Strong and Weak entity set.

Strong entity	Weak entity
1) It has its own prim-any key.	1) it doesn't have sufficient attibuks to form a primary key of it's own.
2) It is represented by a rectangle.	2) It is represented by double rectangle.
3) Contains a primary key represented by underline.	3) Contains a partial k represented by dashed underline.
4) Its member is called dominant entity set.	4) Its member is calle subordinate entity set
5) Primary key is one of attibute that uniquely identifies its member.	5) Primary key is com bination of partial ke & primary key of stron entity set.
6) relationship between two strong entity set is represented by diamond symbol.	6) relationship between o strong and one weak entity set represented by double diamond symbol.

Scanned with CamScanner
CamScanner

21st August 2016

7) single line connects strong entity set with relationship.

7) Double line connects weak entity set with relationship.

ER Diagram

↳ illustrates the logical structure of database.

↳ is a data modeling technique that creates a graphical representation of entities and the relationship between entities within an information system.

↳ major components are:

1) Rectangles - ▭ ⇒ to represent entity set

2) Ellipses - ⬭ ⇒ to represent attributes

3) Diamond - ◇ ⇒ to represent relationship.

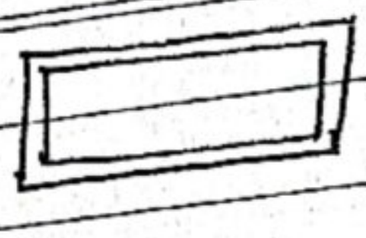
4) Lines - ——— ⇒ to link attribute to entity and entity sets to relationship.

5) Double ellipses - ⬭ ⇒ to represent multivalued attributes.

6) Dashed ellipses - ⬭ ⇒ to represent derived attribute

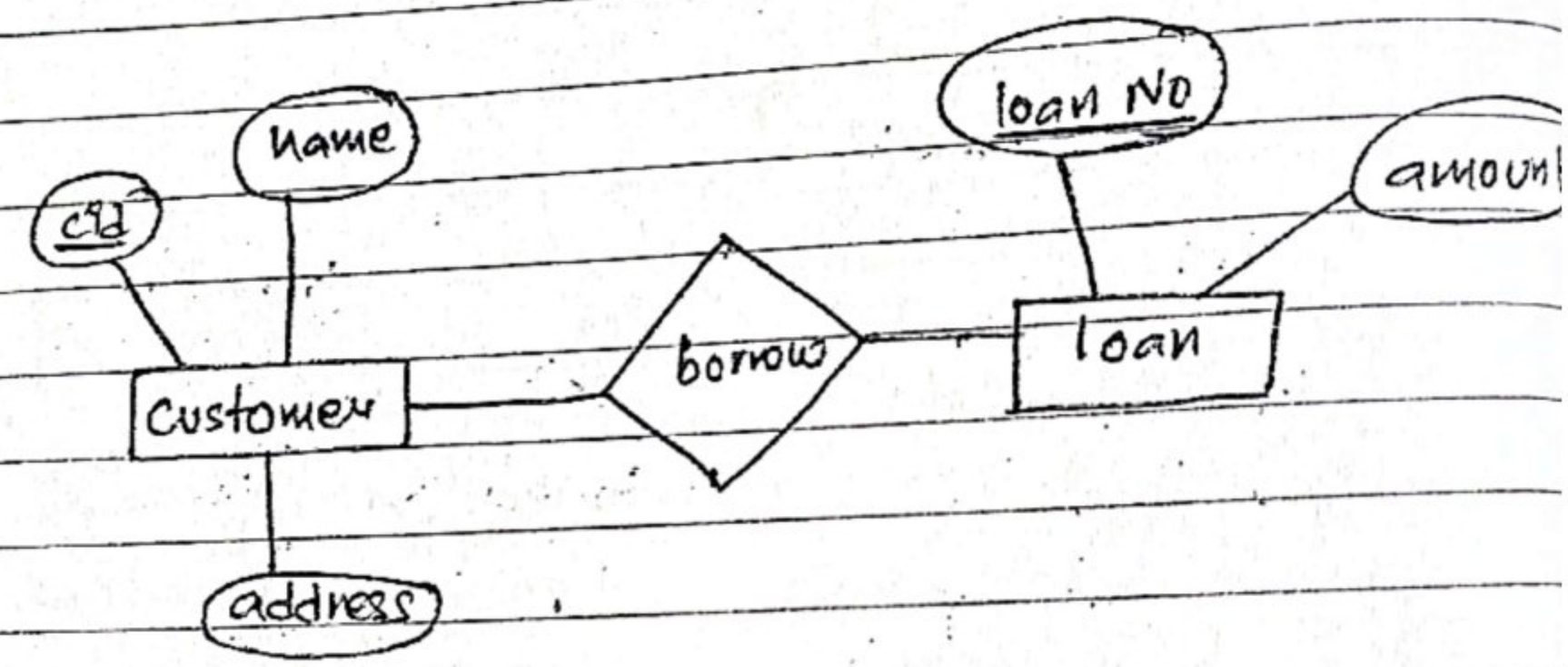
Scanned with CamScanner

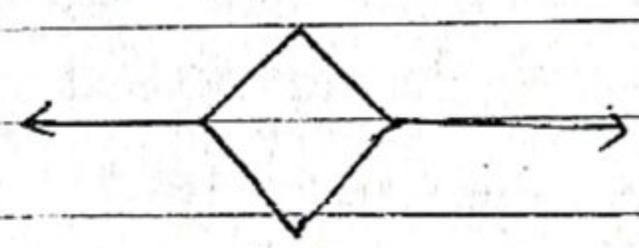
Relationship type –

7) Double rectangle –  ⇒ to represent we entity set.

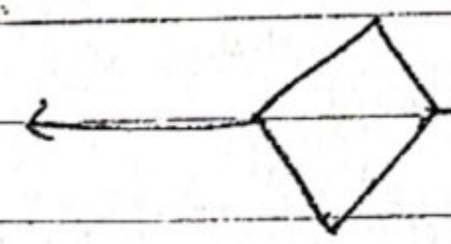
e.g:

i)

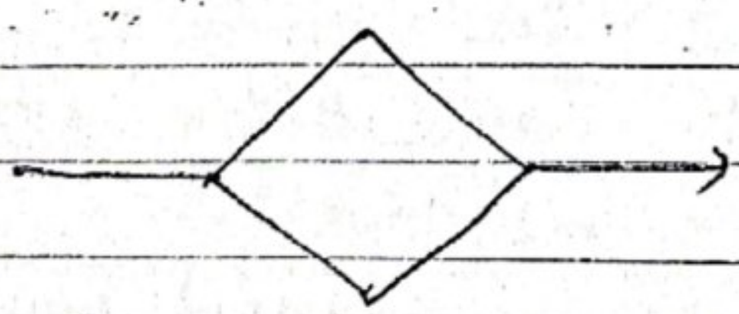




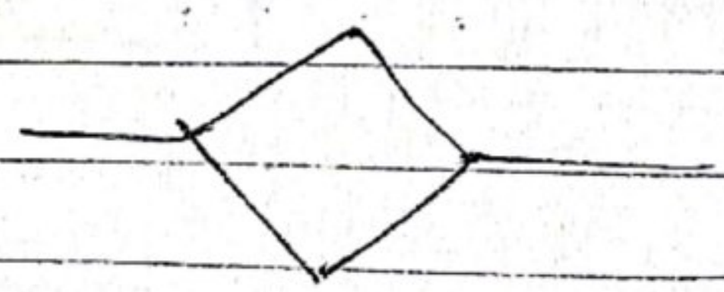
* one to one relationship



* one to many



* many to one

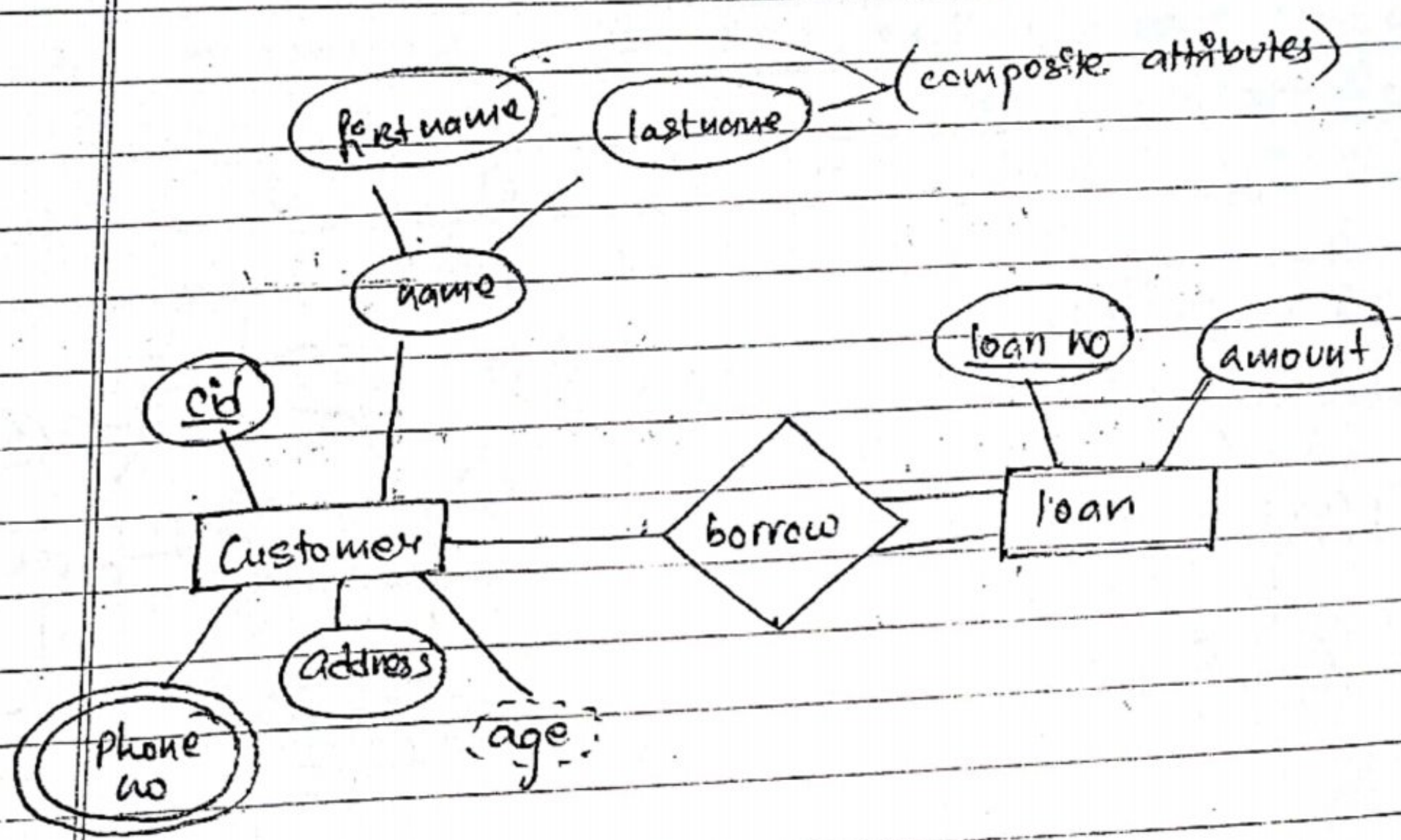


* many to many

Scanned with CamScanner

Date / /

Page No.

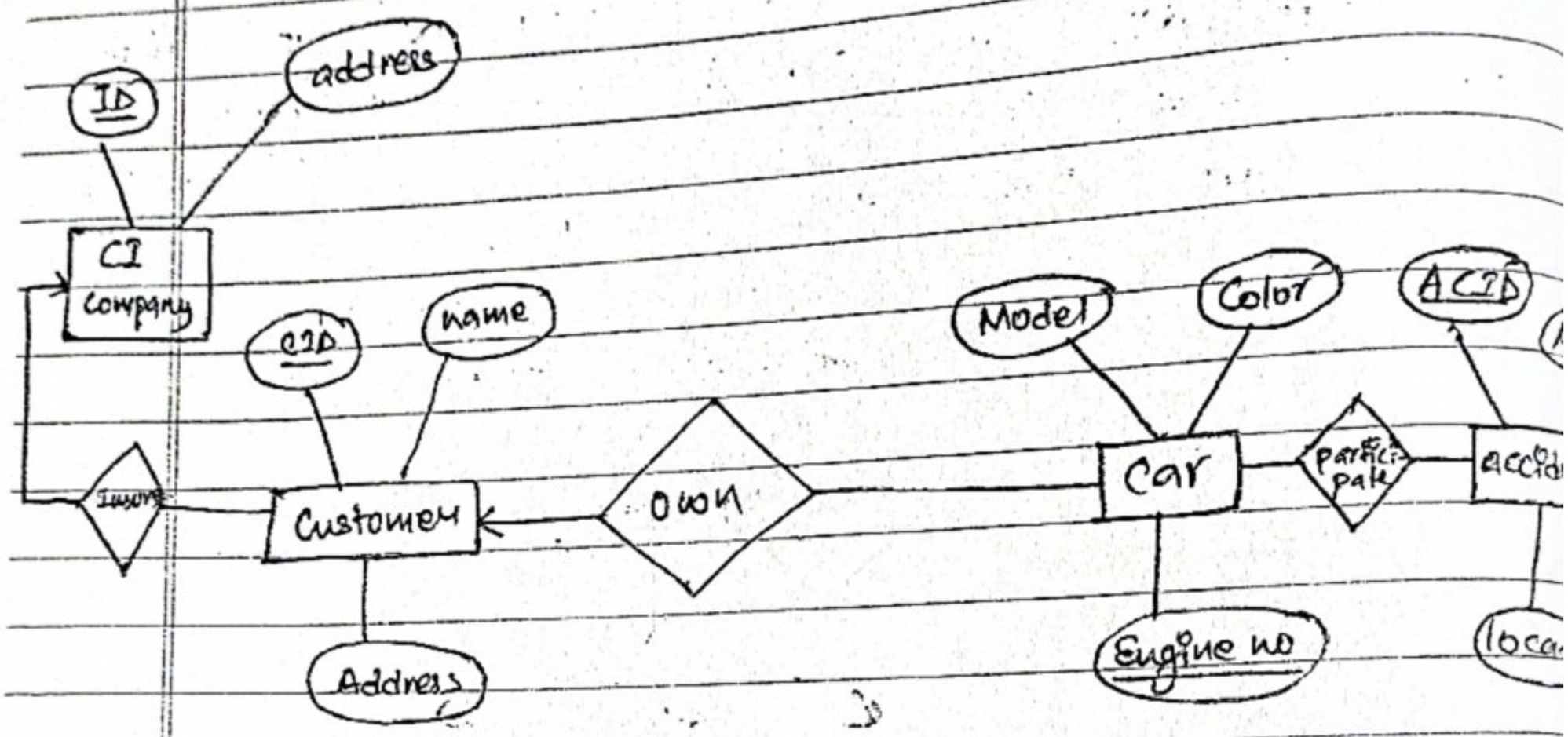


Example

Construct an E-R Diagram for a car - Insurance Company whose customers own one or more cars. Each car has associated with it zero to any no. of recorded accidents. Assume attributes yourself.

Scanned with CamScanner

CamScanner



1) Library Management System
2) Hospital System.

EER Model (Extended or Enhanced ER)

- is the ER model combined with the additi features of semantic concepts.
- shows complex relationships between objects in database.
- Some additional concepts in EER model are

a) Specialization
b) Generalization
c) categorization

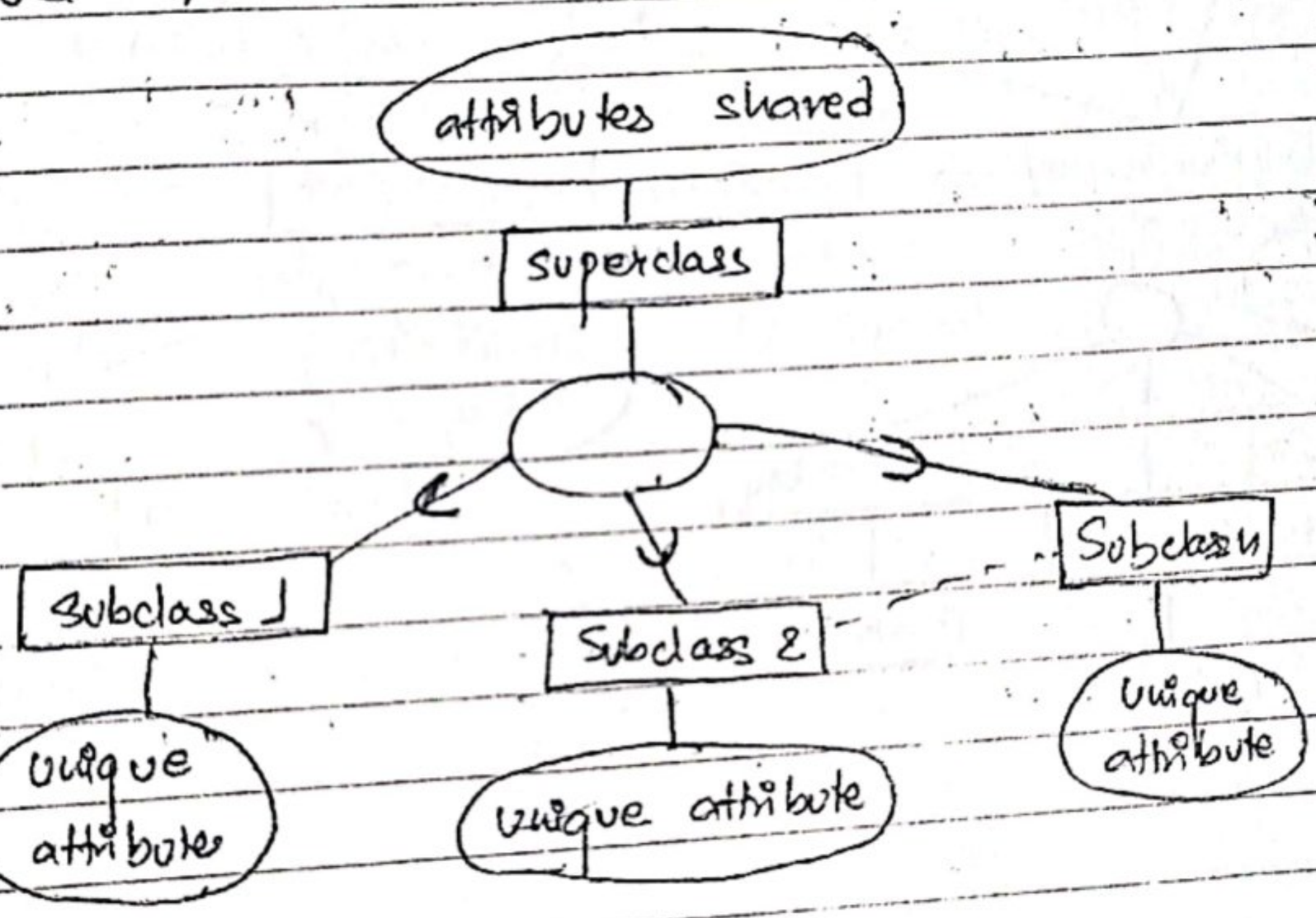
Scanned with CamScanner
CamScanner

Date / /

Page No.

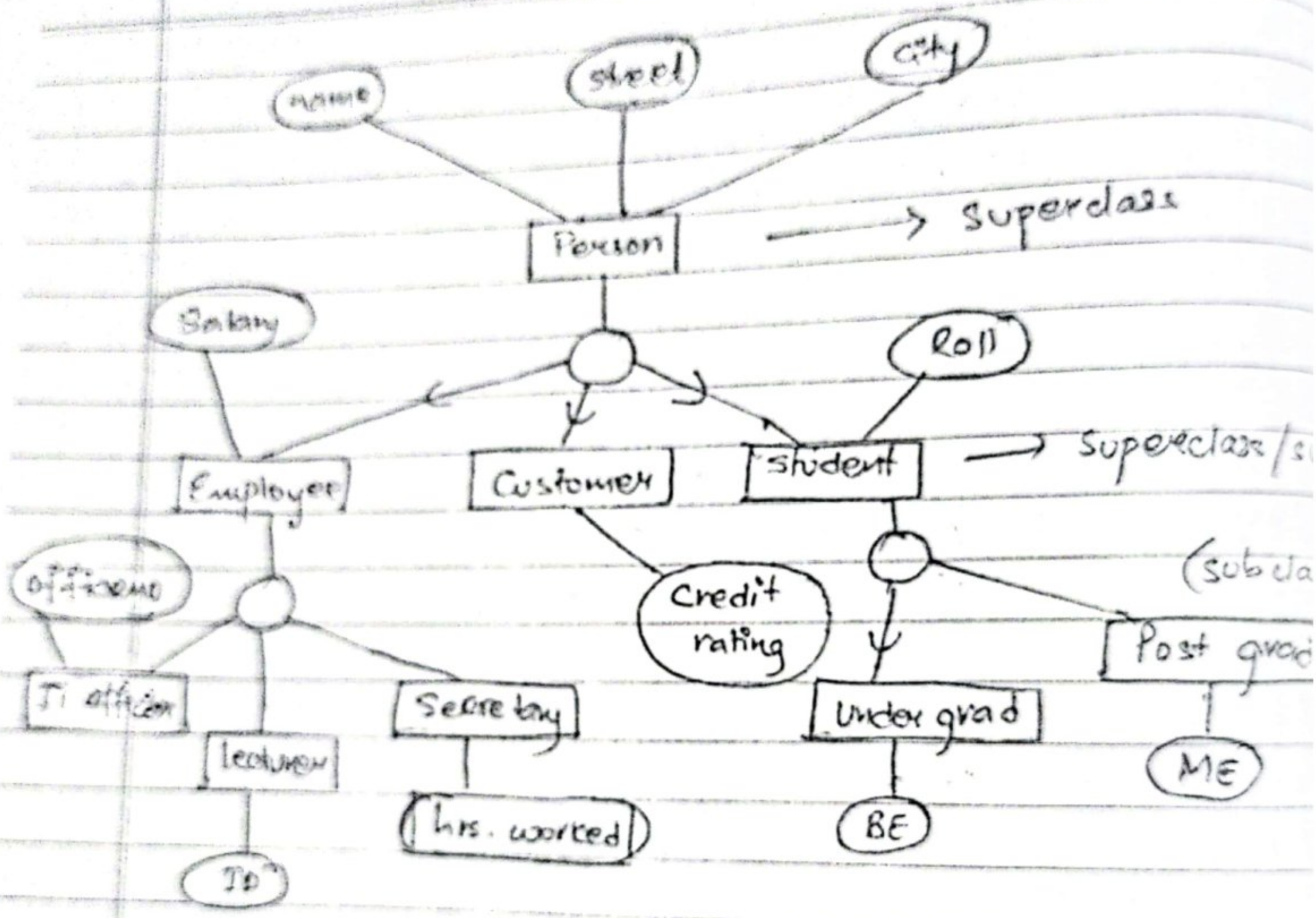
Superclass + Subclass

- Superclass is the entity type whose attributes are shared among subclasses.
- Subclass is the one whose attribute is unique from other subclasses.



e.g: Basic notation for superclass, subclass relationship.

Scanned with CamScanner



Scanned with CamScanner

22nd August 2016

Date / /

Page No.

Specialization

↳ is a means of identifying sub groups within an entity set which have unique attributes.

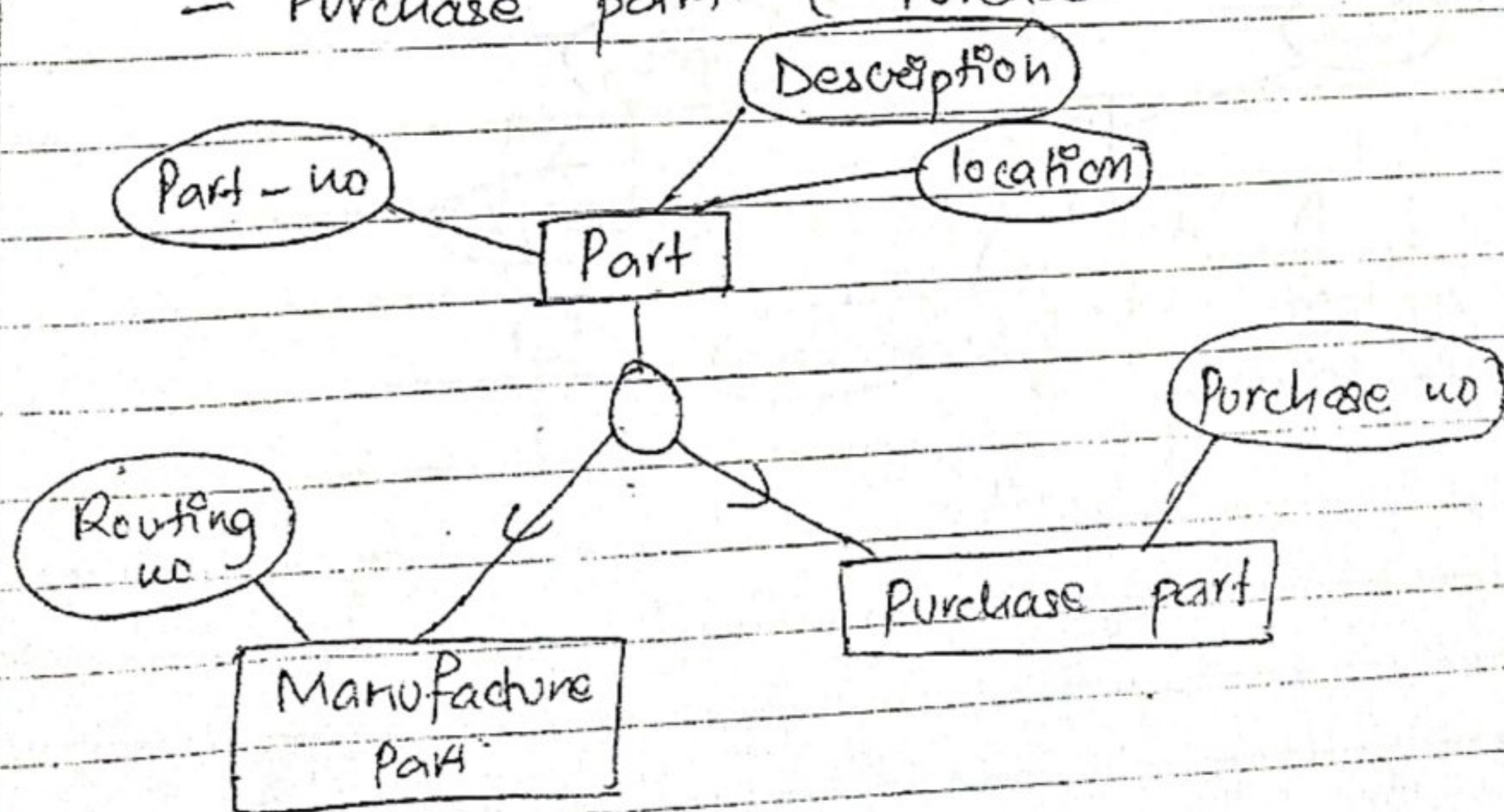
↳ is a top down approach.

