16.12.19
Monday

CS8492 - DATABASE MANAGEMENT SYSTEMS

Unit - 1 : RELATIONAL DATABASES.

Purpose of Database system - Use of data - Data models - Database System architecture - Introduction to relational Database - Relational models - keys - Relational algebra - SQL fundamentals - Advanced SQL features - Embedded SQL - Dynamic SQL.

Unit - 2 : DATABASE DESIGN

Entity Relationship model - ER diagrams - Enhanced ER diagrams - ER to Relational mapping - Functional dependancy - Non loss decomposition - 1st, 2nd, 3rd normal forms. dependancy preservation Boyce/codd Normal form - multivalue dependancy and 4th Normal form - Join dependancy and 5th normal form.

Unit - 3 : TRANSACTION

Transaction Concepts - ACID property - schedules - serializability - Concurrency control - Need for Concurrency - locking protocols - Two phase locking - Deadlock - Transaction recovery - save points isolation levels - SQL facility for concurrency and recovery.

Unit-4: IMPLEMENTATION TECHNIQUES.

RAID - File organisation - Organisation of Records and files - Indexing and Hashing - Ordered Indice - B^+ tree index file - B tree index file - static Hashing - Dynamic Hashing - Query processing overview - Algorithm for SELECT and JOIN operations - Query optimisation using Heuristics and cost estimation.

Unit-5: ADVANCED TOPICS

Distributed DataBase - Architecture - Data storage - Transaction processing - object based Database - object Database Concepts - object Relational features - ODMG object Model, ODL, OQL, XML Databases, XML Hierarchical Model, DTD XML Schema, X query Information Retrieval - IR Concepts - Retrieval models - Queries in IR Systems.

Textbook:

1. Abraham Silberschatz, Henry F. Korth, S. Sudharsan "Database system Concepts" - 6th Edition.

2. Ramez Elmasri, Shamkant B Navathe "Fundamentals of Database System" - 6th Edition.

17.12.19
Tuesday

UNIT - I.

RELATIONAL DATABASE

Introduction:

DATA:

Data is a collection of facts and figures that can be processed to produce information. In otherwords, data can be any character text, word, number, etc.,

INFORMATION:

Information is a collection of data which give a convenient meaning

DATABASE:

DataBase is a collection of inter related data which is use to retrieve, insert and delete the data efficiently

Eg:

Employee ⟶ Table name (or) File name

E-ID	E-Name	E-designation
1001	Birundha	Manager

Header row ⟹ field / attributes.
Data row ⟶ tuples / record

DBMS:

The DBMS is the software that facilitates the creation, storage, retrieval and manipulation of datas in the database.

It also provide protection and security to the database. It also maintain data consistency incase of multiple users.

Some examples of popular DBMS are:

- My SQL
- Oracle
- SQL server
- IBM DB_2
- PostgreSQL
- Amazon Simple DB.

~~Role~~ Goal of DBMS:

The Goal of DBMS is to provide a convenient and effective method of defined storing and retrieving the information contain in a database

Disadvantage of File system over database system:

* <u>Data redundancy</u> (repetition)

Data repetition is possible that the same information may be duplicated in different files this lead to data redundancy result in memory wastage.

* **Data inconsistency**

Because of data redundancy it is possible data may not be in consistent state.

* **Difficult in accessing data**

Accessing data is not convenient, and efficient in file processing system.

* Limited data sharing
* Integrity (combining) problem
* Atomicity problem
* Concurrent access
  ↳ simultaneously many people use at a time
* Security problem

Views of Data:

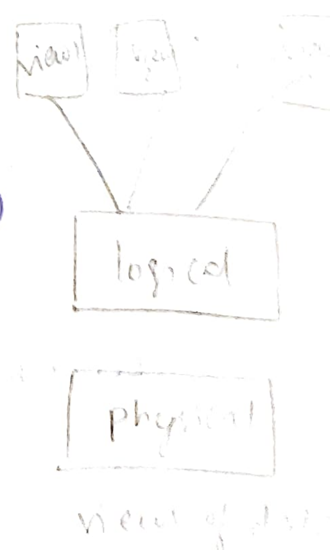
Data abstraction. (hiding the

3 levels.

physical (Internal level)

logical (conceptual level)

view (External level)



Schema:

Design of a db is called Schema.

Instance:

Data stored in the db at the moment of time. Eg stack Market

Data Model:

Object based Logical model.

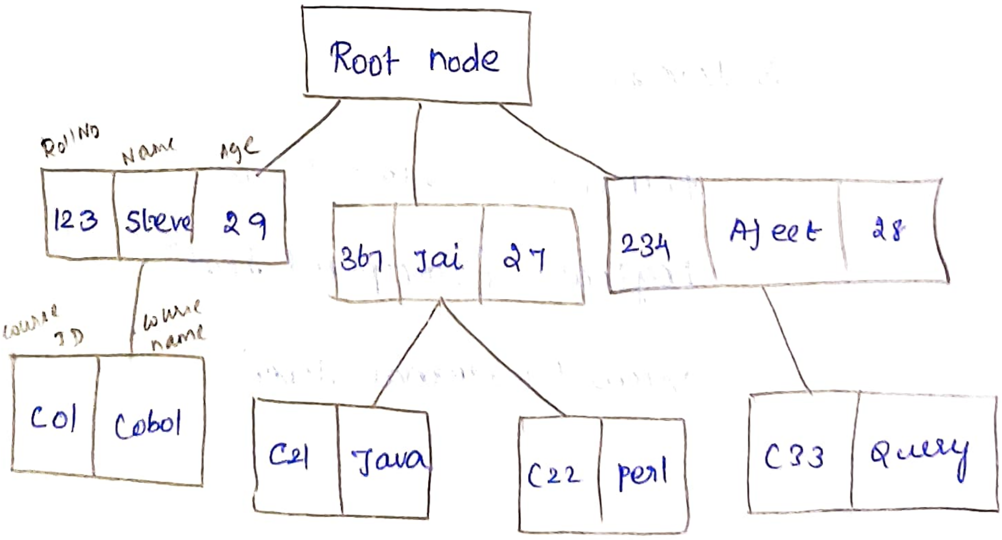
Eg: ER Model
OO Model

Record based logical model

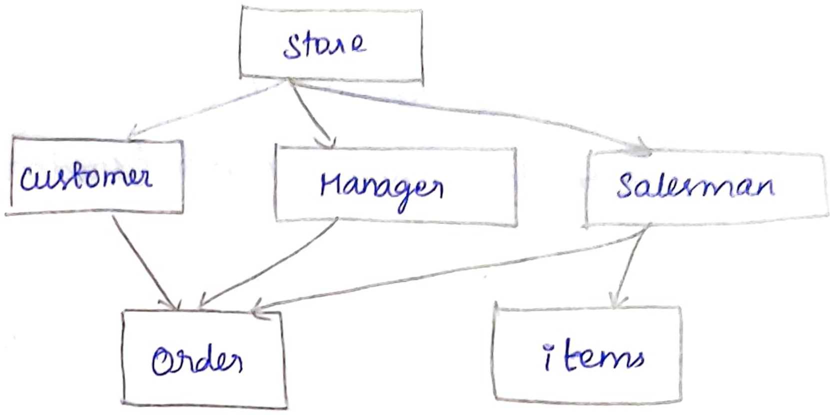
- Hierachical Model
- Network model
- Relational model

⇒ Data
⇒ Data relationship (relation b/w datas)
⇒ Data semantics (whether it gives a meaning for information about the database)
⇒ Consistency

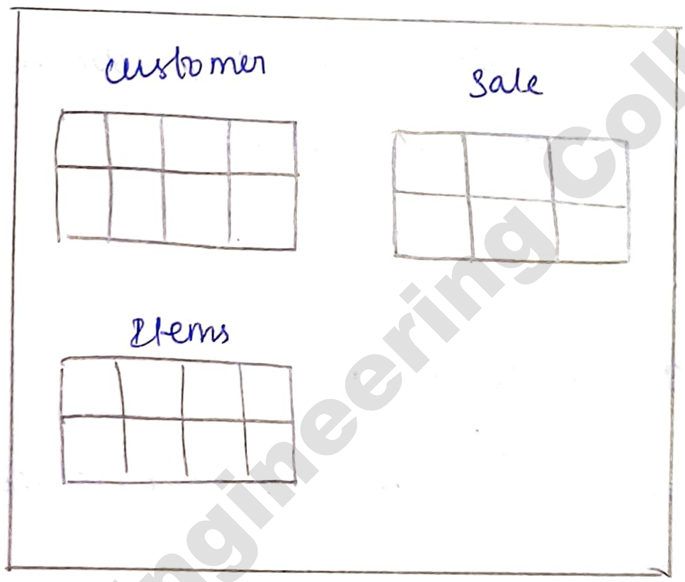
1. Hierarchical Model:



2. Network Model:



3. Relational Model:



19.12.19 Thursday

SQL Commands:

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Control Language (DCL)
4. Transaction control language (TCL)

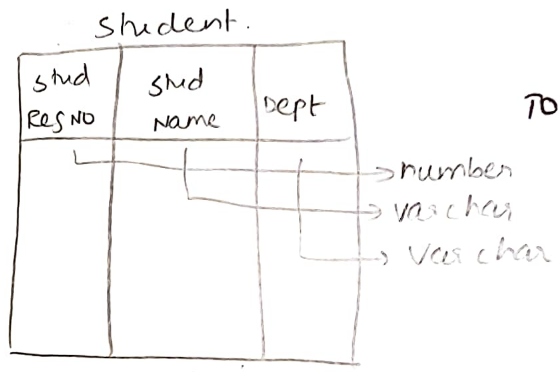
1. DDL.

1. Create - used to create a table
2. Alter → alter the table
3. Drop → used to delete
4. Truncate → used to delete the record but retain the structure.

SQL> Create table student (Stud Regno number (3), studname varchar(20), dept varchar(4));

SQL> alter table student add (age number(2));

alter table student modify (studname varchar(15));



To display:

select * from student
(or)
Table Name

SQL> alter table student drop column age.

SQL> truncate table student.

