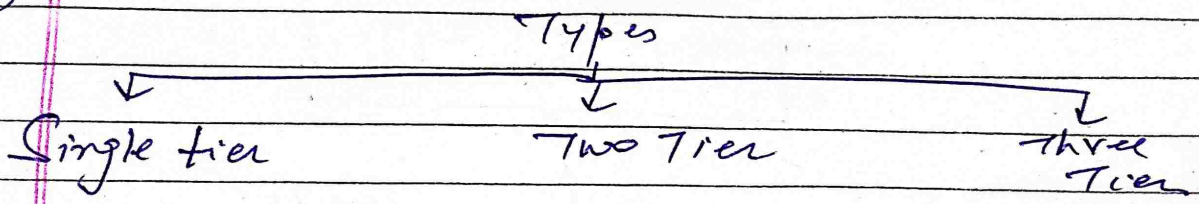
DBMS Architecture :

helps us understand the concepts/components of database system & relation among them.

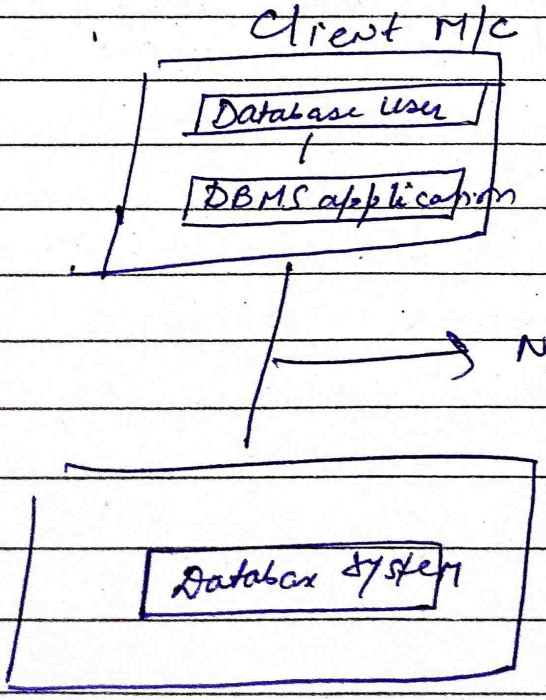
→ Architecture depends upon the computer system on which it runs. for eg Client-Server DBMS architecture.

→



Single Tier : Database is readily available on the client machine, any request made by client doesn't require a n/w connection to perform the action on the database.

Two Tier :

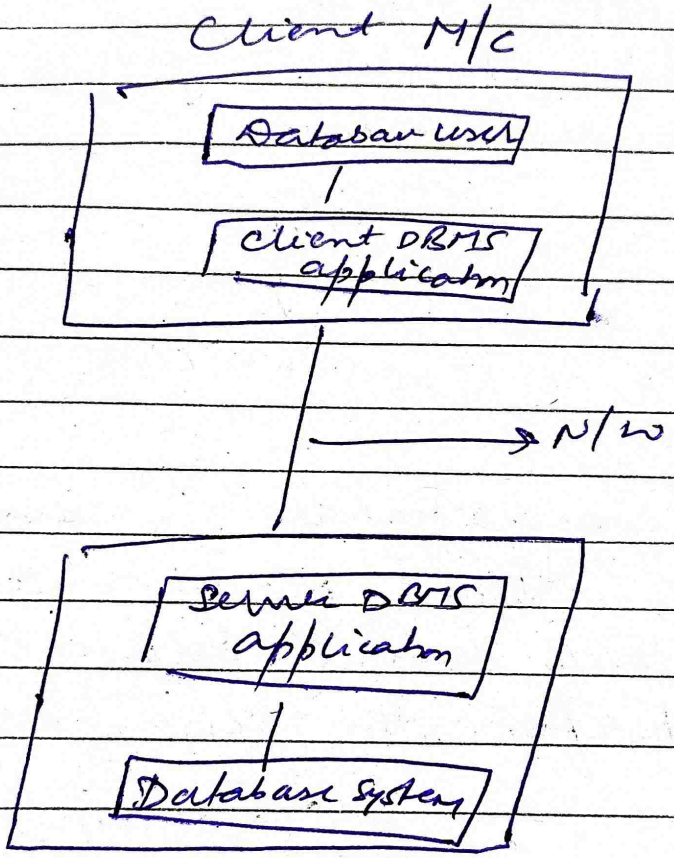


Page :
Date : / /

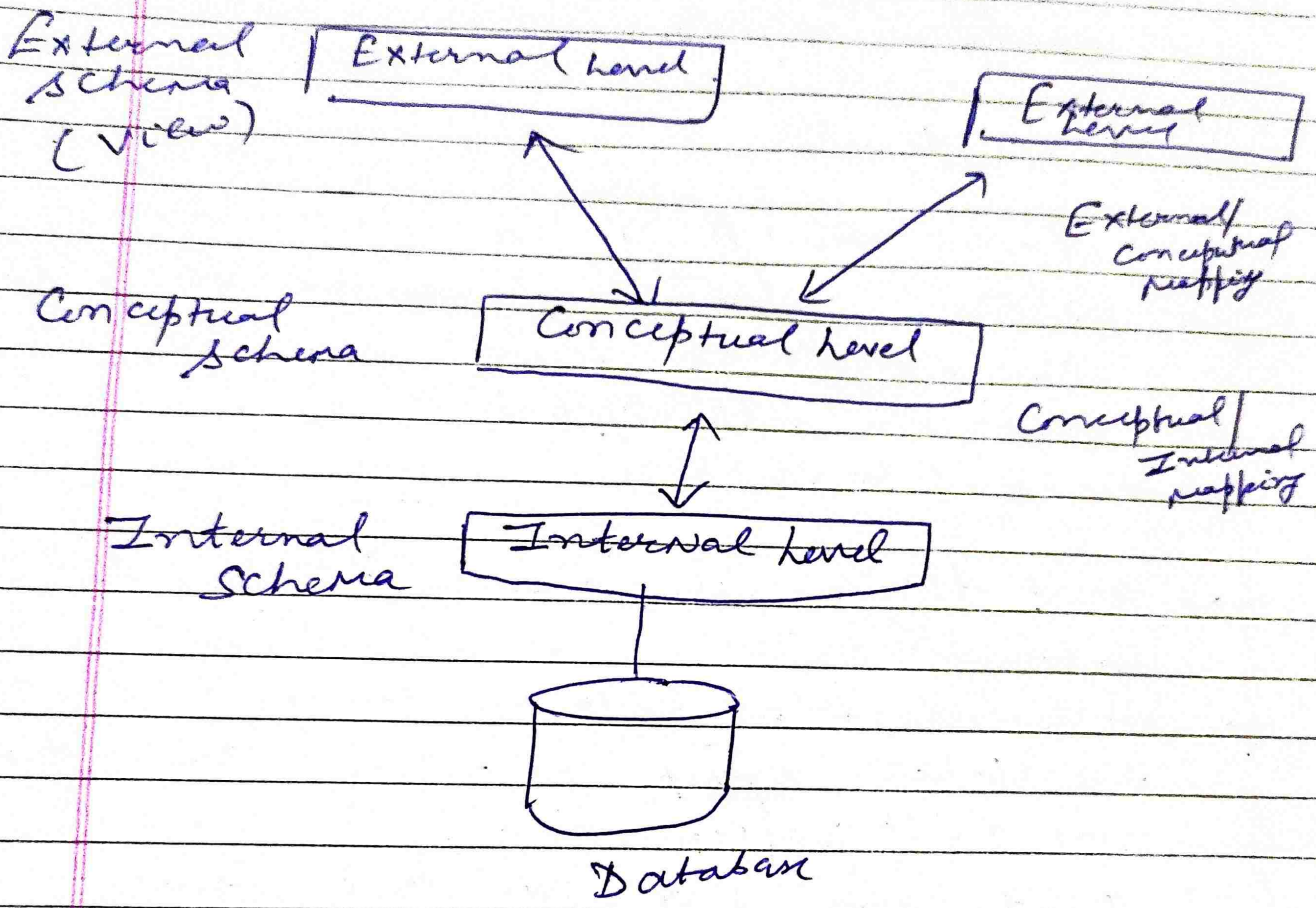
→ Database System is present at server M/c & the DBMS application is present at client M/c.

→ Client communicates using a query language like SQL.

3 Tier



3 level
or
3 Schema Architecture (ANSI/SPARC)



→ describes the structure of a specific database.

⇒ Seperates the user applications & physical database

⇒ Mapping is used to transform the request b/w various database levels of architecture.

Objective :-

enable multiple users to access the same data with a personalized view while storing the underlying data only once.

→ Users of database should not worry about the physical implementation & internal workings of database such as data compression, encryption techniques. etc.

Internal level:

→ lowest level of data abstraction
→ it describes how the data are actually stored on storage devices.
→ also called physical level.

Conceptual level

→ next higher level than internal level of data abstraction.
→ it describes what data are stored in database and what relships exists among those data.
→ also called logical level
→ DBA also works at this level.

External level:

→ highest level of data abstraction.
→ describes a part of entire database that a end user concern.
→ also called view level
→ Different users need different views of database and so there can be many view level abstractions of same database.

Page:
Date: / /

Database Schema: Structure that represents the logical storage of data in a database

→ it represent organization of data & provides info about the r/ships b/w the tables in a given database.

→ Schema objects that may include tables, fields, packages, views, r/ships, keys.

→ In actual data is physically stored in files that may be in unstructured form, but to retrieve it we need a structured form, To do this database schema is used.,

→ provides knowledge about how data is organized in a database & how it is associated with other data.

→ Schema does not contain physicaly data itself, instead it gives information regarding the shape of data & how it is related.

→ Complexity & size vary as per size of project.

Logical schema
Logical constraints that need to be apply to stored data.
view, integrity constraints & table

Physical Schema
how data is stored physically on a storage system or disk storage in form of files or indices

Page :
Date : / /

Database Instance type of Snapshot of an actual database as it existed at an instance of time.

→ It varies or can be changed as per time.

→ can be changed by CRUD operations

Create Read Update Delete

Data Independence: Characteristic of being able to modify the schema at one level of database without altering the schema at next level

Logical

Physical

→ change the conceptual schema without having change the external schema

→ occurs at user Interface level.

→ capacity to change the internal schema without having to change the conceptual schema.

→ occurs at logical interface level

Database Models

→ gives the idea that how the final System will look like after its complete implementation
→ defines data elements & the r/ship b/w the data & elements
→ used to show how data is stored, connected, accessed & updated in DBMS.
→ provides us transparent picture of data which helps in creating an actual database.

Advantages

→ helps in representing data accurately.
→ helps in finding missing data & also in minimizing Data Redundancy.
→ provides data security in a better way.
→ information in data model can be used for defining r/ships b/w tables, keys etc.

Disadvantages:

→ In case of vast database sometimes it becomes difficult to understand the data model.
→ you must have proper knowledge of SQL to use physical model.
→ Even Smaller changes made in structure require modification in the entire application.