↳ is a process of defining one or more subclasses from the superclass and forming superclass subclass relationship.

↳ in EER diagram, it is represented by a triangle component labeled 'ISA' or simply circle with U symbol.

e.g:

- specialization of part as manufactured part and purchase part.
- Part (Part-no, Description, location)
- Manufactured part (Routing no)
- Purchase part (Purchase no)



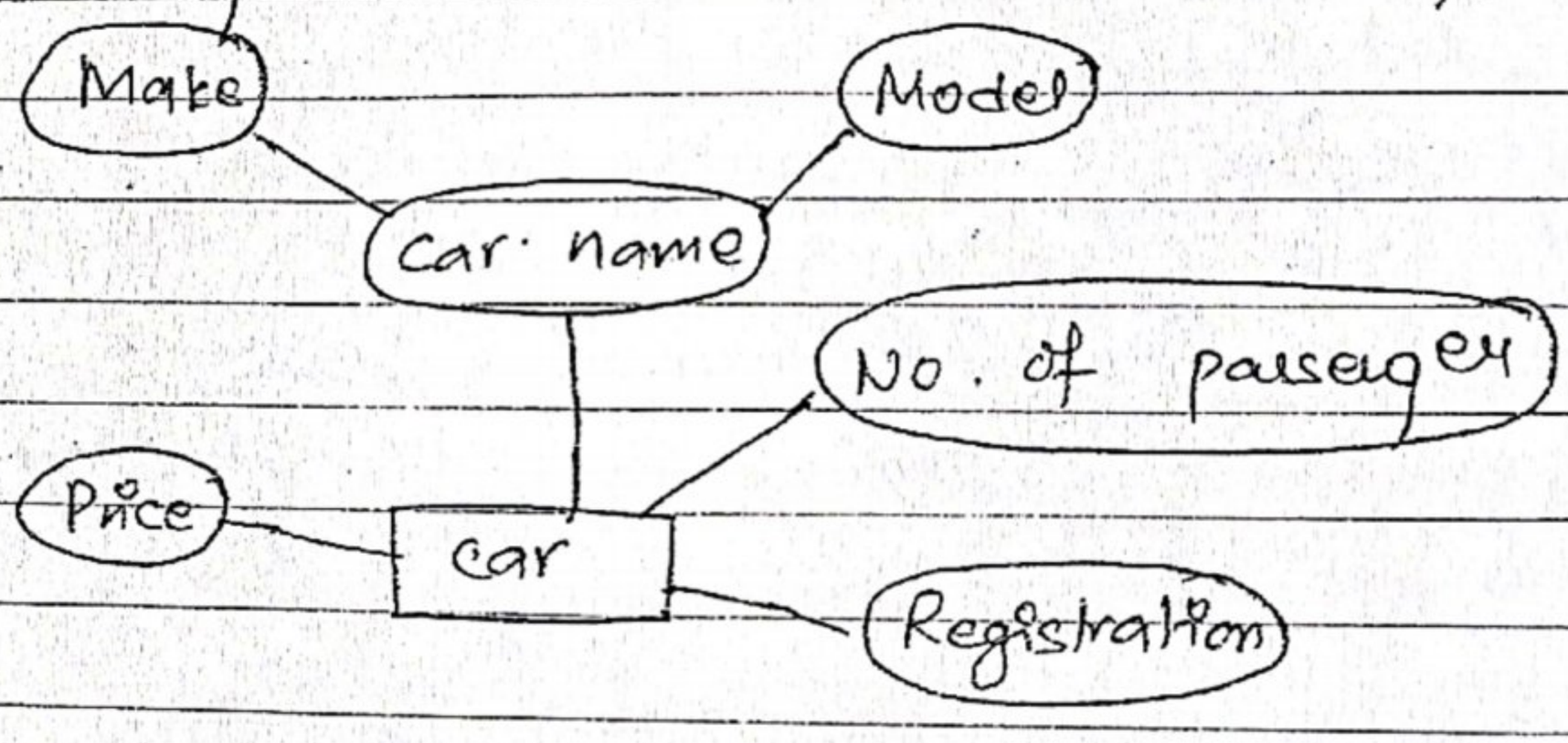
Scanned with CamScanner

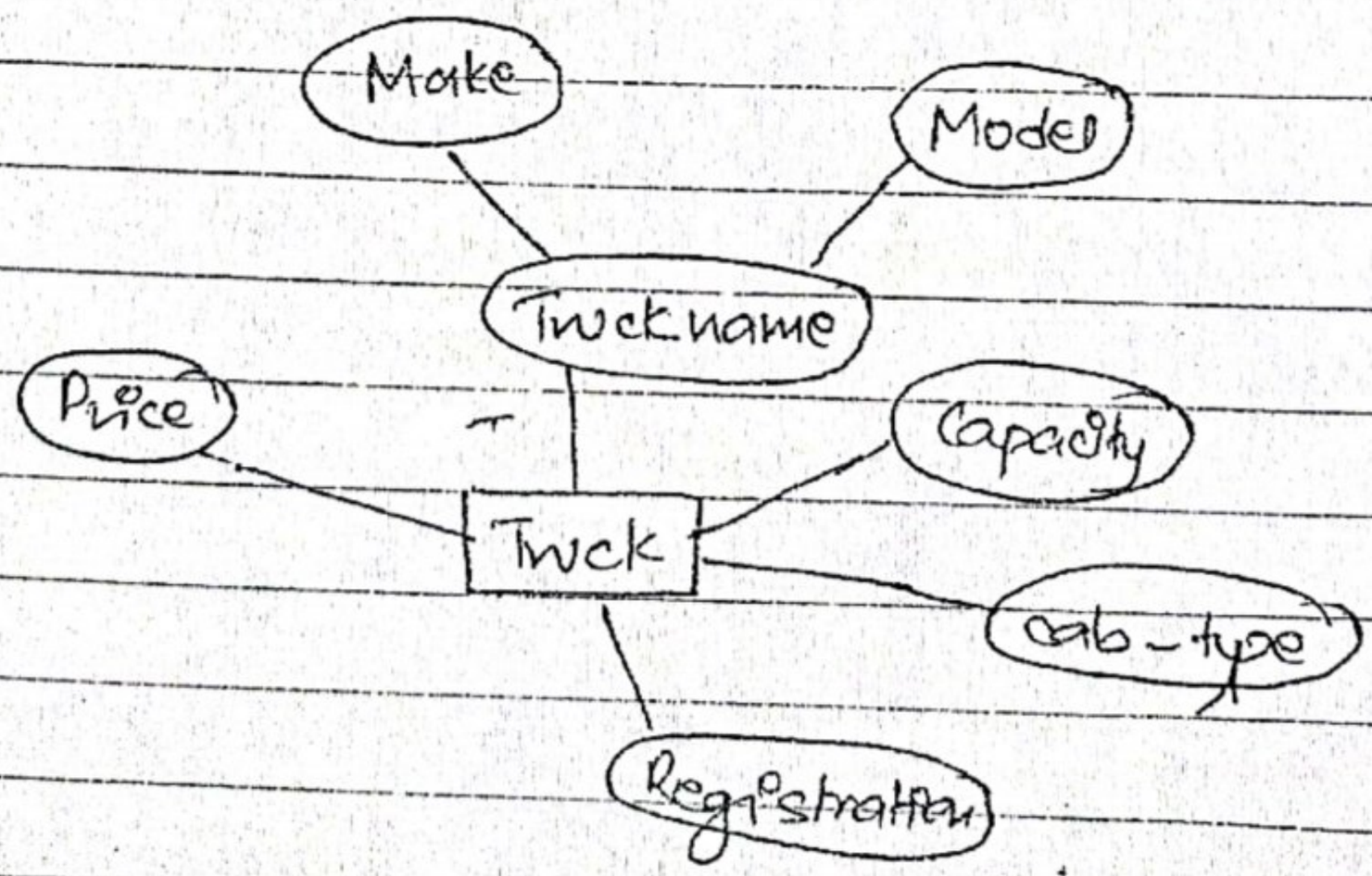
CamScanner

Generalization

- is a bottom up approach.
- is the process of forming superclass.
- it emphasize the similarities among low level entity set and hide the difference

E.g

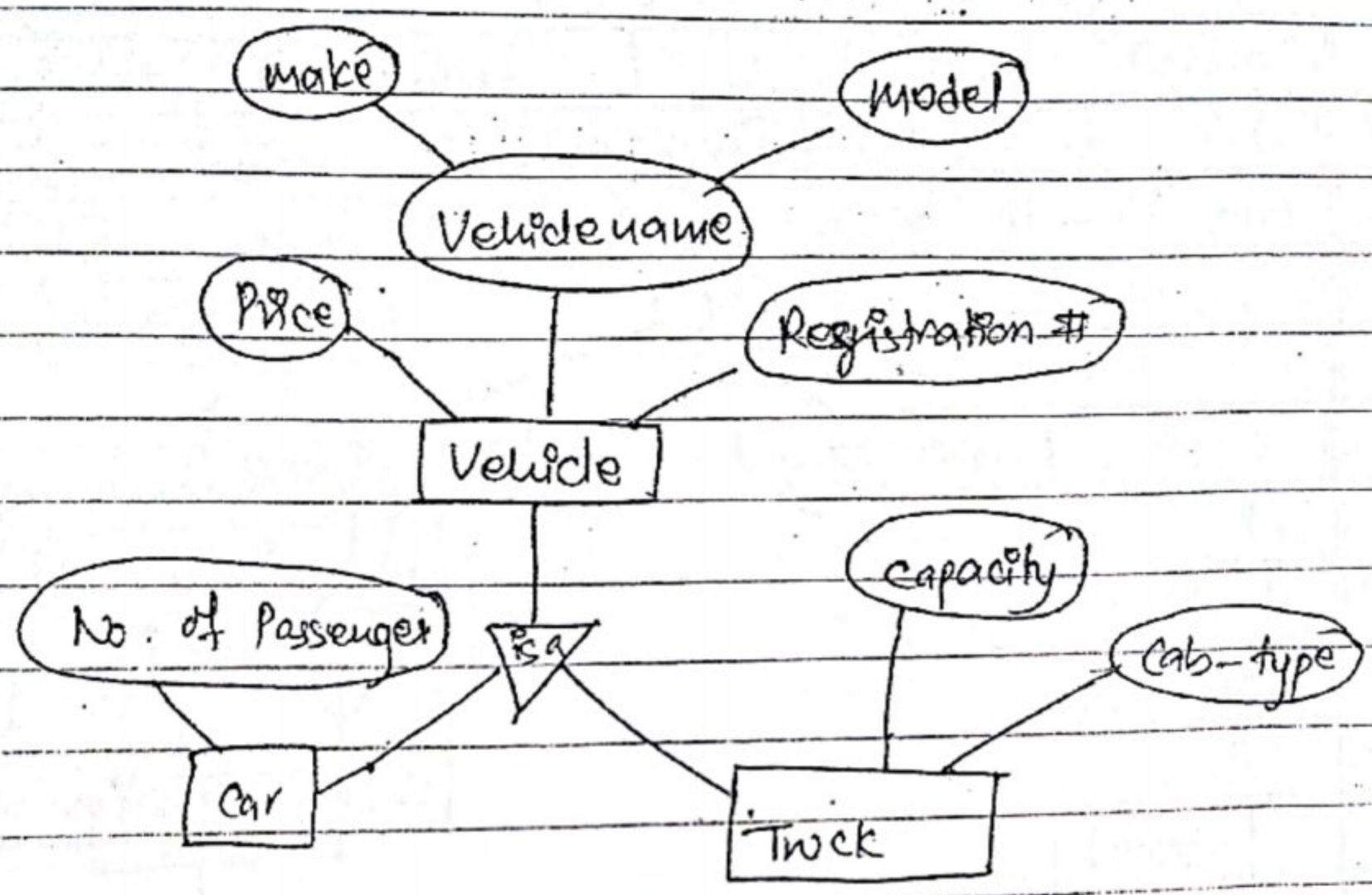




Scanned with CamScanner
CamScanner

Date / /

Page No.



Attribute Inheritance

↳ is a crucial property of superclass and subclass.

↳ attribute of superclass are said to be inherited by subclasses.

⇒ B 'ISA' A

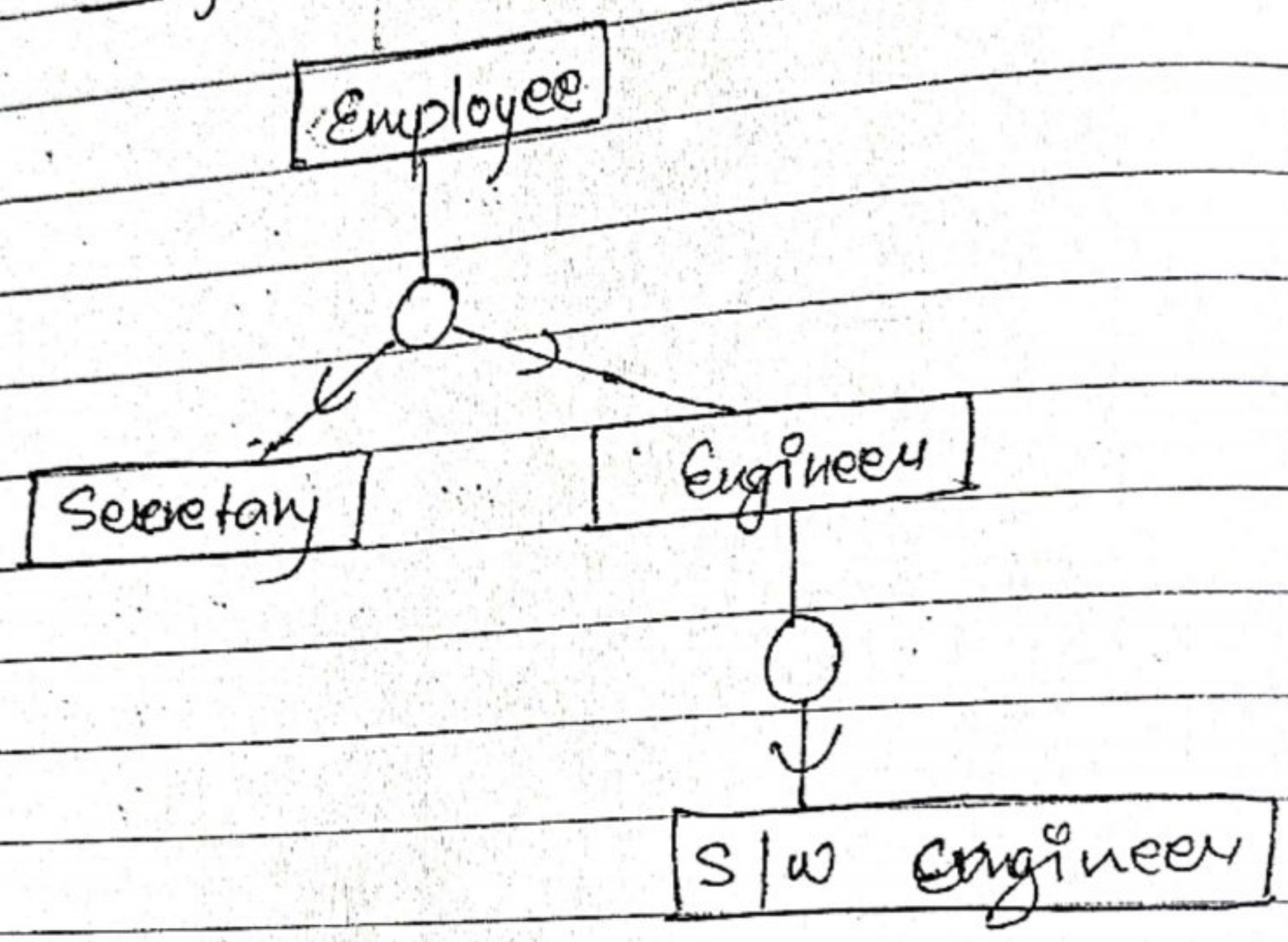
⇒ B inherits features of A

⇒ A is generalization of B.

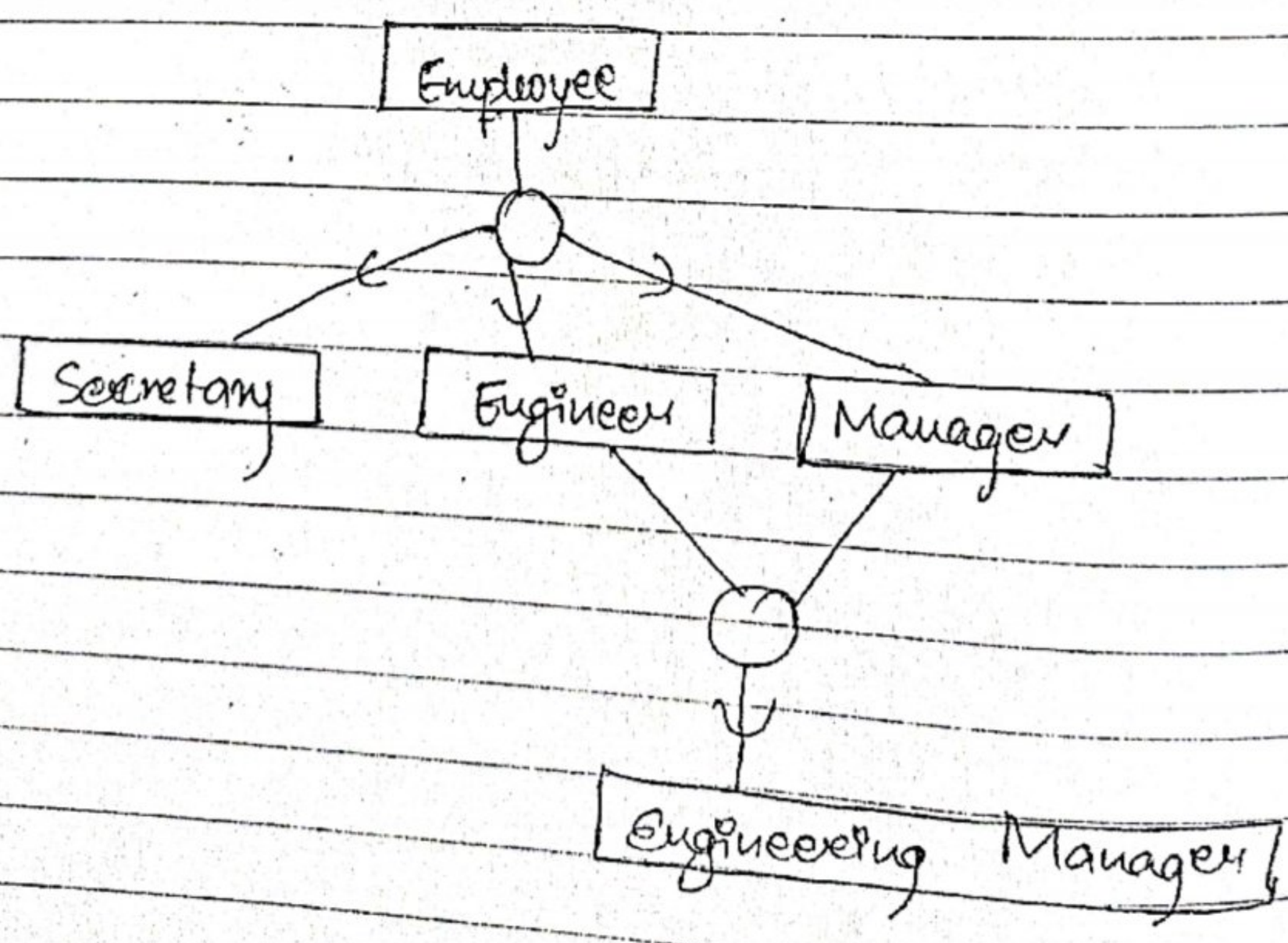
Scanned with CamScanner

CamScanner

Single inheritance (Hierarchy)



Multiple Inheritance (Lattice)



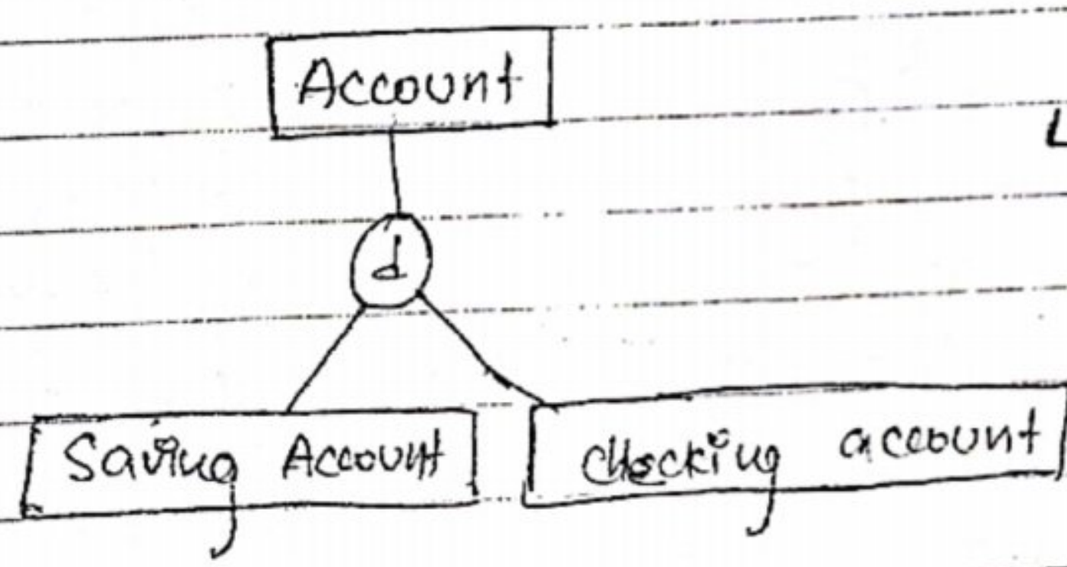
Scanned with CamScanner
CamScanner

Date / /

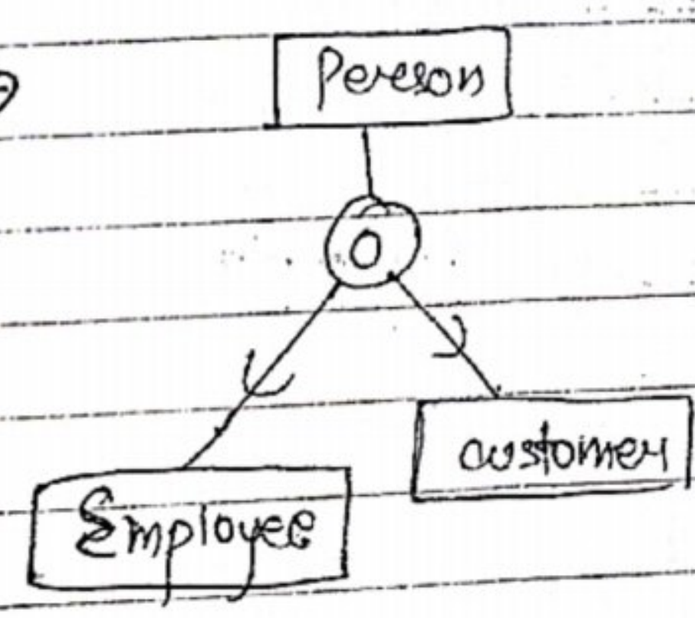
Page No.

Disjoint Constraint	Overlapping Constraint
↳ Subclass belong to no more than one lower level entity set.	↳ Subclass may belong to more than one lower level entity set.
↳ is represented by placing 'd' in circle or simply ISA relationship.	↳ is represented by placing 'O' in circle or simply by ISA relationship.