2. DML:

1. Insert Command

Syntax: insert into <table name> values (a list of data);

Eg:

insert into student values (5001, 'ABI', 'IT');

2. Delete

Syntax: delete from <tablename> where <condition>;

Eg:

delete from Student where dept = EEE;

3. Update Command

Syntax: update <table name> set field = value, ... where <condition>;

Eg:

update Student Set dept = CSE where studname = 'sai';

3. DCL

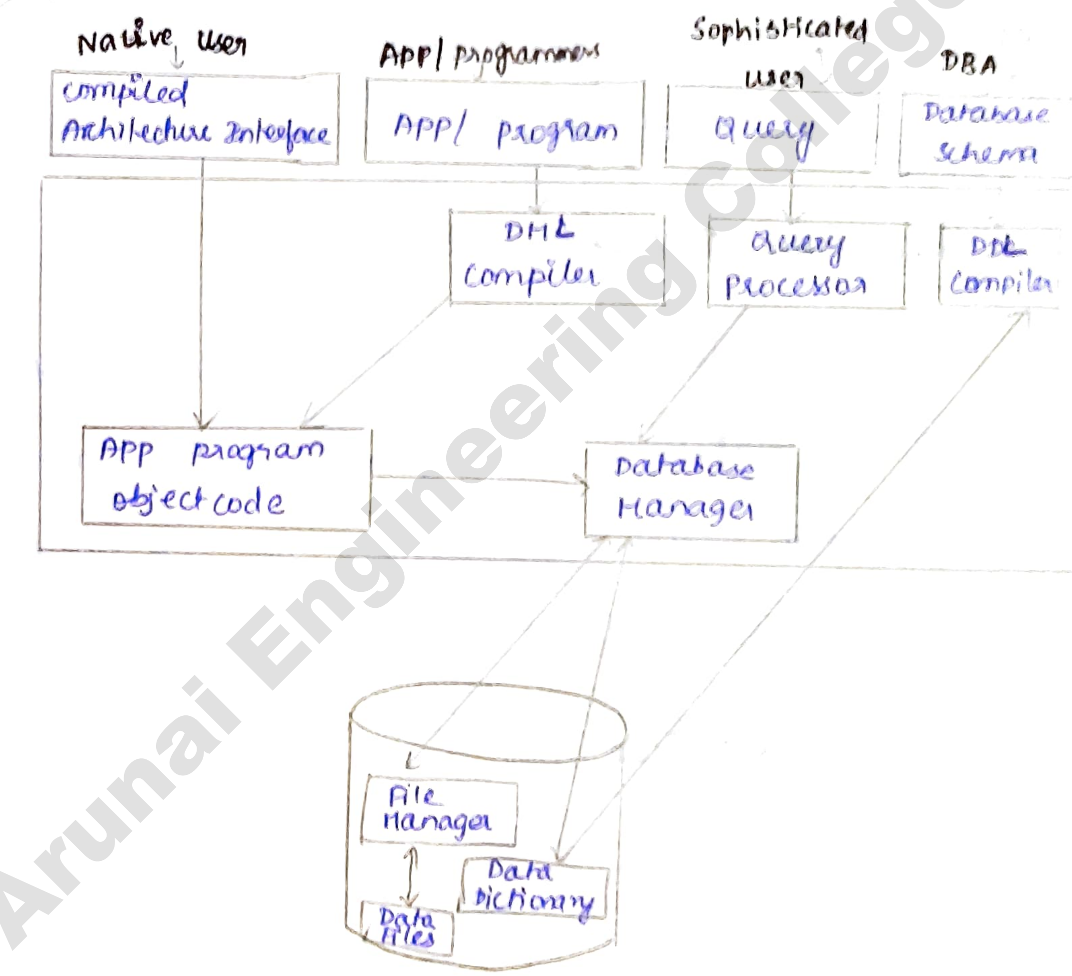
1. Grant
2. Revoke

4. TCL

1. Roll back
2. Commit
3. Savepoint

20/12/19 Friday

DATABASE SYSTEM ARCHITECTURE : (Component)



3 units:

1. Storage Manager

→ Authorization & Integrity manager

→ Transaction Manager

→ File Manager

→ Buffer Manager

2. Query processor:

→ DML Interpreter

→ DML compiler

→ Query Evaluation Engine

3. Database user:

→ Appl programmers

→ Naive user

→ Sophisticated user

→ DBA – DB Administrator

→ Specialized user.

RELATIONAL DATABASE:

A Relational Database Management System (RDBMS) is a database management system based on relational model introduced by E.F. Codd. In this model datas are represented in terms of tuples.

E.F. Codd's Rule:

Rule 1: Information Rule:

All information in a relational database is represented explicitly at the logical level an in exactly one way – By values in the table.

Rule 2: Guaranteed Access Rule:

Each and every atomic value in a relational database is guaranteed to be logically accessible by restoring to a combination of table name, Primary key value and column name.

03-01-2020 Friday.

Rule 3: Systematic treatment of null value

Rule 4: Dynamic online catalogue based on the relational model.

Rule 5: Comprehensive data sublanguage rules.

Rule 6: View updating rules.

Rule 7: High level insert, update and delete

Rule 8: Physical data independency

Rule 9: Logical data independency

Rule 10: Integrity independency

Rule 11: Distribution independency

Rule 12: The Non Sub Version Rule

Keys:

A key allow us to identify a set of attributes and thus distinguishes from each other.

Different types of keys are:

Candidate key

super key

Primary key

Foreign key

Primary key:

An attribute which is unique and not null can identify an instantance of an entity set is called primary key. For Example,

Employee (E-NO, e-name, date-of-join, salary, DOB, Job).

In this e-no is the primary key

Super key:

super key is defined in a relational model of a database organisation as set of attributes of a relational variable for which it holds that all relational assign to that variable there are no two distinguish tuples that have the same value for the attribute in that set.

For example: Consider the student relation Student (Roll-no, Name, Age).

Candidate key:

A table which have more than one attribute that uniquely identify and instance of an entity set. These attributes are called Candidate key.

For example:

Car(License-No, Engine serial-no, make, model, year)

In this relation we can find two candidate key License number and Engine Serial number.

Foreign key:

An attribute is one relation whose value matches the primary key in some other relation is called a foreign key.

For example: Consider two relation, department and employee.

Dept (D-NO, D-name, D-loc)

Employee (E-no, e-name, D-O-J, D-O-B, salary, Job, D-NO)

In the above relations for department D-NO is primary key. For Employee, e-no is the primary key and here we can find that employee relation D-NO matches with Dept D-NO, So that employee relation D-NO is known as foreign key

07-01-2020
Tuesday

Relational Algebra:

Procedural Language:

* The user instruct the system to perform a sequence of operation on the database to compute the desired result.

Eg: Relational Algebra.

Non procedural Language:

* The user describe the desired information without giving a specific procedure for obtaining that information.

Eg: Relational Calculus.

Relational Algebra Definition:

* It consists of set of operation that take one or more relations as input and produce new relations as output.

The operations can be divided into

1. Set operations
2. Algebraic operations
3. Additional operations

~~+ Set operations~~ :

* selection:

The operation is used to fetch rows (or) tuples

Syntax:

$\sigma_{Predicate}$ (relation)

where σ represent the select operation, the peredicate denotes some logic using which the data from the relation is selected.

* projection:

Project operation is used to project only a certain set of attributes of a relation. (i.e used to display particular column from the relation)

Syntax:

Π $c_1, c_2, \ldots\ldots$

where are c_1, c_2 are attributes name.

Eg: $\Pi_{Regno, sname}$ (student)

1. Set Operations:

* Union:

This operation is used to fetch data from two relations. (Table)

For this operation to work the relation specify should have same no. of (column) attributes & same attribute domain.

and also duplicated tuples are automically eliminated from the result.

Syntax:

$A \cup B$

Eg:

$\pi_{sname}(student) \cup \pi_{dept}(college)$

★ Intersection:

This operation is used to fetch data from both the table which is common in both the table.

Syntax:

$A \cap B$

where A and B are relations.

Eg:

$\pi_{sname}(student) \cap \pi_{name}(worker)$

★ cartesian product:

It is used to combine data from two different relations into one and fetch data from Combined relation

Syntax: $A \times B$.

Eg:

$\sigma_{Student.Sid = Reserve.Sid \wedge Reserve.Isbn = 005}$ (student × Reserve)

output

Sid	Sname	age	Isbn	day
1	Ram	21	005	7/7
2	Shyam	18	005	7/7

★ set difference

The result of set difference operation is tuples which are present in one relation but are not in the second relation.

syntax: A – B.

★ Join (⋈)

The operation join is used to combine information from two or more relations Commonly join can be defined as cross product followed by projections and selections. The join operation is used as ⋈.

There are three types of joins

1. Conditional join

This is an operation in which information from truth table is combined using some condition and this condition is specified along the join operation.

syntax:

$$A \bowtie_c B = \sigma_c (A \times B)$$

Eg:

$(\sigma_{(studentid = Reserve\ id) \wedge (Reserve.Isbn = 005)} (student \times Reserve))$

$(student \bowtie_{(studentid = Reserve.id) \wedge (Reserve.Isbn = 005)} Reserve)$

2. Equal join

This is a kind of joint in which there is equality condition between two attribute (column) of relations.

Reserve

Sid	isbn	day
1	005	7/7
2	005	3/7
3	007	8/7

Book

isbn	bname	author
005	DBMS	XYZ
006	OS	PQR
007	DAA	AB

Eg:

$$\sigma_{bname='DBMS'}(Book \bowtie_{(Book.Isbn = Reserve.Isbn)} Reserve)$$

Output

Sid	isbn	day	bname	author
1	005	7/7	DBMS	XYZ
2	005	3/7	DBMS	XYZ.

3. Nature join

When there are common columns we have to equalate. this common columns then we use nature join. the symbol for nature join is ⋈ without any conditions.

Rename Operation:

Syntax:

$$P(Relationnew, Relationold)$$

This operation is used to rename the output relation for any query operation which returns like select, projection

Divide Operation:

The division operation is used when we have to evaluate queries which contain the keyword 'ALL'.

It is denoted by A / B where A and B are instance of relations.

09.01.2020 Thursday.

Advanced SQL features:

Retrieve data.

display the whole table

select * from table name.

select $A_1, A_2, A_3 \ldots$ from $r_1, r_2, \ldots$ where p.

Relational algebra

$$\Pi_{A1, A2, A3} (\sigma_p(r_1, r_2, r_3 \ldots r_n))$$

A - Attributes

r - relation (table)

p - Predicate

Order by

select sname from student where dept = 'CSE' orde by regno;

↳ order by reg no (or) asc (or) desc

Group by

select Name, age FROM patients where age > 40 Group by Name order by name;

Manipulation:

1. Sum

select sum (salary) from Employee where age>35;

2. Avg

select Avg (Price) from product;

3. Count

select count (Reg no) from student;

4. Max

select max (price) from product;

5. Min

select min (price) from product;

NULL values:

select Customer name from bank where loan = null;

* Any absence of values

10.01.2020 Friday.

Substring Comparison:

1. % Percentage
2. (_) underscore.

_a (second char of name)
__a (third char)

Name	percent	Result
kumar	[illegible]	P
[illegible]	9[illegible]	P

select s name from student where address like %Tiruvannamalai

3. AND

select sname from student where Result = pass AND percent > 7[illegible]

View:

A View is a tailored table that is formed at the result of a query. It has tables and rows just like any other table.

It is usually a good idea to run queries in SQL as independent views because this allow them to be retrieved later, to give view the query result rather than computing the same command everytime as a particular set of result.

Creating a view

create view failing student AS select sname, Regno from student where CGPA < 40;

Retrieve or display the view

select * from failing student;

Update view

Replace view [product list] as select PID, PName, category from products where discontinued = no;

Drop

(what could be drop?)

Drop view newtable;

Nested Queries:

↳ in (compare)

select distinct customer name from depositor where customer name in (select customer name from borrower);

↳ not in (complement of in)

not in

select distint customer name from borrower where customer name not in (select customer from depositor);

20.01.2020 Monday

Joins:

- Inner Join
- outer Join
  - Right join
  - Left join
- Natural Join

Inner join:

student (sname, place)

student_mark (sname, dept, mark)

student

sname	place
prajan	chennai
Anand	kolkata
kumar	Delhi
Raju	mumbai

Student mark

sname	dept	mark
prajan	CSE	700
Anand	IT	650
vasu	CSE	680
Raju	IT	600.

select student. sname, studentmark. mark from student innerjoin Studentmark on student. sname = Studentmark .sname

output:

sname	mark
Prajan	700
Anand	650
Raju	600

outer Join:

Left outer join

The left outer join returns matching rows from the table being joint and also non-matching rows from the left table in the result and places Null value in the attributes that comes from the right table.