→ No set data manipulation language in DBMS.

→ To develop data model one should know physical data stored characteristics

Conceptual: → describes database at a very high abstraction & output is understand the needs or requirement of database., Requirement gathering phase i.e ER Model

Representational Data Model

↓ Relational model

Physical Data Model

SQL used to implement Relational model

Hierarchical

N/w Model

Object oriented

Float Data Model basically consist of 2D array of data models that do not contain any duplicate elements in the array. Data Model

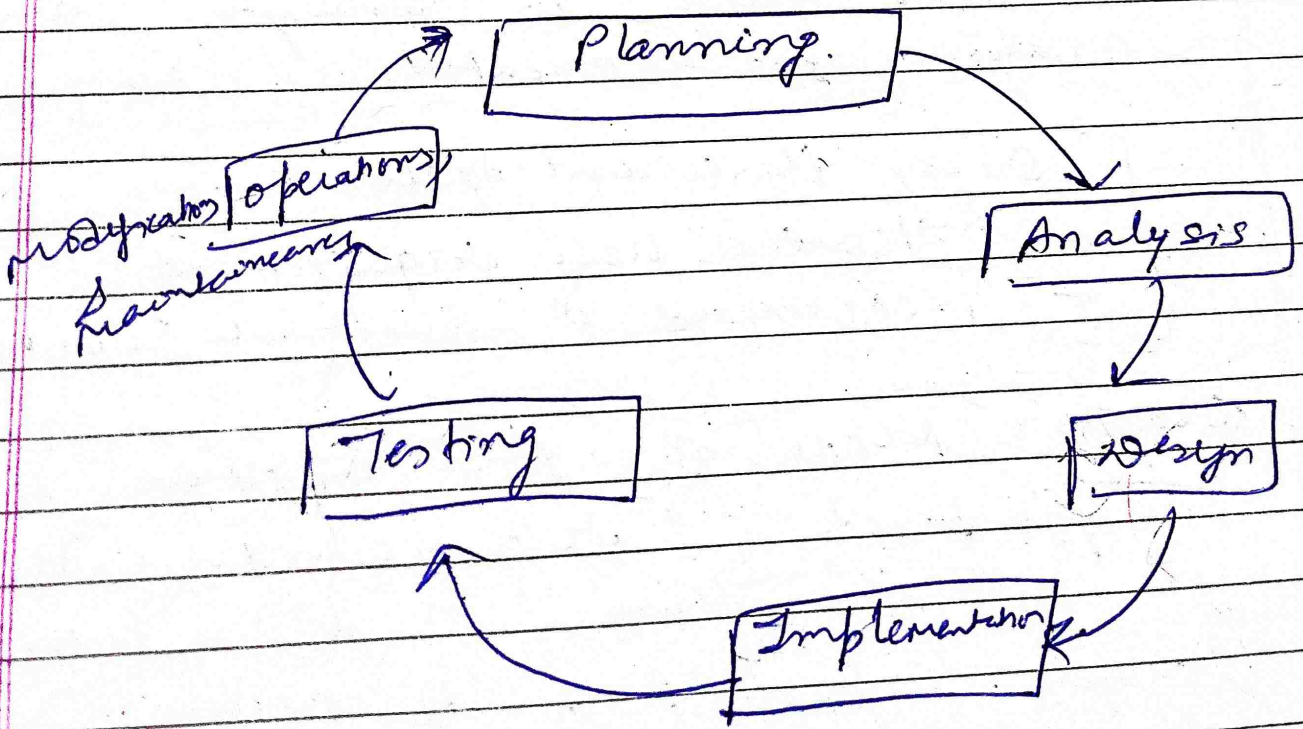
Page:
Date: / /

has one drawback, it can't store large amount of data that is tables can't be of large size.

Context Data Model, a data model which consist of more than one data model.

Semi structured deals data in flexible way. Some entities may have attributes & some have missing attributes.

DDLC (Database Development Life Cycle)



Planning : phase starts when a customer request to develops a system.

→ Identify the resources needed to develop the System.
→ Identify the time limit of completion of System.

Requirement Analysis :

→ Analyze end-user needs.
→ Analysis phase is used to study the current System. Data is collected about System to be developed. The collected data is analyzed and analysis report is created.

- Study the current System
- Determine user requirement
- Recommend a suitable solution.

Design Model of the System is prepared requirement collected in analysis phase is translated into logical representation of system.

- Conceptual Model
- Logical Model.
- Physical Model

Page:
Date: / /

Implementation

→ Final design is implemented.
→ database Mgt System is installed databases are created, then data are loaded or imported,

O/P : S/W

Testing Phase

→ We check the tolerance of our system./S/W.
→ check the acceptable & unacceptable values

Maintenance goes along with the time.

→ Monitoring back up, security of system.
→ Updating & upgrading database (when required)

ER Model is a model for identifying entities to be represented in database & representation of how those entities are related.

→ ER diagram explains the r/ship b/w the entities present in database.

Why use ERD?

→ used to represent the ER Model in database, which makes easy to convert into tables.

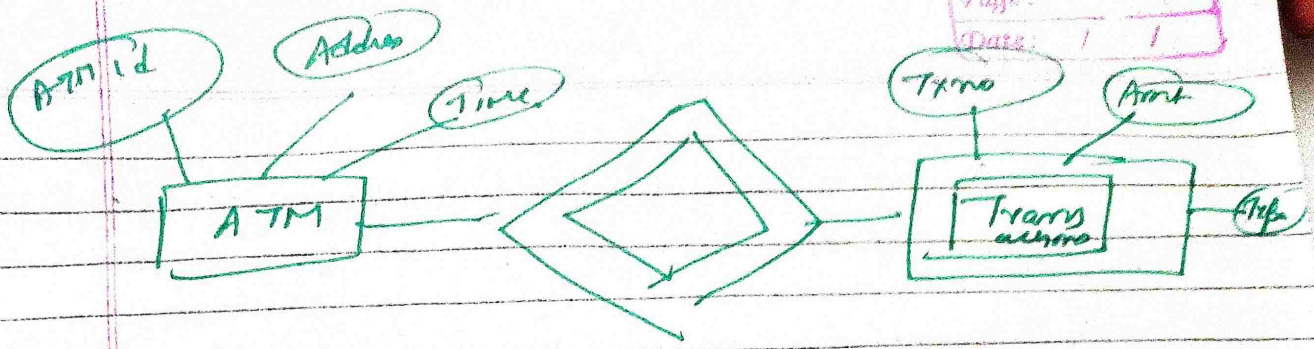
→ purpose of real world objs modeling of objects.

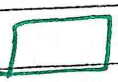
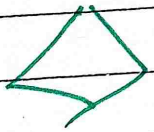
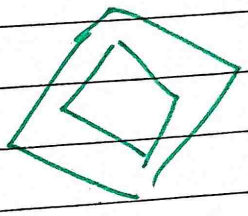
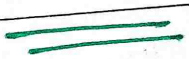
→ requires no technical knowledge & no h/w concept.

→ Standard soln for visualizing data logically

→ GUI representation of logical database

Cardinality no of times an entity of an entity set participates in a r/ship set is known as cardinality.

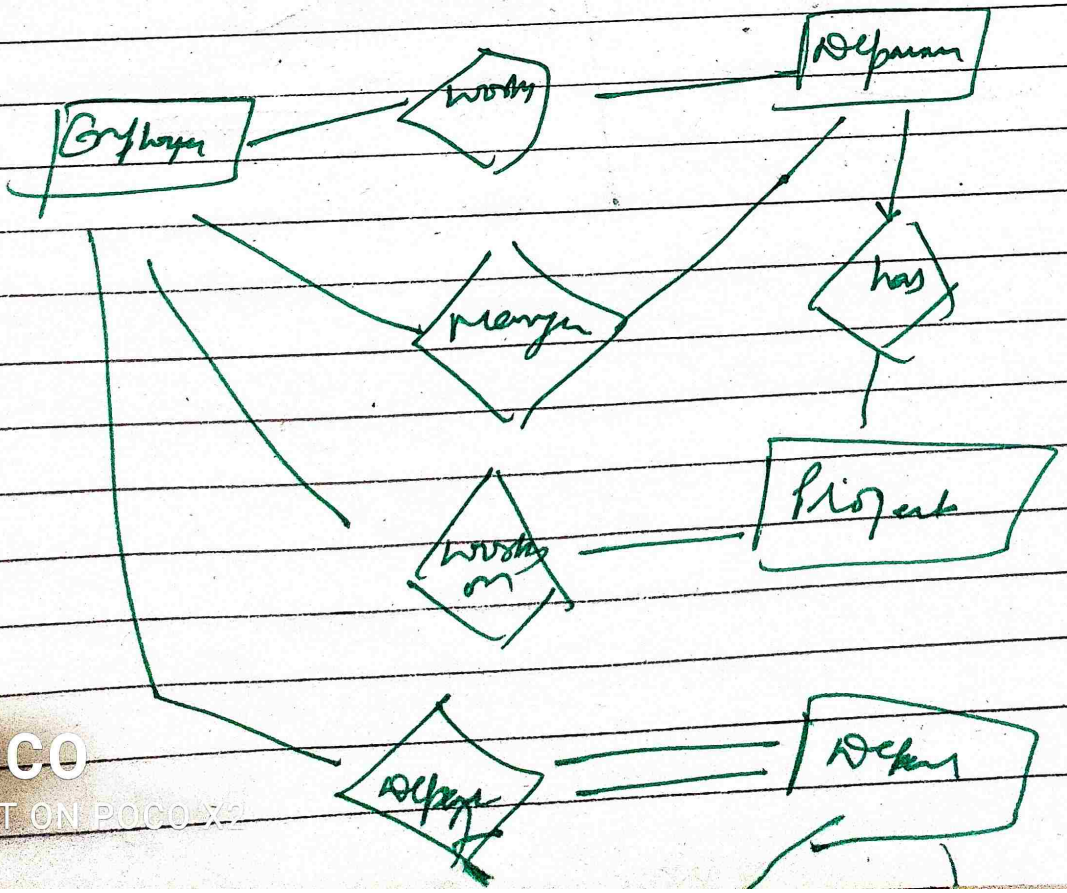


Strong Entity Set	Weak Entity Set
→ always has a primary key.	→ does not have attributes to build a primary key.
→ 	
→ Members are called dominant entity set	→ Sub ordinate entity set.
→ 	
→ 	