↳



↳



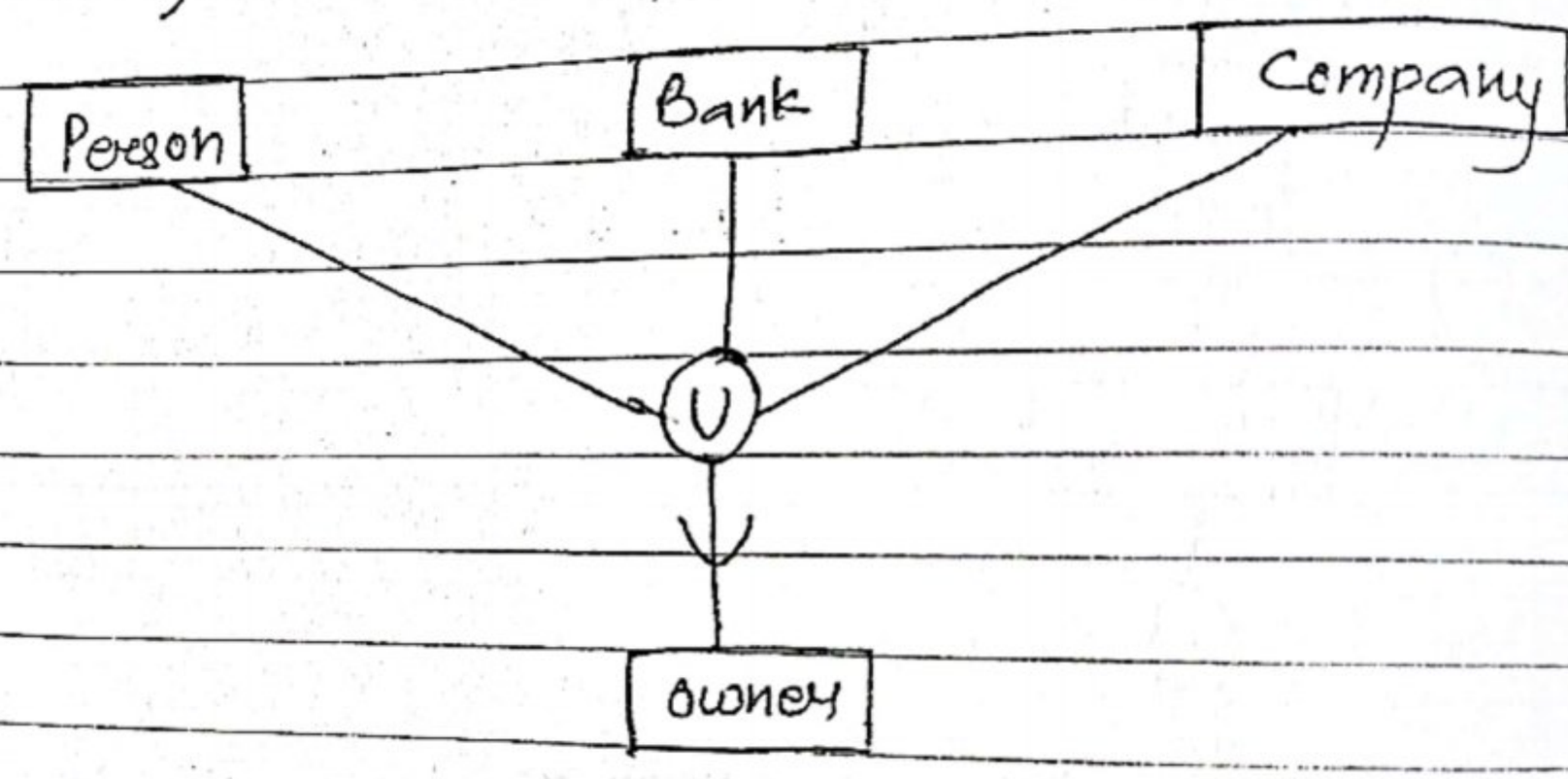
Total generalization / specialization	Partial generalization / specialization.
↳ every entity in superclass must belong to subclass.	↳ every entity in superclass may not belong to subclass,
↳ double lines are used to connect the superclass and generalization or specialization.	↳ single line is used.

Scanned with CamScanner

CamScanner

Categorization (Union type)

↳ model a class | subclass with more than
superclass of distinct entity sets.
↳ Subclass is called category.
↳ only one subclass exists in categorizat



- here, category 'owner' is subclass of u
of person, Bank, Company.
- category member must exist in at
least one of its superclass.

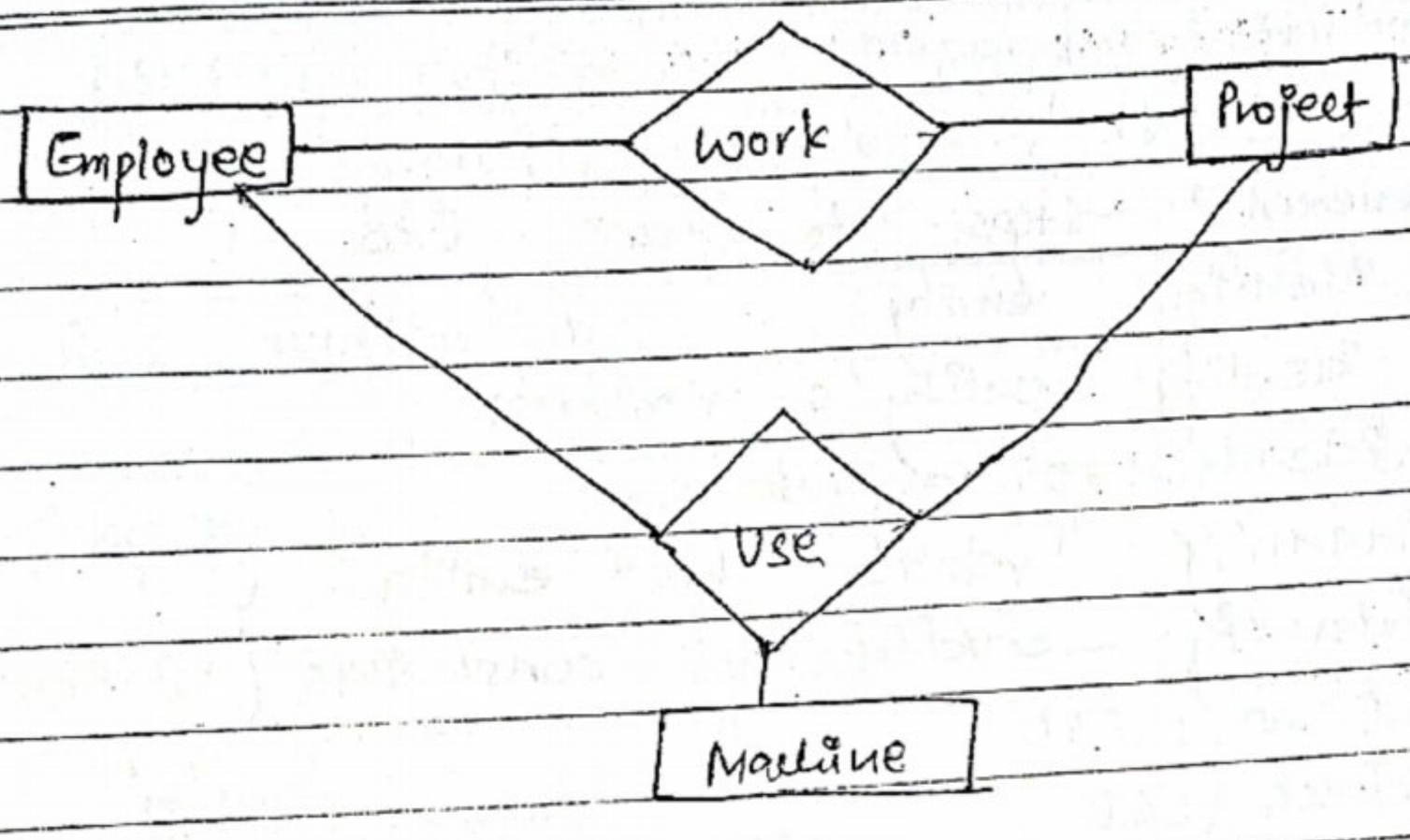
Aggregation

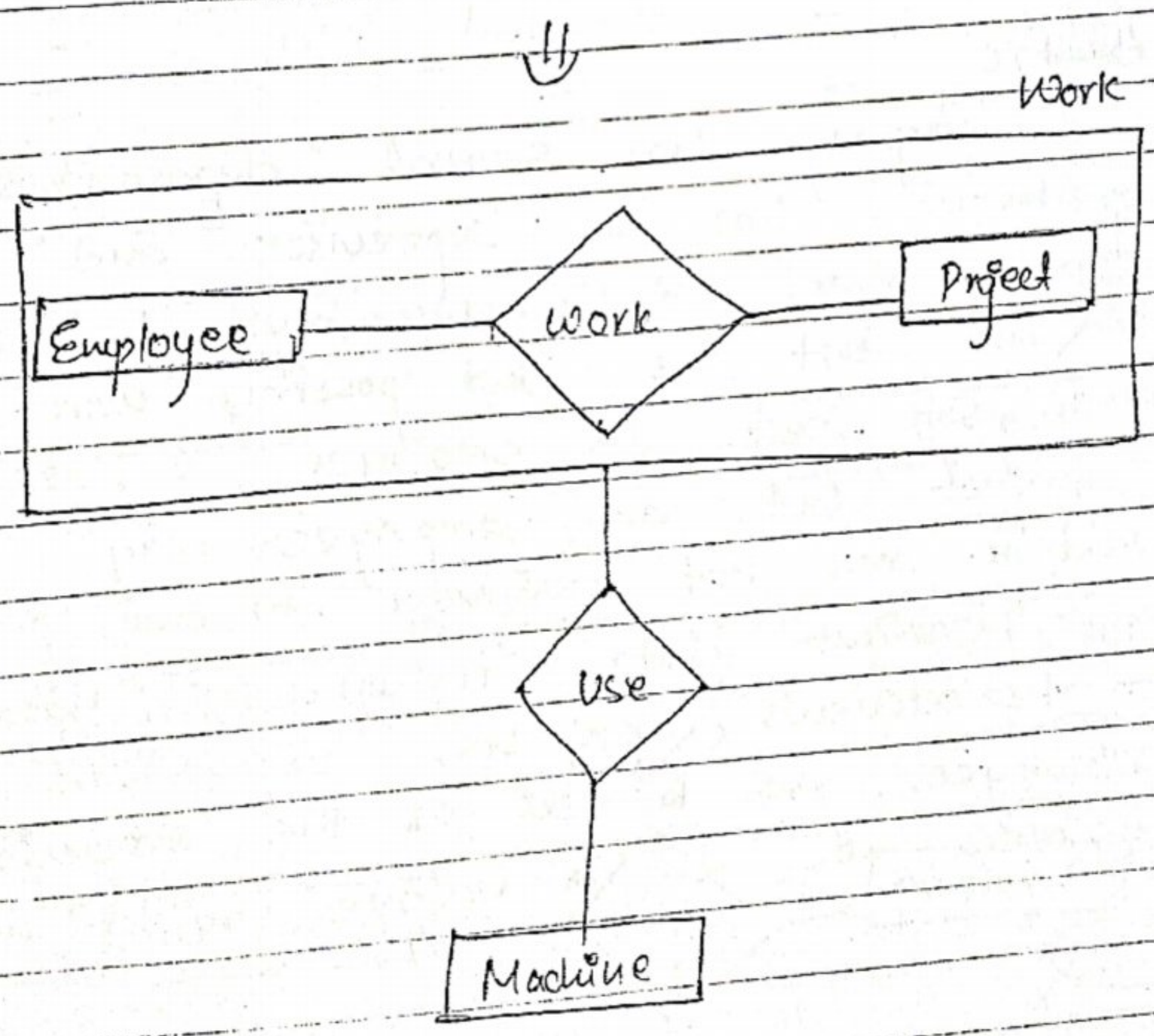
- is an abstraction through which relat
ships are treated as high level entities

Scanned with CamScanner
CamScanner

Date / /

Page No.





Scanned with CamScanner

CamScanner

ER diagram

General steps to create ERD

- ↳ identify entity
- ↳ identify entity's attributes
- ↳ identify primary keys.
- ↳ identify relation betn entities.
- ↳ identify cardinality constraints (relationship
- ↳ Draw ERD
- ↳ check ERD.

Example

A company has several departments. Eac department has a supervisor and at les one employee. Employees must be assigne to at least one but possibly more depa ments. At least one employee is assigned a project but an employee may be o vacation and not assigned to any projects. The important (data attributes fields) are names o of departments, projects, supervisors and employees. As well as the supervisor an employee no. & a unique project number.

Scanned with CamScanner

Date / /

Page No.

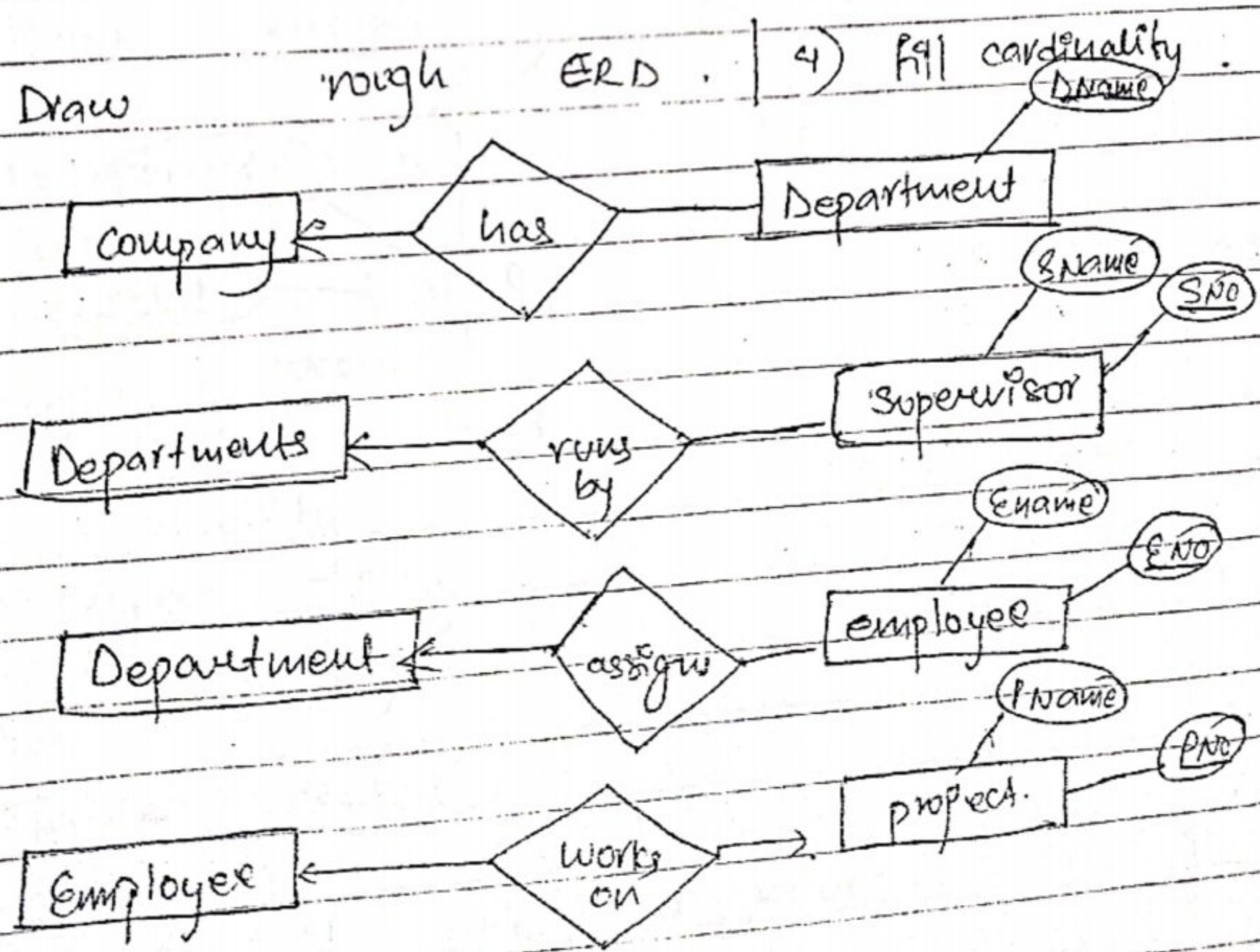
1) identify entities :- Company, Department, Supervisor, Employee, Project.

2) find relationships:

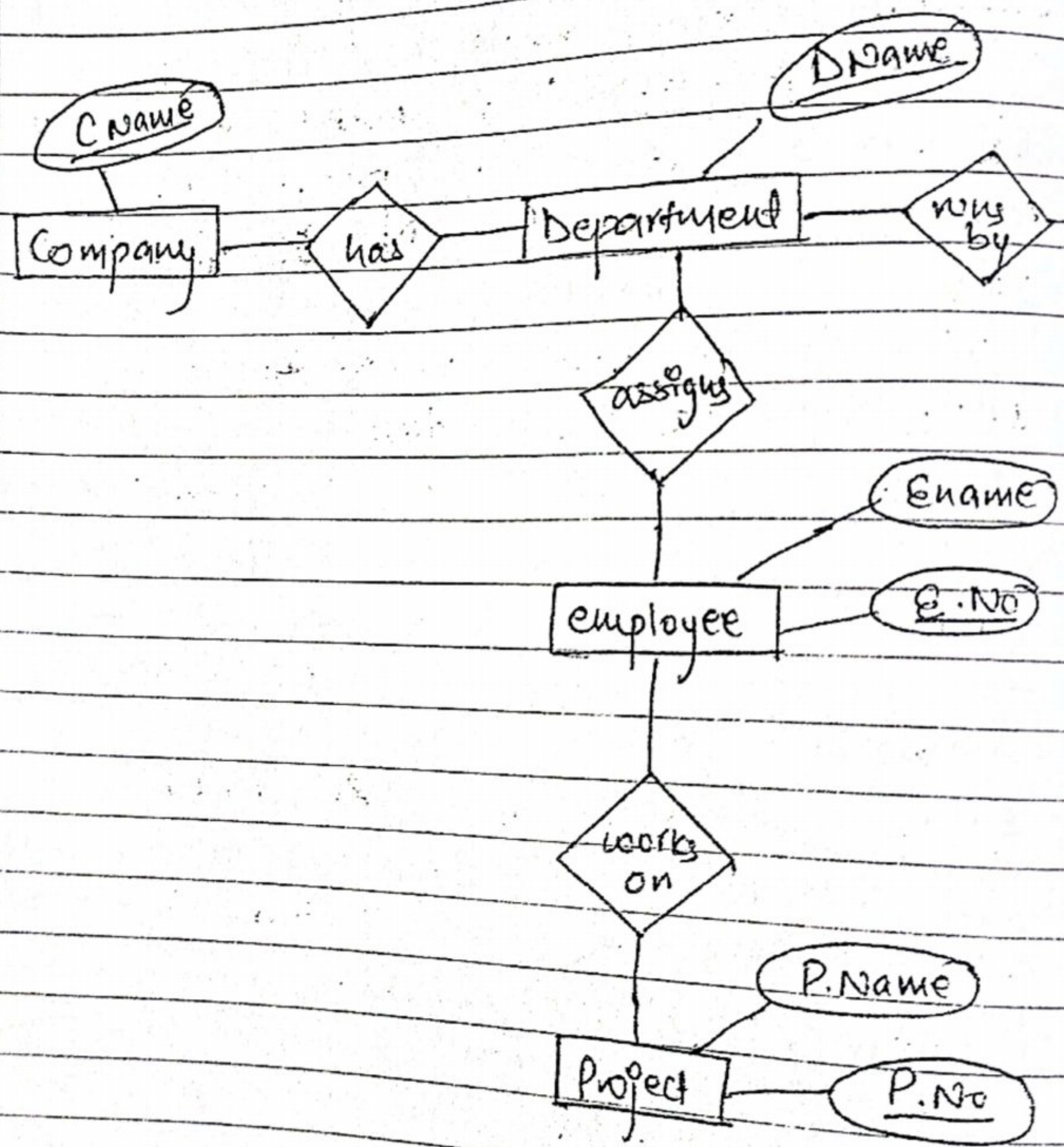
	Company	Department	Supervisor	Employee	Project
Company	—	has	—	—	—
Department	belongs.	—	run by	is assigned	—
Supervisor	—	runs	—	—	—
Employee	—	belongs to	—	—	works on
Project	—	—	—	uses	—

3) Draw rough ERD.

4) fill cardinality.



Scanned with CamScanner



Scanned with CamScanner

CHAPTER 3

Date 2073/5/3 / 4th August

Page No.

Relational * Model

Relation data model

- is the primary data model widely used for data storage and processing.
- was proposed by Dr. EF Codd, IBM in 1970.
- data are organized in ~~the~~ two dimensional tables called relation.
- tables are related to each other.

RDBMS

- is dbms based on relational model

Some terminology for RDBMS.

	record or relation oriented	table oriented
1)	↳ relation	table
2)	↳ Domain	set of permitted values
3)	↳ Tuple, record	Row
4)	↳ Attribute, field	column
5)	↳ Cardinality	No. of rows
6)	↳ Degree, Arity	No. of columns
7)	↳ Primary key	unique + NOT NULL

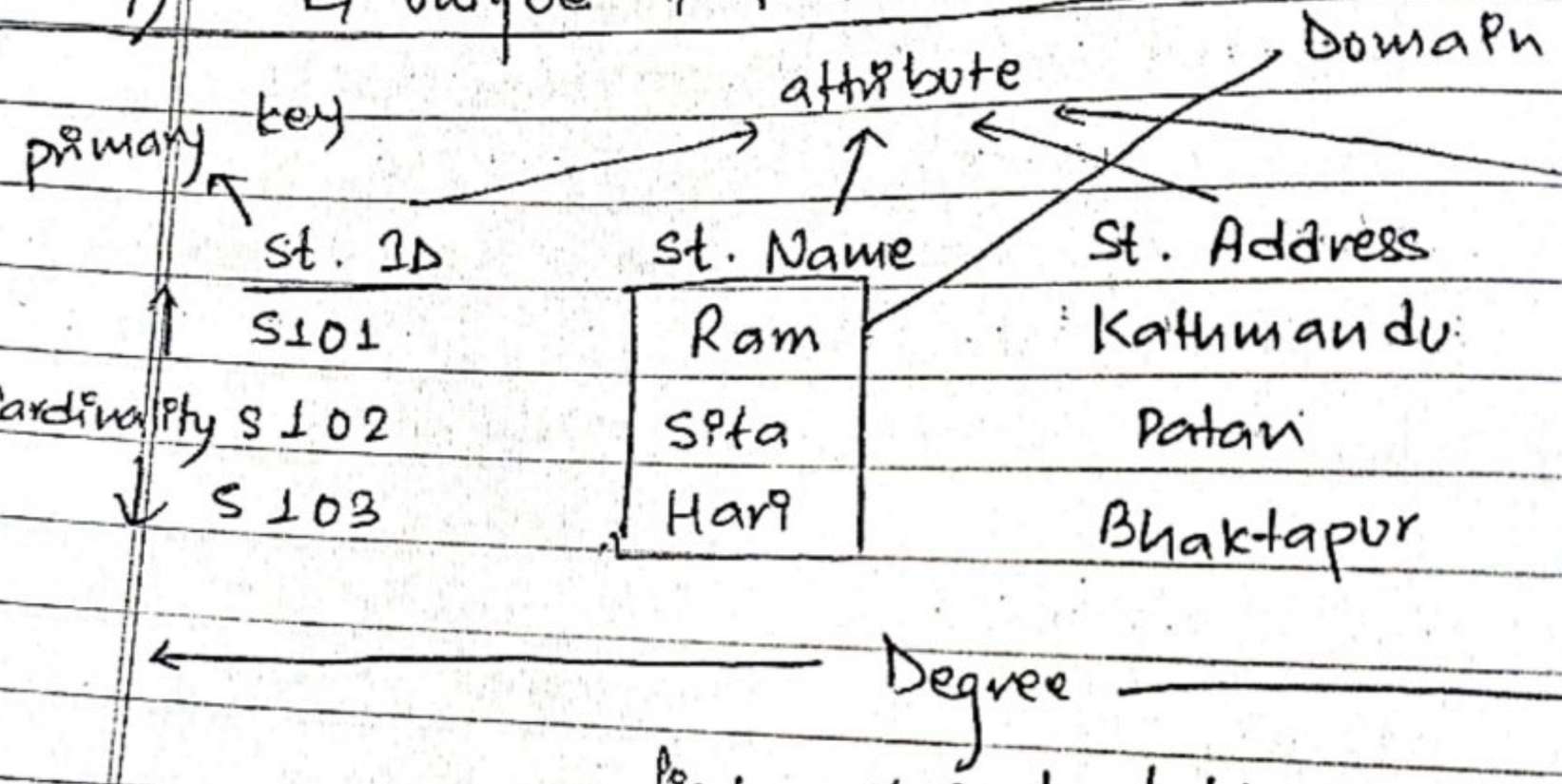
Table oriented

1) ↳ table
2) ↳ set of permitted values
3) ↳ Row

Scanned with CamScanner

CamScanner

4) ↳ ~~code~~ Column
5) ↳ No. of rows.
6) ↳ No. of columns
7) ↳ Unique + NOT NULL



St. ID	St. Name	St. Address	St.
S101	Ram	Kathmandu	471000
S102	Sita	Patan	5567890
S103	Hari	Bhaktapur	660703

fig: student table or student rela

here,

let D_1 is set of st. ID
Let D_2 is set of St. name
Let D_3 is set of st. address
D_4 is set of St. phone

hence,

relation 'Student' is subset of $D_1 \times D_2 \times D_3 \times D_4$

in general,

a table of n attributes must bi subset of

$D_1 \times D_2 \times \cdots \times D_n$

Scanned with CamScanner

Date / /
Page No.

for above eg.
Degree of student = 4
Cardinality of student = 3

→ relation is a set of tuples.
→ $t \in r$ denotes that tuple t is in relation r.
→ for relation r, domains of all attributes of r be atomic.