Eg:

select student.sname, studentmark.mark from student leftouterjoin studentmark on student.sname = studentmark.sname.

output:

Sname	mark
Prajan	700
Anand	650
Raju	600
kumar	NULL

Right-outer join:

The right outer join operation returns matching rows from the table being joint and also non-matching rows from the right table in the table in the result and places Null value in the left attributes that comes from the left table.

Eg:

Select student.sname, student.place, studentmark.mark
from student right outer join
studentmark on student.sname = studentmark.sname

output

Sname	place	Mark
Prajan	chennai	700
Anand	kolkata	650
Vasu	NULL	680
Raju	Mumbai	600

Embedded SQL

host language

EXEC SQL or EXEC SQL BEGIN

END-EXEC (or) EXEC SQL END (or) semicolon.

Embedded SQL in C program:

Eg1: /* Variable Declaration in language C */

```
EXEC SQL BEGIN DECLARE SECTION;
varchar dname [16], fname[16];
char ssn[10], bdate [11];
int dno, dnumber, SQL CODE....;
EXEC SQL END DECLARE SECTION;
```

22.01.2020 Wednesday

DYNAMIC SQL:

Use of dynamic SQL:

It is generally used for situation where data is distributed non uniformly

It enable you to write reusable code that can be adopted for different environment.

25.01.2020 Saturday.

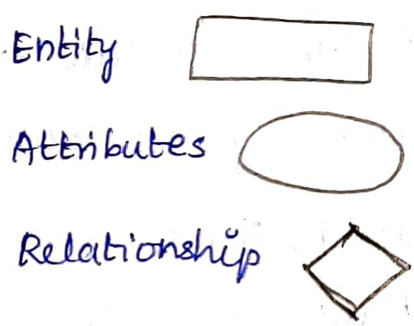
Unit - II

DATABASE DESIGN

Entity Relationship Model:

The entity relationship model which is popular for database high level design provides means for representing relationship between entities.

Notation used



Entity:

An entity is a thing or a object in the real world that is distinguishable from all other objects.

Eg: A particular person, car, house etc., Entity set:

An entity set is a collection of entities having same properties.

The individual entities in an entity set are called the extension of the entity set.

Attributes:

The property that describes an entity are called attributes.

An attributes can be classified into various types:

1. Simple attribute ⇒ an attribute that cannot be divided into further subpart

   Eg: customer id.

2. Composite attribute ⇒ an attribute that can be divided into set of subparts.

   Eg: The attribute name can be divided into first name, middle name & last name

3. Single value attribute ⇒ an attribute having only one value in a particular entity.

   Eg: Id, Street.

4. Multi value attribute ⇒ an attribute having more than one value for particular entity.

   Eg: Phone number.

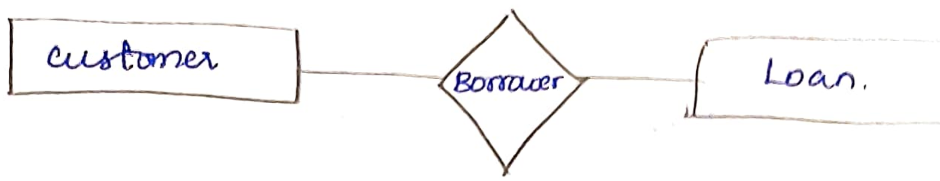
5. Derived attribute ⇒ an attribute derived from other related attributes

   Eg: Age from D.O.B.

Relationship :

Relationship is an association among several entities.

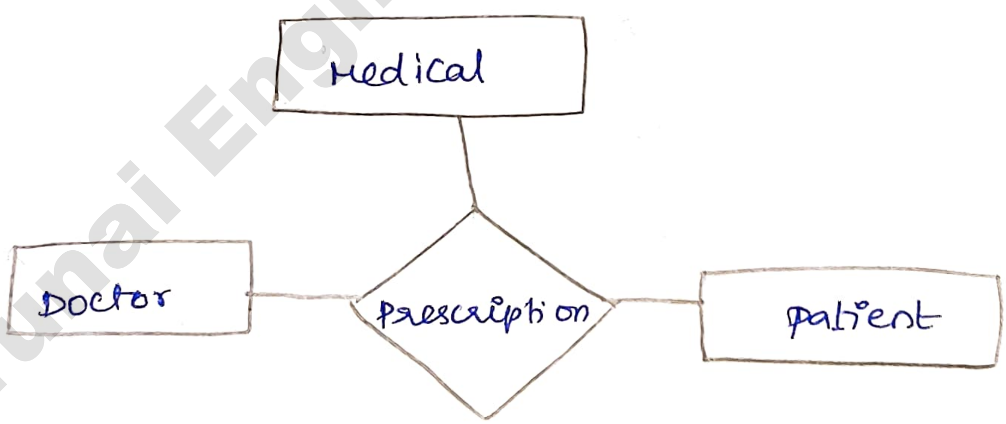
Eg:



The number of entities set that participate in a relationship set is called degree of relationship set.

Relationship between two entity sets are called Binary relationship set.

Relationship between three entities are called Ternary relationship set.



Constraints:

An ER enterprise schema may define certain constraints to which the constraints to which the content of the database must confirm.

Two types of constraints are

Mapping cardinalities

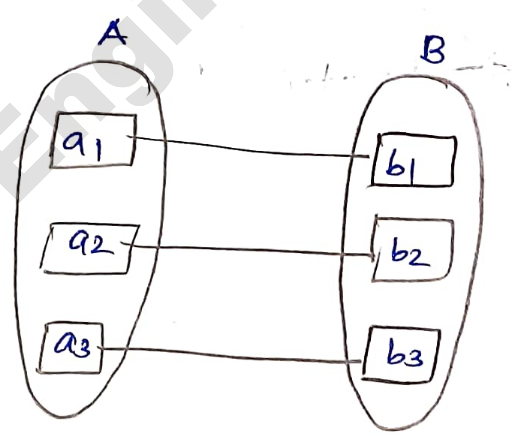
participation constraints.

Mapping cardinalities: (or) Cardinality ratio

Mapping cardinality or cardinality ratio is defined as the number of entities to which other entity can be associated via a relationship set.

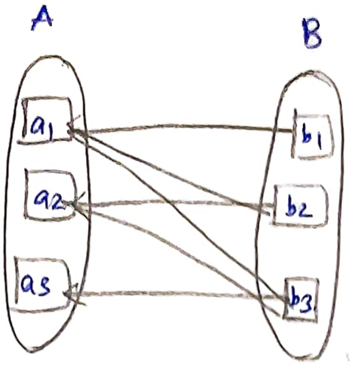
one to one (one : one) (1 : 1)

An entity in A is associated with atmost one entity in B and an entity in B is associated with atmost one entity in A.



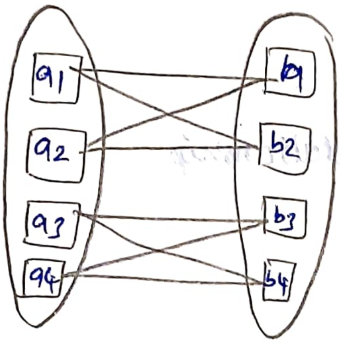
one to many (1 : M)

An entity in A is associated with any number of entity in B. An entity in B can be associated with atmost one entity in A.

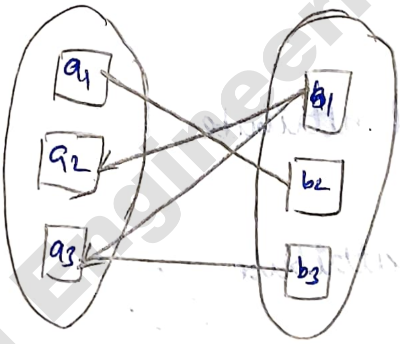


27.01.2020 Monday

Many to Many (M:M)



Many to one (M:1)



Participant constraints:

Total participant constraint

partial participant constraint.

Weak entity set 

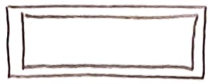
don't have key attribute of own.

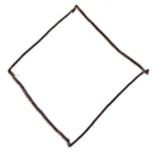
Strong entity set

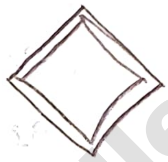
have key attribute.

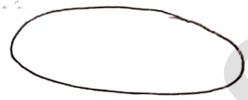
ER diagram notations:

1. Entity – 

2. Weak entity – 

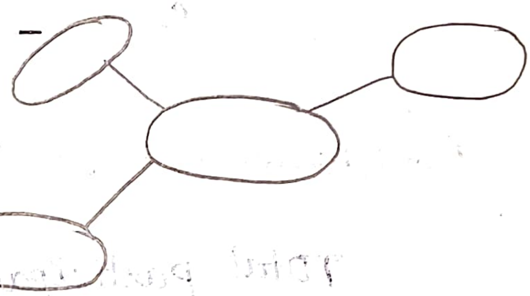
3. Relationship – 

4. Identifying Relationship – 

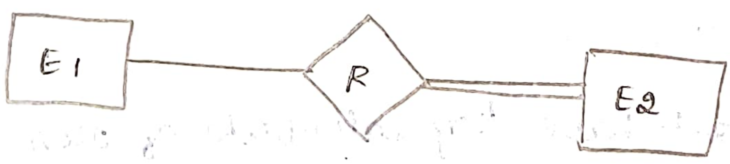
5. Attribute – 

6. Key attribute – 

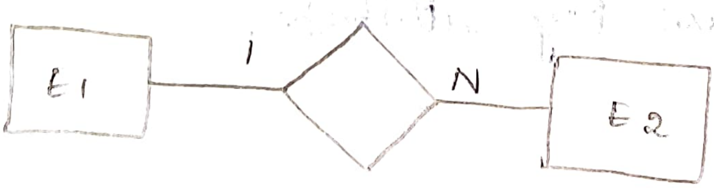
7. Multivalued attribute – 

8. Composite attribute – 

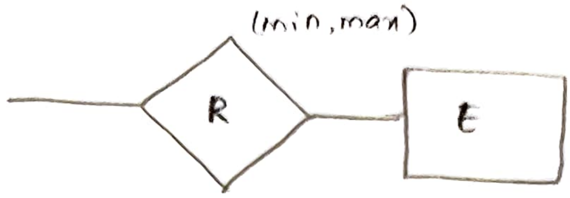
Total participation of E2 in R

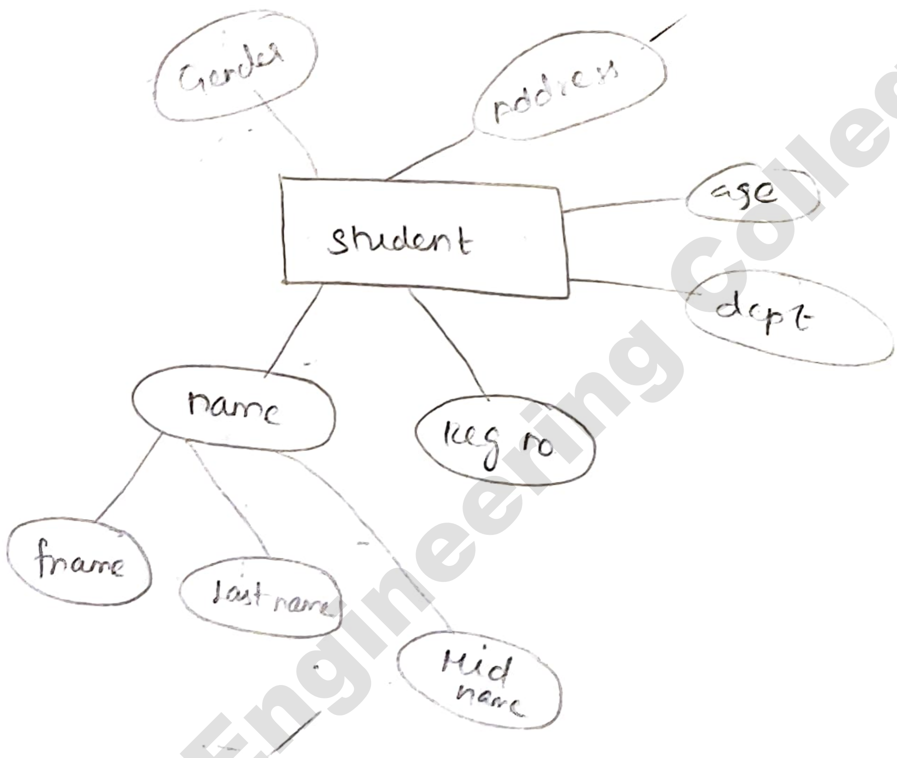


Cardinality ratio 1:N for E1:E2 in R



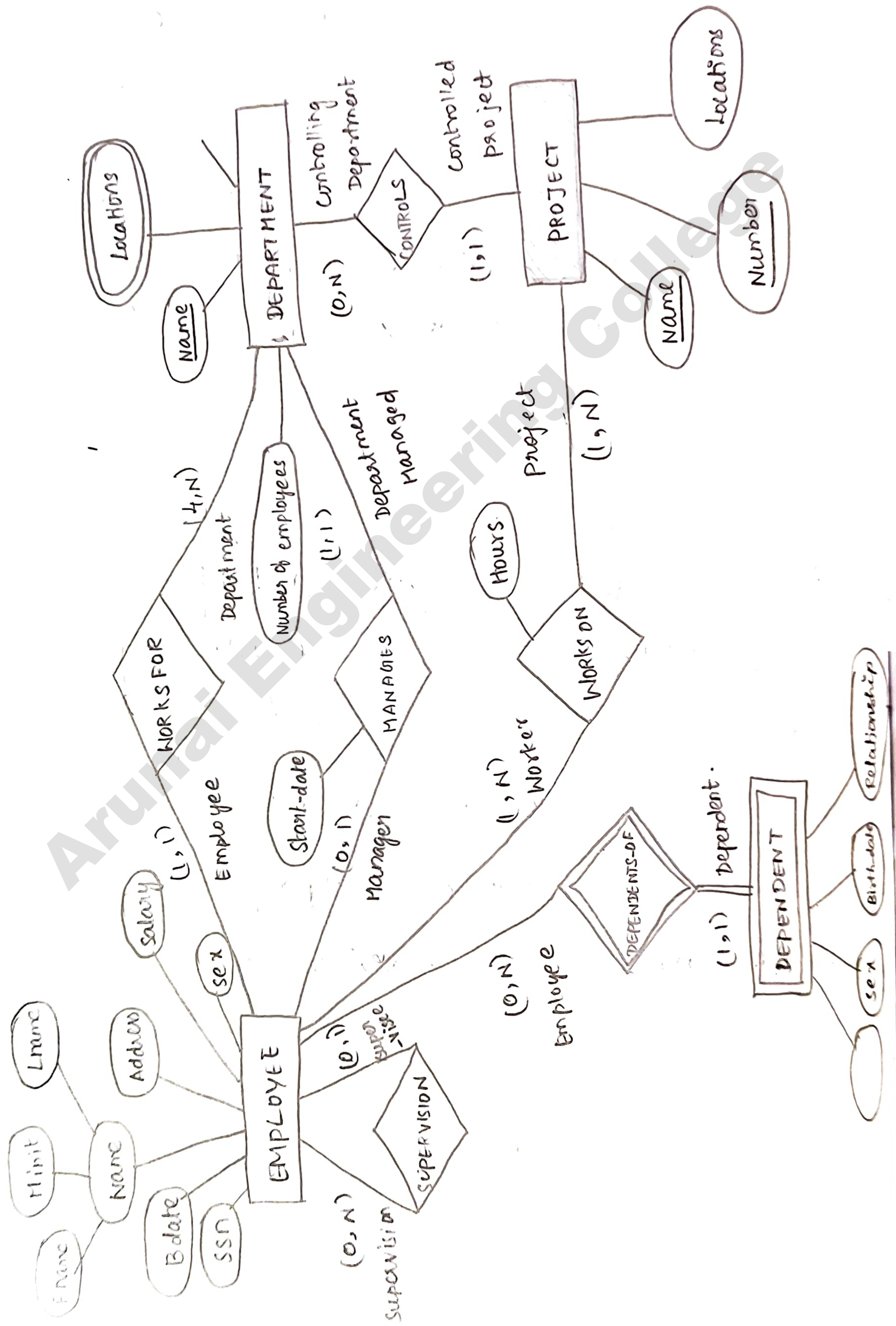
structural constraints (min, max) on participation of E in R



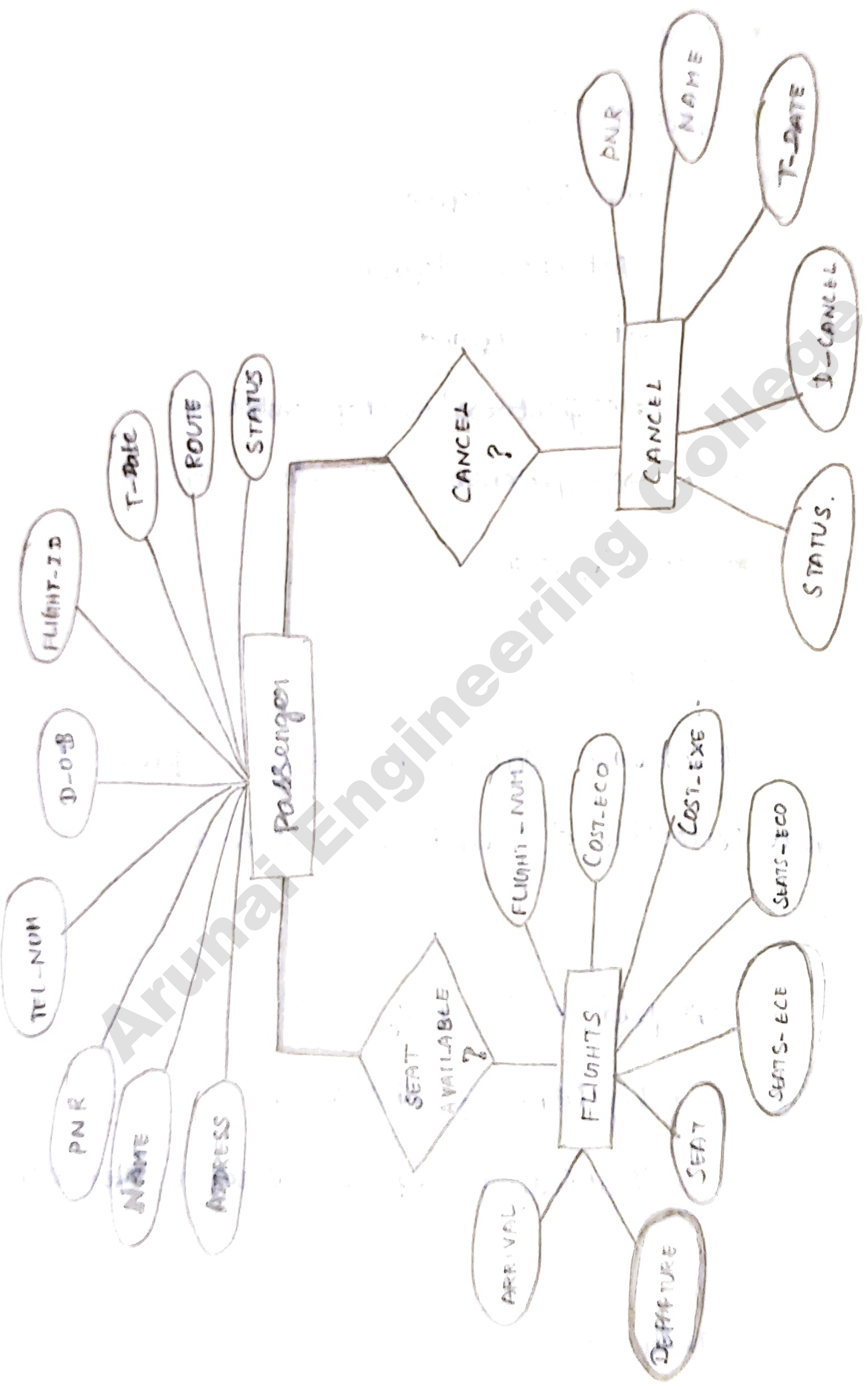


28.01.2020
Tuesday

ER diagram for the company schema.



ER diagram for an Airline Schema.



03.02.2020
Monday.

Enhanced ER diagram or Extended ER diagram

* Subclass
* superclass
* Inheritance
* Specialization
    ⇒ Predicate defined
    ⇒ Attributed defined
    ⇒ User defined
    ⇒ Disjointness / overlap constraint
    ⇒ Completeness constraints
* Generalization.

05.02.2020
Wednesday

Functional Dependency:

redundancy (duplication of datas)

Relational Data Base design requires a good collection of relational schema.

Bit fall in relational database schema:

⇒ A Bad design may lead to
   * Repetition of information
   * Inability to represent certain information

Design goal:

* Avoid redundant data.
* Ensure that relationship among attributes are represented.
* Facilitate the checking of update for violation of database integrity constraints.

Functional dependency:

* It requires that the value for certain set of attributes determines uniquely the value for other set of attributes.

* In a relation R, x and y are attributes. Attribute y is functionally dependent on attribute x if each value of x determines exactly one value of y which is represented as $x \longrightarrow y$. (i.e x determines y or y is functionally dependent on x)

Example: Marks ⟶ Grades

Types:

1. Fully functional dependency (or) Full dependency.

If a relation R, x and y are attribute x functionally determines y. Subset of x should not functionally determine y.

2. partial dependency

Attribute y is partially dependent on the attribute x only if its dependent on a subset of attribute x

3. Transitive dependency

x, y and z attributes in the relation R

$x \to y$
$y \to z$
$x \to z$

Use of functional dependency:

It is used to

1. Test relations to see if they are legal under a given set of functional dependency

2. Specify constraints on the set of legal constraints

3. A functional dependency is trivial if it is satisfy all the instance of the relations

12/02/2020 Wednesday

Normalization:

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy may cause insertion, deletion, updation, anomalies.

Types of Normalization:

First Normal Form (1NF)

Second Normal Form (2NF)

Third Normal Form (3NF)

Boyce Codd Normal Form.

Fourth Normal Form (4NF)

Fifth Normal Form (5NF)

1) First Normal Form (1NF)

To avoid the Multivalued attributes or Composite attributes.
(violate)

Eg:

St.NO	Sname	ph	state
1	Ram	XXXX YYYY	TN
2.	Suresh	ZZZZ	A.P.