eg

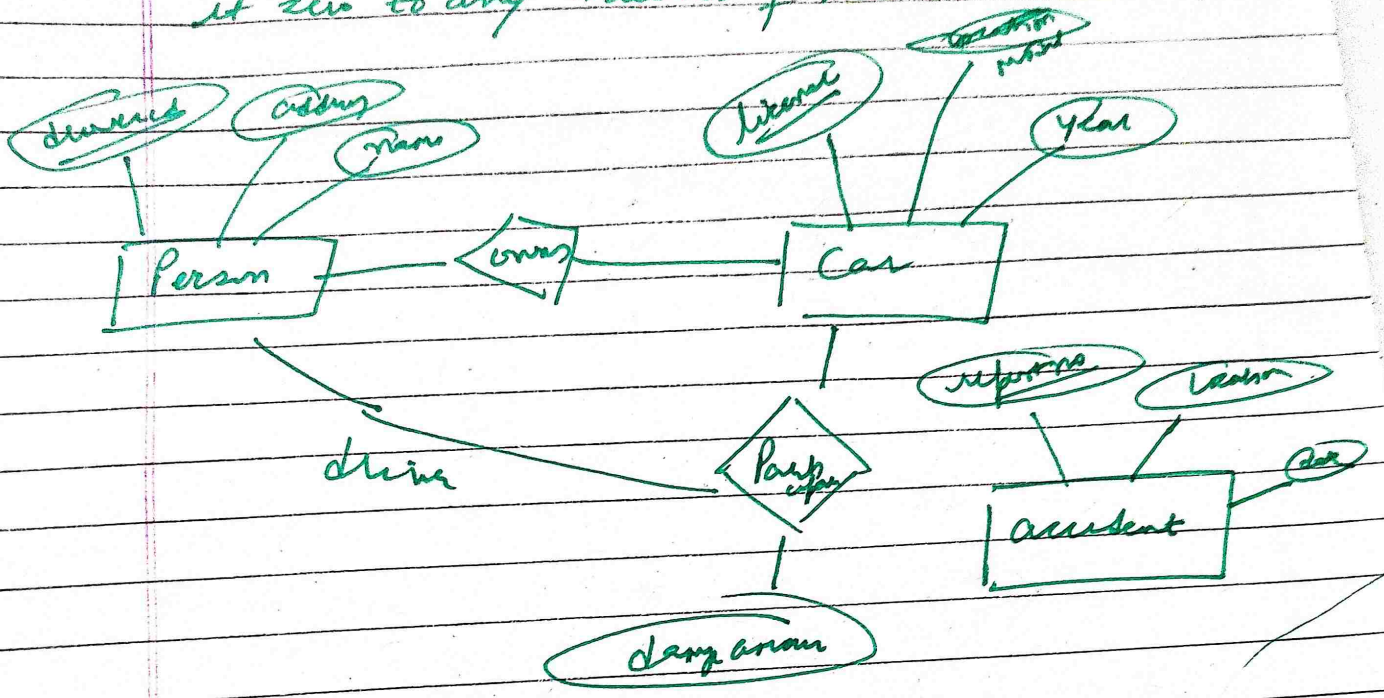
ER digram of Company

→ Company has several departments
→ Each dept may have several locations
→ A manger enters a particular dept
→ Each dept is associates with projects
→ Employees are identified by id, address, dob, doj etc
→ Each employee has dependent.
→ Dependent has D_name, Gender & Relation.



Page:
Date: / /

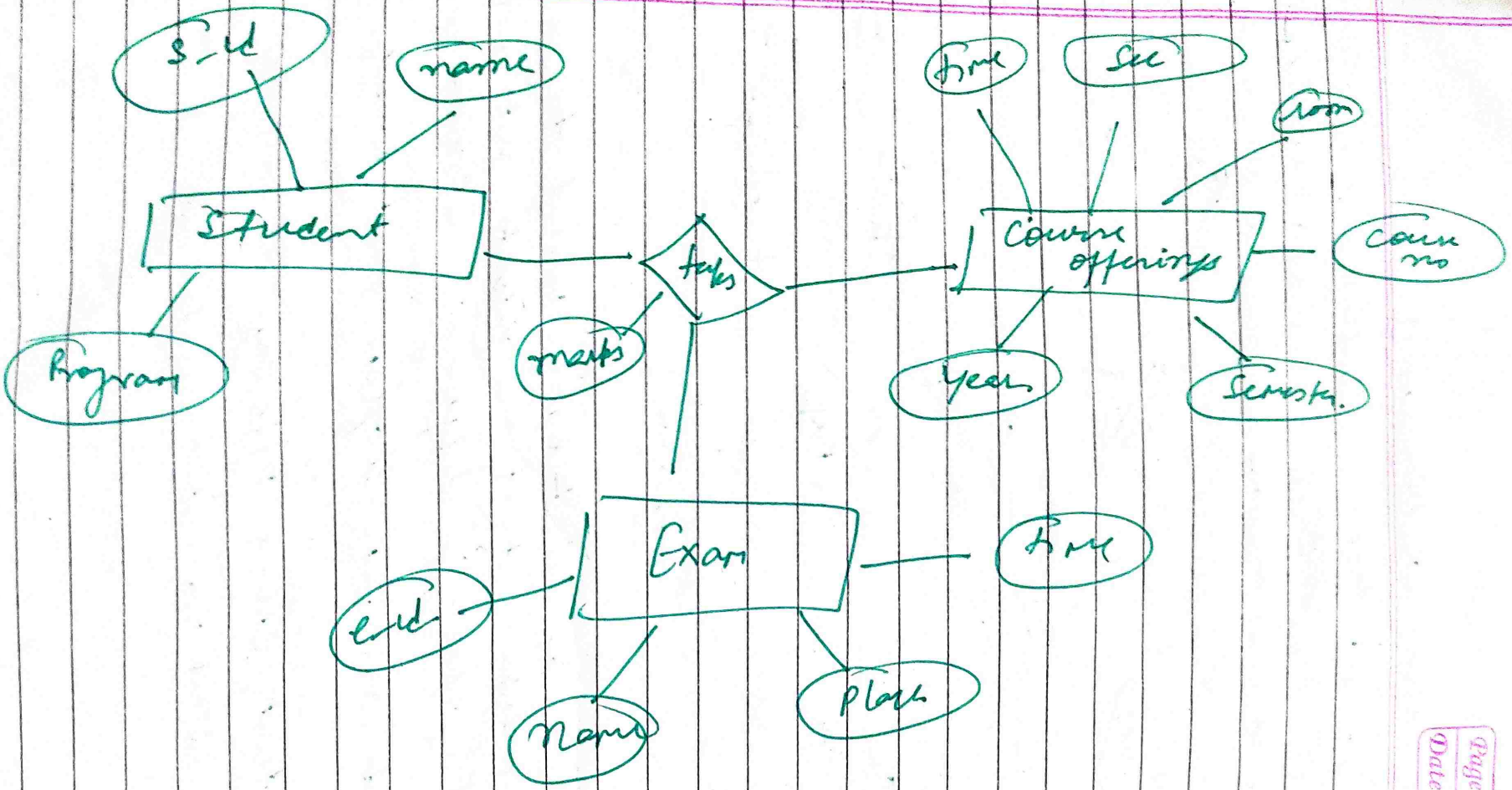
⊗ Construct an ER Diagram of a Car Insurance Company whose customers own one or more cars each. Each Car is associated with zero to any number of recorded accidents.



⊗ Consider a database used to record the marks that students get in different exams of different Course offerings. Construct an E-R diagram that models

→ Student, course offerings as entities

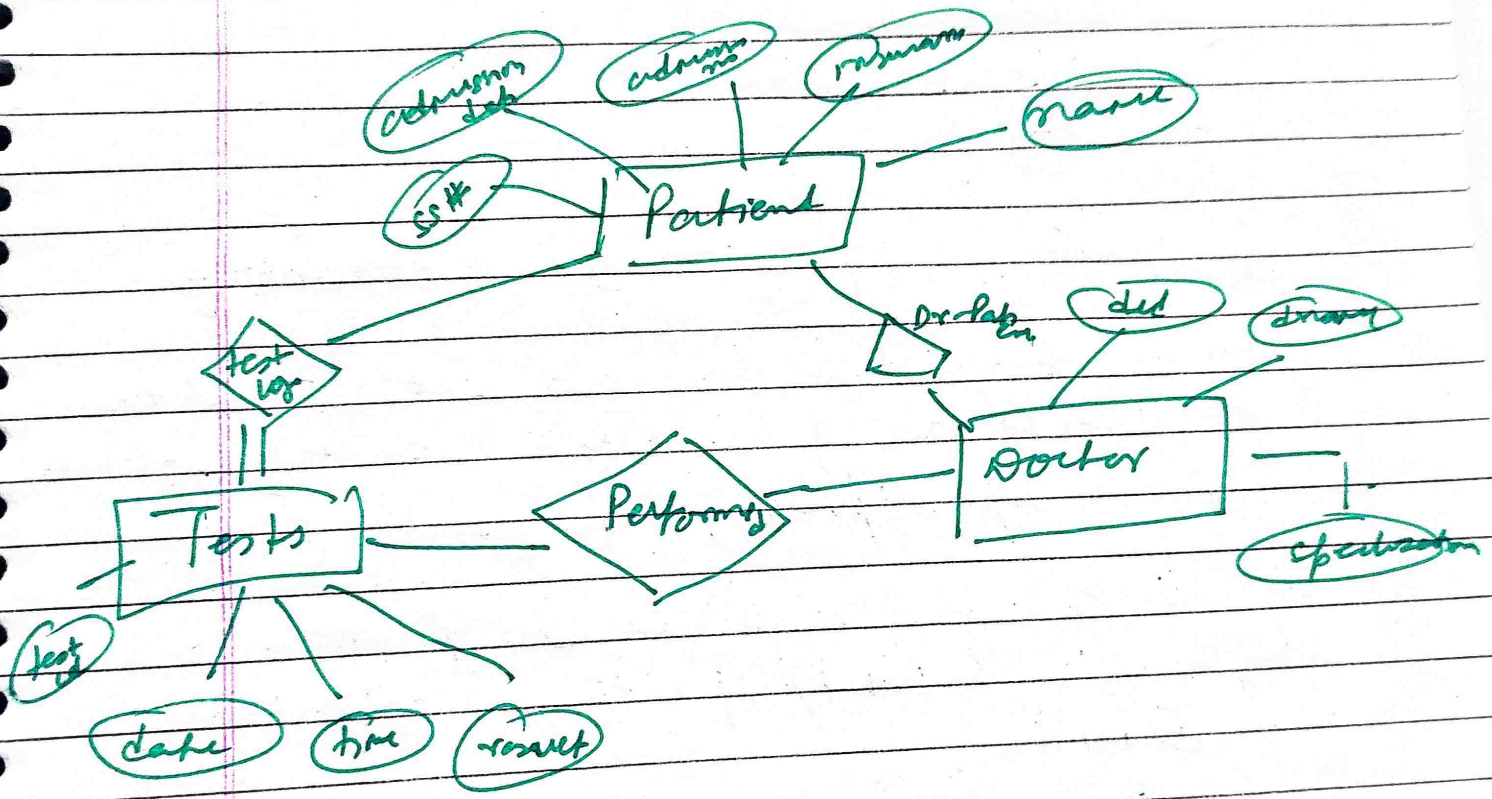
→ use relset for above database.