Integrity Constraints.

- Integrity refers to accuracy or correctness of data in db.
- integrity constraints ensures no result in loss of data. Consistency although changes are made to db by authorized users.
- integrity constraints are defined by some key constraints like primary key, foreign key.
- two general integrity rules are
1) Entity Integrity
2) Referential integrity.

1) Entity Integrity:
- ensures that there are no duplicate records within the table and the field that identifies each record is unique and not null.

Scanned with CamScanner

- it is the mechanism to provide and ma
primary key.
- ensures two properties for primary key
→ primary key for a row is unique
→ primary key is not null.

2) Referential Integrity

- is concerned with foreign key.
- it designates a column or combination column as foreign key and establish relationship between primary key.
- It states that, if in two relations R and if an attribute A, in relation R is primary key and foreign key in S then the value of foreign key in a t tople in R or be equal to primary key be entirely null.

SQL (structured Query language)

→ is the database language designed for mana data in RDBMS.
→ Its scope include data insert, query, upda tion and delete. schema creation and modifi and data access control.

Scanned with CamScanner
CamScanner

Date / /
Page No.

→ can execute queries against databases.
→ Different types of SQL statements are

a) DDL (Data Definition language)
→ CREATE, ALTER, DROP, TRUNCATE

b) DML (Data Manipulation language)
→ INSERT, UPDATE, DELETE

c) DQL (Data Query Language)
→ SELECT

d) DCL (GRANT, REVOKE)
e) TCL (COMMIT, ROLLBACK)

Basic SQL data types

	Datatype	Access	SQL server	Oracle	MYSQL
1)	Boolean	Yes/No	Bit	Byte	N/A
2)	Integer	Number	Int	Number	int, integer
3)	Float	Number	float, real	Number	float
4)	Currency	Currency	Money	N/A	N/A
5)	String (field)	N/A	char	char	char
6)	String (variable)	text	varchar	varchar	varchar
					Text

Scanned with CamScanner

CamScanner

History of SQL

→ SEQUEL (Structured English Query Language) in 1970's

→ developed at IBM

→ SQL - 86 (first ANSI standard)

→ SQL - 89, (SQL1

→ SQL - 92, SQL2 (Most dbms use it)

→ SQL - 99, SQL3 (recursive queries, triggers & OO features)

→ SQL - 2003, windows function, XML related features

→ SQL - 2006, XML query support

→ SQL - 2008

→ SQL - 2011

DDL

→ to create database

Syntax:

CREATE DATABASE dbname;

e.g.

CREATE DATABASE cms;

→ to create table

Syntax

CREATE TABLE tablename

(

column1 & datatype;

Scanned with CamScanner

CamScanner

Date / /
Page No.

```
        :
        column n        datatype ;
    )

CREATE TABLE tablename
(
        column 1        datatype        constraints,
        :
    );
```

Q.9

```
CREATE TABLE STUDENT
(
    SID int,
    name varchar (20),
    address varchar (25)
);
```

```
CREATE TABLE Student
(
    SID int primary key,
    name varchar (20),
    address varchar (25)
};
```

```
CREATE TABLE Student
(
    SID int not null,
    name varchar (20) not null,
    address varchar (25),
    CONSTRAINTS PK_stID
    primary key (SID, name)
);
```

Scanned with CamScanner

To alter table

Syntax

i) ALTER TABLE tablename
ADD Columnname datatype
[e.g: ALTER TABLE student ADD email text]

ii) ALTER TABLE tablename
DROP COLUMN Columnname
[e.g DROP COLUMN address]

iii) ALTER TABLE tablename
// Modify // ALTER COLUMN columnname datatype
[e.g. ALTER COLUMN email text]

→ ALTER TABLE student
ADD primary key (SID)
or
ADD constraint PK - SID Primary key (SID)

→ ALTER TABLE Student
DROP Constraint PK - SID

To delete table

Syntax.

DROP TABLE tablename

To delete data inside table, not table itself

Scanned with CamScanner
CamScanner

Date / /
Page No.

Syntax:

TRUNCATE TABLE tablename

DML (Data Manipulation language)

1) Insert

- to insert record use INSERT statement

Syntax

→ INSERT INTO table name (column1, ...) VALUES (Val1, ...);

→ INSERT INTO tablename VALUES (Val1, ...);

→ INSERT INTO tablename VALUES (Val1, Val2, ...), (Val1, Val2, ...);

→ INSERT INTO tablename VALUES (Val1, ~~Val2~~ NULL, Val2, ... Valn);

Student

SID	name	address	phone

Scanned with CamScanner

CamScanner

2) UPDATE

- to modify records use UPDATE statement

Syntax :

i) UPDATE tablename SET Column1 = Val

ii) UPDATE tablename SET Column1 = Valu
WHERE condition

iii) UPDATE tablename SET Column1 = Valu
Column2 = Value 2 WHERE condition
(SID = 1)

Student

SID	name	address	phone
1	Ram	ktm	9602 . . .
2	Shyam	Pkr	9860 . . .
3	Shyam	Dharan	9814 . . .
4	Sita	Patan	98510 . . .
5	Gita	Bhaktapur	9023 . . .

e.g:

UPDATE student SET phone = 4710346;
UPDATE student SET phone = 4710346.
WHERE SID = 1;
UPDATE student SET name = 'Gita', phone
= 4445678 WHERE SID = 2;

Scanned with CamScanner

Date / /
Page No.

3) DELETE :

- to delete use DELETE statement

Syntax

i) DELETE FROM ~~TABLE~~ tablename

ii) DELETE FROM tablename WHERE condition -

e.g: DELETE FROM student
DELETE FROM Student WHERE SID=5

DQL (Data Query language)

Query:

- a query is an operation that retrieves data from one or more tables or views.
- query nested within another SQL statement is called subquery.
- use 'SELECT' statement to retrieve info from table.

Syntax

i) SELECT * FROM tablename;

ii) SELECT col1, col2 FROM tablename;

iii) SELECT * FROM tablename WHERE condition

Scanned with CamScanner

CamScanner

iv) SELECT * FROM tablename WHERE Col1 = Val1 AND Col2 = Val2.

v) SELECT * FROM tablename WHERE Col1 = 'Val1' OR Col2 = 'Val2'.

vi) SELECT * FROM tablename WHERE Col1 = 'Val1' AND (Col2 = 'Val2' OR Col3 = 'Val3')

~~DSL (Data~~

vii) SELECT * FROM tablename ORDER BY Columnname ASC | DESC ;

viii) SELECT Column1 as New Col FROM tablenam
optional

ix) SELECT TOP number | percent Columnname(FROM tablename;

e.g: SELECT TOP 50 percent * FROM student

Scanned with CamScanner

CamScanner

Date / /
Page No.

Wild Card Characters:

→ are used with SQL LIKE operator.
→ are used to search for data within table.
→ % ⇒ a substitute for zero or more characters
→ _ ⇒ a substitute for single character.
→ [character list] ⇒ sets and range of characters to match
→ [! character list] ⇒ Match only characters not specified in brackets.

Syntax:
select columnname (s) FROM tablename WHERE Columnname LIKE pattern;

e.g:
→ select * FROM customer WHERE city LIKE '%s';
→ select * from customer where country like '% land %';
→ select * from customer where country NOT like '% land %';
→ select * from customer where city like '% ber %'.
→ select * from customer where city like '_erlin';
→ select * from customer where city like 'L_N_ON';

Scanned with CamScanner

28th August

Date / /
Page No. 27

[ ] → for multiple options.

→ select * from customer where city like '[bsp
→ " " " " " " " '[a-c
→ " " " " " " " '[!bsp
→ " " " " " " " NOT lic
⇒ " " " " '[bsp] %'

Employee

EID	Name	Address	Age	Salary
1	Ram	Patan	35	20000
2	Hari	Pokhara	40	25000
3	Sita	Dharan	32	22000
4	Shyam	Kathmandu	40	20000
5	Gita	Palpa	28	17000

1. Find name of employee whose age is less th 40

⇒ SELECT name from employee where Age <

2. Display all employee whose name starts with S.

⇒ SELECT * FROM Employee where name LIKE 'S%';

Scanned with CamScanner

CamScanner

सुगम स्टेसनरी सप्लायर्स एण्ड फोटोकपी सर्भिस
बालकुमारी, ललितपुर ९८४१५९९५९२
NCIT College

Date / /
Page No.

3) Display all employees whose age is less than 40 and salary greater than 20,000.

⇒ SELECT * FROM employee WHERE age < 40 AND Salary > 20000.

EID	Name	Address	Age	Salary
3	Sita	Dharan	32	22000

4) Display name of employee whose address starts with either P or K.

⇒ SELECT name FROM employee WHERE address LIKE '[PK]%'

5) Display all employees whose salary not starts with 2.

SELECT * FROM employee WHERE salary NOT LIKE '2%'.

Scanned with CamScanner

XI) Select Columnname (S) from tablename where Columnname IN (Val1, Val2, ...)

e.g:

Select * from Customer where city IN ('London', 'Paris'); [where city = 'London', city 'Paris'].

XII) Select Columnname (S) from tablename where Columnname BETWEEN Value 1 AND Value 2

e.g:

→ Select * from products where price between 10 and 20.

→ Select * from ~~ord~~ products where price not between 10 and 20.

→ Select * from products where (price between 10 and 20) and not category IN (1, 2, 3);

→ select * from products where productname between 'C' and 'M'.

→ Select * from products where productname not between 'C' and 'M'.

→ SELECT * FROM Orders WHERE Orderdate between # 07/04/2016 # and # 08/05/2016 #.

change tablename temporarily.

XIII) Select * from tablename As Newtablename;

e.g. select * from Customer As c;

Scanned with CamScanner

CamScanner

XI) Select Columnname (S) from tablename where Columnname IN (Val1, Val2, . . .)

e.g:

Select * from Customer where city IN ('London', 'Paris'); [where city = 'London', 'Paris']

XII) Select Columnname (S) from tablename where Columnname BETWEEN Value 1 AND Value 2

e.g:

→ Select * from products where price between 10 and 20.

→ Select * from products where price not between 10 and 20.

→ Select * from products where (price between 10 and 20) and not category IN (1, 2, 3);

→ select * from products where productname between 'C' and 'M'.

→ Select * from products where productname not between 'C' and 'M'.

→ SELECT * FROM Orders WHERE Orderdate between # 07/04/2016 # and # 08/05/2016 #.

XIII) change tablename temporarily.
Select * from tablename As Newtablename;
e.g. select * from Customer As C;

Scanned with CamScanner

Date / /
Page No.

SQL Functions

⇒ AVG ( ) → returns average value of numeric column.
— Select AVG (columnname) from tablename.

⇒ Count ( ) → returns the no. of rows.
— Select count (columnname) from tablename;
— Select Count (*) from tablename;

⇒ Max( ) → returns the maximum value of selected column.
— Select max (columnname) from tablename;

⇒ Min( ) → return the minimum value of selected column.
→ select min (columnname) from tablename;

⇒ Sum( ) → returns the total sum of numeric column.
— SELECT Sum (columnname) from tablename;

⇒ Upper ( ) → converts value of field to uppercase
— SELECT upper (Columnname) from tablename;

⇒ lower ( ) → converts value of field to lower case

Scanned with CamScanner

CamScanner

- select lower (columnname) from tablename;

⇒ SQRT () → finds square root of any no.
- select sqrt (columnane) from tablename;

⇒ RAND () → to produce random numbers betn 0 and 1.
- Select * from employee order by RAND();

⇒ concat() → to concatenate two strings to from single string.
- Select concat (Column1, Column2) from tablename.

Groupby and "Having" ~~Cla~~ Clause

- used to divide the rows in table in groups.
- is used with SQL aggregate functions like Sum(), Avg()

e.g:
- Select job, sum (salary) from employee group by job.
- Select job, max (salary) from employee group by job having max (salary) >= 30000

Scanned with CamScanner

Date / /

Page No.

Subquery:

- is a query within another SQL query and embedded within 'where' clause.
- is used to return data that will be used in main query as a condⁿ to further restrict data to be retrieved.
- can be used with select, insert, update and delete statements along with operators like =, >=, <=, <, >, in, between etc.

Scanned with CamScanner

CamScanner

Subquery

Rules for subquery:

→ must be enclosed within parenthesis
→ can have only one column in 'select' claus
→ An 'order by' can't be used but 'Group b can be.
→ Subqueries that return more than one row can only be used with multiple value operator like 'IN'.
→ A Subquery can't be immediately enclosed in a set function.
→ 'Between' operator can't be used with subquery, however it can be used within subquery.

Subqueries with select statement
Syntax

Select Columnname (s) from table where columnname operator
(select Columnname (s) from table where..)

e.g:

Select * from customer
where ID IN
(select ID from customer where salary > 4500)

Scanned with CamScanner

Subqueries with INSERT statement

e.g:

Insert into customer1 select * from Customer where ID IN (Select ID from Customer);

Subqueries with update statement

e.g:

UPDATE Customer SET Salary = Salary * 0.25 where age in (Select age from customer where age >= 27);

Subqueries with Delete Statement

e.g:

Delete from Customer where age in (select age from customer where age>27)

Customer

ID	name	age	Salary
1	Ram	28	10000
2	Shyam	32	5000
3	Hari	49	3000
4	Sita	23	7000

Scanned with CamScanner

CamScanner

SQL using join

- join is used to combine records from two or more tables in a database based on a common field between them.

Different types of join

a) Inner join (or simply join)

Select all rows from both tables as long as there is a match between the columns in both tables.

Syntax

select Columnname (s) from table1
inner join table2
on table1 column1 = table2. column2

eg:

Select Customer. Customer Name, Order. Order
from customer inner join order.
on customer. Customer ID = Order. Customer ID.

b) Left join (left outer join)

- returns all rows from the left table (table1) with the matching rows in right table (table2)
- The result is null in right side where there is no match.

Syntax

Select Columnname (s) from table1
left join table2

Scanned with CamScanner

e.g

Select Customer. Customer Name, Order. Order ID from customer left join order on customer. Customer ID = Order. CustomerID

c) Right join (right outer join) ~~(second table)~~

- returns all rows from right table (table 2) with the matching rows in left table (table 1)
- the result is null in leftside where there is no match.

Syntax

Select columnname(s) from table1
right join table2
on table1. Column1 = table2. Column2

e.g:

Select Customer. Customername, Order. OrderID from Customer right join order on customer. Customer ID = Order. Customer ID

d) Full outer join

- returns all rows from left table and from right table.
- combine the result of both left and right join.

Syntax:

Select Columnname(s) from table1
Full outer join table2

Scanned with CamScanner

CamScanner

on table1. column1 = table2. Column2

e.g:

Select Customer. Customermame , Order. OrderID
from Customer full outer join order
on customer. CustomerID = Order. CustomerID.

Set operation

1) Union Operation

Syntax

a) Select Columnname (s) from table1
Union
Select Columnname (s) from table2.

b) Select Columnname (s) from table1
Union All
Select Columnname (s) from table 2.

2) Intersect operator

Syntax:

Select Columnname (s) from table1
Intersect
Select Columnname (s) from table2

3) Except operator

Syntax:

Select Columnname (s) from table1
Except
Select Columnname (s) from table2.

Scanned with CamScanner

CamScanner

Views

→ View is a virtual table based on the result set of an SQL statement.

→ Views are logical representation of data.

→ It contains rows and columns just like real table.

→ SQL functions, where and join statements can be added to a view and data can be presented as if data are coming from a single table.

→ View allows users to

- structure data in a way that users want.
- restrict access to data such that users can see and sometimes modify exactly what they need and no more.
- summarize data from various tables, which can be used to generate reports.

Syntax to create view.

```
CREATE VIEW viewname AS
SELECT columnname (s) FROM TABLE ; /
WHERE CLAUSE
```

to query view

```
SELECT * FROM viewname ;
```

to update view

```
UPDATE viewname SET column1 = Val1
WHERE - . . .
```

Scanned with CamScanner

CamScanner

to drop view

DROP VIEW viewname ;

Note:

It shouldnot contains aggregate function or Distinct or Groupby clause while updating view

Example

Customer (cid , cname , Caddress)

order (oid , cid , date)

⇓

CREATE VIEW V customer

SELECT Cname , date FROM CUSTOMER

INNER JOIN ORDER

ON CUSTOMER . cid = ORDER . cid .

Relational Algebra: 8 marks.

→ is a procedural query language which takes instances of relations as input and gives instance of relation as output.

→ helps to understand query execution and optimization in RDBMS.

→ Basic operations of relational algebra are

a) selection

b) projection

c) Union

d) set difference

e) cartesian

Date / /
Page No.

1) Selection

- It selects a subset of rows from relation that satisfies some condition.
- It is denoted by σ.
- here, schema of result is identical to schema of input relation.
- Syntax:

$\sigma_{condition}$ (relation)

- result relation can be input for another relational algebra operation.

employee (eid, ename, eaddress, salary)
find all employees whose salary is greater than 10000.

$\sigma_{salary > 10000}$ (employee)

eid	ename	eaddress	salary
1	Ram	ktm	15000
2	Shyam	Patan	10000

2) Projection:

- It deletes the attribute that are not in project list.
- is denoted by π.
- Schema of result contains exactly the

Scanned with CamScanner

fields in projection list, with the same names that they had in input relation.

- Syntax:

Π attribute list (relation)

e.g:

$\Pi_{ename, salary}$ (employee)

eid	ename	eaddress	Salary
1	Ram	Ktm	15000
2	Shyam	Patan	10000
3	Hari	Bhaktapur	12000

$\Pi_{ename, salary}(\sigma_{salary > 10000}(employee))$

$\sigma_{salary > 10000}(\Pi_{ename, salary}(employee))$

3) Union

- it builds a relation from tuples appearing in either or both of the specified relation.
- it eliminates duplicates.
- is denoted by $\cup$.
- two relation are union compatible if
  - → they have same no. of attributes.
  - → attribute domain must be compatible.

Scanned with CamScanner

Date / /
Page No.

→ names of attribute are same in both.

Syntax:

$\Pi_{\text{attribute, list}}$ (relation) $\cup$ $\Pi_{\text{attribute list}}$ (relation)

→ $R \cup S = S \cup R$

4) Set Difference // Except

→ if results tuples present in 1st relation but not in second relation.

→ PS denoted by.

→ union compatible should exists.

→ Syntax:

$\Pi_{\text{attributelist}}$ (relation) − $\Pi_{\text{attributelist}}$ (relation)

→ $R - S \neq S - R$

Scanned with CamScanner

CamScanner

5) Cartesian Product

↳ It builds a relation with concatenation of every row in first relation with every row in second relation.

↳ also known as Cross product or ~~Cross~~ Cross join

↳ is denoted by 'X'.

Syntax :-

R X S

e.g:

Book

BID	BName	Price	PID
B10	BName	Price	PID
B1	DBMS	550	P1
B2	Math	600	P2
B3	SE	750	P3

Publisher

PID	PName	Paddr
P1	ABC	ktm
P2	XYZ	Patam

Book X Publisher

BID	BName	Price	Book. PID	Publisher. PID	PName	Paddr
B1	DBMS	550	P1	P1	ABC	ktm
B1	DBMS	550	P1	P2	XYZ	Patan
B2	Math	600	P2	P1	ABC	ktm
B2	Math	600	P2	P2	XYZ	Patan
B3	SE	750	P3	P1	ABC	ktm
B3	SE	750	P3	P2	XYZ	Patan

Scanned with CamScanner

CamScanner

B - name, price, publisher

$\Rightarrow$ $\Pi_{Bname, price, Pname}$ (Book $\times$ publisher)

$\Downarrow$ condition, (Book PID relation P. PID)

$\Pi_{Bname, price, Pname}$ ($\sigma_{Book.PID = publish.PID}$ (Book $\times$ publisher))

~~Relationshat~~ A

Intersection

→ results tuples that appear in both relation. → denoted by $\cap$.

→ relation must be union compatible

→ Syntax:

$\Pi_{attributelist}$ (relation1) $\cap$ $\Pi_{attributelist}$ (relation2)

→ $R \cap S = S \cap R$

Scanned with CamScanner

CamScanner

Join operation (⋈) { cartisian product with selection }

↳ allow users to combine two relation in a specified way.

↳ The join operation can be stated in terms of a cartesian product followed by a selection operation.

↳ is very important as used frequently while specifying database queries.

↳ three types of join:

a) Theta join (Condition Join)
b) Equijoin (inner join)
c) Natural join

a) Condition Join (Theta join)

↳ join two relation together on basis of some comparision operator.

↳ θ is a predicate & consists of one of comparision operator (=, <, >, <=, >=, <>) and specifies join condition.

↳ if θ is =, then it is called equijoin.

↳ defined as cross product followed by selection.

↳ for any two relations R and S, theta join is

$$R \bowtie_\theta S \quad \text{OR}$$

$$\sigma_\theta (R \times S)$$

Scanned with CamScanner

e.g.

R1

Sid	bid	day
22	101	10/10/2015
58	103	9/9/2016

S1

Sid	S name	rating	age
22	Ram	7	45
31	shyam	8	50
58	Hari	10	35

Now,

S1 ⋈ R1

S1.Sid < R1.Sid

R1.Sid	S1.Sid
22	22
22	31
22	58
58	22 ✓
58	31 ✓
58	58

o/p

sid	sname	rating	age
22	Ram	7	45
31	Shyam	8	50

sid	bid	day
58	101	9/9/2016
58	103	9/9/2016

b) Equi join or Inner Join:

↳ if θ is '=', then theta join becomes equijoin.

↳ the (=) condition is performed by primary and foreign keys.

↳ The result must include two attributes with property that values of these two attributes are equal every tuple in relation.

↳ ~~Sy~~ ~~Sy~~ Syntax:

$$R \bowtie_{(R.PK = S.FK)} S$$

e.g:

$$R1 \bowtie_{R1.sid = S1.sid} S1.$$

sid	bid	day
22	101	10/10/2015
58	103	9/9/2016

sid	sname	Rating	Age
22	Ram	7	45
58	Hari	10	35

Scanned with CamScanner

CamScanner

e) Natural join:

↳ is represented by symbol ⋈

↳ only relates those rows which have common attribute value.

↳ For relation R and S

R⋈S = Column (R) U column (S)

Employee

EID	name	Deptname
101	Ram	Finance
102	Shyam	Marketing
103	Hari	Finance

Department

Deptname	Manager
Finance	Gopal
Marketing	Gita

employee ⋈ Department.

EID	name	Deptname	Deptname	Manager
101	Ram	Finance	Finance	Gopal
102	Shyam	Marketing	Marketing	Gita
103	Hari	Finance	Finance	Gopal.

Scanned with CamScanner

Hierarchical Model:

↳ it organizes data in tree structure where there is a hierarchy of parent and child data segments.

↳ It has a single root segment connected to lower level segments.

↳ each segment may connect to other lower level segments.

↳ Here, rule is that one parent can have many children but children are allowed only one parent.

↳ this allows information to be repeated through parent child relation.

Advantages

↳ allows easy condition and deletion of new information.

↳ fast access to data at top.

↳ relates well to anything that works on one to many relationship.

Disadvantages:

↳ allows data redundancy.

↳ Searching for lower level information is slow.

↳ Many to many relationship are not supported.

Scanned with CamScanner

Date / /
Page No.

Network Model:

↳ replaces the hierarchical tree with graph.
↳ allows to have

Advantages

↳ simple and easy to implement.
↳ can handle 1:1 and many to many N:N relationship.
↳ data access is easier than hierarchical model.

Disadvantages:

↳ System complexity.
↳ Lack of structural independence.
i.e. change in structure makes change in application too.

Relational Model

↳ all data are stored in tables where each table consists of rows and columns.
↳ here, keys are used to order data or relate data to other tables.
↳ defines set of rules to enforce data integrity known as integrity constraints.
↳ also defines how data are manipulated.
↳ defines special feature called normalization to ensure efficient data structure.

Scanned with CamScanner

Advantages :-
- easy of use
- flexibility
- security
- data independence.

Disadvantages :
- use of design can lead to ~~data~~ bad design
- H/w overheads.

QBE (Query By Example)

- developed at IBM in 1970's.
- QBE has a two dimensional syntax.
- Queries looks like table.
- QBE queries are expressed by example instead of giving a procedure.
- Queries in QBE are expressed by using skeleton tables, which show the relational schema of database.
- Skeleton table looks like.

Relation name	Column1	Column2	. . .	Column N
Tuple or row operation.				

- For relation College (CollegeId, Name), QBE is as:

College	College Id	Name.

Scanned with CamScanner

Advantages:-

- easy of use
- flexibility
- security
- data independence.

Disadvantages:

- ease of design can lead to ~~data~~ bad design.
- H/W overheads.

QBE (Query By Example)

- developed at IBM in 1970's.
- QBE has a two dimensional syntax.
- Queries looks like table.
- QBE queries are expressed by example instead of giving a procedure.
- Queries in QBE are expressed by using skeleton tables, which show the relational schema of database.
- Skeleton table looks like.

Relation name	Column1	Column2	. . .	Column N
Tuple or				
row operation.				

- For relation College (collegeId, Name), QBE is as:

College	College Id	Name.