(unnormalized form)

↓

Normalised form

St.NO	Sname	ph	state
1	Ram	XXXX	TN
2	Ram	YYYY	TN
3	Suresh	ZZZZ	A.P.

Eg:

Ecode	Dept	Proj code	Hour
E101	system	P27 P51 P20	90 101 60
E305	Sale	P27 P22	101 80.

↓

Normalized form

Ecode	Dept	proj code	Hour
E101	system	P27	90
E101	system	P51	101
E101	system	P20	60
E305	Sale	P27	101
E305	Sale	P22	80.

2) Second Normal Form (2NF)

Relation must not contain any partially dependency i.e every attribute in the row is functionally dependent upon the whole key and not just the part of the key.

A Table is said to be in ~~too(1) enough~~ 2NF and when it is ~~one(1) enough~~ 1NF.

Eg:

Ecode	Proj code	Dept	Hour
E101	P27	S/m	90
E305	P27	Finance	10
E508	P51	Admin	null
E101	P51	S/m	101
E101	P20	S/m	66
E508	P27	Admin	70.

↳ Ecode → Dep.

Ecode + projcode → Hour.

Dep id	Dept name	Emp id	Empname	Emp desig	Emp salary

PD → Dept id
   ↘ Empid } partially dependent.

Now, 2NF

Dept

Dept id → Dept name.

Employee

Emp id → empname, emp designation, emp salary

Salary

dept id → Empid, salary

3) Third Normal Form (3NF)

A relation is said to be in 3NF if the following two conditions are satisfied:

1. A Relation must be in 2NF
2. A relation must not have any transitive dependency.

Eg:

Employee Table.

Ecode	Dept	Dept Head
E101	S/m	E901
E305	Finance	E909
E402	Sales	E906
E508	Admin	E908
E607	Finance	E909
E608	Finance	E907.

Ecode → Dept $x \rightarrow y$

Ecode → Depthead $x \rightarrow z$

Dept → Depthead. $y \rightarrow z.$

3NF,

Employee — Ecode, dept

Dept — Ecode → DeptHead.

4) Boyce Codd Normal Form (BNF):

Ecode, Name, ProjecCode, Hours

Ecode } pk
Pcode }

→ Name + projcode → ck.(candidate key)

→ Ecode + projCode → hour }
→ Name + projcode → hour } M-CK Multiple candidate key.

overlapped factor.

Conversion BNF

Employee
Ecode name

Project
Ecode projcode hour

5) Fourth Normal Form (4NF):

A relation is in 4NF if for any non trivial multivalue dependency $x \rightarrow\rightarrow y$ in F^+, x is a superkey

Eg:

EMP.

Ename	Pname	Dname
Dharshan	x	prajan
Dharshan	y	Pradeep
Dharshan	x	Prajan
Dharshan	y	Pradeep.

Ename	Pname
Dharshan	x
Dharshan	y

Ename	Dname
Dharshan	Prajan
Dharshan	Pradeep.

14.02.2020 Friday

6) Fifth Normal Form (5NF):

The fifth Normalization form (5NF) deals with joint dependency which is generalization of the MVD (multiple value dependency)

The aim of the 5NF is for all joint dependency atleast one of the following holes.

1. Trivial joint dependency
2. Every R_i is a candidate key for R

Denormalization:

It is the process of attempting to optimise the read performance of a database by adding redundant data or by grouping data

UNIT- III

TRANSACTION

Transaction:

Collection of operation that form a single logical unit of work is called Transaction.

Process of Transaction:

The transaction is executed as a series of read and write of database object

1. Read operation : READ (x)

TO read a database object it is first brought into main memory from disk and then its value is copied into a program variable.

2. Write operation : WRITE (x)

TO write a database object an inmemory copy of an object is first modified and then written to disk.

ACID PROPERTY:

1. Atomicity (all or nothing):

* A transaction said to be atomic if a transaction always executes all its action in one step or not execute any action at all.

2. Consistency (No violation of integrity constraints) breaking the rules

* A transaction must preserve the consistency of a database after the execution the DBMS assumes that this property holds for each transaction.

3. Isolation (concurrent changes invisible)

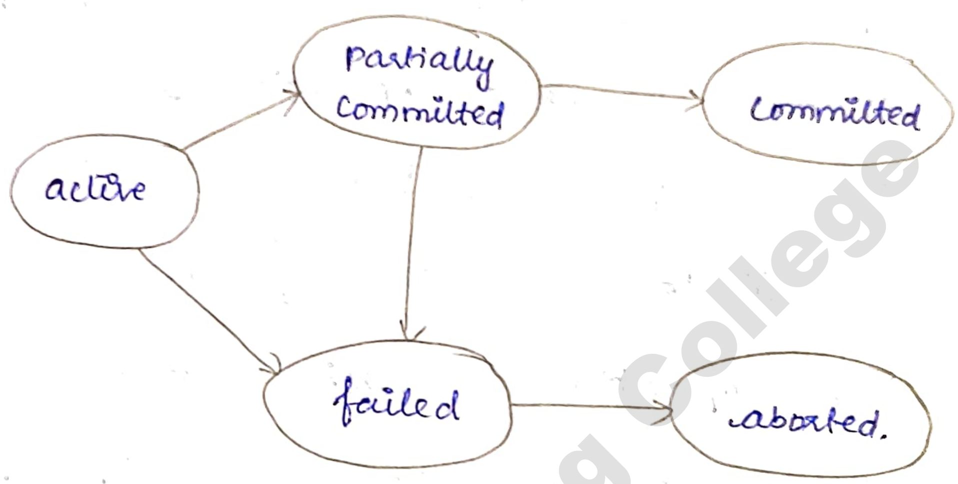
* The transaction must behave as if they are executed in isolation it means that several transactions are executed concurrently the result must be same as if they were executed serially in some order.

4. Durability (commited, update, persist)

* The effect of completed or committed transaction should persist even after a crash. It means once a transaction committ the system must guaranteed with the result of its operation will never be lost.

States of Transaction:

* Active (initial state)
* Partially committed ([illegible])
* Failed (work is not done)
* Aborted (the work are cancelled or rejected)
* Committed (completed)



17.02.2020
Monday.

Schedules:

Schedules (Histories) of Transaction

- → Conflict schedule
- → Complete schedule.

Recoverability

- → Recoverable schedules
- → Cascadeless (non cascading) schedules
- → Strict schedule

cascade → continuous flow of actions

Conflict schedule:

Two operation in a schedule are said to be conflict if they satisfy three condition

(i) they belong to different transaction
(ii) they access the same item X
(iii) atleast one of the operation is a write_item(X

Complete Schedule:

A Schedule S on n transaction $T_1, T_2 \ldots T_n$ is said to be complete schedule if the following condition holds

(i) The operation in S are exactly those operation in $T_1, T_2 \ldots T_n$ including a committ or abort operation as the last operation for each transaction in the schedule.

(ii) For any pair of operation from the same operation T_i their order of apperance is S is the same as a order of appearance in T_i

(iii). For any two conflicting operation one of the two must occurs before the other in the schedule.

Recoverable Schedule:

A recoverable schedule is a one where for each pair of transaction T_i and T_j such that T_j read a data item previously written by T_i. The committ operation of T_i appears before the committ operation of T_j

cascadeless schedule:

Even if a schedule is recoverable to recover correctly from the failure of a transaction Ti we may have to roll back several transaction such situation occurs if transaction have to read data written by Ti

Cascading Rollback:

Cascading Rollback is a one in which a single transaction failure leads to a serious of transaction rollback.

Schedule with cascading Rollback or undesirable schedules should not contain cascading schedules are called Cascadeless schedule.

19-02-2020 Wednesday

Serializability:

Serial schedule
Non Serial schedule
serializability

Types of Serializability:

Conflict serializability
View Serializability

1) conflict serializability:

I_i = read (Q) & I_j = read (Q)

I_i = read (Q) & I_j = write (Q)

I_i = write (Q) & I_j = read (Q)

I_i = write (Q) & I_j = write (Q)

I_i = read (Q) & I_j = read (Q)

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

— conflict occurs.

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

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

2) View serializability : (equivalence of the process)

Initial condition

1. T_1 = read (Q) in S
   T_1 = read (Q) in S_1

2. T_1 = read (Q) & write (Q) in S
   T_1 = read (Q) & write (Q) in S_1.

Final condition

3. T_1 = write (Q) in S

T_1 = write (Q) in S_1.

Testing for serializability

Precedence graph

$G = (V, E)$

V - vertices - all transaction

E - Edges - $T_i \rightarrow T_j$

3 conditions hold

T_i execute write (Q) before T_j execute read(Q)

T_i execute read(Q) before T_j execute write(Q)

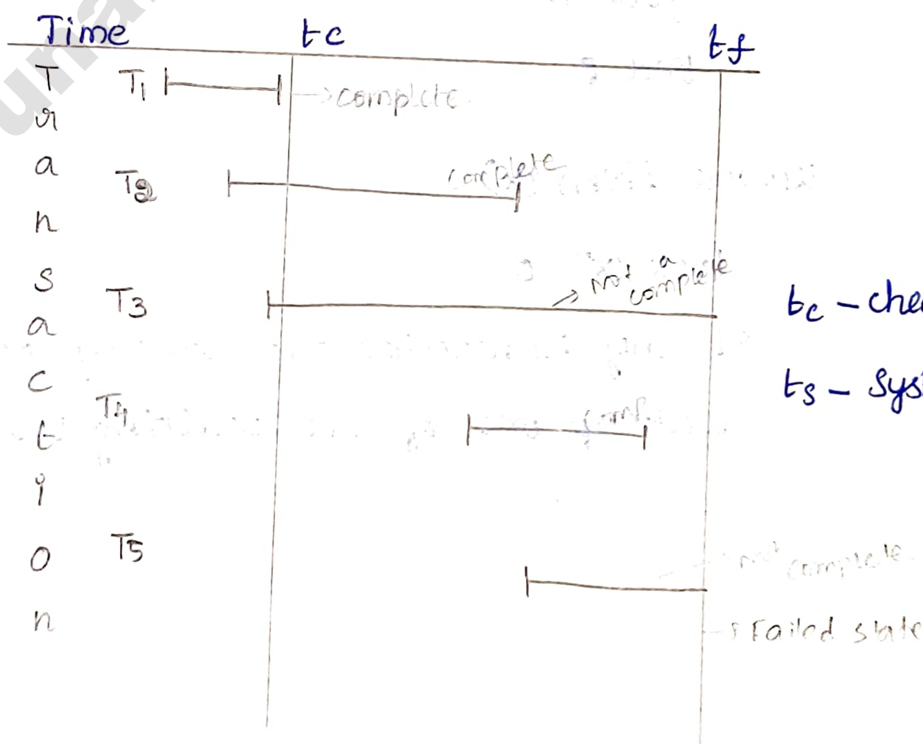
T_i execute write (Q) before T_j execute write(Q)

21.02.2020 Friday.

System Recovery

Failure

* System failure ⇒ Soft crash
* Media failure ⇒ Hard crash



t_c – checkpoint

t_f – System failure

Two phase Committ:

→ Committ

→ Roll back.

2 phase.

1. Prepare → ask ok or not ok

2. Committ.

Concurrency Control:

⇒ Lock based protocols.