S_id
name
Student
Program
takes
marks
time
sec
room
Course offerings
Coun no
year
Semest.
Exam
time
eid
Name
Place

Page :
Date : / /

Q Construct an ER diagram for a hospital with a
→ set of Patients
→ set of Medical Doctors.

Associate with each patient a log of various tests & examinations conducted.



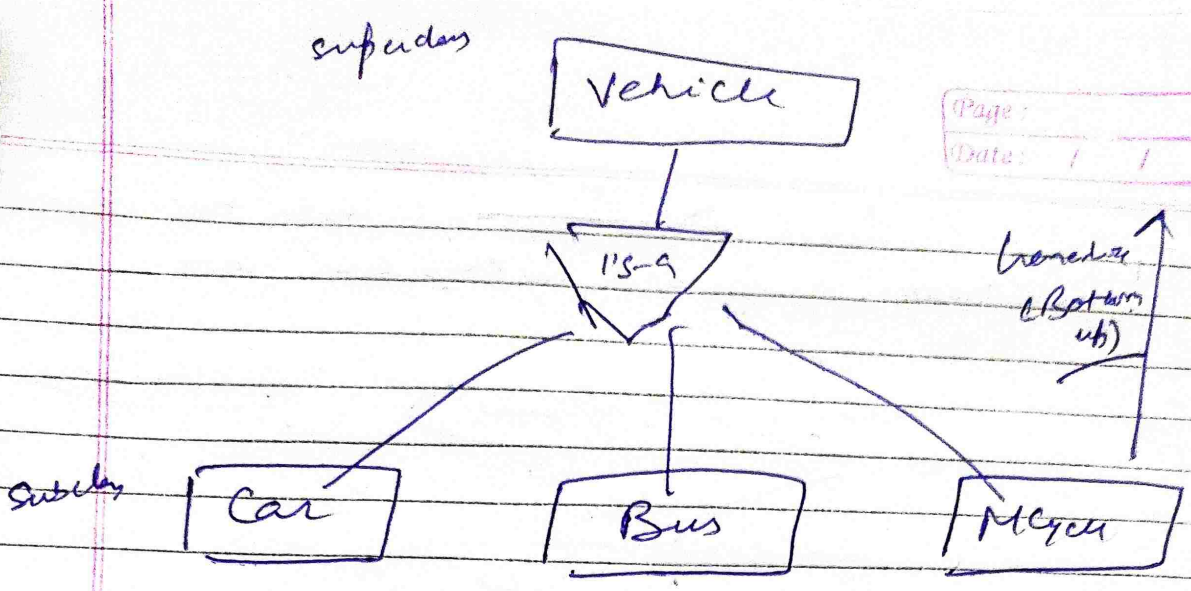
Page :
Date : / /

Extended ER Model

→ As complexity of data increases, more difficult to use traditional ER Model.
→ Improvements to existing ∧ for complex applications.
→ 3 new concepts
  - Generalization
  - Specialization
  - Aggregation

Generalization: process of extracting common properties from a set of entities & create a generalized entity from it.
→ Bottom up approach, in which two or more entities can be combined to form a higher level entity if they attributes in common.
- Subclasses are combined to make Superclass.

→ used to emphasize similarities among lower level entity set & to hide differences within there
→ depicted by triangle component labeled as g



Specialization: opposite of Generalization

→ an entity is broken down into sub entities based on their characteristics

→ Top down approach - when higher level entity is specialized into two or more lower level entities.

→ can be repeatedly applied to refine the design of a schema

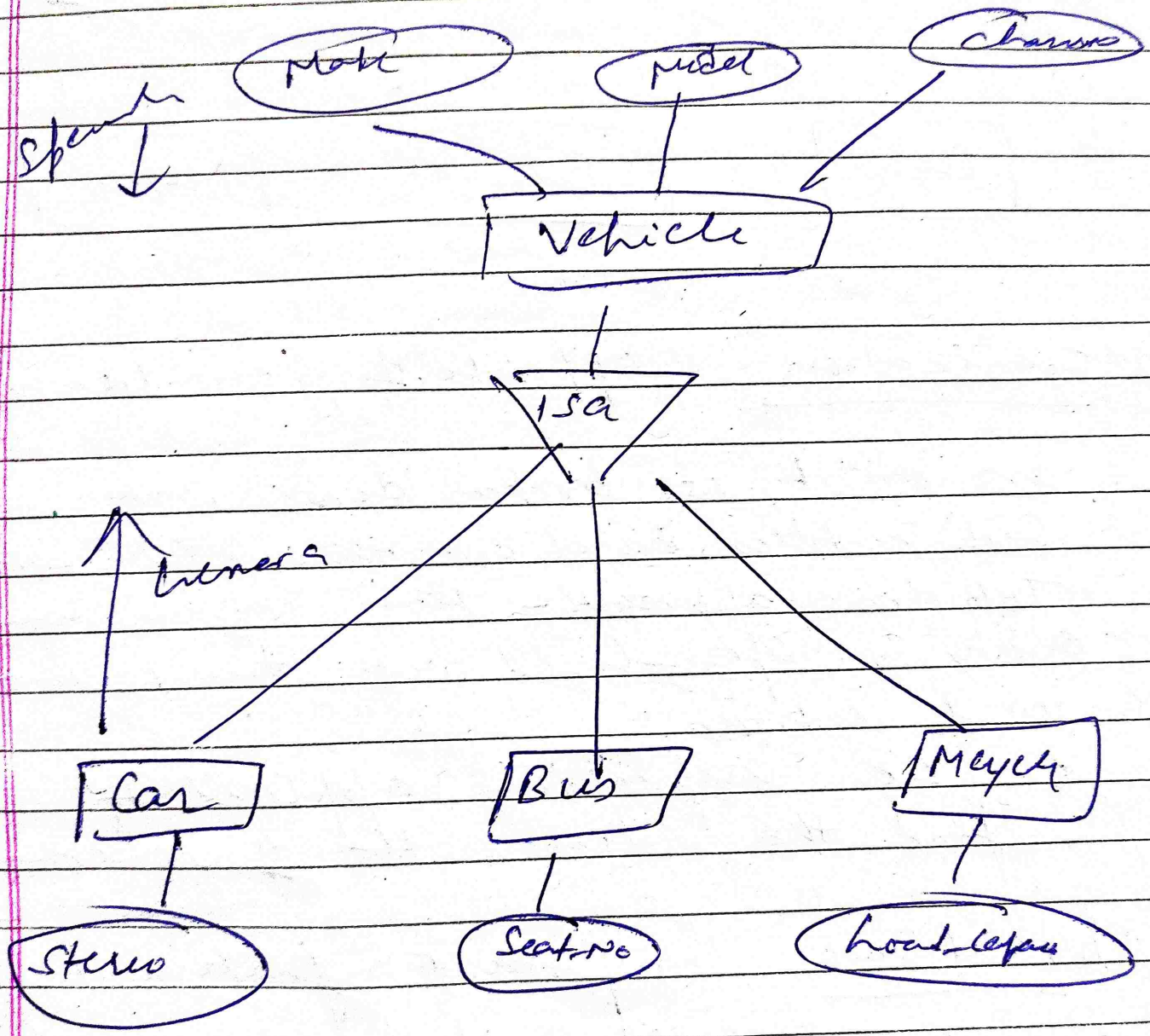
Inheritance important feature of generalization & specialization.

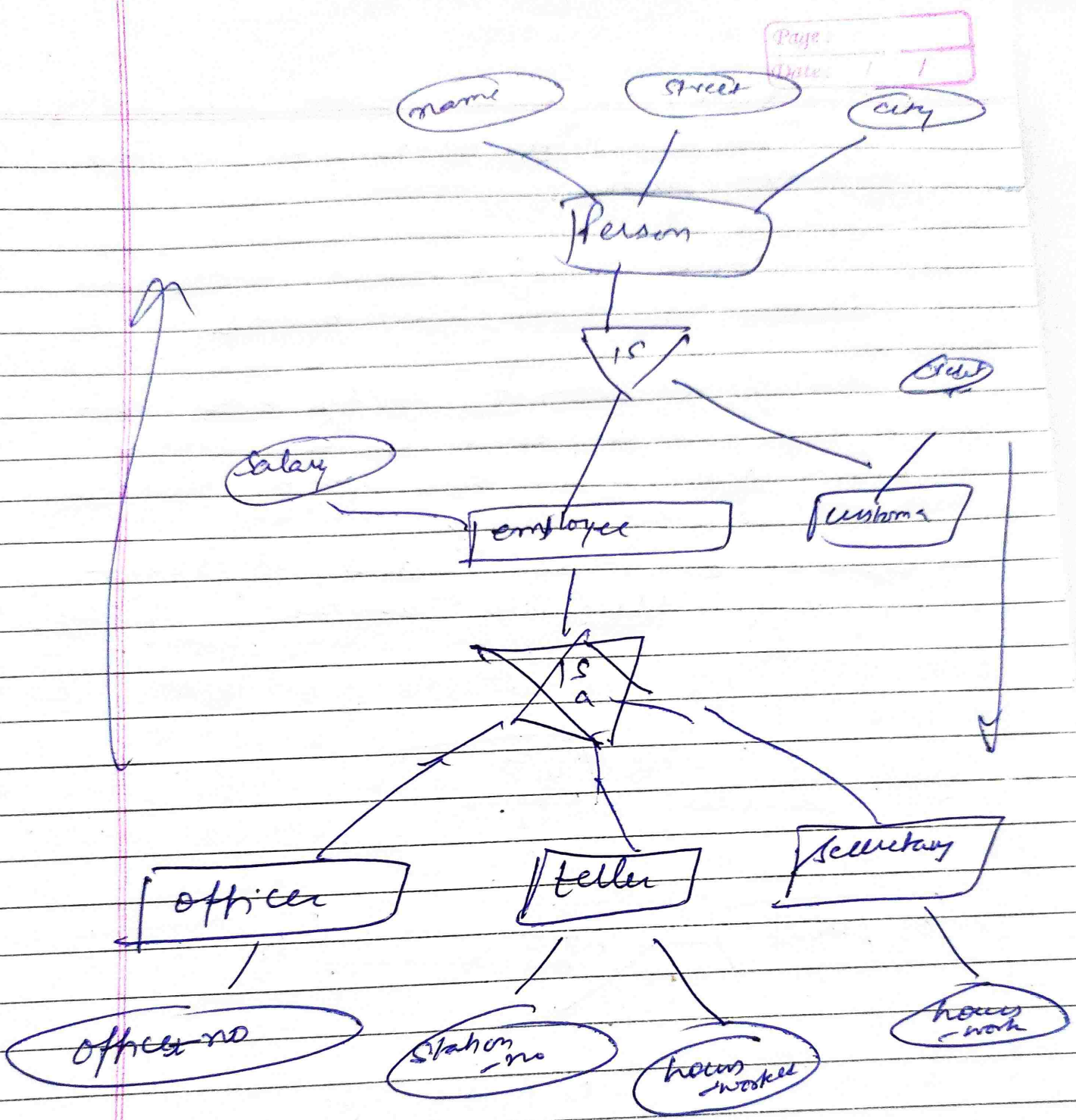
→ Attribute inheritance allows lower level entities to inherit the attributes of higher level entities.