Scanned with CamScanner

Date / /
Page No.

↳ use of P to get list of relation.
↳ P. for retrieval command
↳ data command is P. <variablename>
where variablename is optional.
↳ QBE automatically eliminates duplicate values
↳ to see duplicate values, P. ALL
↳ QBE supported by microsoft Access is Graphical QBE.

e.g:
employee (ename, city, salary, rating)

1) to find all employee names who live in city kathmandu.

employee	ename	city	salary	rating
	P. or P. ALL	kathmandu		

2) to find entire employee

employee	ename	city	salary	rating
P.				

3) to find employee name where salary is greater than 10000.

employee	ename	city	salary	rating
	P. or P. ALL		>10000	

Scanned with CamScanner

सुगम स्टेसनरी सप्लायर्स एण्ड फोटोकपी सर्भिस
बालकुमारी, ललितपुर ९८४१५९९५९२
NCIT College

4) to find all employees whose salary between 5000 and 10000.

employee	ename	city	salary	rating
P.			-x	

Condition
-x >= 5000
-x <= 10000

Database modification in DBE.

1) Insert

Employee	ename	city	salary	rating
I.	Ram	Pokhara	15000	5
I.	Shyam			6

2) Delete

employee	ename	city	salary	rating	~~rating~~
D.	Ram.				

employee	ename	city	salary	rating
	Ram	D.		

सुगम स्टेसनरी सप्लायर्स एण्ड फोटोकपी सर्भिस
बालकुमारी, ललितपुर ९८४१५९९५९२
NCIT College

Scanned with CamScanner

Date / /
Page No.

3) Update

employee	ename	city	salary	rating
	Ram			U. 10

Relational Calculus

- is non procedural query language that describe the set of answers, without being explicit about how they should be computed.
- two types,
  - ↳ Tuple relational calculus (TRC)
  - ↳ Domain " " (DRC)

(I) TRC

- a TRC query has the form $\{T \mid P(T)\}$ where
  - T is a tuple variable.
  - P(T) is a formula that describes T.
- it results set of all tuples t to which P(T) evaluates to true when T = t.

e.g:

a) $\{T.name \mid Author(T)$ and $T.article = 'dbms'\}$

- it returns tuples with name from Author who has written article on 'dbms'.

Scanned with CamScanner

b) { t | student (t) and t.percentage > 60 }

c) { t | employee (t) and t.deptNo = 10 }

DRC

1) ↳ it uses domain variables that take on values from attribute domain rather than values for an entire tuple.

↳ query is expressed in the form of

{ < x_1, x_2, ... x_n > | p (x_1, x_2, ... x_n) }

where x_1, x_2, ... x_n represent domain variables and p represents formulas.

↳ an atom in DRC has the form.

→ < x_1, x_2, ... x_n > ∈ r, where r is relation of n attributes and x_1, x_2, ... x_n are domain variables.

→ X ⊖ Y, where X and Y are domain variable and ⊖ is comparision operator.

→ X ⊖ c, where X is domain variable, ⊖ is comparision operator, c is a constant.

Scanned with CamScanner

CamScanner

Date / /
Page No.

⇒ We build up formula from atom by using following rules

↳ an atom is a formula.

↳ If P1 is a formula then so are ¬P1 and (P1)

↳ if P1 and P2 are formula then so are P1 ∧ P2, P1 ∨ P2 and P1 = P2.

↳ if P1(x) is a formula in x, where x is domain variable then ∃x (P1(x)) and ∀x (P1(x)) are also formula.

e.g:

i) $\{ < l, b, a > \mid < l, b, a > \in loan \wedge a > 20000 \}$

ii) $\{ < c, a > \mid \exists_l (< c, l > \in borrower \wedge \exists_b (< l, b, a > \in loan \wedge b = 'ktm')) \}$

Note:

Expression power of TRC and DRC is equivalent to relational algebra.

Stored Procedure

↳ is a set of SQL statements with an assigned name that is stored in db in compiled form so that it can be shared by no. of programs.

↳ is nothing more than prepared SQL code that is saved and can be reused over and over again

Scanned with CamScanner

CamScanner

Creating Simple Stored Procedure

Syntax:

```
CREATE PROCEDURE procedureName
AS
  SQL Statement
```

e.g:

```
CREATE PROCEDURE getInfo
As
select * from Employee
```

To execute stored procedure

→ getInfo

→ EXEC getInfo

Stored procedure with a parameter

```
CREATE procedure getInfo @city varchar(25)
As
  select * from employee
  where city = @city
```

e.g:

getInfo @city = "Berlin"

⇒ employee (eid, name, salary)

department (did, dname, eid).

```
CREATE PROCEDURE getInfo
As
SELECT name, salary, dname from employee Inner
  join department
  on employee.eid = department.eid
```

Scanned with CamScanner

CamScanner

CHAPTER 5

Database Constraints and Relational Database Design.

↳ Constraints are nothing but rules on data that help to define what data is valid and what is invalid, helps to maintain integrity of data.

↳ Database constraints are user defined structure that let you restrict the behaviour of columns.

↳ Different types of database constraints are
i) primary key constraint
ii) foreign " "
iii) NOT NULL " "
iv) Unique " "

Integrity Constraints:
- ensures that changes are made to the database by authorized users do not result in loss of data consistency.
- guards against accidental damage of database

Referential Integrity
- ensures that a value that appears in one relation for a given set of attributes also appear for certain set of attributes in another relation.
- ensures that foreign key value = primary key value.

Scanned with CamScanner

Functional dependencies:

- are the fundamental to the process of normalization.
- FD plays key role in differentiating good database design from bad db design.
- FD describes the relationship between attributes in a table.
- for attributes X & Y in relation R, functional dependencies between X & Y is shown as

$$X \longrightarrow Y$$

here,

X is determinant
Y is functionally dependent on X.

- The functional dependency $X \longrightarrow Y$ holds on table R if in any legal relation r(R), for all pair of tuples t_1 & t_2 in r such that $t_1[X] = t_2[X]$, it is also case that $t_1[Y] = t_2[Y]$

eg:

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	d2
a2	b2	c2	d2
a2	b2	c2	d3
a3	b3	c2	d4

here,
$A \rightarrow C$ is satisfied
$C \rightarrow A$ is not satisfied
$\because$, $t_1[C] = t_2[C]$
but
$t_1[A] \neq t_2[A]$

Scanned with CamScanner

CamScanner

Date / /
Page No.

E.g:

eid	ename	age	salary
101	Ram	25	10000
102	Shyam	28	15000
103	Hari	30	18000

here,

eid → ename ; True as eid uniquely determines ename

ename → eid ; false as it is not always true

eid → age ; True

eid → salary ; false

age → eid ; false.

eid	project No	hrs	ename	project name	project location

here,

eid → ename

project No → { project Name , project location }

{eid , project No} → hrs.

Types :

Scanned with CamScanner

CamScanner

CHAPTER 10

Transaction Processing and Concurrency Control

25th Sept
Page 42

Transaction System

→ group of operations that form a single logical unit of work is called transaction.

→ a transaction is a logical unit of work that must be either entirely completed or aborted.

→ a transaction is a unit of program execution that access and possibly updates various data items.

→ a transaction is the DBMS abstract view of a user program that consists of sequence of reads and writes.

→ transaction access data using two operations
  i) read (X)
  ii) write (X)

→ Transaction processing is information processing that is derived into individual, indivisible operations called transactions.

→ each transaction must succeed or fail as a complete unit but can't remain in intermediate state.

→ Transaction processing maintains a system in as consistent state.

Scanned with CamScanner

CamScanner

Date / /
Page No.

e.g.:

T1 : read (A)
A = A - 100
write (A)
read (B)
B = B + 100
write (B)

Properties of Transaction (ACID properties)

1) **Atomicity**: either all operations of transaction are reflected properly in database or none are.

2) **Consistency**: execution of a transaction in isolation preserves the consistency of db.

3) **Isolation**: even though multiple transactions may execute concurrently, the system guarantees that for every pair of transactions T_i & T_j for T_i either T_j finished execution before T_i started or T_j started execution after finished.

4) **Durability**: after successful completion of transaction, the changes in db persist, even if there are system failures.

Scanned with CamScanner

CamScanner

Transaction State:

i) Active State:
- It is the initial state.
- transaction stays in this state while it is executing.

ii) Partially committed:
- transaction is in P. C state after the final statement has been executed.

iii) failed:
- transaction is in failed state after normal execution can no longer proceed.

iv) Aborted:
- transaction is in aborted state after it has been rolled back & the database has been restored to its state prior to the start of transaction.

v) committed:
- transaction is in committed state after successful completion.

Scanned with CamScanner

Date / /

Page No.

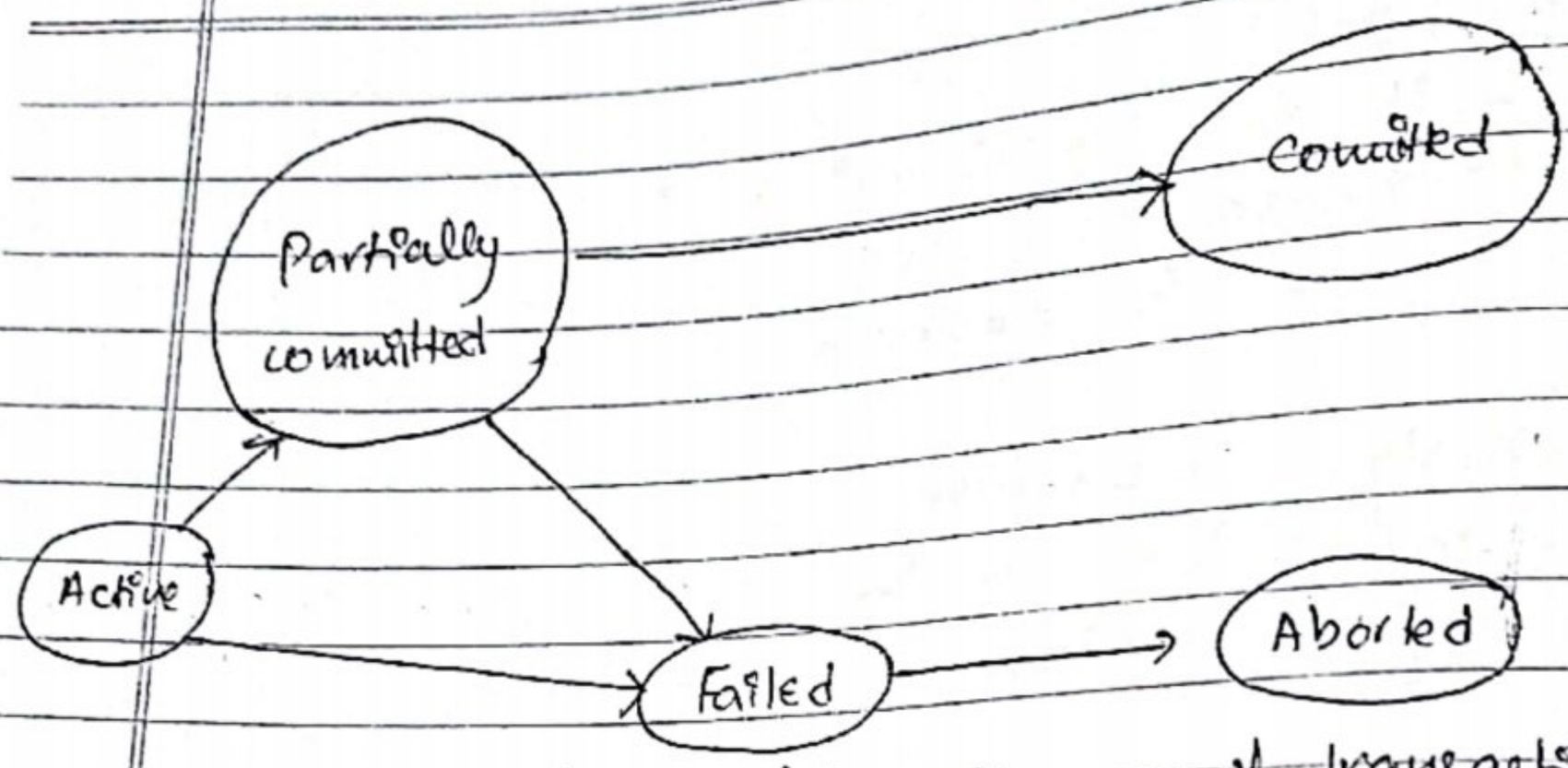


fig: state diagram of transaction.

→ a transaction has committed only if it enter committed state

→ a transaction has aborted only if it enter aborted state.

→ a transaction has terminated if either committed or aborted.

→ after transaction being in aborted state, system has two options either it can restart transaction.

→ Transaction processing system usually allow multiple transactions to run concurrently.

→ This cause several complications with consistency of data.

→ though, transactions can be run serially to ensure consistency, concurrent execution of transaction has following advantages:

Scanned with CamScanner

CamScanner

i) Improved throughput and resource utilization.

ii) Reduced waiting time.

→ database system control the concurrent transactions to prevent from the interaction among destroying consistency of db through variety of mechanisms called concurrency control scheme

→ when transactions are executing concurrently in an ~~unbalanced~~ interleaved fashion, then the ~~act~~ order of execution of operations from various transactions is known as schedule.

↳ A schedule is specifies the chronological order in which the operations of concurrent transactions are executed.

↳ e.g:

Let Transaction T1 transfer Rs 100 from account A to B.

Transaction T2 transfer 20 % of balance from A to B.

let A = 1000, B = 1000

Now,

T1 :- read(A)
A = A - 1000
write(A)
read(B)
B = B + 100
write(B)

Scanned with CamScanner

Date / /

Page No.

T2 : read (A)
temp = A * 0.2
A = A - temp
write (A)
read (B)
B = B + temp
write (B)

Serial Schedule

i)

T1.	T2
read (A)	
A = A - 100	
write (A)	
read (B)	
B = B + 100	
write (B)	
	read (A)
	temp = A * 0.2
	A = A - temp
	write (A)
	read (B)
	B = B + temp
	write (B)

ii)

T1	T2
	read (A)
	temp = A * 0.2
	A = A - temp
	write (A)
	read (B)
	B = B + temp
	write (B)
read (A)	
A = A - 100	
write (A)	
read (B)	
B = B + 100	
write (B)	

Scanned with CamScanner

CamScanner

Non - Serial Schedule

1)

T1	T2
read (A)	
A = A − 100	
write (A)	
	read (A)
	temp = A × 0.2
	A = A − temp
	write (A)
read (B)	
B = B + 100	
write (B)	
	read (B)
	B = B + temp
	write (B)

2)

T1	T2
read (A)	
A = A − 100	
	read (A) → 1000
	temp = A × 0.2 → 200
	A = A − temp → 800
	write (A) → 800
	read (B) → 1000
write (A) → 800	
read (B) → 1000	
B = B + 100 → 1100	
write (B) → 1100	B = B + temp

Scanned with CamScanner

CamScanner

Date / /

Page No.

here, both (i) & (ii) are non serial schedule.

i) results correct state

but

ii) doesn't result correct state.

Scanned with CamScanner

CamScanner

Serializability.

→ Basic assumption in each transaction preserves data consistency.

→ Thus serial execution of a set of transaction preserves database consistency.

→ Main objective of serializability is to search non serial schedules that allow transaction to execute concurrently without interfering one another transaction and produce the result of db state that would be produced by serial execution.

→ We can conclude that a non serial schedule is correct if it produce same result as serial execution.

→ a schedule is serializable if it is equivalent to a serial schedule.

Rules:

→ Ordering of read/write is important

→ if two transactions only read data item they do not conflict and order is not important.

→ If two transactions either read or

write completely separate data items, they do not conflict and order is not important.

→ if one transaction write a data item and another reads or write same data item, order of execution is important

Schedule 2

Scheduled			Schedule 2		
T1	T2	T3	T1	T2	T3
R(a)			R(a)		
	R(b)		W(a)		
		R(c)		R(b)	
W(a)				W(b)	
	W(b)				R(c)
		W(c)			W(c)

→ Here, actions of transactions in schedule 1 are not executed as same as in schedule 2 but at the end schedule 1 gives same result as that of schedule 2.

→ Thus it is considered as serializable.

Scanned with CamScanner

CamScanner

Conflict serializability.

→ Instructions I_i and I_j of transactions T_i and T_j respectively conflict if and only if there exists same item Q accessed by both I_i & I_j and at least one of these instruction wrote 'Q'.

→ If a schedule S can be transformed into a schedule 'S' by a series of swaps of non conflicting instructions, we say that S and S' are conflict equivalent.

→ We say that a schedule S is conflict serializable if it is conflict equiv. to a serial schedule.

→

Instruction I_i	Instruction I_j	Result.
Read (Q)	Read (Q)	No conflict
Read (Q)	Write (Q)	Conflict
Write (Q)	Read (Q)	"
Write (Q)	Write (Q)	"

Scanned with CamScanner

Eg. 1) let we have schedule A as.

T1	T2
read (A)	
write (A)	
	read (A)
	write (A)
read (B)	
write (B)	
	read (B)
	write (B)

and schedule B as,

T1	T2
read (A)	
write (A)	
read (B)	
write (B)	
	read (A)
	write (A)
	read (B)
	write (B)

Scanned with CamScanner

CamScanner

→ Here, schedule A can be transformed into serial schedule B by series of swaps of non conflict instructions.

→ Hence, schedule A is conflict serializable.

eg.2) Let we have,

Schedule X

T1	T2
read (Q)	
	write (Q)
write (Q)	

Schedule Y.

T1	T2
read (Q)	
write (Q)	
	write (Q)

Here, we can't transform schedule X into schedule Y by swapping non conflict instruction.

∴ Schedule X is non conflict serializable.

View serializability:

→ two schedules are said to be view equivalent if following 3 condition holds.

1) For each data item Q, if transaction T_i reads initial value of Q in schedule S, also read initial value of Q in schedule S'.

Scanned with CamScanner

2) For each Q, if T_i executes read (Q) in S and if value was produced by write (Q) of T_i then read (Q) of T_i must in S' also read value of Q produced by same write (Q) of T_i.

3) For each Q, the transaction that performs the final write (Q) operation in S; must perform final write (Q) operation in S'.

→ a schedule is view serializable if it is view equiv. to a serial schedule.

eg: let,

Schedule A			Schedule B		
T_1	T_2	T_3	T_1	T_2	T_3
read(Q)			read(Q)		
	write(Q)		write(Q)		
write(Q)		write(Q)			
				write(Q)	
					write(Q)

→ here, schedule A is view serializable as because it is view equivalent to serial schedule B.

Scanned with CamScanner

→ here write (W) of ~~the~~ T_2 and T_3 are called blind writes because it is performed without having performed read (R).

→ every conflict serializable schedule is also view serializable but not vice versa.

Testing for serializability

- construct a directed graph called precedence graph from S.
- precedence graph consists of a pair $G = (V, E)$ where
  - V is set of vertices, it consists of all transactions in schedule.
  - E is set of edges.
    - set of edges consists of all edges $T_i \to T_j$ for which one of 3 conditions hold.

①

②

③

Scanned with CamScanner

CamScanner

→ if an edge $T_i \rightarrow T_j$ exists in precedence graph then in any serial schedule S equivalent to S, T_i must appear before T_j.

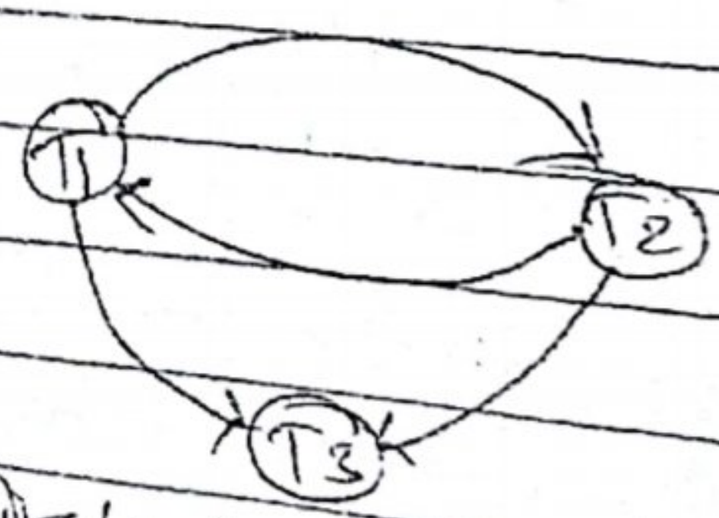
→ if the precedence graph has a cycle, then the schedule is not conflict serializable.

→ if the precedence graph has not a cycle then schedule is conflict seri

eg. 1)

T1	T2	T3
read (A)		
	write (A)	
write (A)		write (A)

Now,



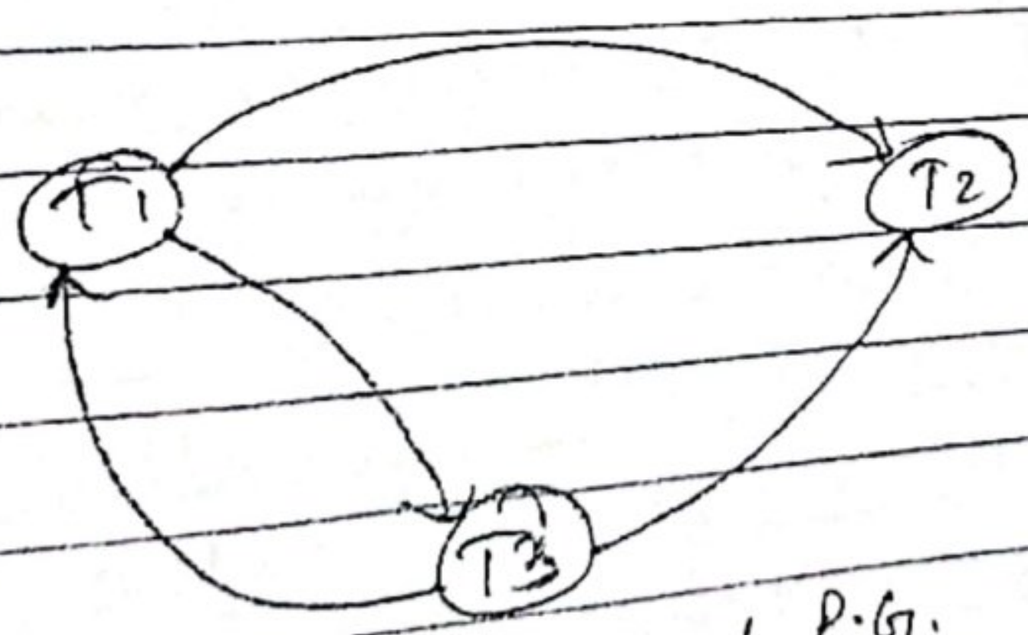
Equivalent precedence graph

Scanned with CamScanner

→ here graph contains a cycle, so above schedule is non conflict serializable.

eg. 2)

T1	T2	T3
write (A)		
	read (A)	
		write (B)
write (B)		
		write (B)
	write (A)	
		read (B)
	read (B)	



equivalent P.G.

- Here also, above schedule is non conflict serializable.

Scanned with CamScanner

Test for following.

1. $r_1(x)$, $w_2(x)$, $r_1(x)$.
2. $r_1(A)$, $r_3(A)$, $w_1(A)$, $r_2(A)$, $w_3(A)$
3. $r_3(A)$, $r_2(A)$, $w_3(A)$, $w_1(A)$, $w_1(A)$
4. $r_1(A)$, $w_3(A)$, $w_3(A)$, $w_1(A)$, $r_2(A)$.

Concurrency ~~symbol~~ control.

→ ensures that database ~~connections~~ transactions are performed concurrently without violating data integrity of database.

→ Different concurrency control schemes can be used to ensure the isolation property when multiple transactions are executed in parallel.

→ DBMS must guarentee that only serializable schedule are generated and that no effect of commited transaction is lost and no effect of aborted transaction remains in db.

→ Different methods like locking method, timestamp methods, etc. are used.

Scanned with CamScanner

Lock based protocol

→ a lock is a mechanism to control concurrent access to data item.

→ Data items can be locked in two modes.

a) Exclusive mode (X)

→ Data item can be both read as well as written.

→ X-lock is requested using lock-x instruction

b) Shared mode (S)

→ Data item can only be read.

→ S-lock is requested using lock-S instruction.

→ lock requests are made to concurrency control manager.

→ Transaction can only be proceed once request is granted.

→ A transaction may be granted a lock on an item if the requested lock is compatible with locks already held on item by other transactions.

	S	X
S	True	False
X	False	False

fig: lock compatibility matrix.

Scanned with CamScanner

- here, shared mode is compatible only with shared mode.
- if any transaction holds an exclusive lock on item, no other transaction may hold any lock on the item.

→ If a lock can't be granted, the requesting transaction is made to wait till all incompatible locks held by other transactions have been released the lock is then granted.

- A transaction can unlock a data item Q by instruction unlock (Q).
- transaction must hold a lock on data item as long as it access item.

e.g:

T: lock_s (A)
read (A)
unlock (A)
lock_s (B)
read (B)
unlock (B)
display (A+B)

- A locking protocol is set of rules followed by all transactions while requesting and releasing locks.
- locking protocol restricts the set of possible schedules.
- locking protocol must ensure serializability.

Scanned with CamScanner

CamScanner

e.g:

let, value of account A and B are 100 and 100 respectively.

T1 transaction transfer 20 from A to B.