↳ Types of locks

* Binary lock
  - 0 → unlock
  - 1 → lock

Lock (item)

lock (x) = 0

lock (x) = 1 - lock

* Shared & Exclusive locks.
  - Shared ↳ Read only ↳ lock - S(Q)
  - Exclusive ↳ read write ⇒ lock - x(Q)

lock item (x):

```
B : if lock (x) = 0
    then lock (x) 1
    else begin
        wait (until lock (x) = 0 and the lock
            manager wakesup the transaction);
    goto B;
    End;
```

unlock item (x)

lock (x) 0

If any transaction are waiting then wakeup one of the waiting transaction.

24.02.2020
Monday

Shared & Exclusive lock:

T1	T2	Concurrency control Manager
lock - x (B)		grant - x (B, T1)
read (B)		
B : B - 50		
write (B)		
unlock (B).	lock - S(A)	grant - S (A, T2)
	read (A)	
~~lock - S(A)~~	unlock (A)	
~~read (A)~~	lock - S(B)	grant - S (B, T2)
~~unlock~~	read (B)	
	unlock (B)	
	display (A+B)	
	lock - x(A)	grant - x (A, T1)
T1 ←	read (A)	
	A : A+50	
	write (A)	
	unlock (A)	

LOCKING PROTOCOL:

→ Set of rules ⇒ locking protocol
→ Granting of locks

Ex: T_2 - S
T_1 - X
T_3 - S

(I: T_2 - S and T_3 - S; II: T_1 - X)

↳ starved

Two phase locking protocol.

1. Growing phase - obtain locks
2. Shrinking phase - release locks.

Growing phase:

Transaction may obtain locks but may not release any lock.

Shrinking phase:

A Transaction may release lock but may not obtain any new lock.

Types:

1. Strict 2 phase locking protocol – X

2. Rigorous 2 phase locking protocol.

↳ Lock conversion

↳ upgrade – S to X

↳ downgrade – X to S

TIMESTAMP BASED PROTOCOLS:

→ Time stamp ⇒ Eg TS(Ti)

→ systemclock

→ logical counter

* W – timestamp(Q)

* R – timestamp(Q)

Time stamp Ordering Protocol:

1. Suppose that transaction Ti issue read(Q)

↳ TS(Ti) < W – timestamp(Q)

↳ read reject – Ti rolled back.

↳ TS(Ti) > W – timestamp(Q)

↳ read execute

2. Suppose the transaction Ti issue write(Q)

↳ TS(Ti) < R – timestamp(Q)

write - reject, roll Ti back.

↳ TS(Ti) < W – time stamp(Q)

write - reject, roll Ti back.

26.02.2020 Wednesday

Deadlock :

→ Deadlock

→ Deadlock prevention

→ Deadlock Detection.

Deadlock prevention

2 approaches

→ either all or none — (disadvantages) wastage of time

→ Preemption

* timestamp.

→ 2 different deadlock prevention scheme using timestamp are

① wait - die (older wait for younger)

② wound - wait (older never waits for younger)

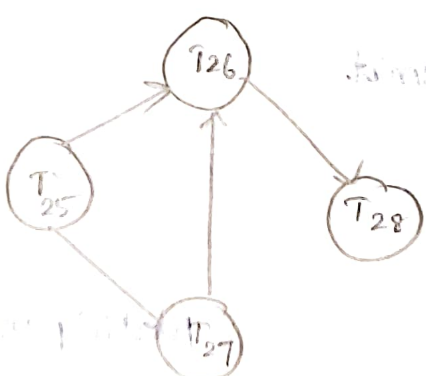
Deadlock Detection:

→ wait for graph

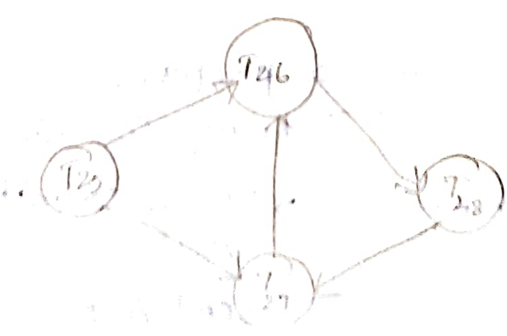
$G = (V, E)$

V – set of vertices

E – set of Edges.



→ no deadlock



deadlock occur

Recovery from Deadlock.

3 action

1. selection of a victim
2. Roll back < Total rollback / Partial rollback
3. starvation.

04.03.2020 Wednesday

Transaction Recovery:

Coordinator (Transaction manager)

participants (Resource Manager)

2 phase commit (2PC)

① Votting phase

② Decision phase

* Unilateral abort
* Global commit.

Phase 1.

→ prepare

→ wait - period

phase 2

① Abort

② Ready commit

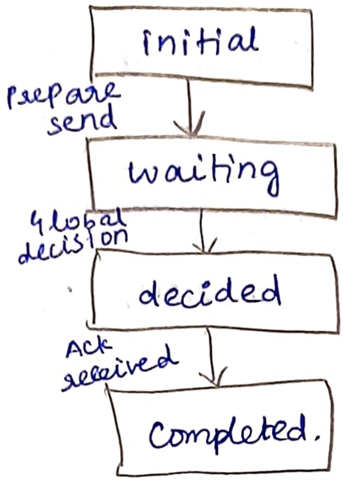
③ Ack.

Voting commit

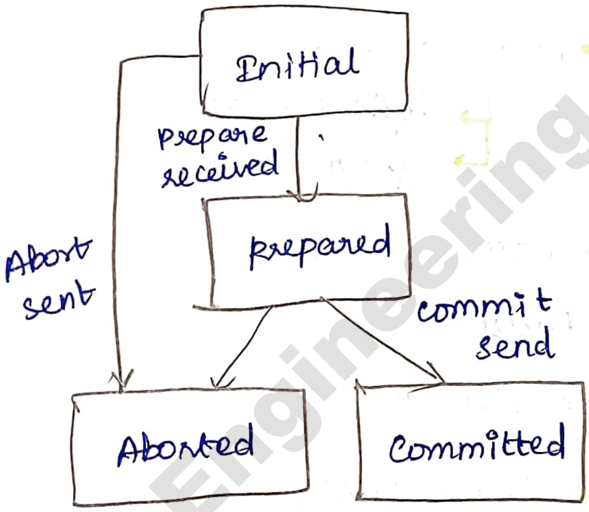
Coordinator		participant
Commit		
	——→	prepare
	←——	Ready Commit
ReadyCommit Global commit	——→	Global commit write
Ack all	←——	send ACK

Termination Protocol for 2PC.

Coordinator.



Participant



Save Point:

STUDENT.

ID	NAME	DEPT	MARK.
1	X Y	IT	75
2	X Y	CSE	85
3	Z Z	ECE	90
4	X Z	EEE	90
5	Y Z	MECH	75

SAVE POINT SP1;

delete from student where ID = 1; (1) deleted

SAVE POINT SP2; (save 2, 3, 4, 5)

delete from student where ID = 2;

SAVE POINT SP3; (save 3, 4, 5)

Rollback to SAVEPOINT SP2; (2, 3, 4, 5)

ISOLATION LEVEL:

3 SQL Isolation

→ Dirty Ready - uncommitted / modified record created by another transaction.

→ Non-Repeatable - read the same row in a table twice

→ phantoms - retrieves a range of data value twice

SQL Facilities for Concurrency and Recovery:

→ Crash Recovery

→ Failure classification

- Transaction failure
  - logical error
  - system error.

→ System crash

→ Disk Failure

→ Storage structure

- Volatile storage
- Non volatile storage.

→ Recovery and Atomicity

→ Log based Recovery

→ Recovery with concurrent transaction

- Check point
- Failure

09.03.2020
Monday.

UNIT-4.

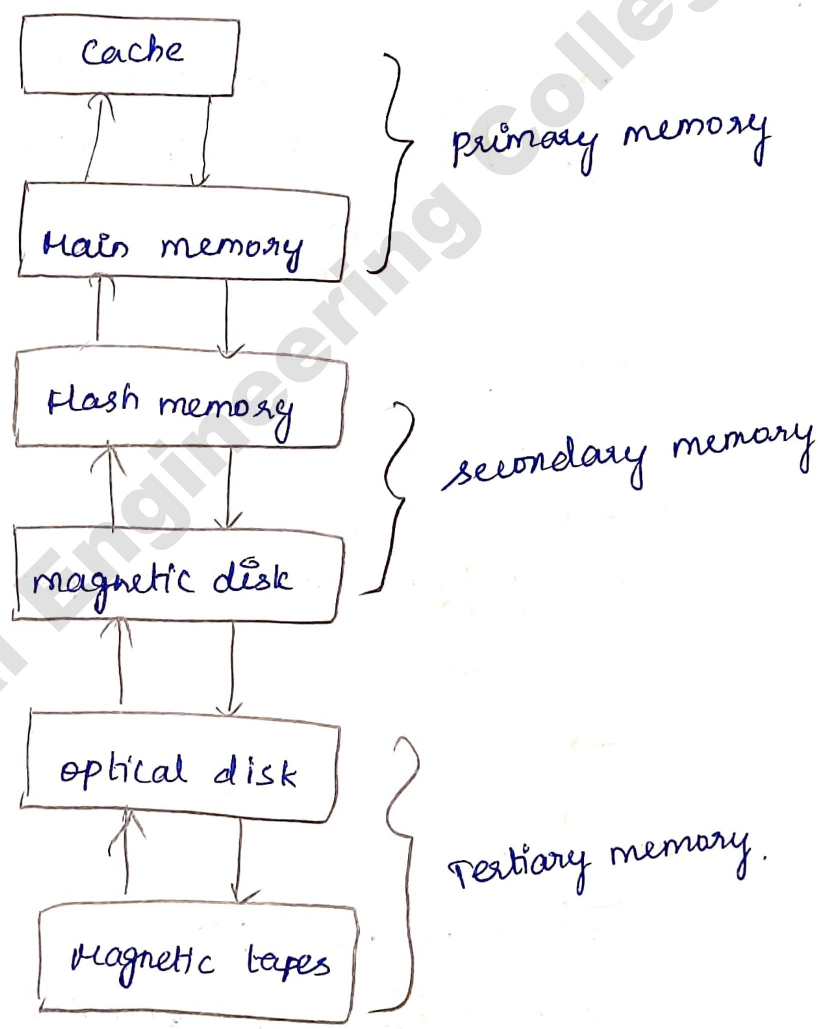
IMPLEMENTATION TECHNIQUES

Storage and File Organization:

Physical storage media

- Accessing speed
- Cost per unit of data
- Reliability

Storage device hierarchy



RAID (Redundant Arrays of Independent Disks)

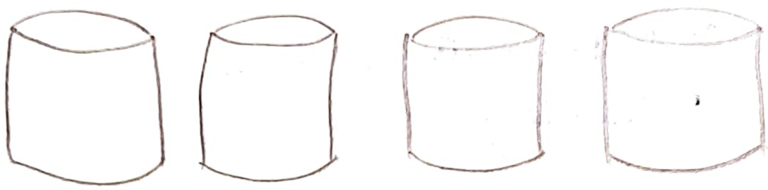
- mirroring or shadowing
- Bit level skipping
- Block level skipping

RAID levels

a) RAID level 0.

non redundant block skipping

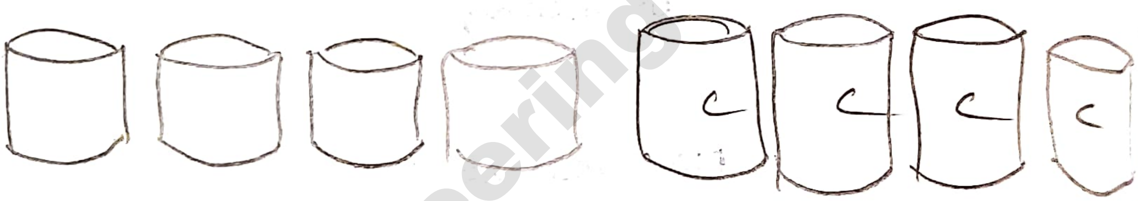
data lost is not critical.



b) RAID level 1.

mirroring disk with block skipping

storing log files in db sys.



