R S I T कम्प्यूटर कॉलेज करकेली

शिव मंदिर के पास, रेलवे स्टेशन रोड़ करकेली
जिला- उमरिया (म.प्र.) 484661

RUDRA SOFTECH INSTITUTE TECHNOLOGY
Transforming Education Transforming India

Subject – DBMS

Session 2018-19

- **PGDCA**
- **DCA**
- **DCPA**
- **ADCP**
- **MCIT**

एवं अन्य 8 वीं पास सर्टिफिकेट कोर्सेस

Prepared by: -
Prakash Dwivedi (BE) 8982505087
Abhilash Pathak (MCA) 8517906324, 7974259812

Website: www.rudrasoftech.in

UNIT-1

❖ *DBMS*: -

DBMS का पूरा नाम Data Base Management System होता है। DBMS, programs का पूरा collection होता है, जो कि user's को Database को create and maintain करने के योग्य बनाता है, तो हम कह सकते हैं कि DBMS एक General Purpose softwere system है।

Data के बारे में हमने पिछले भाग में सीखा है, अब बात आती है Base management system की। Base मतलब data का प्रकार । हमारे पास जो data है वो किस format में है। अगर हम अपने system की बात कहें तो उसमें हम अपने data को अगल-2 managed करते हैं, जैसे Mp3 base वाला data etc.

Managemenet का मजलब simple है, हमें अपने data को इस तरह से रखना है कि उसे जल्दी से जल्दी खोज जा सके और System का मतलब एक ऐसी machine से या method से है जिससे मि अपने data को systematic रुप से रख सके।

Database management system को हम संक्षिप्त रुप में बाले तों data को सही और व्यवस्थित तरीके में store करना ताकि समय पर उसे जल्दी से जल्दी recall किया जा सके।

अब बात करते हैं, हमें DBMS की जरुरत क्यों पडी। हमारे पास और भी Database management system हैं, जैसे- Ms-Access, Ms- excelor Register (Manual) etc. पर जैसे-जैसे हमारा database बडा हुआ वैसे ही excel से आप data को आसानी से पुनः प्राप्त नहीं कर सकते और अगर database थेडा बडा हुआ जैसे किसी company का हो तो वहाँ access भी data को आसानी से प्राप्त नहीं कर पाता।

इसलिए हम कह सकते हैं कि अगर हमें किसी बडे data को व्यवस्थित करना है तो हमें Oracle का ज्ञान होना चाहिए। Oracle सिर्फ data को manage करने के लिए ही नहीं जाना जाता बल्कि data का सुरक्षा के लिए भी जाना जाता है।

अगले भाग में हम पढ़ेंगे Oracle का architecture, कि कैसे oracle data को save करता है, कैसे process करता है और कैसे data को पुनः प्राप्त कराता है।

Advantages of Database: -

1) **No Data Redundancy & Inconsistency: -** एक ही तरह के डाटा का बहुत सारी जगह नकल को हम data redundancy कहते हैं, और एक तरह के डाटा का बहुत जगह होना data inconsistency का कारण बनती है जिससे storage and cost बढ़ता है लेकिन DBMS में हमें इससे छुटकारा मिलता है।

2) **Restriciting Unauthorized Access: -** DBMS में DBA (Database Administrator) security and unauthorized sub-system का use account restrictions को specify करने के लिए करता है।

3) **Data Integrity and Security: -** DBMS में सुरक्षा और अखण्डता का पूरा ध्यान रखा जाता है। Database में किसी भी प्रकार का मान को insert करने से पहले कुछ conditions को satisfy करना आवश्यक होता है। Data base में user को सभी data को access करने की अनुमति नहीं होती है। जिससे data integrity बढ़ती है।

4) **Simple Access: -** DBMS में database को आसानी से access किया जा सकता है। आसानी से access करने के लिए API (Application programming interface) का use किया जाता है।