T2 transaction display sum of A and B

Now, here,

T1 :	T2 :-
lock - X(A)	lock - S(A)
read (A)	read (A)
A = A - 20	unlock (A)
write (A)	lock - S (B)
unlock (A)	read (B)
lock - X (B)	unlock (B)
read (B)	display (A+B)
B = B + 20	
write (B)	
unlock - X(B)	

→ If T1 and T2 are executed serially, we get correct result.

→ but concurrent execution of T1 and T2 may result incorrect value.

like

Concurrency Control Manager

T1	T2
	grant - X(A, T1)
lock - X(A)	
read (A)	
A = A - 20	
write (A) //update	
unlock (A)	

Scanned with CamScanner

Date / /
Page No.

T1	T2	Concurrency Control Manager
	lock - S(A)	
		grant - S(A, T2)
	read (A)	
	unlock (A)	
	lock - S(B)	
		grant - S (B, T2)
	read (B)	
	unlock (B)	
	display (A+B)	
lock - X(B)		
		grant - X (B, T1)
read (B)		
B = B + 20		
write (B)		
unlock (B)		

Fig: Schedule X.

→ Assume that unlocking is delayed at the end of transaction. then ~~T1 becomes~~ ..

T1 becomes

T3 : lock - X(A)
read (A)
A = A - 20
write (A)
lock - X(B)
read (B)
B = B + 20

T2 becomes

T4 : lock - S(A)
read (A)
lock - S(B)
read (B)
display (A+B)
unlock (A)
unlock (B)

Scanned with CamScanner

CamScanner

write (B)
unlock (A)
unlock (B)

with T3 and T4, we can't get schedule x as above and for any other schedule, q produce correct value IRE:

T1	T2	~~T3~~	~~T4~~
lock- x(A)			
read (A)			~~read (B)~~
A = A - 20			~~display (A+B)~~
write (A)			~~unlock (A)~~
lock - X(B)			~~unlock (B)~~
read (B)			
B = B + 20			
write (B)			
unlock (A)			
	lock - S(A)		
	read (A)		
unlock (B)			
	lock - S(B)		
	read (B)		
	display (A+B)		
	unlock (A)		
	unlock (B)		

Scanned with CamScanner

Date / /
Page No

Pitfalls of lock based protocol

- consider partial schedule as

T3	T4
lock - X(B)	
read (B)	
B = B - 50	
write (B)	
	lock - S(A)
	read (A)
	lock - S(B)
lock - X(A)	

fig: Scheduler

- here, neither T3 nor T4 can make progress executing lock - S(B) cause T4 to wait for T3 to release its lock on B.
  while executing lock - X(A) cause T3 to wait for T4 to release its lock on A.
- Such situation is called deadlock.
- to handle deadlock one of transactions T3 or T4 must be rolled back and its lock must be released.
  if we do not use locking, we may get inconsistent state.
  Similarly, if we do not unlock data item before requesting lock on another item,

Scanned with CamScanner

deadlock occurs.
However, deadlocks are necessary evil, more preferable than inconsistent states.
The potential for deadlock exists in most locking protocol.
Starvation is also possible,
→ if a transaction may be waiting for an X-lock on item while, a sequence of other transactions request S-lock and are ~~guarantee~~ granted on same item.
→ Same transaction is repeatedly rolled back due to deadlock.

Two phase locking protocol

→ ensures serializability.
→ here, transaction issue lock and unlock requests in two phases

i) Growing phase :-
→ transaction may obtain locks.
→ transaction may not release locks.

ii) Shrinking phase:
→ transaction may release locks.
→ transaction may not obtain locks.

→ Initially, transaction is in growing phase
→ after transaction release a lock, it enters shrinking phase.

Scanned with CamScanner

e.g. T3 and T4 are two phase but not T1 and T2.

→ two phase locking does not ensure freedom from deadlocks.

→ cascading roll back is possible under two phase locking.

the phenomenon in which a single transaction failure leads to series of transaction rollbacks is called cascading roll back.

e.g:

T1	T2	T3
read (A)		
read (B)		
write (A)		
	read(A)	
	write (A)	
		read(A)

↳ here, T2 is dependent on T1, and T3 is dependent on T2.

↳ if T1 failed, T1 must be rolled back. Similarly, as T2 is dependent on T1, T2 also be rolled back. again as T3 is dependent on T2. T3 also must be rolled back.

Scanned with CamScanner

T_a	T_b	T_c
lock-X(A)		
read(A)		
lock-S(B)		
read(B)		
write(A)		
unlock(A)		
	lock-X(A)	
	read(A)	
	write(A)	
	unlock(A)	
		lock-S(A)
		read(A)

fig: partial schedule X under two phase locking

- in above schedule X,
- if T_a fails after read(A) of T_c then there must be cascading rollback of T_b & T_c

Strict two phase locking protocol

→ here, transaction must hold all its exclusive mode locks till it commits.

→ this prevents any other transaction from reading data.

→ No cascading rollback.

Scanned with CamScanner

Date / /
Page No

2) Rigorous two phase locking
→ here, all locks (S and X) are held till commits.
→ No cascading roll back.
→ here, transaction can be serialized in order in which they commit.

Deadlock Handling

↳ System is deadlocked if there is set of transactions such that every transaction in the set is waiting for another transaction in the set.

↳ Let,

T1 :	T2 :
write (X)	write (Y)
write (Y)	write (X)

↳ here, Schedule with deadlock

T1	T2
lock - X (X)	
write (X)	
	lock - X (Y)
	write (Y)
	write (X)
write (Y)	

↳ to deal with deadlock, we can use
1) Deadlock prevention protocol

Scanned with CamScanner

सुगन स्टेसनरी सप्लायर्स एण्ड फोटोकपी सर्भिस
ललितपुर १८४१५९९५९२
NCIT College

- ensure that system will never enter deadlock state.

2) Deadlock detection and recovery scheme
- try to recover system once it entered deadlock state.

→ Both methods may result in transaction rollback.
→ prevention is used if probability of system entering deadlock state is relatively high.
→ otherwise, detection and recovery are more efficient.

Deadlock prevention

↳ ~~dead~~ deadlock prevention protocol ensure that the system will never enter into deadlock state.

↳ Some prevention strategies.

i) require that each transaction locks all data item before it begins execution.

ii) impose partial ordering of all data item.
- require that a transaction can lock data items only in order specified by partial order.

iii) Timeout based schems.
- a transaction waits for a lock only for specified amount of time.

सुगन स्टेसनरी सप्लायर्स एण्ड फोटोकपी सर्भिस
बालकुमारी, ललितपुर १८४१५९९५९२
NCIT College

Scanned with CamScanner

Date / /
Page No.

- after the wait time is out, transaction is roll back.
→ Simple to implement but starvation is possible.
→ following schemes use transaction timestamps for deadlock prevention.

i) Wait - die scheme (Non preemptive)
→ older transaction may wait for younger one to release data item.
→ younger transactions never wait for older ones; they are rolled back instead.
→ a transaction may die several times before acquiring needed data item.

2) ~~Deadlo~~ Wound - wait Scheme (pre emptive)
→ older transaction wounds (force rollback) Younger transaction instead of waiting for it.
→ younger transaction may wait for older ones.
→ may be fewer rollbacks than wait die scheme.

↳ in both schemes, rollback transaction is restarted with its original timestamp.
↳ older transaction thus have precedence over newer ones in those schemes and starvation is avoided.

Scanned with CamScanner

Deadlock Detection and Recovery

is used if no protocol is used to ensure deadlock freedom.

some algo. are used to check whether deadlock occurs, if deadlock occurs, then must be recovered.

Deadlock detection:

deadlock can be described as wait for graph which consists of pair $G = (V, E)$

→ V is set of vertices (transaction)

→ E is set of edges, each edge is ordered pair $T_p \rightarrow T_j$

when transaction T_i request a data item held by T_j then $T_i \rightarrow T_j$ is inserted in wait for graph.

this edge is removed only when T_i no longer holding data item needed by T_j.

the system is in deadlock state if and only if wait for graph has a cycle.

the system invokes deadlock detection algorithm periodically to look for cycle.

Scanned with CamScanner

CamScanner

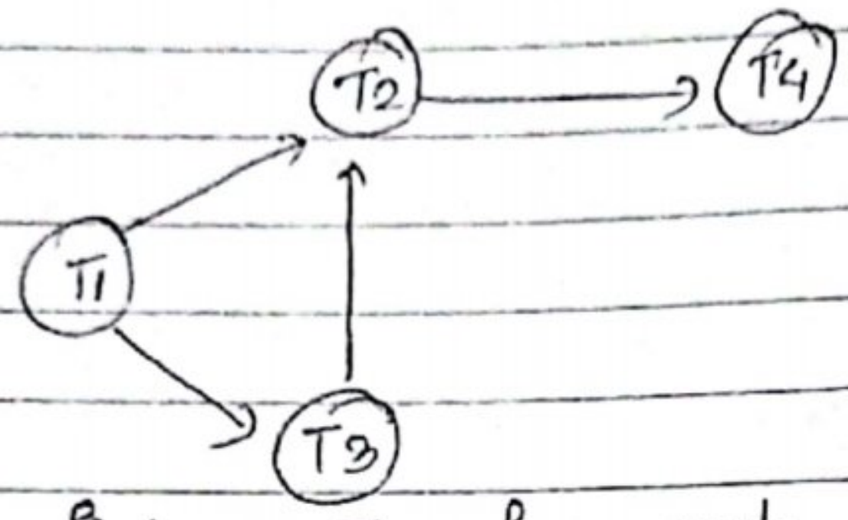


fig: wait for graph without cycle.

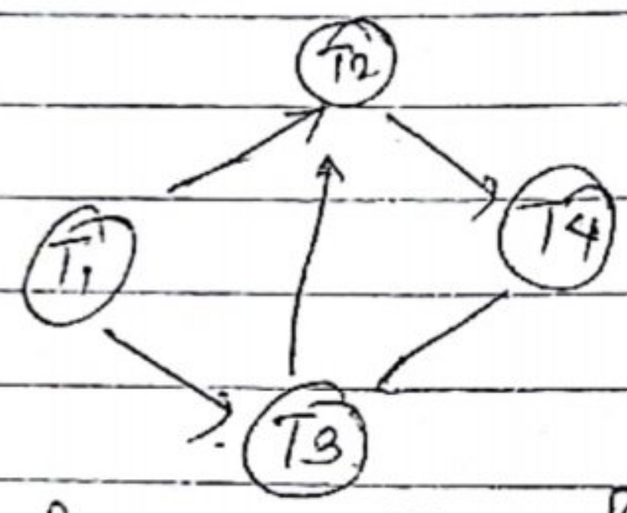


fig: wait for graph with cycle.

Deadlock Recovery

when deadlock is detected

→ Some transactions will have to roll back to break deadlock (i.e. victim selection)

→ Rollback :- determine how far to rollback transaction

a) total rollback

- abort transaction and then restart it.

b) Partial rollback

- rollback transaction only as far as necessary to break deadlock.
- is more effective.

Scanned with CamScanner

- Starvation happens if same transaction is always chosen as victim.
- must ensure that transaction can be picked as a victim only a small no. of times.

CHAPTER 9: CRASH RECOVERY

DBMS is highly complex system with hundreds of transactions being executed every second. availability of DBMS depends on its complex architecture and underlying h/w or system s/w. if it fails or crashes amid transactions being executed, it is expected that the system would follow some sort of algo. or techniques to recover from crashes or failures.

There may be several causes of failure, including power failure, s/w bugs etc.

Types of failure

1) Transaction Failure
   → logical errors – transaction can't complete due to some internal condition.

Scanned with CamScanner

→ System errors :- The database system must terminate an active transaction due to an error condition (like deadlock).

System Crash

→ a power failure or other h/w, s/w failure cause the system to crash.

→ fail stop assumption :-

- non volatile storage contents are assumed not to be corrupted by system crash.
- DB system have numerous integrity checks to prevent corruption of disk data.

Disk Failure

- a head crash or similar disk failure destroys all part of disk storage.
- destruction is assumed to be detectable, disk drives use check sums to detect failure.
- Multiple copies or archival tapes are solution.

Scanned with CamScanner

Recovery and atomicity

→ modifying the database without ensuring that the transaction will commit may leave the database in an inconsistent state.

→ Consider ~~ta~~ transaction T_i that transfers Rs ~~B~~.. 100 from account A to B here goal is either to perform all database modification made by T_1 or none at all.

→ If T_i performed multiple database modification several output operations may be required for T_i and a failure may occur after one of these modification have been made but before all of them are made.

→ To ensure atomicity despite failures, we first o/p information describing the modification to stable storage without modifying database itself.

→ two approaches

i) log based recovery

ii) shadow paging

here, assume, that transaction run serially.

Scanned with CamScanner

CamScanner

Log based Recovery

The log is a sequence of log records and maintains a record of update activities on the database, which is kept in a stable storage.

It works as follows:

→ the log file is kept on stable storage media.

→ when a transaction enters the system and starts execution, it write a log about it.

$< T_i$ start $>$.

→ when the transaction modifies an item x, it writes log as follows:

$< T_i , x , V_1 , V_2 >$

i.e. T_i has changed value of x from V_1

→ when the transaction finishes.

- Two approaches using log

i) Deffered database modification

ii) Immediate " "

Deffered database Modification:

it ensures transaction atomicity by recording all db modification in the log. But deferring execution of all write operations of a transaction until the transaction partially commits.

Scanned with CamScanner

→ When transaction partially commits, the information on the log associated with transaction is used in executing deferred values.

→ If the system crashes before transaction completes its execution, then the information on the log simply ignored.

→ Transaction starts by writing < Ti, start > record to log.

→ A write (X) operation results in a log record < Ti, X, V > being written where V is new value of X. Here, old value is not needed

→ Deffered update ~~statement~~ steps

i) 'write' operation is not performed on X at this time but is deffered.

ii) when Ti partially commits < Ti, commit > is written to log.

iii) finally, by records are read and used to actually execute previously 'deffered writes.

→ <u>REDO only scheme</u>:-

↳ during recovery after crash, a transaction needs to be redone if and only if both < Ti, start > and < Ti, commit > are there in log.

↳ redoing a transaction Ti (i.e. redo(Ti)) sets the value.

Scanned with CamScanner

CamScanner

→ ~~Ex~~

→ Crashes can occur while
- the transaction is executing the original updates or while recovery action is being taken.

e.g:
transaction T_0 and T_1 such that T_0 executes before T_1.
Let $A = 1000$, $B = 1000$, $C = 500$

T_0	T_1
read (A)	read (C)
A = A − 50	C = C − 100
write (A)	write (C)
read (B)	
B = B + 50	
write (B)	

log	Database
<T_0, start>	
<T_0, A, 950>	
<T_0, B, 1050>	
<T_0, Commit>	
	A = 950, B = 1050 } // write operations executed
<T_1, Start>	
<T_1, C, 400>	
<T_1, Commit>	
	C = 400

Scanned with CamScanner

Below is the log as it appears at three instances of time.

(a)	(b)	(c)
<T0, Start>	<T0, Start>	<T0, Start>
<T0, A, 950>	<T0, A, 950>	<T0, A, 950>
<T0, B, 1050>	<T0, B, 1050>	<T0, B, 1050>
	<T0, commit>	<T0, commit>
	<T1, Start>	<T1, Start>
	<T1, C, 400>	<T1, C, 400>
		<T1, commit>

If log on stable storage at time of crash is as in case

(a) → No redo action need to be taken

(b) → redo(T_0) must be performed

(c) → {redo(T_0), redo(T_1)} must be performed

Immediate Database Modification:

allows database update of an uncommitted transaction to be made as the writes are issued.

Since, undoing may need, update logs must have new and old values.

update log record must be written before database

item is written.

→ output of updated blocks can take place at any time before or after transaction commit.

→ order in which blocks are output can be different from order in which they are written in buffer.

→ Example as before.

then,

log	Database
<To, start>	
<To, A, 1000, 950>	
<To, B, 1000, 1050>	
	A = 950
	B = 1050
<To, commit>	
<T1, start>	
<T1, C, 500, 400>	
	C = 400
<T1, commit>	

→ here recovery procedure has two operations

i) Undo (Ti) :- restores value of all data items updated by Ti to old values, going backward from last log record for Ti.

ii) Redo (Ti) :- Sets the value of all data item updated by Ti to new values, going forward from first log record from Ti.

Scanned with CamScanner

→ Both operations must be idempotent.
undo (T_i) = ~~undo(T_i)~~ = undo(undo (T_i)) = undo (undo(T_i))

i.e. even if the operation is executed multiple times the effect is the same as if it is executed once.

— It is needed since operations may get re-executed during recovery.

→ While recovering after failure, Transaction T_i needs to be redone if the log contains ~~both~~ both the record <T_i, start> and <T_i, commit>

→ Undo operations are performed first then only redo.

→ Before, we show log as it appears at three instance of time.

(a)	(b)	(c)
<T_0, start>	<T_0, start>	<T_0, start>
<T_0, A, 1000, 950>	<T_0, A, 1000, 950>	— —
<T_0, B, 1000, 1050>	<T_0, B, 1000, 1050>	— —
	<T_0, commit>	— —
	<T_1, start>	— —
	<T_1, C, 500, 400>	— —
		<T_1, Commit>

Scanned with CamScanner

- recovery actions in each case of above are

(a) :- Undo(T_0)

(a) :- undo (T_1) then redo(T_0)

(a) :- redo(T_0) and redo(T_1)

Checkpoints

→ when more than one transactions are being executed in Parallel, the logs are interleaved.

→ at the time of recovery it would become hard for recovery algo. to backtrack all logs and then start recovering.

→ To ease this situation, most modern DBMS use concept of checkpoints.

→ problems in recovery procedure as discussed earlier

i) searching entire log is time consuming.

ii) might necessarily redo transactions which have already output their updates to database.

→ Streamline recovery procedure by periodically performing checkpointing

i) output all log records currently residing in main memory to stable storage.

ii) output all modified buffer blocks to disk.

iii) write a log record < checkpoint > on stable storage.

Scanned with CamScanner

→ During checkpointing, other transactions cannot be processed.

→ During recovery we need to consider only the most recent transaction T_i that started before checkpoint and transactions that started after checkpoint.

→ Scan backward from end of log to find the most recent <checkpoint> record.

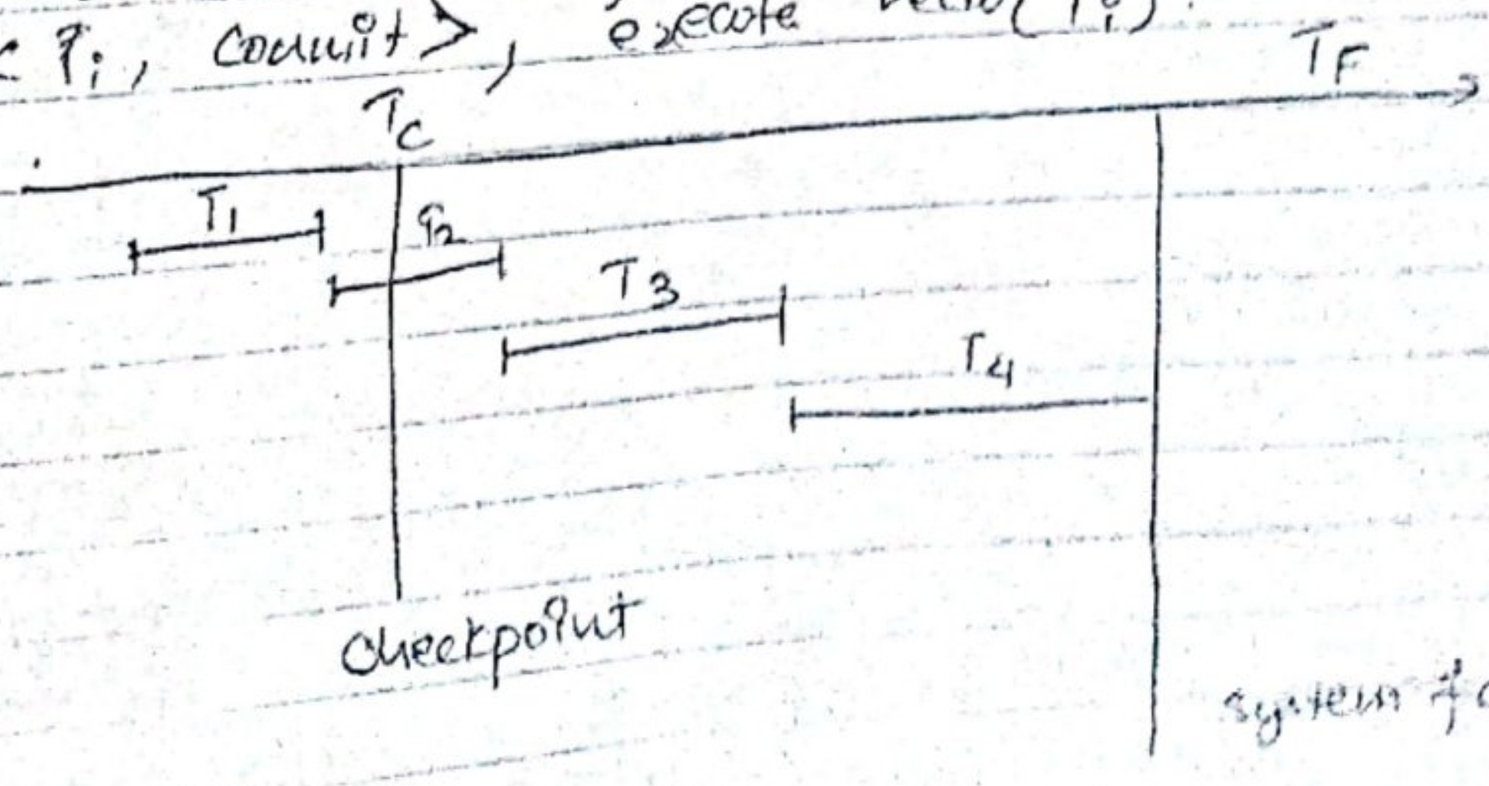
→ Continue scanning backward till a record <T_i, start> is found.

→ Need only consider the part of log following above 'start' record.

- earlier part of log can be ignored during recovery and can be erased whenever desired.

→ for all transactions (starting from T_i or later) with no <T_i, commit> execute undo(T_i).

→ Scanning forward in the log, for all transactions starting from T_i or later with <T_i, commit>, execute redo(T_i).



Scanned with CamScanner

here,

- Ignore T_1
- undo T_4
- Redo T_2, T_3

Shadow Paging

Is an alternative to log based recovery.
is useful if transaction execute serially.
idea is maintain two page tables during the lifetime of transaction, on is current page table, another is shadow page table.
Store shadow page table in non-volatile storage, such that state of the db prior to transaction execution
- here, shadow page table is never modified during execution.

To start with, both page tables are identical.
- only current page table is used for data item access during execution of transaction.

whenever any page is about to be written for first time
- a copy of this page is made onto an unused page.
- the current page table is then made to point to copy.
- the update is performed on the copy.

Scanned with CamScanner

To commit transaction.
flush all modified pages in main memory to disk.
output current pagetable to disk.
Make current pagetable the new shadow page table as follows.
keep a pointer to shadow page table at a fixed known location on disk.
simply update pointer to point current page table on disk.
once pointer to shadow page table has ~~to~~ been written, transaction is committed.
No recovery is need after the crash, new transaction can start right away using shadow page table.
→ Pages not pointed to from current / shadow page table should be free.

Shadow Paging

→ Advantages of shadow paging over log based
→ no overhead of writing log records.
→ recovery is trivial.

→ Disadvantages:
→ Copying the entire page is very expensive.
→ Commit overhead is high.
→ Data gets fragmented (related pages get

Scanned with CamScanner

→ To commit transaction.

i) flush all modified pages in main memory to disk.

ii) output current pagetable to disk.

iii) Make current pagetable the new shadow page table as follows.

a) keep a pointer to shadow page table at a fixed known location on disk.

b) Simply update pointer to point current page table on disk.

→ once pointer to shadow page table has been written, transaction is committed.

→ No recovery is need after the crash, new transaction can start right away using shadow page table.

→ Pages not pointed to from current / shadow page table should be free.