→ also extend to participation inheritance in which relationship higher level entity sets are inherited by lower level entity sets.

Page :
Date : / /

→ Lower level ~~Inheritance~~ entity set can participate in its own R/ship set, too.





Classes

Customer (name, street, city, credit rating)

officer (name, street, city, salary, office no)

teller (name, street, city, salary, station_no, hours worked)

Secretary (name, city, street, salary, hours worked)

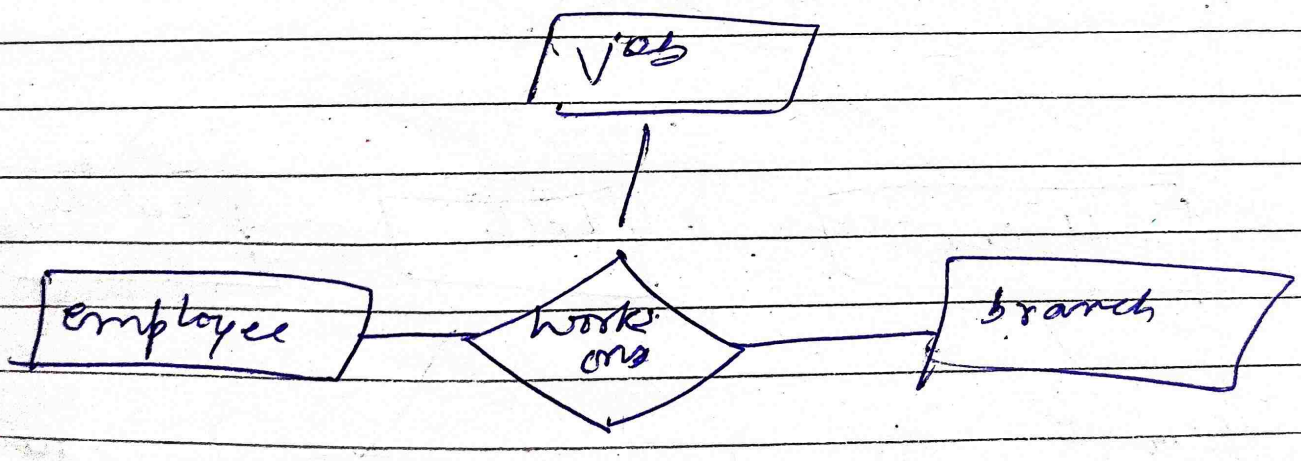
Page :
Date : / /

Aggregation : is a ~~feature~~ used when we need to express a M-ship among M-ships.

→ abstraction through which M-ship are treated as higher level entities.

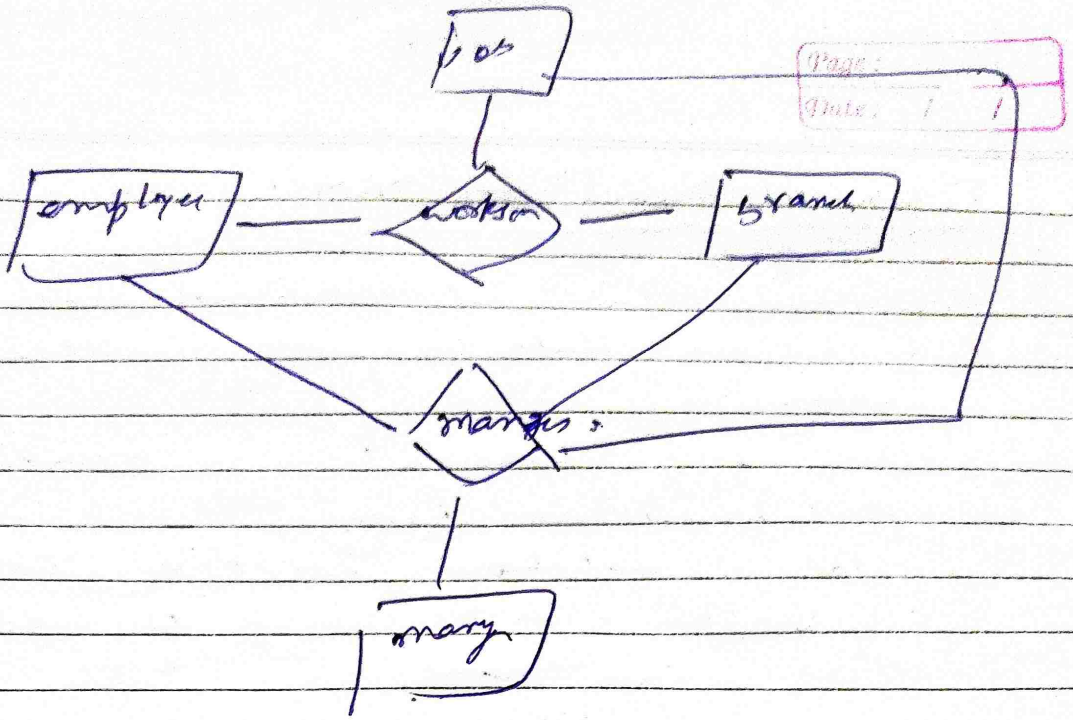
→ is a process when a M-ship b/w two entities is considered as a single entity and again this has a M-ship with another entity

→ Basic ER model can't represent M-ships involving another M-ships.



↓ extended

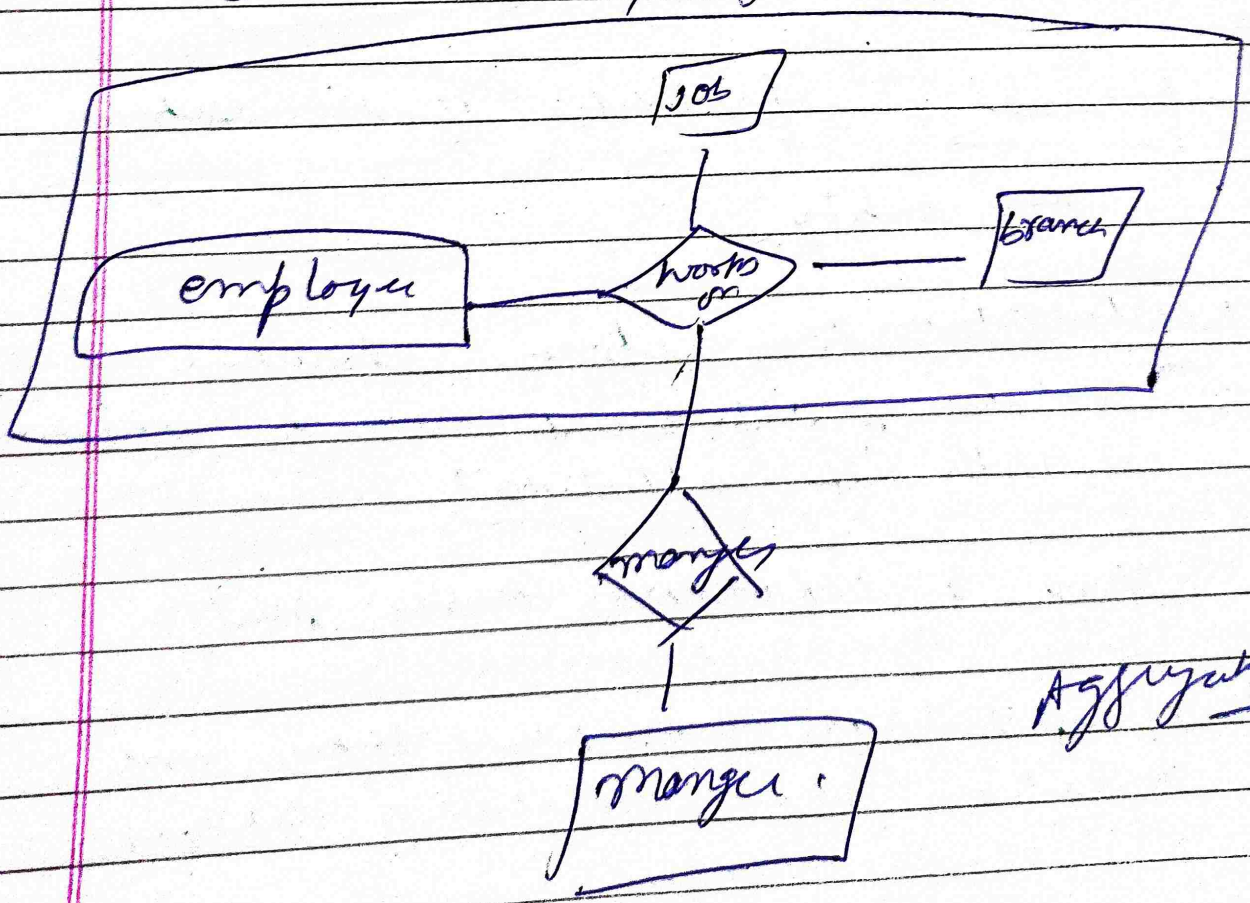
If we want assign manager to each branch, employees, jobs



Redundancy ↑

↓

Combine reduce by aggregation



Aggregation

Page :
Date : / /

Relational Model in DBMS:-

E F codd proposed relational Model to model the data in form of relations or tables.

→ represents how data is stored in Relational databases.

→ relational database consist a collection of tables each of which is assigned a unique name.

RollNo	Name	Add	Phone	Age
1	Ram	Mathura	–	18
2	Rani	Hathras	–	20
3	Ayushi	Rajasthan	–	19

Attribute: property that defines an entity.

Relation Schema: defines structure of relation & represents name of relation with its attributes.

Student (RollNo, Name, Add)

* if a schema has more than one relation then relational schema

Tuple: each row in a relation is called tuple.

Relation Instance: set of tuples of a relation at a particular instance of time is called a relation instance.

Degree: no of attributes in the relation is called degree.

Cardinality: no of tuples in a relation.

Null values: Values which is not known or unavailable called null value
→ represented by blank space.

Relation key: keys that are used to identify the rows uniquely or helps in identifying tables.

- Primary
- Candidate
- Super
- Foreign
- Alternate
- Composite

Constraints: Some conditions which must hold for data present in database called constraints.
→ If there is a violation of any constraints operation fails.