5) Data Backup & Recovery की सुरक्षा प्राप्त है, जिससे data के loss होने पर recover किया जा सके।
6) **Data Sharing facility: -** Data sharing facility से data central में रखा जाता है, जिससे कोई unauthorized person access भी नहीं कर सकता है।

7) **Multiple User Interface: -** Data central में रखे जाने से multiple user at a time access कर सकते हैं तथा user की जरुरत के अनुसार अलग-2 interface होता है, जिसे उपयोग कर सकते हैं।

Example: Query, Form, Menu, Programming etc कार्य के अनुसार user के द्वारा use होते हैं।

Dis-Advantages of Database: -

1) अधिक खर्चीला होता है क्योंकि new software development, storage cost, time to time new technique and hard ware upgrading करने में ज्यादा खर्च होता है।

2) Complexity of Security and Integrity का manage सही तरीके से नहीं हो पाता।

3) Centralized data problems (General Backup): - Data duplication problem से बचने के लिए data को central में रखा जाता है जिससे उसका backup बनाया जाता है ताकि Centralized data के loss होने पर भविष्य में काम आ सके, अर्थात् costly होता है।

Characterisitics of DBMS: -

DBMS की कुछ विशेषताएं नि0 लि0 हैं–

1) DBMS की सबसे बडी विशेषता यह है कि इसमें data redundancy को control किया जा सकता है।

2) DBMS में data को share कर सकते हैं।

3) DBMS में security का पूरा ध्यान रखा जाता है।

4) DBMS में processing की speed अच्छी होती है।

5) DBMS की एक और विशेषता यह है कि इसमें data independent होता है।

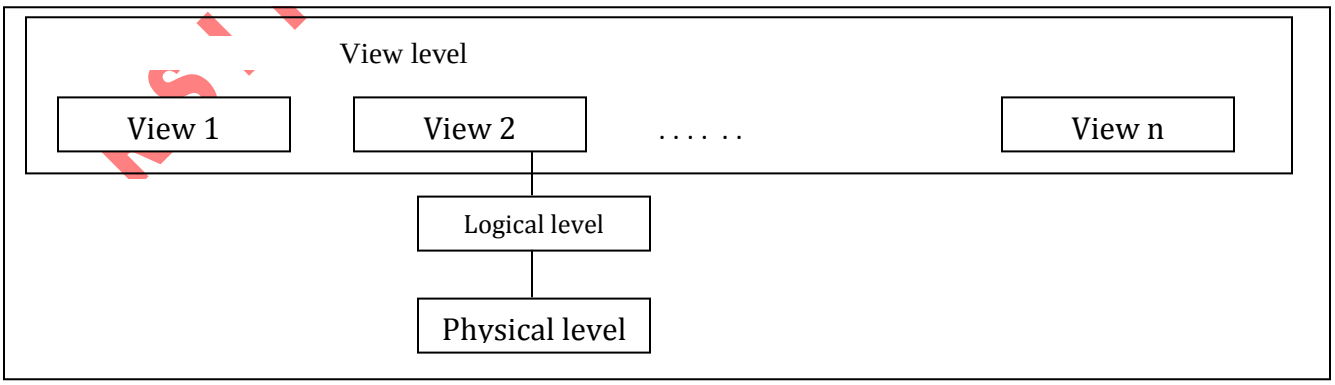
View of Data: - Data को view को दो भाग में विभाजित किया जाता है–

1) Logical Stracture: - जैसा कि हम जानते हैं कि data को central में रखा जाता है जिससे हम अलग-2 user को अलग-2 view में data के भग को दिखाते है कि कौन सा data का भाग किस level के user के लिए है जिससे data का reliability बना रहता है।