Shadow Paging

→ Advantages of shadow paging over log based
→ no overhead of writing log records.
→ recovery is trivial

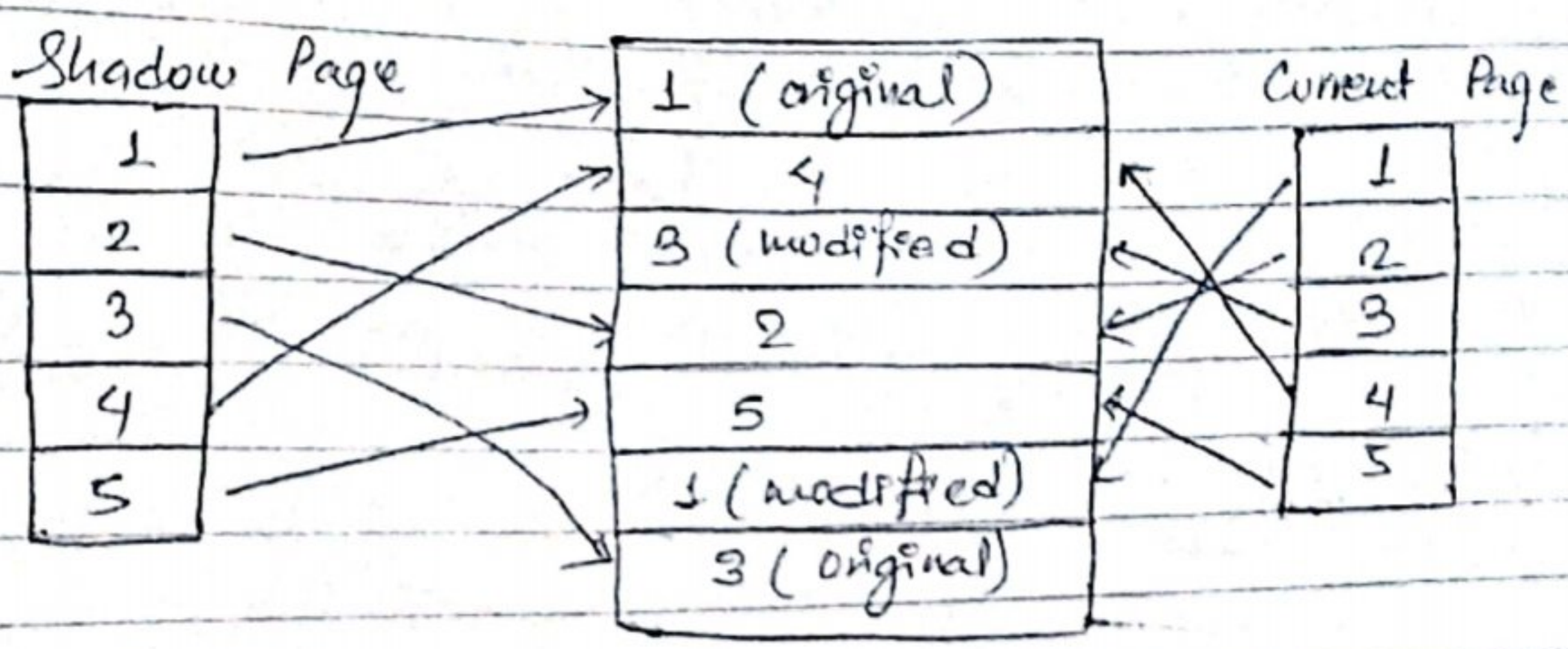
→ Disadvantages:
→ Copying the entire page is very expensive.
→ Commit overhead is high.
→ Data gets fragmented (related pages get

Scanned with CamScanner

separated on disk).

→ after every transaction, old version of modified data need to be garbage collected

→ hard to extend algo to allow transaction run concurrently.



Data Backup and Recovery

→ is to protect data.

→ backup files can protect against accidental loss of data, database corruption and other failures.

→ with backup and recovery plan, we can easily recover out important data from any type of disasters.

→ To create backup plan, deal with following

i) how important is data on system?

ii) what type of information does data contain?

iii) How often does data change?
iv) How quickly data need to recover?
v) Have the equipment to perform backup?
vi) Who will be responsible for backup and recovery plan?
vii) What is best time to schedule backup?
viii) Is there need of storing backup off site?

Backup plans depends on factors like
a) Capacity
b) Reliability
c) Extensibility
d) Speed
e) cost

common backup solution are
a) Tape drive
b) Disk drive
c) Magnetic optical drive
d) removable disk etc.

Remote Backup or Online Backup

Here, files, folders or entire contents of hard drive are regularly backed up on a remote server with n/w connection.

Scanned with CamScanner

CamScanner

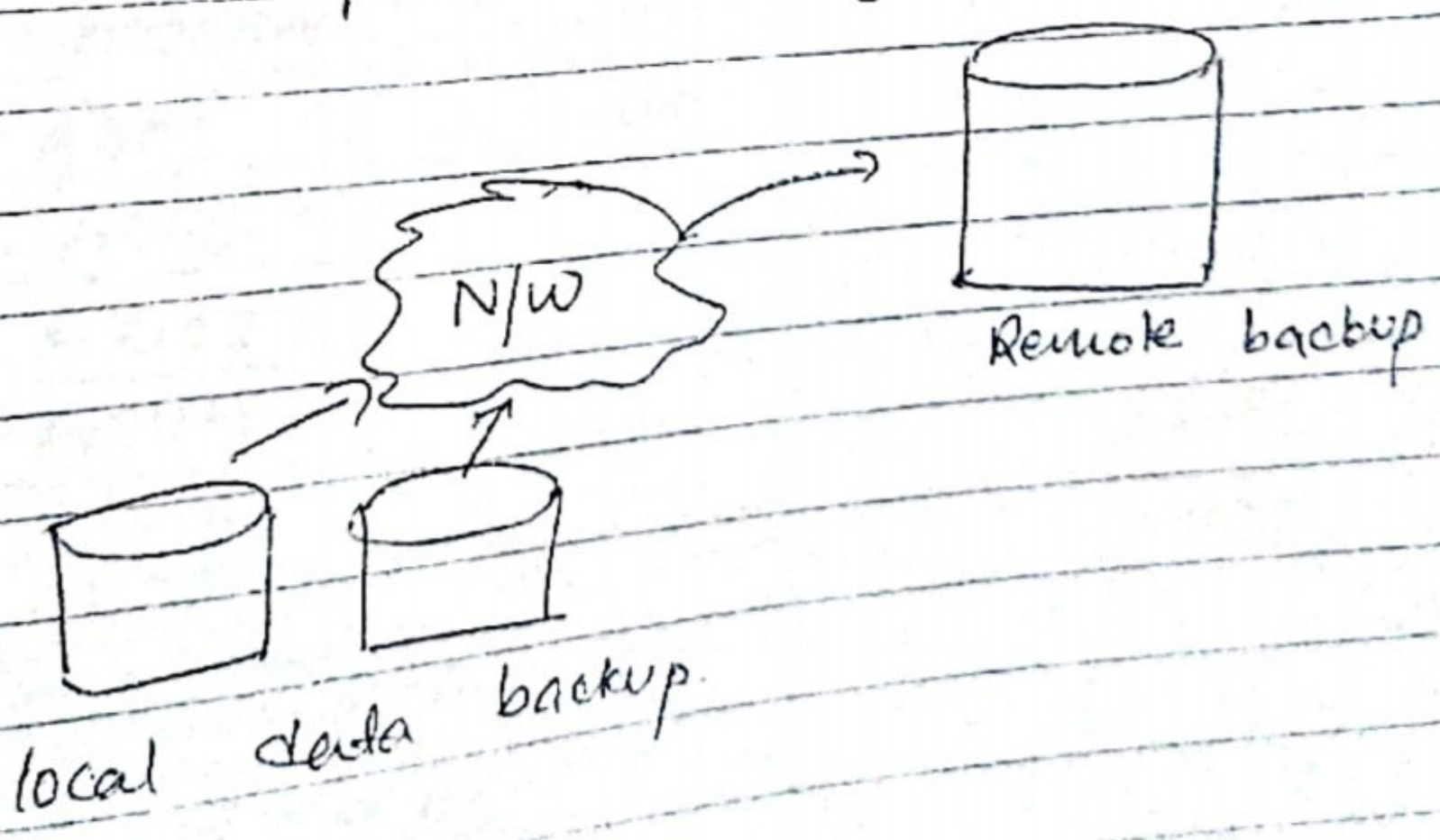
→ here, risk of catastrophic data loss as a result of fire, theft, file corruption or other disaster is eliminated.

→ encryption and password protection system helps to ensure privacy and security.

→ For large organization and for practically valuable data, online backup strategy is wise investment.

→ It is backup performed on data even though it is actively accessible to users and may currently be in state of being updated.

→ Cloud computing, Remote login method are for remote backup and recovery.



Scanned with CamScanner

Chapter 5

Types of functional dependency.

1) **Full functional dependency.**

$X \rightarrow Y$ is a full function dependencies if the removal of any attribute 'A' from X removes the dependency.

↳ A full functional dependency occurs when it is ~~are~~ already functional dependency and the set of attributes on the left side of functional dependency cannot be reduced anymore.

e.g.

orderno.	lineno	qty	price
A001	001	10	200
A002	001	20	400
A002	002	30	800
A004	001	15	300

Here,
{orderno, lineno} → qty
{orderno, lineno} → price
} are full functional dependency.

2) **Partial functional dependency:**

→ A functional dependency $X \rightarrow Y$ is partial dependency if some attribute 'A' belongs to X can be removed from X and the dependency still holds.

Scanned with CamScanner

CamScanner

e.g:

{eid, phone} ⟶ name

here,

removing phone.

eid ⟶ name

Transitive Dependency

Occurs when there is an indirect relationship that causes a functional dependency.

if $X \rightarrow Y$ and $Y \rightarrow Z$ then transitive dependency exists is

$X \rightarrow Z$.

~~Trivial~~ Trivial Dependency

⟶ Occurs when functional dependency of an attribute is described on a collection of attributes that includes the original attribute.

Functional dependency is trivial if RHS is subset of LHS,

e.g.

{name, SSN} ⟶ SSN

Non Trivial Dependency

are the one that is not trivial.

e.g:

Scanned with CamScanner

CamScanner

{S#, P#} → SH — Trivial
{S#, P#} → {S#, qty} — Non trivial

Axioms or Rules of inference for functional dependency.

Reflexivity Rule:
if Y is subset of X (Y ⊆ X) then X → Y holds.

~~Trans~~ Augmentation Rule:
if X → Y holds and Z is set of attributes then ZX → ZY holds.

Transitivity Rule:
if X → Y and Y → Z holds then X → Z holds.

Union Rule
if X → Y and X → Z holds then X → YZ holds.

Decomposition Rule:
if X → YZ holds then X → Y and X → Z holds

Pseudo transitivity Rule:
if X → Y and ZY → P holds then XZ → P holds.

Scanned with CamScanner

CamScanner

Date / /

Page No.

Self determination

$A \rightarrow A$

e.g:

$R = (A, B, C, G, H, I)$

set of F.D $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$

Then we can have

$A \rightarrow H$; with transitivity Rule

$CG \rightarrow HI$; " Union "

$AG \rightarrow I$; " pseudotransivity rule -

e.g:

$F = \{A \rightarrow B, C \rightarrow X, BX \rightarrow Z\}$

~~Use~~ prove or disprove

$AC \rightarrow Z$

$\Rightarrow$ $C \rightarrow X$, $BX \rightarrow Z$ (Pseudo transitivity)

$A \rightarrow B$ and $BC \rightarrow Z$

$AC \rightarrow Z$

e.g. $F = \{A \rightarrow B, C \rightarrow D, C \subseteq B\}$

prove or disprove $A \rightarrow C$.

$B \rightarrow C$

$\Rightarrow A \rightarrow C$.

Scanned with CamScanner

CamScanner

Closure of a set of functional dependencies

The set of functional dependencies that is logically implied by F is called closure of F and written as F^+.

The closure of F denoted by F^+ is set of all FD's entailed by F.

$$F^+ = \{X \to Y \mid F \models X \to Y$$

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

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

list several members of F^+.

Sol^n

i) $A \to B$, $A \to C$

$\therefore$, $A \to BC$

ii) $A \to BC$, $BC \to D$

$A \to D$

iii) $A \to C$, $BC \to D$

$\therefore$, $AB \to D$

$F^+ = \{A \to BC, A \to D, AB \to D\}$

e.g.:

$R = \{A, B, C, G, H, I\}$

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

list element of F^+.

Scanned with CamScanner

CamScanner

Date / /

Page No.

i) $A \to B$, $A \to C$.

$\Rightarrow A \to BC$

ii)

iii) $R = (A, B, C, D, E, F)$

$F = \{A \to BC, B \to E, CD \to EF\}$

list F^+ & prove $AD \to F$

Scanned with CamScanner

CamScanner

Closure of attibute set

Given a set of attributes $A_1 \ldots A_n$ the closure $\{A_1 \ldots A_n\}^+$ is set of attibutes B such that

$$A_1 \ldots A_n \rightarrow B$$

An algorithm to compute α^+

```
result = α,
while (changes to result) : do
  for each Functional dependency β → γ
  begin
      if β ⊆ result then
          result = result ∪ γ
  end
```

e.g:

$R = (A, B, C, G, H, I)$

$F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$

Compute $(AG)^+$

Sol^n

result = AG

for $A \rightarrow B$

$A \subseteq AG$ true ; result = {ABG}

for $A \rightarrow C$

$A \subseteq ABG$, true ; result = {ABCG}

Scanned with CamScanner

CamScanner

Date / /

Page No.

for $CG \to H$

$CG \subseteq ABCG$ true ; result $= \{ABCGH\}$

for $CG \to I$

$CG \subseteq ABCGH$ true ; $\therefore$ result $= \{ABCGHI\}$

for $B \to H$

$B \subseteq ABCGHI$ true ; $\therefore$ result $= \{ABCGHI\}$

$\therefore (AG)^+ = ABCGHI$

eg. 2.

$R = (A, B, C, D, E, F)$

$F = \{A \to BC, E \to CF, B \to E, CD \to EF\}$

$\{AB\}^+ = ?$

Scanned with CamScanner

CamScanner

Imp

Normalization

- Database normalization is a technique of organizing data in the database. It is the process of removing data redundancy and undesirable characteristics like insertion, update and deletion anomalies.
- Mainly, two purposes of normalization are
  i) Eliminating redundant data
  ii) Ensuring data dependencies make sense.
- It involves dividing table into two or more tables and defining relationship between the tables.
- Normal forms represents guidelines for design
- Different types of normal forms are 1NF, 2NF, 3NF, BCNF, 4NF etc.

Unnormalized Form.

- It contains one or more repeating groups or each row may contain multiple set of values for some columns.
- Multiple values in a single row are called no atomic values.

e.g:

CID	CName	Caddress	Cphone
C101	Ram	Kathmandu	98 03346813, 5 3346814
C102	Shyam	Patan	9851010257
C103	Hari	Pokhara	55 94832, 9803343 434, 9813003274.

Fig ①, UNF

Scanned with CamScanner

Date / /

Page No.

First Normal Form (1NF)

A relation is in 1NF if and only if all columns are atomic i.e. no repeating items in columns

↳ here, fig. 1 will become.

CID	Cname	Caddress	Cphone
C101	Ram	kathmandu	9803346813
C101	Ram	kathmandu	55346814
C102	Shyam	patan	9851010257
C103	Hari	pokhara	9803343437
C103	Hari	pokhara	9813003244
C103	Hari	pokhara	55346832
		~~pokhara~~	

fig. 1NF

↳ There shouldn't be any repeating columns.

id	name	email	child1	child2	child3
1	Ram	---	Hari	Sita	Gita
2	Shyam	---	Shyam	John	rita.

Scanned with CamScanner

CamScanner

Second Normal Form (2NF)

For a relation to be in 2NF,

a) it must be in 1NF

b) all attributes depend on full primary key.

eg. table person project

Person ID	Proj ID	PersonName	Project Code	ProjName	Mob.
1	1	Ram	Pro 1	DB	981...
2	1	Shyam	Pro 1	DB	980...
1	2	Ram	Pro 2	Web	981...
2	2	Shyam	Pro 2	Web	980...

1. Person

Person ID	Project Name	Phone
1	Ram	981...
2	Shyam	980...

2. Project

Proj ID	Proj Code	Proj Name
1	Pro 1	DB
2	Pro 2	Web

3. PerProj

Person ID	Project ID
1	1
2	1
1	2
2	2

Scanned with CamScanner

CamScanner

Date / /
Page No.

Third Normal Form (3NF)

- For a relation to be in 3NF,
  a) it must be in 2NF,
  b) There is no transitive functional dependency i.e. there shouldn't be case that non prime attribute is determined by another non-prime attribute.

eg. Student info

StID	StName	DOB	City	Zip
1	Ram	1984/1/1	Ktm	01
2	Shyam	1985/2/1	Pkr	05
3	Hari	1989/2/1	Pkr	05
4	Gita	1990/5/5	Ktm	01

1. Student

St.ID	St.Name	DOB	Zip
1	Ram	1984/1/1	01
2	Shyam	1985/2/1	05
3	Hari	1989/2/1	05
4	Gita	1990/5/5	01

2. Address

Zip	City
01	Kathmandu
05	Pokhara

Scanned with CamScanner

CamScanner

Date / /
Page No.

BCNF (Boyce Codd Normal Form)

For a relation to be in BCNF
- it must be in 3NF
- Every determinant in that table is superkey.

if table contain only one candidate key, 3NF and BCNF are equivalent.

eg. Student

Student	Course	Teacher
Ram	DB	John
Ram	PT	Tedd
Shyam	DB	John
Shyam	PT	Tedd

where, Key = {student, course}
{student, course} → Teacher
Teacher → course
but, Teacher is not superkey, but determine course. So, not in BCNF.

Table1

Teacher	Course
~~Fe~~ John	DB
Tedd	PT

Table 2

Student	Course
Ram	DB
Ram	PT
Ram	DB
Shyam	PT
Shyam	

Scanned with CamScanner

CamScanner

Date / /

Page No.

Table3

Course

DB

PT

Denomalization

- It is process of attempting to optimise the performance of a database by adding redundant data or by grouping data.
- In relational database, denormalization is an approach to speed up read performance in which the administrator selectively adds back specific instances of redundant data after the data structure has been normalized.
- It is needed when multiple joins in same query can have negative impact on performance.
- Denormalization should take place after satisfactory level of normalization has taken and that any required constraints or rules have been created to deal with inherent anamolies in design.

Scanned with CamScanner

CamScanner

Date / /
Page No.

Decomposition

- refers to breaking down of one table into multiple tables.
- desirable properties of decomposition are:
  i) attribute preparation
  ii) Lossless description
  iii) Dependency prevention
  iv) Lack of redundancy

Assertion

- A condition that we want database to always follow.
- It means to specify integrity constraints.
- An assertion in SQL takes the form:

CREATE ASSERTION <assertion_name> CHECK <predicate>

- When an assertion is made, the system tests it for validity and ~~test~~ it again on every update that may ~~update~~ violate assertion.
- Assertion should be used with great care as testing may cause significant amount of overhead.

eg. the sum of all loan amounts for each branch must be less than sum of all account balances at branch.

```
CREATE assertion sum-constraints check (not exists (select * from
branch where (select sum(amount) from loan where loan.branchname
= branch.branchname) >= (select sum(balance) from account where
account.branchname = branch.branchname
```

Scanned with CamScanner

Date / /
Page No.

Trigger

- It is a special kind of stored procedure that executes in response to certain action on table like insertion, deletion or update of data.
- It is db object bound to table and is executed automatically.
- We cannot explicitly invoke triggers.

Multivalued Dependency

- It is a functional dependency where determinant can determine more than one value.
- For MVD,
  i) There must be 3 attributes in relation say. A, B, C.
  ii) Given A one can determine multiple values of B and also one can determine multiple values of C.
  iii) B & C are independent of one another.

eg.

SID	Major	Activities
1	DB	Football
2	DB	Basketball
2	Math	Football
1	Math	Basketball

Here, SID → Major, SID → Activities.

Scanned with CamScanner

CHAPTER 6: Security

Date / /
Page No.

→ unauthorized access or manipulation of database creates problem for organization.

→ Security refers to protection of data against unauthorized disclosure, alteration or destruction

→ Main objectives of designing secured database system are integrity, availability and secrecy.

↳ Database security is about controlling access to information. i.e. some information be available freely and other informations be available to certain authorized people or groups.

↳ Database security system stores authorization rules and enforces them for database access.

↳ Database security can be

1) Physical security

It refers to security of hardware and protection of sites where computer resides.

2) Logical Security

It refers to software safeguards for organization, systems, including user id diffication, password access, access rights and authority levels.

Scanned with CamScanner

CamScanner

Date / /
Page No.

Security Violation:

↳ Among the forms of malicious access are unauthorized ~~the~~ reading of data, unauthorized modification of data and unauthorized destruction of data.

Database Security levels

↳ To ensure database security, security at different levels must be maintained.

1) **Database System:**
Database system have to ensure that authorization, restrictions are not violation.

2) **Operating System:**
Weakness in operating system security is concerned with unauthorized ~~exce~~ access to database

3) **Network**
Software ~~to~~ level security within network software is important.

4) **Physical**
Sites with computer systems must be physically secure against armed entry by intruders.

Scanned with CamScanner

CamScanner

Date / /
Page No.

5) Human

Users must be authorized carefully.

↳ Security is protecting data against unauthorized users.

↳ Integrity is protecting data against authorized users.

Authorization :

↳ Authorization is a security mechanism used to determine user/client privledge or access level related to system resources.

↳ During authorization system verifies authenticated users access role and either grant or revoke resource access.

↳ Authorization includes

i) Permitting only certain users to access, process or alter data.

ii) Applying different limitations on users access or actions. Here limitations placed on users can apply to object such as tables, rows etc.

↳ Authorization on data include

i) Authorization to read data

ii) Authorization to insert new data

iii) Authorization to update data.

iv) Authorization to delete data.

↳ Each of these type of authorization is called

Scanned with CamScanner

CamScanner

Date / /
Page No.

privilege.

- Users are authorized all or none or combination of these types of privileges on specified parts of database such as relation or views
- In addition to authorization data, users may be granted database schema, allowing them to create modify or drop relations.
- The ultimate form of authority is that given to a DBA (Database Administrator).
- DBA may authorize new users, restructured the database etc.

Granting and Revoking of privilege

- SQL standard include the privileges select, insert, update, delete.
- all privileges can be used for all allowable privileges.
- a user who creates a new relation is given all privileges automatically.

- Grant statement:

```
grant < privilege list >
on < relation or view >
to < user | role list >
```

e.g:

1)
```
grant select
ON employee
to Ram, Hari,
```

Scanned with CamScanner

to read tuples in relation select authorization is required.

2) i) grant update on employee to Ram, Hari;

ii) grant update (salary) on employee to Ram, Hari;

* to update any tuple in relation 'update' authorization is used.
* 'update' authorization may be given either on all attributes of relation or only some.

To insert tuples into relation, 'insert' authorization is used.

'insert' privilege may also specify ~~only those~~ a list of attributes.

any inserts to the relation must specify only these attributes.

The system either gives default value or NULL for remaining attributes.

3) i) grant insert on employee to Ram, Hari;

ii) grant insert (name, address) on employee to Ram, Hari;

To delete tuples from relation 'delete' authorization is used.

Scanned with CamScanner

CamScanner

Date / /
Page No.

eg 4) grant delete
on employee
to Ram, Hari;

↳ privileges granted to 'public' are implicitly granted to all current and future users.

↳ To revoke an authorization, use revoke statement

revoke < privilege list>
on <relation or view>
from <user/role list>;

eg:

i) revoke select
on employee
from Ram, Hari;

ii) revoke update (salary)
on employee
from Ram, Hari;

↳ System can support user groups also known as 'roles' and can thus provide a way of allowing all with same role to show same object

like:

- create role instructor;
- grant select
  on employee
  to instructor;

↳ role is a database object that groups one or more privileges.

Scanned with CamScanner

CamScanner

Date / /
Page No.

- roles can be assigned to users or groups or other roles by using Grant statement.
- users that are members of roles have privileges that are defined for the role with which to access data.

Data Encryption / Crypto System.

→ is storing and transmitting data in encrypted form.
→ original data is called plain text.
→ plaintext is encrypted using encrypted ~~alog~~ algorithm (whose inputs are plaintext and encrypted key)
→ output is called ciphertext.
→ Encryption refers to the process of transforming data into a form that is unreadable unless the reverse process of decryption is applied.
→ encryption algorithm use an encryption key to perform encryption and require decryption key to perform decryption.
→ encryption is widely used today for protecting data in transit in a variety of applications such as data transfer in internet, cellular phone n/ws.
→ encryption is also used to carry out other tasks like authentication.

Scanned with CamScanner

Date / /
Page No.

→ In db, encryption is used to store data in secure way so that even if data is acquired by unauthorized users the data will not be accessible without decryption key.

→ a good encryption technique has following properties

a) It is relatively simple for authorized users to encrypt and decrypt data.

b) It depends on encrypts key used to encrypt data.
- In symmetric key encryption, encryption key is also used to decrypt.
- In assymetric key encryption, two different keys public and private key are used to encrypt and decrypt data.

c) Its decryption key is extremely difficult for an intruder to determine even if intruder has encrypted data.
- In assymetric key encryption, its difficult to infer private key even if public key is available.

e.g:

AES (Advanced Encryption Standard)
DES (Data Encryption Standard)

Scanned with CamScanner

सुगम स्टेसनरी सप्लायर्स एण्ड फोटोकपी सर्भिस
बालकुमारी, ललितपुर ९८४१५९९५९२
NCIT College

Date / /
Page No.

→ in symmetric key encryption,
authorized users must be provided with encrypted key via secure mechanism.

→ in public key encryption
- two keys private & public.
- public key are published.
- private key is known only to user to whom key belongs.