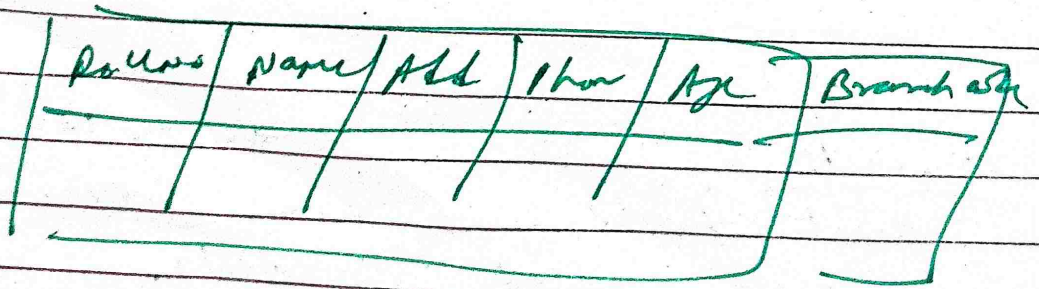
→ Constraints are used to ensure accuracy and consistency of data in relational database.

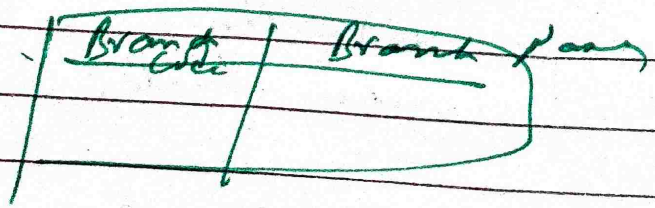
Page:
Date: / /

Domain Constraints : Attribute level constraints.
An attribute can take values only lies in its domain range.
eg Age ?0 , if ge -ve results in failure.

Key Integrity : every relation in a table database must have at least one set of attributes that defines a tuple uniquely. Set of attributes is called ~~Primary~~ key.
key has properties
→ unique for all tuples
→ can't have null values.

Referential Integrity : When one attribute of a relation can only takes values from another attributes of same relation or any other relation is called Referential Integrity.





Page
Date / /

Integrity constraints are set of rules that database is not permitted to violate.

→ ensures that changes (updation, deletion & insertion) made to database by authorized users do not result in loss of data consistency

→ Integrity constraints guard against accidental damage to database.

- Domain (Attribute)
- Entity Integrity (Attribute) (Primary key)
- Key (Attribute)
- Referential Integrity (R/S between Relation)

Keys :- is an attribute or set of attributes that uniquely identifies any record (tuple) from relation/table

→ allows you to find the relation b/w tables

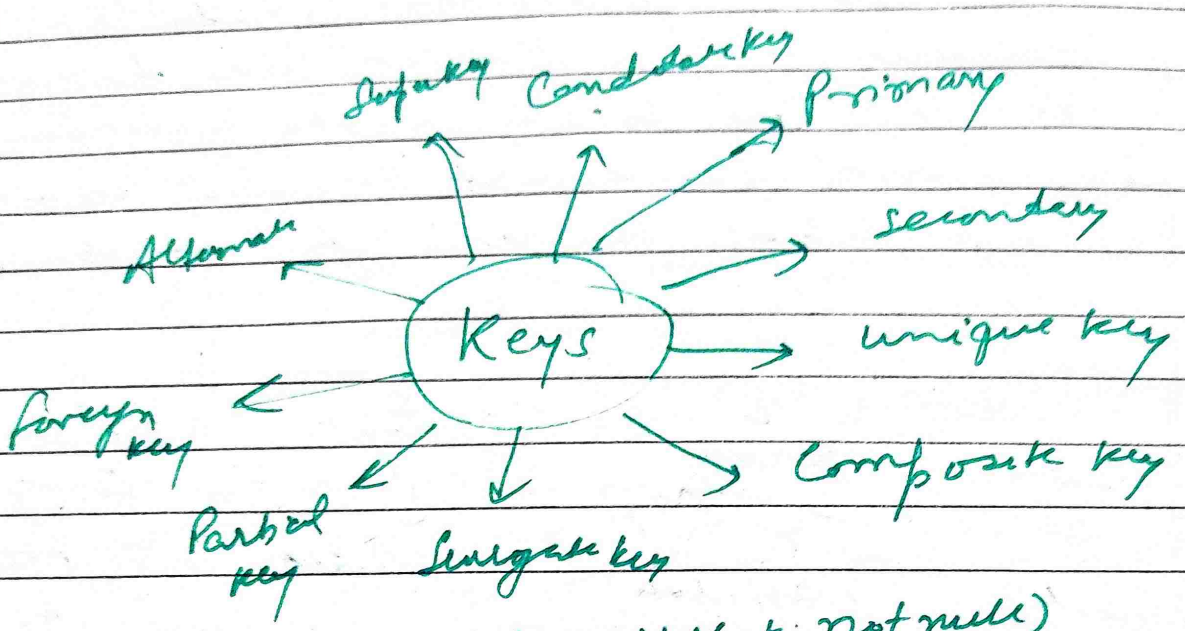
→ helps you to identify a row in a table by a combination of one or more columns in that table.

Purpose

→ to establish & identify the relation b/w tables.

→ Helps to enforce identity & integrity in the relationship

Page:
Date: / /



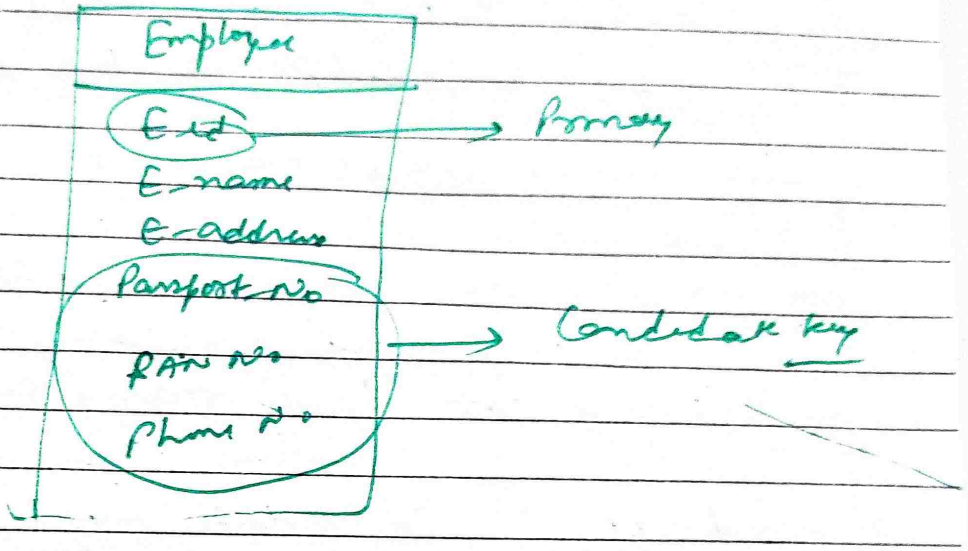
Primary key: (unique + not null)
is one of the candidate key chosen by the database designer to uniquely identify the tuple in a relation

→ Value of primary key can never be null.

→ value of primary key must always be unique (not duplicate).

→ values of primary key can never be changed i.e no updation is possible

→ Value of primary key must be assigned when inserting a record.
→ A relation is allowed to have only one primary key.



Candidate key: attribute or set of attribute which can uniquely identify a tuple

→ remaining attribute except for primary key are considered as candidate key.
Candidate key are as strong as primary key.
→ A table can have multiple candidate keys but only a single primary key.

Properties
→ must contain unique values.
→ CK may have multiple attributes.
→ uniquely identify each record in a table.

Stu_id	RollNo	Fname	Lname	Email

Superkey: → combination of all possible attributes which can uniquely identify two tuples or table
Set of an attribute which can uniquely identify a tuple.

Superkey → superset of Candidate key.

→ may have additional attributes that are not needed for unique identification
→ A table can have many superkeys.

Alternate key: Candidate keys that are left unimplemented or unused after implementing the primary key called alternate key.

Emp_id	Name	Adhar No	Email id	Dept no
				↓ Foreign key

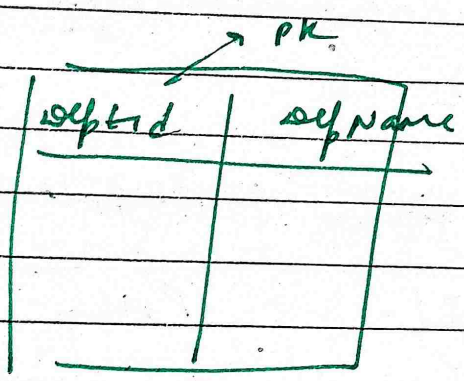
CK → ?
Alternate key → Adhar, email id
PK → emp_id.

Page :
Date : / /

Foreign key. key used to link two tables together.

→ An attribute (or set of attributes) in one table that refers to primary key in another table.

→ Purpose of foreign key is
- to ensure or maintain referential integrity of data.



→ foreign key references the primary of table

→ foreign key can take only those values which are present in primary key of referenced relation

→ Foreign key may have a name other than that of a primary key

Date: / /

→ foreign key can take the null value.

→ no restriction on foreign key to be unique.

→ Referenced relation may also be called as master table or primary table

→ Referencing Relation may also be called as foreign table.

Composite key :

Key that has more than one attributes is known as composite key. also called compound key

Cust id	order id	Product Code	Product Qty
c01	001	P111	5
c02	012	P111	8
c02	012	P222	6
c01	001	P333	9

Composite key :

{ Cust.-id , Product – code }

eg Let a Relation R have attributes (a_1, a_2, a_3), $a_1 \rightarrow ck$. Then how many super keys?

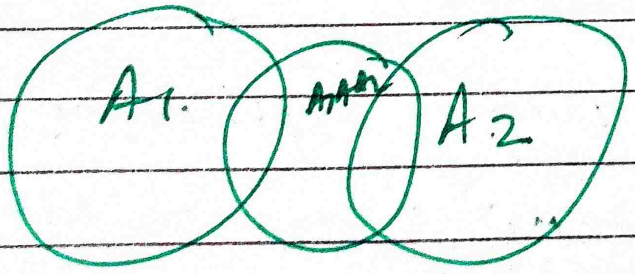
$\{a_1\}, \{a_1, a_2\}, \{a_1, a_3\}, \{a_1, a_2, a_3\}$

2^{n-1} → in General.

eg $R = [a_1, a_2, a_3]$, $ck\{a_1 a_2 a_3\}$

2^{n-3}

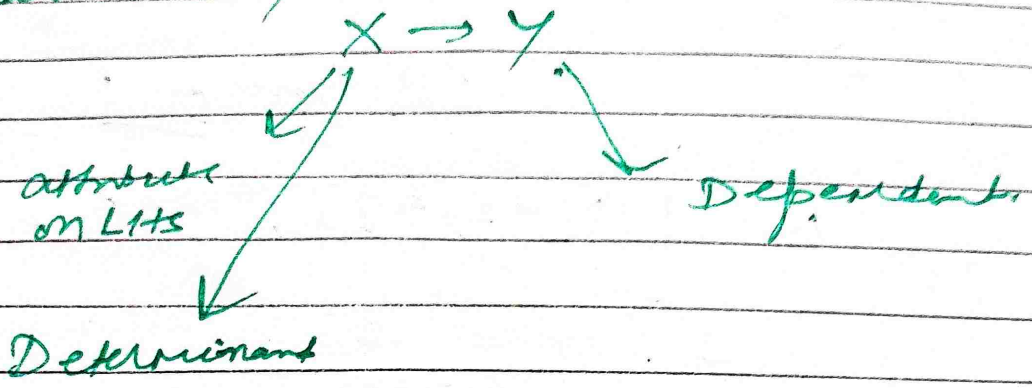
Let a Relation R have attributes $[a_1, a_2, a_3 \ldots a_n]$ & ck as $[a_1, a_2]$ possible no of SK?