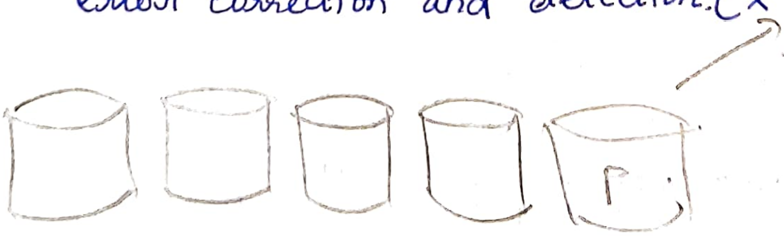
RAID level 2.

memory style error correcting code with bit stripping

RAID level 3.

Bit interleaved parity

error correction and detection. (x or R)



RAID level 4.

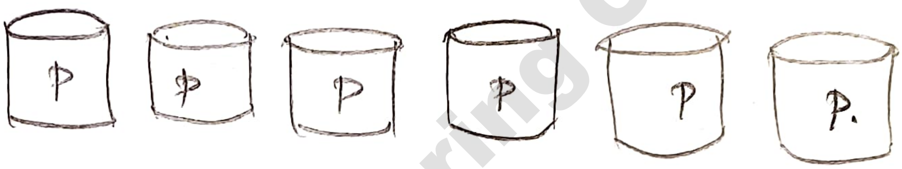
Block Interleaved parity.

RAID level 5.

Block Interleaved distributed parity

RAID level 6.

P + Q Redundancy scheme.



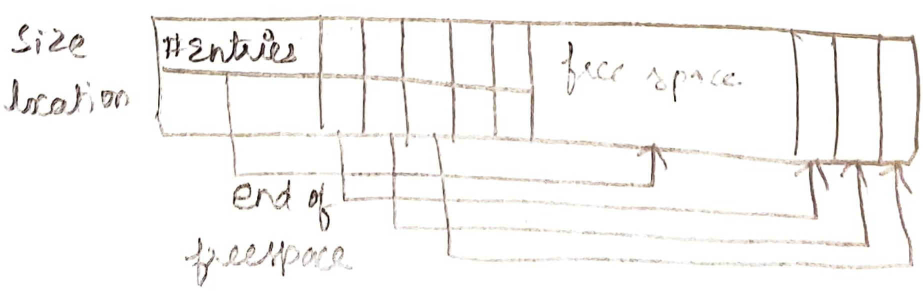
12.03.2020
Thursday.

File Organization:

→ Fixed length Record

Header			
Record	A102	parrot	400
	A215	Crow	700
	A101	Hen	600

→ Variable length Record



Organization of Records in Files

* Access → Sequential access / Direct access

5 methods

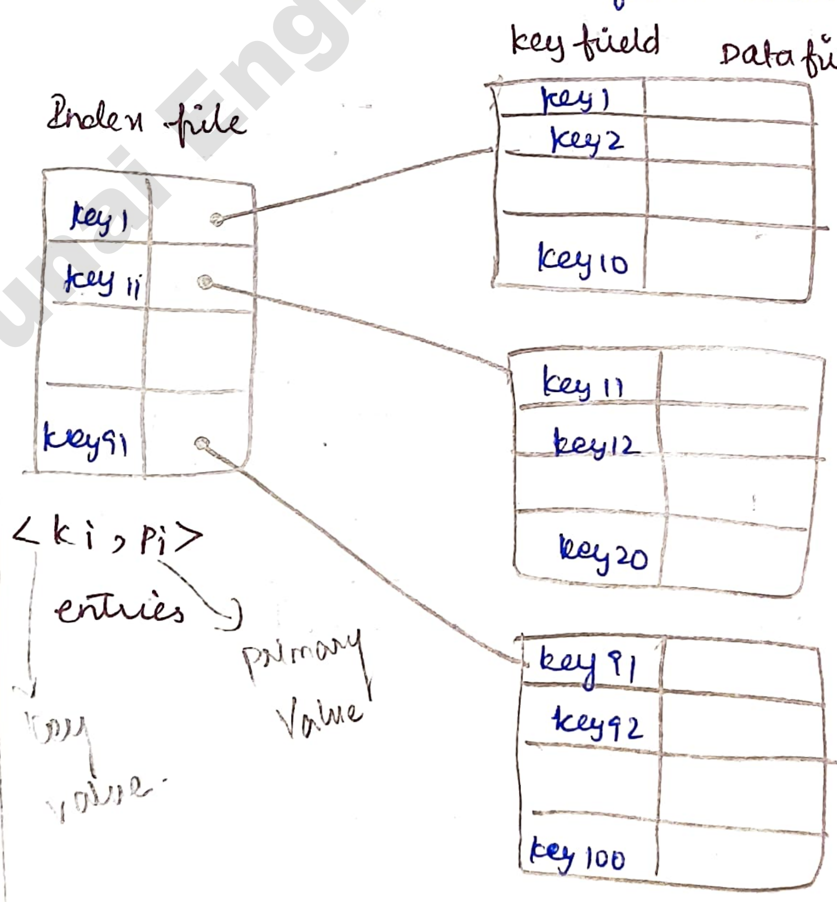
1. sequential organization
2. Indexed sequential organization
3. Direct organization
4. Heap File organization → os allocated memory
5. clustered Organization → grouping of allocating memory

Indexing and Hashing:

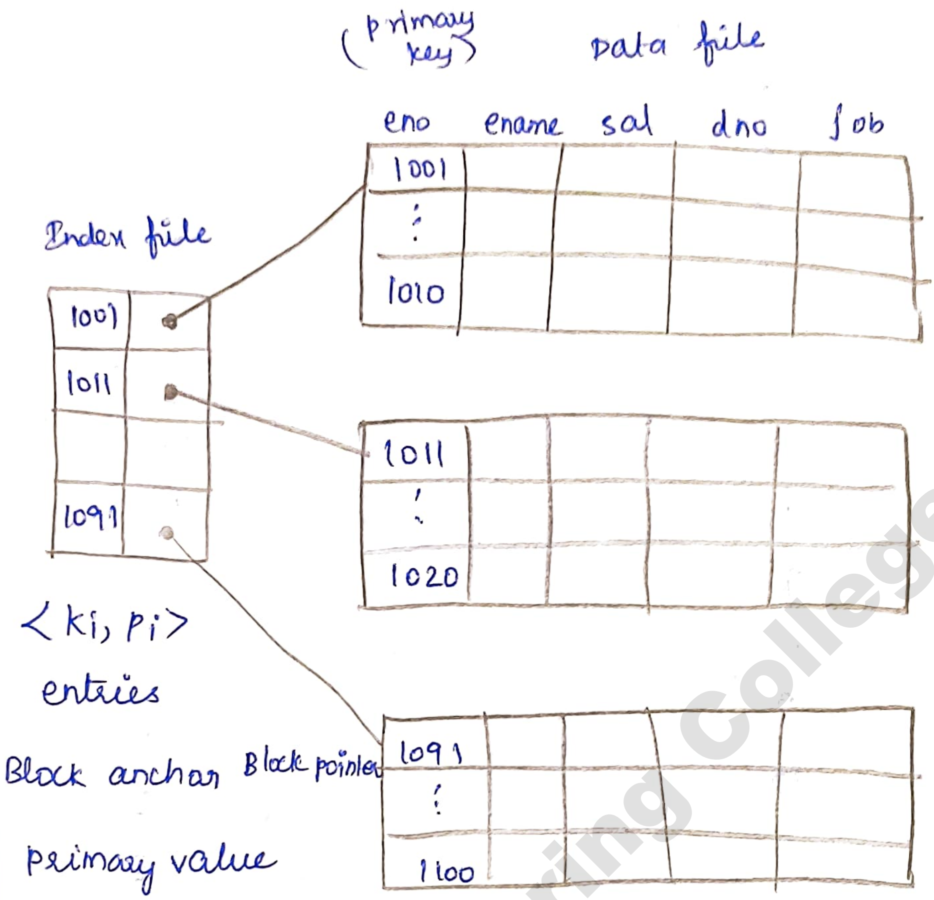
Indexing

2 Indices

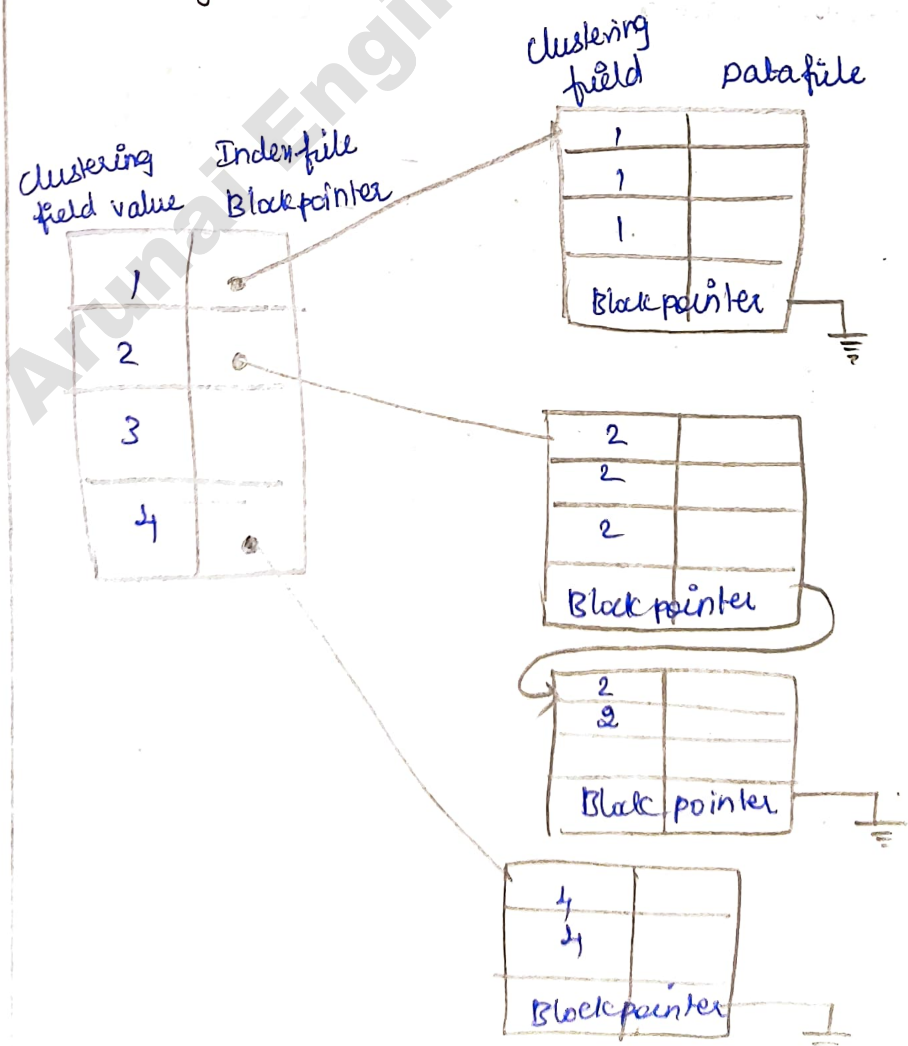
1. Ordered Indices – stored ordering.
2. Hash Indices – uniform distribution.