Scanned with CamScanner

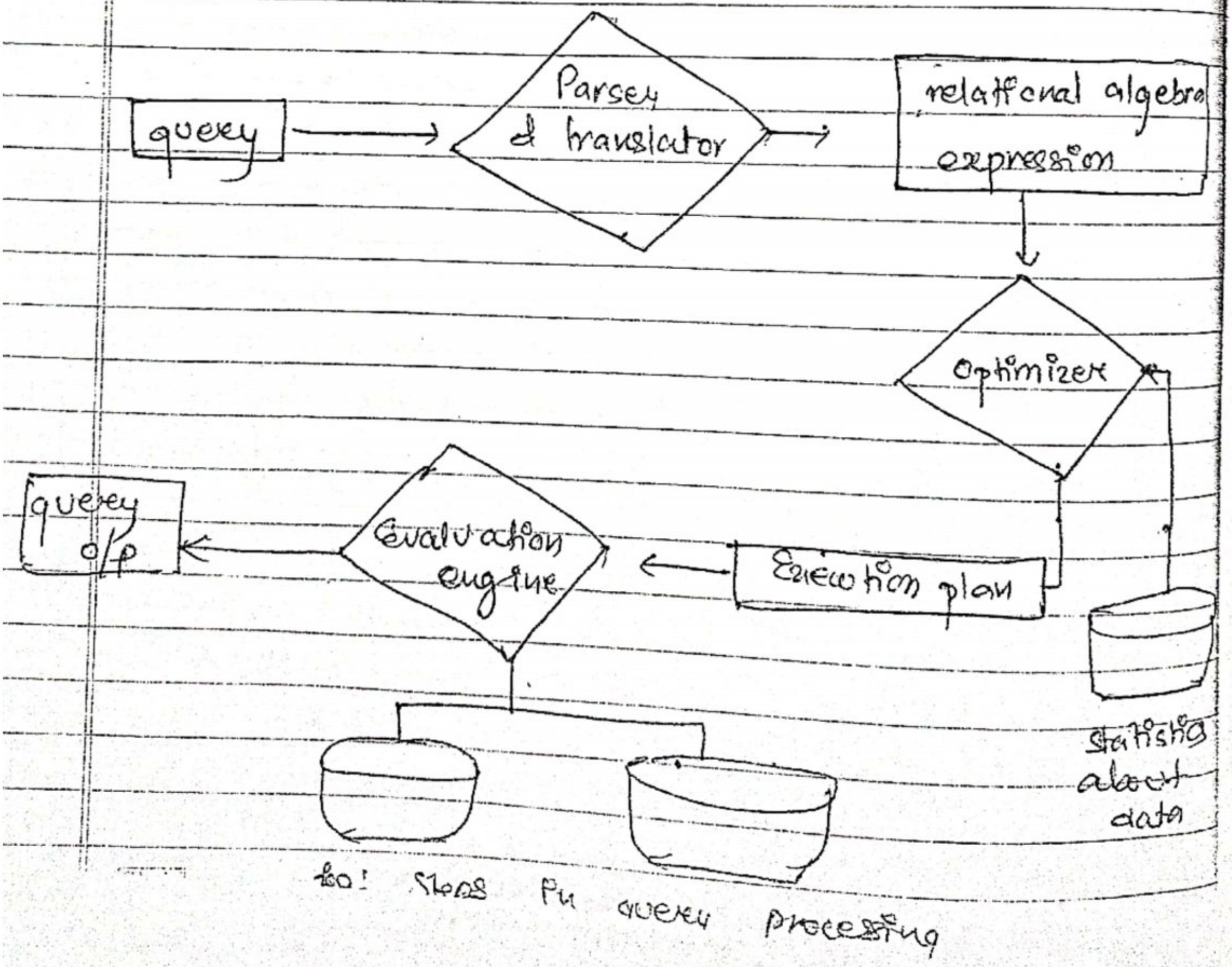
CamScanner

CHAPTER 7: QUERY PROCESSING

Date / /
Page No.

Introduction to Query processing

- is transforming a query written in high level language typically in SQL into correct and efficient execution strategy expressed in a low level language and to execute the strategy to retrieve required data.
- Basic steps in query processing
  i) Parsing and Translation
  ii) Optimization
  iii) Evaluation.



Scanned with CamScanner

Parsing and Translating:

→ translate the query into e internal form which is then translated into relational algebra.

→ parser checks syntax, verifies relation.

→ the relational algebra representation of a query ~~sep~~ specifies only partially how to evaluate a query.

→ a relational algebra expression may have many equivalent expressions.

Consider a query

– Select salary from employee where salary > 20000. Now, equivalent relational algebra expressions are

$$\sigma_{salary > 20000}(\Pi_{salary}(employee))$$

or

$$\Pi_{salary}(\sigma_{salary > 20000}(employee))$$

→ a sequence of primitive operations that can be used to evaluate a query is called query execution plan or query evaluation plan.

Evaluation

– the query execution engine takes a query evaluation plan, executes that plan and return answers to query.

Scanned with CamScanner

Date / /
Page No.

Query optimization

- amongst all equivalent evaluation plans choose the one with lowest cost.
- cost is estimated using statistical information from database catalog.
  e.g. no. of tuples in each relation, size of tuple etc.
- once the query plan is choosen, the query is evaluated with that plan and result of query is output.

Construction of Parse Tree

→ a leaf node is created for each base relation in query.

→ a non leaf node is created for each intermediate relation produced by relational algebra expression.

→ the root node represents result of query.

→ the sequence of operation directed from leaves to root.

i) $\Pi_{salary}(\sigma_{salary > 20000}(employee))$

$\Downarrow$

Π_{salary}

|

$\sigma_{salary > 20000}$

|

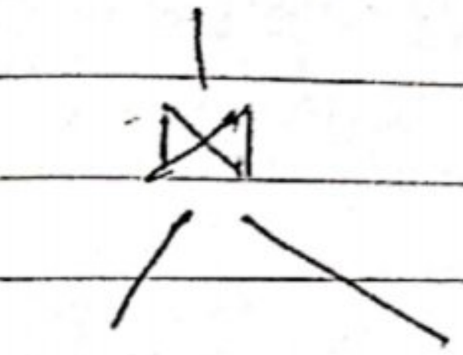
employee

Scanned with CamScanner

CamScanner

e) $\Pi_{name, street}(\sigma_{status='open'}(order \bowtie customer))$

$\Downarrow$



Measure of Query Cost:-

↳ Cost is generally measured as ~~the~~ total elapsed time for answering query.
- many factors contribute to time cost like disk accesses, CPU or even n/w communication

↳ Typically, disk access is the predominant cost and is also relatively easy to estimate. It is measured by taking into account:
- Number of seeks * average seek cost.
- Number of blocks read * average block read cost
- Number of blocks written

↳ Cost to write a block is greater than cost to read a block because data is read back after being written to ensure that the write was successful

Scanned with CamScanner

Date / /
Page No.

↳ For simplicity we just use number of block transfers from disk and the no. of seeks as the cost measures

- t_T = time to transfer one block.
- t_S = time to one seek
- cost for b block transfers plus S seeks

$$b * t_T + S * t_S$$

↳ We ignore CPU cost for simplicity as real systems do take CPU cost into account.

↳ We do not include cost to writing o/p to disk in our cost formula.

Query Operation

* Selection Operation

- The lowest level query processing operator for accessing data is file scan.
- Search and retrieve records for a given selection condition.

Linear Search

- Scan each file block and test all records to see whether they satisfy selection condition.
- Cost estimation = br block transfer + 1 seek

  here, br denotes no. of block containing records from relation r.

Scanned with CamScanner

Date / /
Page No.

→ if selection is on a key attribute, can stop on finding record.
- Cost = $(b_r/2)$ block transfer + 1 seek.

→ Linear Search can be applied regardless of
- selection condition or
- ordering of records in file or
- availability of indices.

Selections using indices.

→ Index Scan
- Search algorithm that use an index.
- selection condition must be on search key of index.

→ A2 (primary Index, equality on key)
- retrieve a single record that satisfies the corresponding equality condition
- cost = $(h_i + 1) * (t_T + t_S)$

→ A3 (primary Index, equality on non key)
- retrieve multiple records.
- records will be on consecutive blocks.
- let b = no. of blocks containing matching record.
- cost = $h_i * (t_T + t_S) + t_S + t_T * b$.

Scanned with CamScanner

Date / /
Page No.

→ A4 (Secondary Index, equality on non key)
- retrieve a single record if search key is candidate key.
  * cost = $(h_i + 1) * (t_T + t_S)$
- retrieve multiple records if search key is not candidate key.
- each of n matching records may be on different block
- cost = $(h_i + n) * (t_T + t_S)$
- can be very expensive.

Sorting

- A query may specify that the output should be sorted.
- the processing of some relational query operations can be implemented more efficiently based on sorted relations (join operation).
- for relations that fit into memory, Techniques like quick sort can be used.
- for relations that do not fit into memory, external merge sort can be used.

External Merge Sort

- let M denote memory size (in pages).

1. Create sorted runs:
   - let i be 0 initially.

Scanned with CamScanner

Date / /
Page No.

repeatedly do the following till the end of relation:
ⓐ Read M blocks of relation in memory.
ⓑ Sort the in-memory blocks.
ⓒ Write sorted data to Run R_i, increment i.
Let the final value of i be N.

2. Merge the runs (N-way merge):-
- we assume $N < M$.
- use N blocks of memory to buffer input runs and 1 block to buffer output. Read the first block of each run into its buffer page.
- repeat.
  * Select the first record (in sort order) among all buffer pages.
  * Write the record to output buffer. If output buffer is full write it to disk.
  * Delete the record from its input buffer page If the buffer page becomes empty then read next block of run into buffer.
- until all input buffer pages are empty.

3. If $N \geq M$, several merge passes are required.
- In each pass, contiguous groups of $M-1$ runs are merged.
- A pass reduces the number of runs by factor of $M-1$ and creates runs longer by same factor.

Scanned with CamScanner

Date / /

Page No.

e.g: if ~~M = 4~~ M = 11, and there are 90 runs, one pass reduces the no. of runs to 9, each 10 times the size of initial runs.

→ Finally, repeated passed are performed till all runs have been merged into one.

g	24
a	19
d	31
c	33
b	14
e	16
r	16
d	21
m	3
p	2
d	7
a	14

Initial relation

Create runs

a	19
d	31
g	24

b	14
c	33
e	16

runs

Merge pass-1

a	19
b	31
c	33
d	21
e	16
g	24

runs.

d	21
m	3
r	16

a	14
d	7

runs

Merge Pass-1

a	14
d	7
d	21
m	3
p	2
r	16

runs

Merge pass-2

Scanned with CamScanner

CamScanner

Date / /

Page No.

Query optimization

→ is an important aspect of query processing.

→ is the activity of choosing an efficient execution strategy for processing a query.

→ aim is to choose one of memory equivalent transformation that minimizes resource usage.

→ Reduce the total execution time of the query, which is the sum of execution times of all individual operators that make up the query.

→ There can be enormous difference in terms of performance between different evaluation plan for same query.

e.g:- seconds vs days to execute same query.

→ cost based query optimization.

→ generate logically equivalent expressions by using set of equivalent rules

- annotate the expressions to get alternative query evaluation plans.

- select the cheapest plan based on estimated cost.

→ Estimation of query evaluation cost based on statistical info from catalogue manager in combination with expected performance of algo.

Scanned with CamScanner

Date / /
Page No.

equivalent Rules:

1 Conjunctive selection operations can be deconstructed into sequence of individual selections.

- $\sigma_{\theta_1 \times \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$

2 Selection operations are commutative.

- $\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$

3 Selection can be combined with cartesian products and theta joins.

- $\sigma_{\theta}(E_1 * E_2) = E_1 \bowtie_{\theta} E_2$
- $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \cap \theta_2} E_2$

4. Theta join (and natural join) operations are commutation

- $E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$

5 Natural join operations are associative

- $(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$

6 Union and intersection operations are commutative

- $E_1 \cup E_2 = E_2 \cup E_1$
- $E_1 \cap E_2 = E_2 \cap E_1$

7. Union and intersection operations are associative

- $(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$
- $(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$

Scanned with CamScanner

CamScanner

Date / /
Page No.

Selection operation distributes over union, intersection and set difference

— $\sigma_p(E_1 - E_2) = \sigma_p(E_1) - \sigma_p(E_2)$

projection distributes over union.

— $\Pi_A(E_1 \cup E_2) = (\Pi_A(E_1)) \cup (\Pi_A(E_2))$

Heuristic Optimization

→ Cost based optimization is expensive even with dynamic programming.

→ System's may use Heuristics to reduce the no. of choices that must be made in cost based fashion.

→ Heuristic optimization transforms the query tree by using a set of rules that typically improve execution performance.

— perform selection early. (reduces the no. of ~~re~~ tuples)

— perform projection early (reduces the no. of attributes)

— perform most restrictive selection and join operations ~~selection~~ before other similar operations

— some systems use only heuristics, others combine heuristic with partial cost based optimization.

Scanned with CamScanner

CamScanner

Date / /

Page No.

Steps in Heuristic Optimization

- Deconstruct conjunctive selections into a sequence of single selection operations.
- Move selection operations down the query tree for the earliest possible execution.
- Execute first those selection and join operation that will produce the smallest relation.
- replace cartesian product operation that are followed by selection condition by join operations.
- Deconstruct and move as far down the tree as possible list of ~~poss~~ projection attributes, creating new projections where needed.
- Identify those sub-trees whose operations can be ~~part~~ pipelined & execute them.

1) Consider a relational schema,

Teacher (TeacherID, TeacherName, office)

- write SQL Statement for the following task;

i) To create a table.

ii) To eliminate duplicate rows.

iii) To add new column 'Gender' in table.

iv) To sort data in table.

v) to delete row.

⇒

Scanned with CamScanner

CamScanner

create table Teacher (
TeacherID INT PRIMARY KEY,
TeacherName text,
Office text,
);
~~INSERT INTO Teacher VALUES~~

Select distinct (*) from Teacher

ALTER table Teacher
Add Gender text

Select * from Teacher
order by TeacherID desc.

delete from Teacher.

Employees (emp-ID, Fname, Lname, email, phone, hiredate, jobid, salary, manager-id, Department id)

Departments (Departmentid, dname, manager-id, location-id)

locations (location-id, street, postcode, city, country-id, state)

Scanned with CamScanner

Date / /

Page No.

i) Show lastname, salary and department id of employee whose salary is between 5000 & 8000.

ii) Show the records of those employees whose name starts with letter 'S' and works in department 100.

iii) List all employees firstname, lastname, email, department name whose department name is 'programming'.

iv) List records of all employees department name, managerid, street whose employeeid = 110.

I) ⇒ select ~~e~~ Employees.lastname, Employee.salary, Departments.Departmentid. ~~Manager~~ ~~join~~ from ~~e~~ emplo

→ Select lastname, salary, departmentid from employees where salary between 500 & 800.

ii) Select * from employee where fname like 's%' and departmentid = 100.

iii) Slect ~~e~~ employees.fname, employees.lname, employee.email, department.departmentname from ~~e~~ employees inner join department on

Scanned with CamScanner

CamScanner

Date / /

Page No.

~~department~~ ~~dname = 'programming'.~~

employees. ~~emp~~ department_ID = department. department_ID where department. dname = 'programming'.

iv) Select department. ~~departmentname~~ dname, department. manager_id, location. street. from ~~department~~ location inner join ~~location~~ department on department. location_id = locations. location_id inner join employee ~~where~~ on department. department_id = employee. department_id where employees. empid = 110.

3) Employee (EID, Name, Post, Age),
Post (Post-title, Salary)
Project (PID, PName, Duration, Budget)
Work (PID, EID, joinDate)

i) List name of employees whose age is greater than average age of employees.

ii) Display all employee numbers of those employees who are not working in any project.

iii) List name of employee and their salary who are working in project 'DBMS'.

iv) Update database so that 'Rishab' now lives in 'Butwal'.

Scanned with CamScanner

CamScanner

Date / /

Page No.

Answers

select name from employee where
age > ~~avg (~~ (select avg (age) from employee);

Select count (employee.EID) from Employee
inner join work
on employee. EID = work. EID
where work. P.ID = NULL.

select employee. name , post. salary from
employee inner join ~~employee~~ post on
employee. postID = post. postID
inner join ~~~~
Work on ~~employee~~
work. EID = employee. EID
inner join ~~work on~~ project
on work. PID = Project. PID where
Project. Pname = ' DBMS '.
ALTER table employe add location text ,
UPDATE employee set location = ' Butwal '
where employee. name = ' Rishab '.

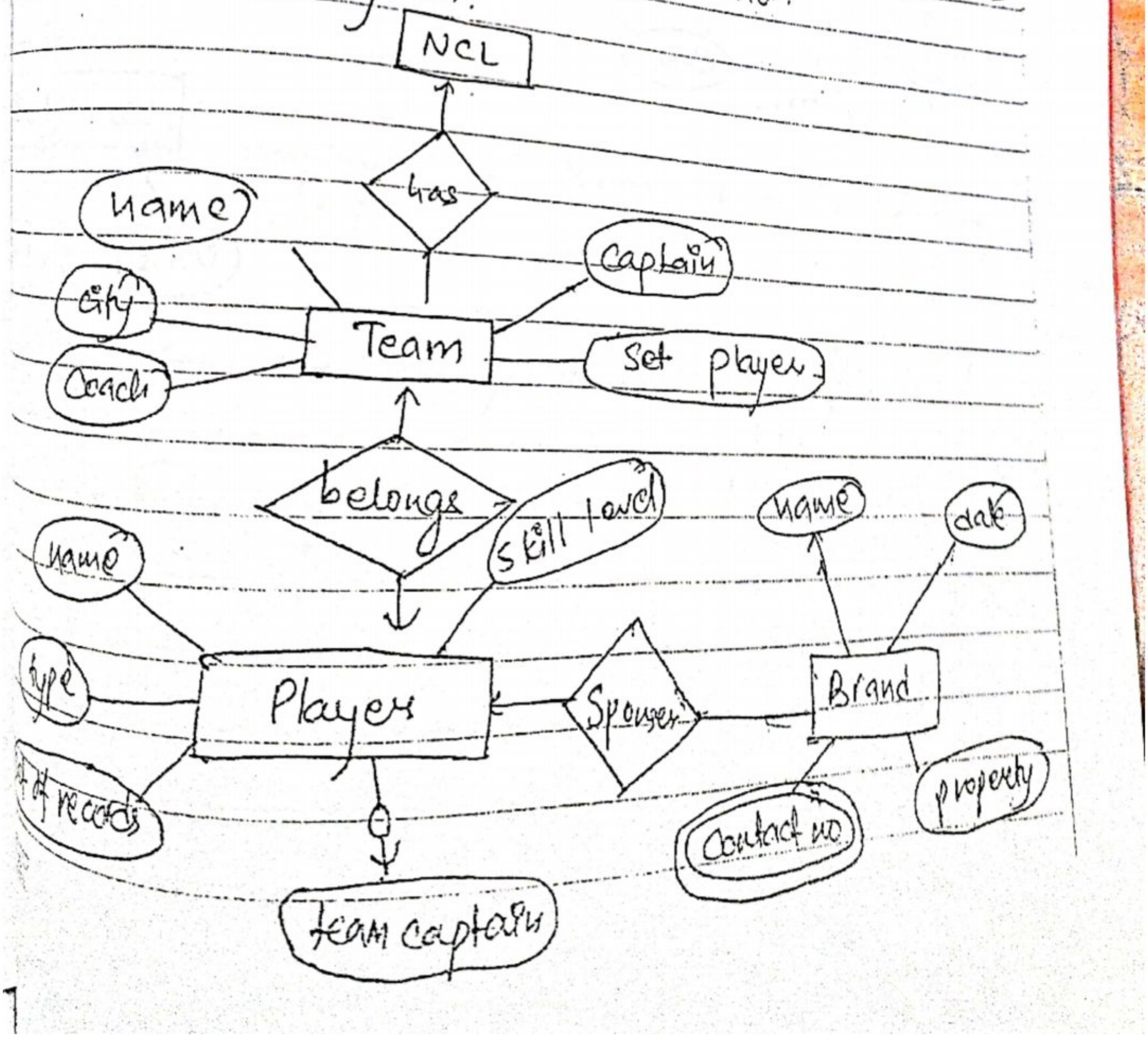
Scanned with CamScanner

CamScanner

Q. Suppose you are given following requirements for simple database National Cricket League (NCL)

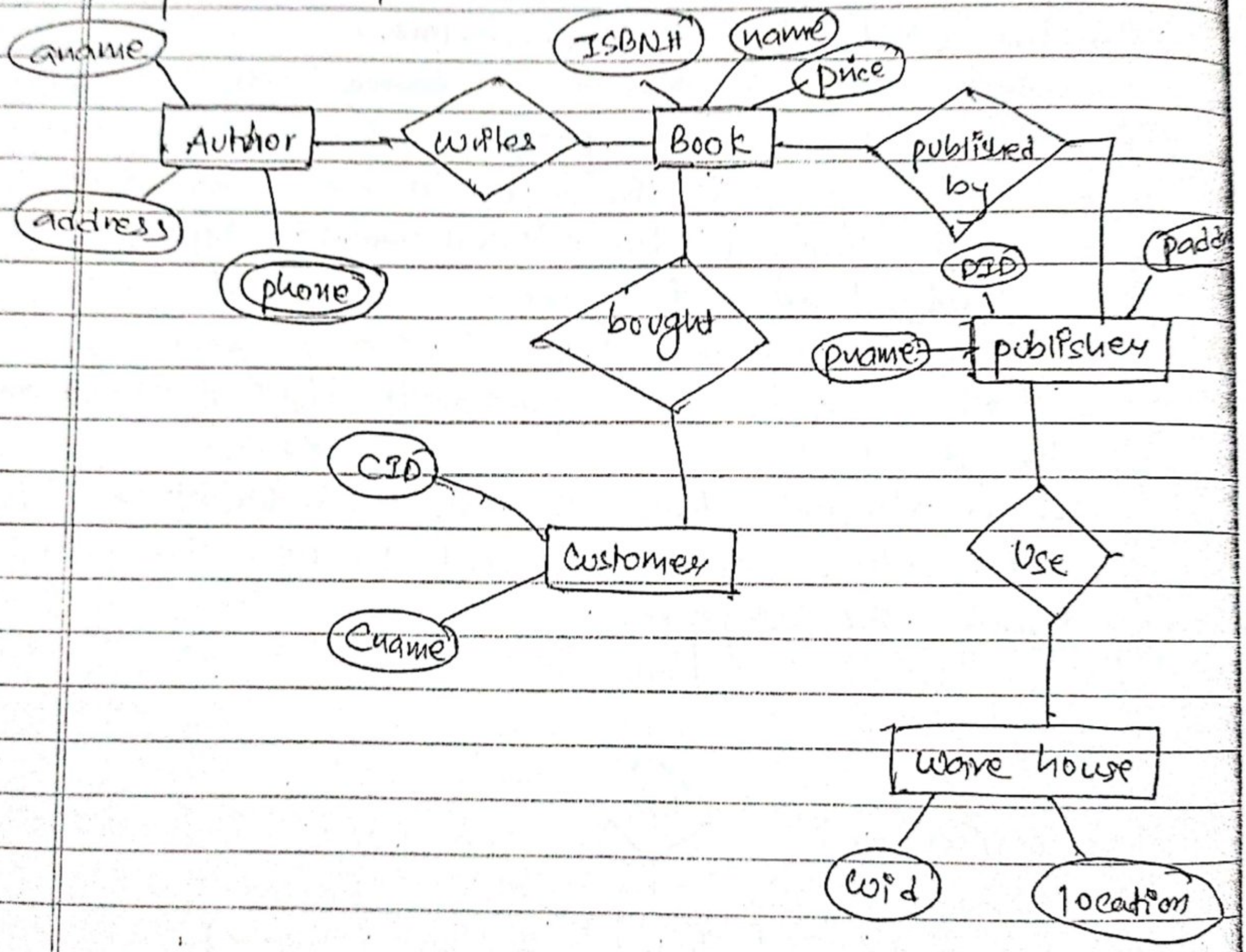
- the NCL has many teams.
- each team has a name, city, coach, captain, set of players
- each player belongs to only one team, has a name, type, skill level, set of records.
- each player is also a player.
- a team captain is sponsored by at least one brand.
- a brand has its name, established date, property, multiple contact no.

Draw ER diagram:



Scanned with CamScanner

7) Construct an E-R diagram for online Book store where entities are author, book, publisher, customer & ware house.



Scanned with CamScanner

Consider following relational database schema:

Employee (EName, street, city)
Works (EName, CompanyName, Salary)
Company (CompanyName, City)
Manager (EName, ManagerName)

Write relational algebra for following

Find names of all employees who work for Hamro Bank.

$\Pi_{EName}(\sigma_{CompanyName = 'HamroBank'}(Employee \bowtie Works))$

OR

Equ Jopn.

$(Employee \bowtie_{Employee.Ename = Works.ename} Works)$

Give all employees of Hamro Bank a 5 percent salary ~~rise~~. raise.

$\Pi_{EName, CompanyName, salary * 1.05}(\sigma_{CompanyName = 'HamroBank'}(Employee \bowtie Works))$

(r1)

Scanned with CamScanner

CamScanner

~~Insert~~ record

Date / /

Page No.

→ Delete all tuples in works relation for employee of Hamro Bank.

Delete

Works ← Works − σ companyName 'HamroBank' (Works)

new

(Original − selection)

↳ Insert record into employee

employee ← employee ∪ ('Hari', 'new record', 'ktm')

↳ Modify database so that Harish now lives in Biratnagar.

⇒

employee ← $\pi_{EName, Street, 'Biratnagar'}$ ($\sigma_{EName = 'Harish'}$) ∪ (employee − $\sigma_{name = 'Harish'}$ (employee))

Or

r1 ← $\pi_{EName, Street, City}$ ($\sigma_{EName = 'Harish'}$ (employee))

or employee.

r2 ← $\pi_{EName, Street, 'Biratnagar'}$ (r1) (employee).

Scanned with CamScanner

CamScanner

सुगम स्टेसनरी सप्लायर्स एण्ड फोटोकपी सर्भिस
बालकुमारी, ललितपुर ९८४१५९९५९२
NCIT College

Date / /
Page No.

Given,

Student (CRN, Name, gender, Address, Telephone)
Course (CourseID, CName, hour, TID)
Teacher (TID, TName, Office)
Registration (CRN, CourseID, Date)

2) Count the number of student registered subject in year 2015 genderewise.

ζ count (gender) (Student $\bowtie_{\text{Date}=2015}$ Registration)