$2^{n-1} + 2^{n-1} - 2^{n-2}$

Date: / /

Functional dependencies: concept that specifies the r/ship b/w two sets of attributes where one attribute determines the value of another attribute denoted by



→ used to express mathematically relations among database entities and are very important to understand advanced ~~concepts~~ concepts in RDBMS.

eg (roll_no, Name, dept_name, dept_bulding)

roll_no → {Name, dept_name, dept building}

rollno → Name ✓
rollno → dept_name ✓
dept rollno → dept bulding ✓
} valid

name → dept_name } Y
dept_bulding → dept_name } Y

Page:
Date: / /

Armstrong properties

① **Reflexivity**: if Y is subset of X, then $X \to Y$ holds by reflexivity rule

eg [roll_No, name] → name valid.

② **Augmentation**: if $X \to Y$ is valid dependency then $XZ \to YZ$ is also valid

eg [roll No, name] → dept_~~name~~ builder is valid

then [roll no, name, dept~~name~~] → dept_builder, dept_name is also valid

③ **Transitivity** $X \to Y$

$Y \to Z$

then $X \to Z$ is also valid

eg roll no → dept_name

dept_name → dept_buildy

then

roll no → dept_buildy

Page:
Date: / /

FD Types

- Trivial
- Non Trivial
- Multivalued
- Transitive FD
- fully
- Partial

Trivial FD: a dependent is always a subset of determinant.

i.e. $X \to Y$ ($Y \subseteq X$) then called trivial

{roll no, name} → name → valid

Non Trivial: dependent is strictly not a subset of determinant.

$X \to Y$, Y not subset of X.

roll_no → name ✓

{roll no, name} → age.

Multivalued: entities of dependent set or not dependent on each other.

If $a \to \{b, c\}$, and there exists no FD between b & c then ✓

roll name → {name, age}

q MD

Page:
Date: / /

Transitive FD: dependent indirectly dependent on determinant If a → b, b → c then a → c

Fully FD: an attribute or set of attributes uniquely determines another attribute or set of attributes.

R(X, Y, Z)

X → Y, X → Z full FD

Partial FD: a non key attribute depends on a part of Composite key rather than whole key.

R(X, Y, Z)

Z → non key

X, Y → Composite key

X → Z q partial FD

Page :
Date : / /

Q Consider relation schema
R = {E, F, G, H, I, J, K, L, M, N}

FD are
{E, F} → {G}
{F} → {I, J}
{E, H} → {K, L}
{K} → {M}
{L} → {N}.

What is key for R?

(a) {E, F}

(b) {E, F, H} ✓

(c) {E, F, H, K, L}

(D) {E}

closure set
{E, F} = {E, F, G}
{E, F, H} = {E, F, K, L, M, N, H, G, I, J}?

{E, F, H, K, L} = {E F H G I J K L M N}

{E}+ = E

POCO SHOT ON POCO X2

Page:
Date: / /

Q. R(A, B, C, D, E, H)

$A \rightarrow B$

$BC \rightarrow D$

$E \rightarrow C$

$D \rightarrow A$

What is CK?

(a) AE, BE

(b) AE, BE, DE

(c) AEH, BEH, BCH

(d) AEH, BEH, DEH

(AE) = [AEBCD]

(BE) - [BCD.

formal language
(query language
(what to do &
how to do)

1970 By E.F. Codd

Page :
Date : / /

Relational Algebra: procedural query language.

→ provides a theoretical foundation language for RDBMS & SQL.
→ main purpose is to define operators that transform one or more i/p relations into an o/p relation.
→ pure mathematics, no use of english keywords in Relational Algebra and only operators are represented using symbols.

Relational Algebra Operators

Basic

→ Projection (Π)
→ Selection (σ)
→ Cartesian product or Cross product ($\times$)
→ Union ($\cup$)
→ Set Difference ($-$)
→ Set Intersection ($\cap$)
→ Rename (ρ)

Derived

→ Natural Join ($\bowtie$)
→ Conditional Join

always works on columns

Relation → Student

RollNo	Name	Age
1	A	20
2	B	24
3	A	19

Projection (Π)

for data retrieval (columns).

Retrieve roll no from Student.

→ always in distinct format.

Project → Display

$\Pi_{RollNo}(Student)$

Output

RollNo
1
2
3

by default removes duplicate values

$\Pi_{rollno, name}(Student)$

RollNo	Name
1	A
2	B
3	A

$\Pi_{name}(Student)$

Name
A
B

→ It projects (displays) that columns that satisfy any given predicate.

Here B_1, B_2, B_n represent attributes of Relation R

$\Pi_{(B_1, B_2, B_n)}(R)$

→ Remember that duplicate rows are eliminated automatically, since relation is a set.

Works on row only

Selection (σ)

Query: Retrieve the name of student whose Roll no = '2'

$\sigma_{rollno=2}(Student)$

→ It selects the tuples from a relation that satisfy the given predicate.

notation is $\sigma_{p}(r)$

p → selection predicate

r → relation

Proposition logical formula that may use connector such as or, and, not

→ relational operator $=, \neq, \geq, <, >, \leq$

eg $\sigma_{Subject = \text{"Information"}}(Novels)$

Note: Selection operator only selects the required tuples but does not display them. For display data, projector operation is used.

eg $\Pi_{name}(\sigma_{rollno=2}(Student))$

→ It helps in combining data & info of two different relations into a single one.
notation :- (R × S)
S × R = { t ∈ S and ... }

Cross product : (Join)

R_1

A	B	C
1	2	3
2	1	4

R_2

C	D	E
3	4	5
2	1	2

$R_1 \times R_2$

A	B	C	C	D	E
1	2	3	3	4	5
1	2	3	2	1	2
2	1	2	3	4	5
2	1	4	2	1	2

{ If m column in R_1
n column in R_2
(m+n) → no of columns

Rows 2 in R_1
2 in R_1
no of tuples (2×2) = 4

→ at least one column should be common otherwise no join

1,2,3 S_1

3,4,5 S_2

Page :
Date : / /

Union : $S_1 \cup S_2$

1,2,3,4,5

→ no of columns must be same in number.
→ Domain of every column must be same
→ R ∪ S
→ If relations don't have same set of attributes, union results in null. → Binary operator

RollNo	Name
1	A
2	B
3	C

Student

Emp no	Name
7	E
1	A

Employee

(Student) ∪ (Employee) =

RollNo	Name
1	A
2	B
3	C
7	E

Π_{Name} (Student) ∪ Π_{name} (Employee)

Name
A
B
C
E

Set Difference (−)

→ name indicates the difference b/w two relations (R−S). denoted by hyphen (−) and it returns all the tuples which are in Relation R but not in relation S.

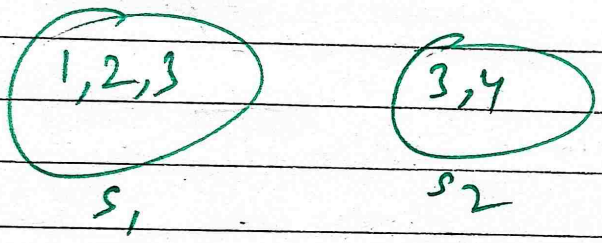
→ Binary operator.

→ No of columns must be same in number

→ Domain of every column must be same

(A−B) = A but not B

= $A \cap B'$



$S_1 - S_2 = \{1, 2\}$

→ not Commutative i.e $S_1 - S_2 \neq S_2 - S_1$

Query: Find name of person who is student but not an employee.

$$\Pi_{name}(Student) - \Pi_{name}(Employee)$$

Rename (ρ)

→ Rename operation denoted by "Rho" (ρ).

→ As its name suggests it is used to rename the O/P relation.

→ Also a binary operator.

$P(R, S)$ → old relation
↓
New relation

Page :
Date : / /

ρ (Student_name , Π_{name} (Student))

Intersection ($\cap$)

Rollno	name
1	A
2	B
3	C
4	D
5	E

Student

Emp_id	Name
1	B
2	E
3	G
4	O
5	C

emp

Π_{name} (student) $\cdot \cap$ Π_{name} (Emp log n)

Name
B
C
E

Commutative

$A \cap B = B \cap A$.

Join = Cross product + some condition
Or
Cross Product + select Statement

Page:
Date: / /

Join: are binary operations that allow us to combine two or more relations
→ There must be some common attribute to perform join.

Join
- Inner
  - → Equi
  - → Theta
  - → Natural
- Outer
  - → Left outer
  - → Right outer
  - → Full outer

Employee

E_No	E_Name	City	Experience
E_1	Ram	Delhi	04
E_2	Varun	Chandigarh	09
E_3	Rani	Noida	03

Department

D_No	D_Name	E_No	Mon_Exp
D_1	HR	E_1	03
D_2	IT	E_2	05
D_3	Sales	E_3	02

Inner Join: When we perform inner join only those tuples returned that satisfy the certain condition.

Page:
Date: / /

Theta (θ) joins conditions two relations using a condition. Condition is represented by symbol (θ).
↓ >, <, >=, <=

Notation — R ⋈θ S
R: First Relation, S: Second Relation

e.g. Suppose we want a relation where experience from Employee >= Min-Experience from dept

Employee ⋈θ Employee.Experience >= Department.Min_Experience

Equi Join: Special case of theta Join where condition can contain only equality comparisons.

e.g. Employee ⋈ Employee.E_no = Department.E_no
Department

Natural Join:- a comparison op. operator is not used in a natural join. It does not concatenate like Cartesian Product.

⇒ Natural join can be perform only if two relations share at least one common attribute.

→ Operates on matching attributes where values of attributes in both relations are same & removes duplicate ones.

→ Preferably Natural Join is performed on foreign key.

R ⋈ S

Outer Join: Outer join also includes some/all tuples which don't satisfy the given condition.

Left Outer: Returns matching tuples & tuples which are only present in left relation, R. However if matching tuples are null, then attributes/columns of right relation are made null in S relation.

eg Employee ⟕ Employee.E_no = Department.E No Department

Right outer ⟖

Full outer join ⟗

Division operation: ÷ or /

→ used in queries that involve keywords "every", "all" etc.