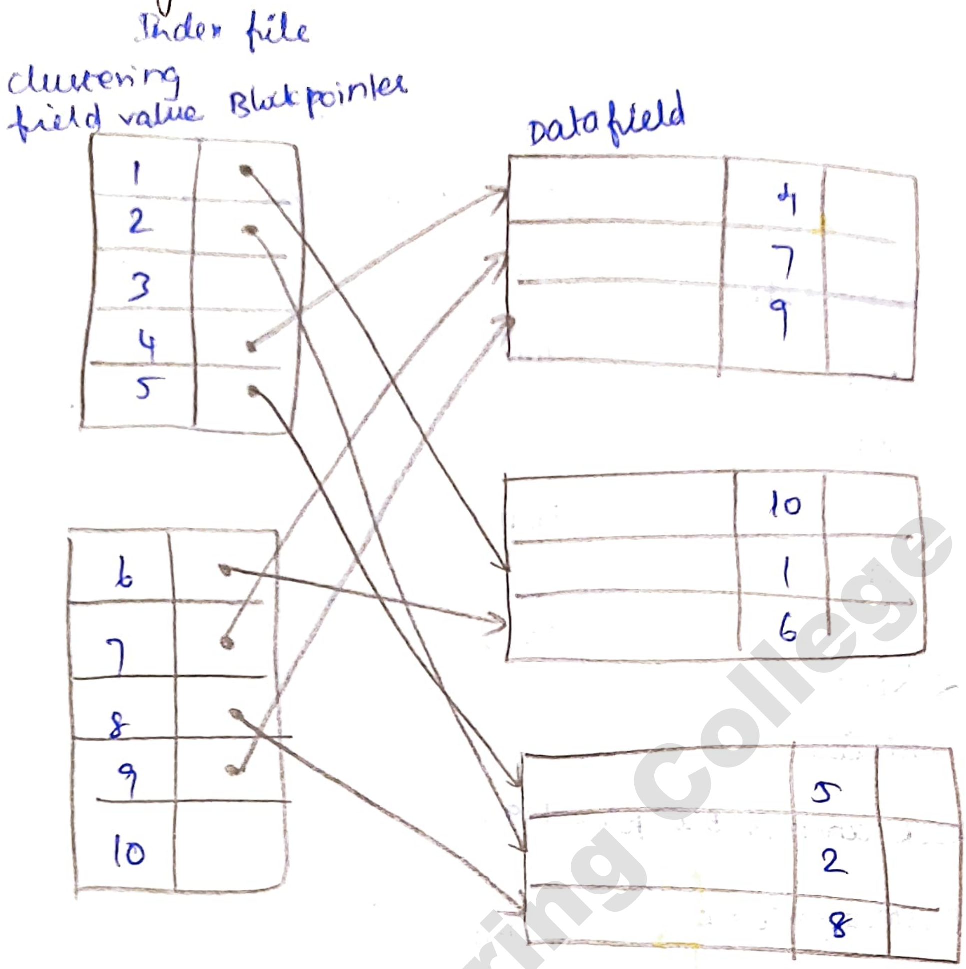
Single Level Ordered Indices



Clustering Index

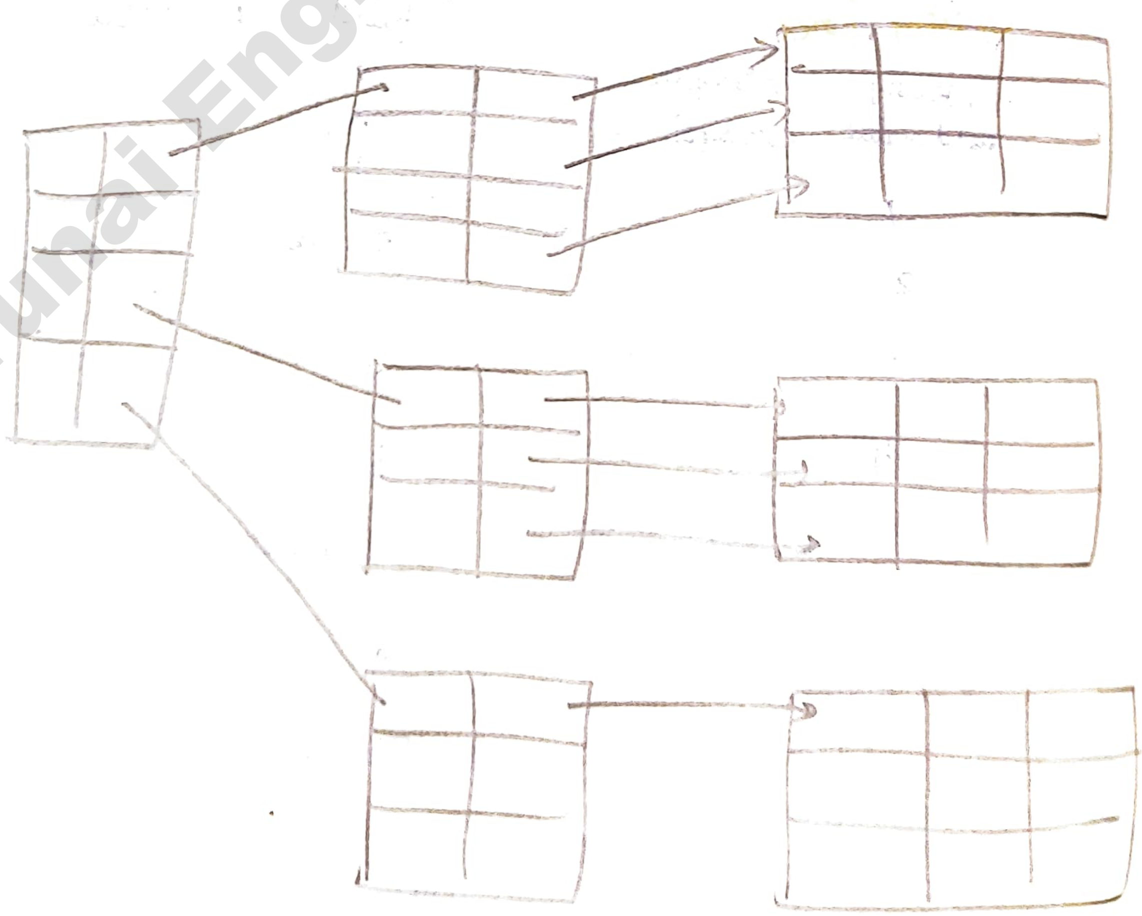


Secondary Index



entries (K(i), P(i))

Multilevel Index



15.03.2020
Monday

HASHING.

1. Internal Hashing for Main Memory

$h(k) = k \bmod M$ function

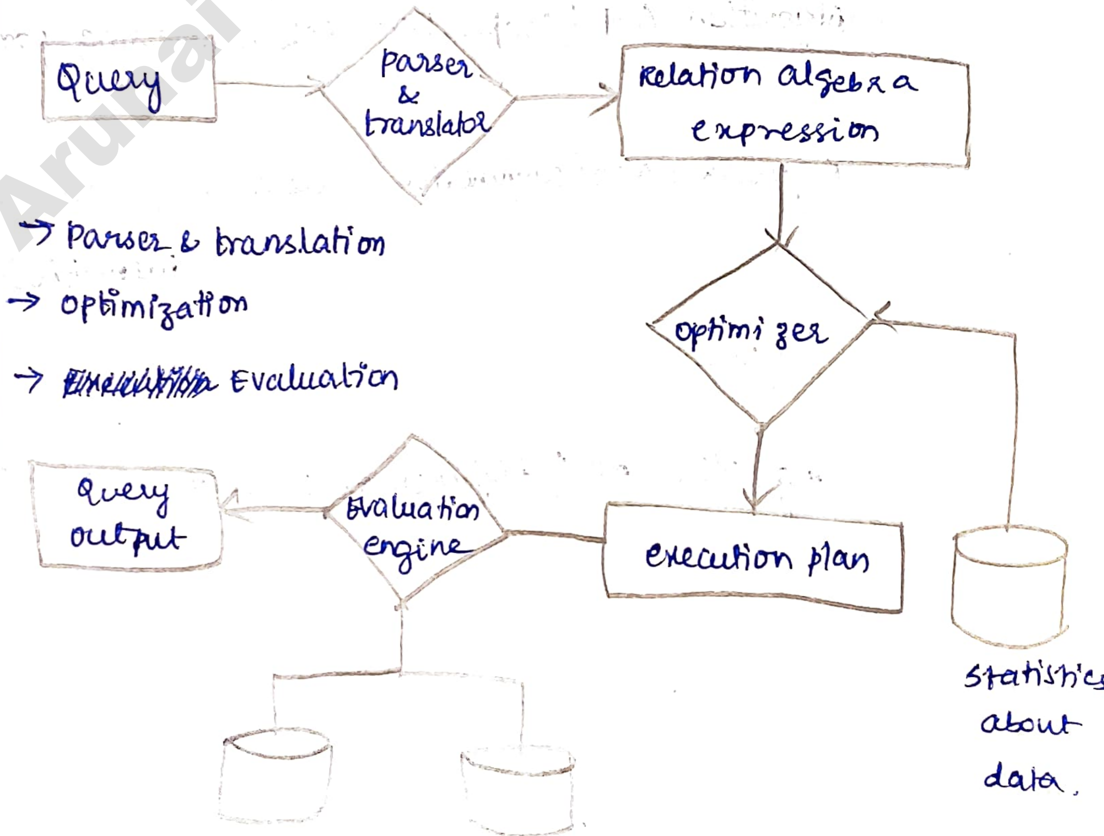
* Collision resolution

method
- → open addressing
- → chaining
- → multiple hashing

2. External Hashing for Disk files

3. Hashing techniques that allow dynamic file expansion.
- → Extensible Hashing
- → searching
- → Insertion
- → Linear Hashing.

Query Processing:



→ Parser & translation
→ optimization
→ ~~Execution~~ Evaluation

Query optimization

→ Measure of query cost

→ Selection operation - file scan.

→ Algorithm A1 (Linear Search)

→ Algorithm A2 (Binary search)

→ Algorithm A3 (primary Index, equality on key)

→ Algorithm A4 (Primary Index, equality on nonkey)

→ Algorithm A5 (secondary Index, equality)

Selection Involving Comparisons:

→ Algorithm A6 (Primary Index, comparison)

→ Algorithm A7 (Secondary Index, comparison)

Complex Selection

→ Algorithm A8 (conjunctive selection using one index)

→ Algorithm A9 (Conjunctive selection using composite Index)

→ Algorithm A10 (conjunctive selection by Intersection of identifier)

Disjunction

→ Algorithm A11 (disjunctive selection by union Identifier)