$R(X,Y) / S(Y)$

Enrolled(S_id, C_id) / Course(Course_id)

Page :
Date : / /

Q Consider the relational schema

Passenger (Pid, Pname, Pgender, Pcity)
agency (aid, aname, acity)
flight (fid, fdate, time, src, dest)
booking (pid, aid, fid, fdate)

(a) Get the complete details of all flights to Newdelhi

$\sigma_{dest = Newdelhi}$ (flight)

(b) Get all flights from chennai to newdelhi

$\sigma_{src = "chennai" \wedge dest = "newdelhi"}$ (flight)

(c) Find the passenger name who have booking for at least one flight

Π_{Pname} (passenger $\bowtie$ flight)

Page:
Date: / /

Q Consider relational database

Student (RollNumber, StudentName, Address)
Teachers (TeacherID, TeacherName, TeachingSubject)
College (RollNumber, TeacherID).

a) Find name of students who live in Lalitpur.
b) Find name of teacher who teach DBMS subject.
c) Find name of teacher who teaches CO subject to JohnSmith student)
d) Insert a tuple into relation teacher.
e) Delete records of students whose address is "Pokhara"

Soln a) $\Pi_{StudentName}(\sigma_{Address=Lalitpur}(Student))$

b) $\Pi_{TeacherName}(\sigma_{TeachingSubject=DBMS}(Teachers))$

c) $\Pi_{TeacherName}(\sigma_{TC="CO" \wedge StudentName="JohnSmith"}(Student \bowtie College \bowtie Teachers))$

d) $Teachers \leftarrow T \cup \{"101", "Ram", "PCPH"\}$

e) $S \leftarrow S - (\sigma_{Address=Pokhara}(Student))$

Page :
Date : / /

Q Branch (branch-name , branch-city, assets)

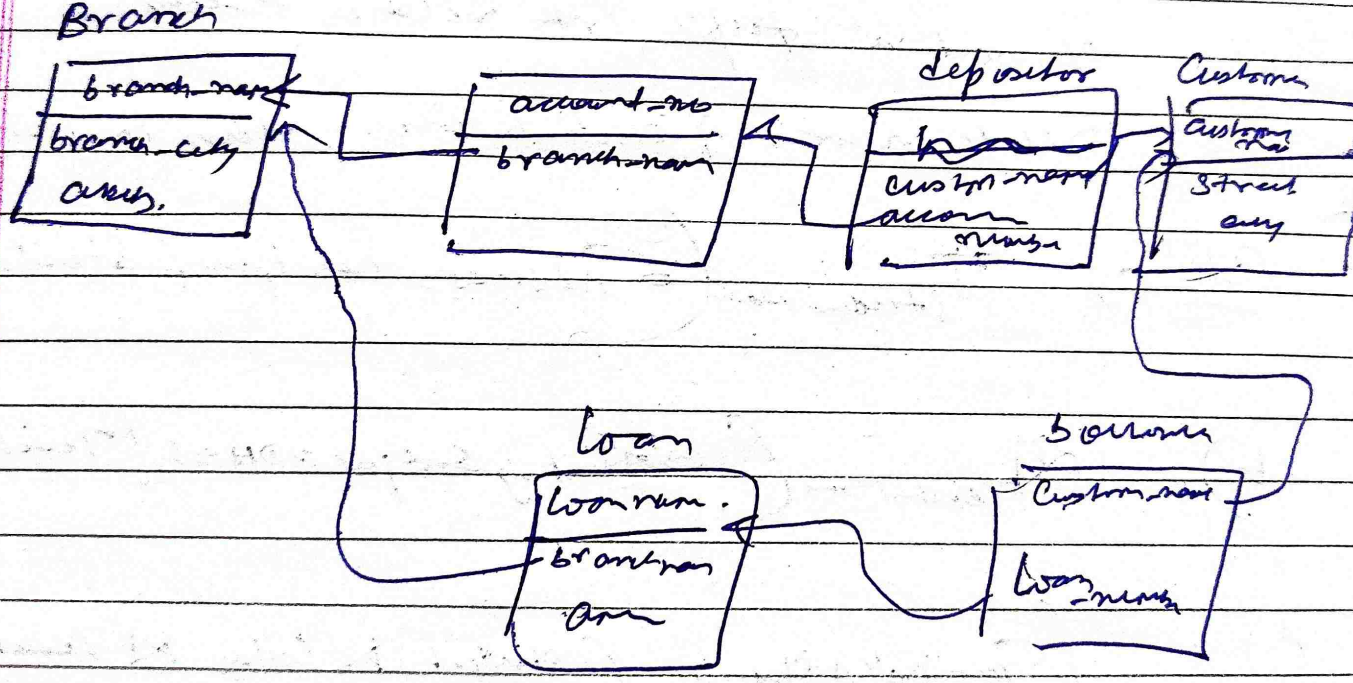
Customer (Customer-name , customer-street , customer-city)

account (account-number , branch-name , balance)
F

loan (loan-number , branch-name , amount)
F

depositor (Customer-name , account-number)
P, F F

borrower (Customer-name , loan-number)
P, F F



a) Find all loans made at "Perryridge" branch

$\Pi_{}$ (σ Branch-name = Perry (loan)

b) Find all loans of over $1200

$\sigma_{amount > 1200}(loan)$

c) Find all tuples who have taken loans of more than $1200 made by "Perryridge" branch

$\sigma_{branch\ name = "Perryridge" \wedge amount > 1200}(loan)$

d) Find all loan-number & amount of loans.

$\Pi_{loan\ number,\ amount}(loan)$

e) Find loan number for each loan of amount greater than $1200

$\Pi_{loan\ number}(\sigma_{amount > 1200}(loan))$

f) Find those customers who live in "Harrison"

$\Pi_{customer\_name}(\sigma_{Customer\_city = "Harrison"}(Customer))$

g) Find names of all customers who have a loan, an account, or both from bank.

$\Pi_{customer\_name}(borrower) \cup \Pi_{Customer\ name}(depositor)$

h) Find the names of all customers who have a loan & an account at bank.

$\Pi_{customer\_name}(borrow) \cap \Pi_{customer\_name}(depositor)$

i) Find names of all customers who have an account but no loan from bank.

$\Pi_{customer\_name}(depositor) - \Pi_{customer\_name}(borrow)$

j) Find names of all customers who have a loan at Perryridge branch.

$\Pi_{customer\_name}$

$(\sigma_{branch\_name = "Perryridge"}$

$(\sigma_{borrower.loan\_number = loan.loan\_number}$

$(borrower \times loan)))$

Anomaly :→ Flaw in database that occurs bcoz of poor planning & redundancy.

Idea : In table Student info we have tried to store entire data about student

Student info

S_id	Name	Age	Branch code	Branch name	HOD name
1	A	18	101	CS	Y42
2	B	19	101	CS	Y42
3	C	18	101	CS	Y42
4	D	19	102	EC	PQR
5	E	20	102	EC	PQR
6	f	21	103	ME	KLM

Result. entire branch data of a branch must be repeated for every student of branch.

Redundancy : When data is stored multiple times unnecessarily in a database

Disadvantages : (1) Insertion, deletion & modification anomalies
(2) Inconsistency
(3) Increase in database size & increase in time (slow)

Insertion Anomaly : When certain database cannot be inserted into database without presence of other data

Spiral

Page No.
Date

Deletion anomaly : If we delete some unwanted data (attributes), it cause deletion of some other data (wanted).

Updation/Modification Anomaly : When we want to update a single piece of data, but it must be done at all copies.

Normalization : process of organizing the data and attributes of a database.
→ It is performed to reduce the redundancy in a database & to ensure that data is stored logically.
→ process of decomposing relations into relations with fewer attributes.

Why do we need Normalization ?
→ To remove data anomalies
→ removal of data redundancy.
→ Maintain consistency & Integrity of data.

Normalization : can be done through a series of stages called Normal forms.

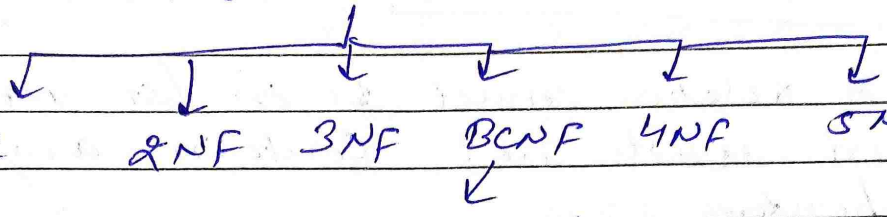
→ N.F can be applied to individual relations.
→ Relation is said to be in a particular Normal form if it satisfies constraints.

Spiral

Advantages:
→ Reduced Data Redundancy
→ Improved Data consistency
→ Simplified db design
→ Improved Query Performance, Easier db maintenance

Page No. _____
Date

Normal forms



First Normal form: A relation is in 1st NF if it does not contain any Composite or Multivalued attribute.
→ A relation is in 1NF if attribute is only a single valued attribute.

Student

Sid	S_Name	S_course
Omkar	1	Java, Python
Rani	2	C
James	3	DBMS, OS

↑ Not in 1st

S_id	S_Name	S_course
1	Omkar	Java
1	Omkar	Python
2	Rani	C
3	James	DBMS
3	James	DBMS

(1)

S_id	S_name	course

S_id	SName

Sid	Scourse
1	

Spiral

Page No. _____
Date | | | |

Second Normal form :

→ A relation must be in 1st N.F. and must not contain any partial dependency.

→ No PD (i.e no non prime attribute is dependent on any proper subset of any candidate key of table. fully

→ All / non prime attribute should be functionally dependent on CK, PD → if proper subset of CK determines non prime attribute, called PD.

Stud-No	Course-No	Course fee

Q R(A, B, C, D)
AB → C
BC → D

(AB)

Q Find highest NF R(A, B, C, D, E)
BC → D
AC → BE
B → E

$AC^+ = [A\ C\ B\ E\ D]$

Customer

Customer_ID	Store_Id	location
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Banglore
4	3	Mumbai

CK → [Customer_ID, Store_ID] → composite

Prime Attri

Non Prime → location

after Decomposing

CK

Cust_Id	Store_ID
1	1
1	3
2	1
3	2
4	3

PK

Store_Id	location
1	Delhi
2	Baglor
3	Mumbai

Cust_Id, Store Id → location

Store_Id → location

eg

Partial CK

Non Prime

AB

A C

AB →

A →

B →

(*) LHS should be proper subset of CK and RHS should be a Non Prime
↑ condition for Partial Dependency

Page No. _____
Date

Q R (A B C D E F)

FD: C → F
E → A
EC → D
A → B

Step 1 Find CK
EC
EC+ = {A B C D E F}

Step 2 Prime Attributes
{E C}

Step 3 Non Prime attributes
{A B D F}

Now check one by one

C → F → Hence PD
↓ ↓
proper subset of CK
Non Prime Attr

E → A (PD)

EC → D ✓

A → B
↓ ↓
PP NP ✓