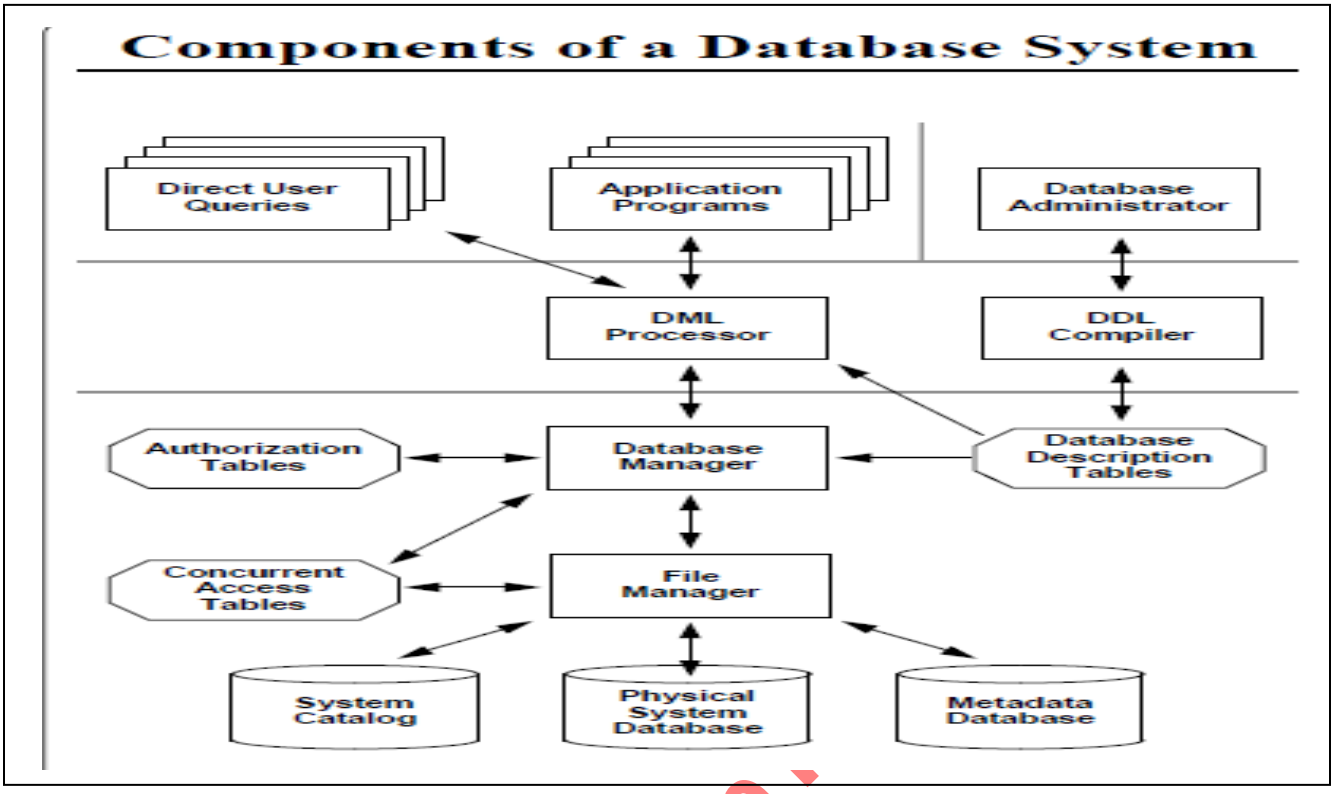
a) Physical/Internal level: - इस level में data को केसे store करना है तथा उसका physical relation बनाना तथा सभी file और उनके संरचना की जानकारी होती है। इस level की जानकारी database creator/designer को होती है। Normal user को इस level से कोई मतलब नहीं होता हे इसलिए इस level को सबसे आखरी में रखा जाता है।

b) Logical /Conceptual level: - Database में store की गई data file में शामिल data field और उनके बीच के relation का description logical view में होता है। इस view में programmer nad database administrator involve होता है। DBA database को management and maintenance करता है और programmer application and database के बीच में coding के द्वारा relation स्थापित करवाता है। इस level में normal user को कोई मतलब नहीं होता है।

c) User /External level: - इस level में user को सिर्फ अपनी आवश्यकता से मतलब होता है बाकि अन्य दोनो level में क्या working and process होता है उससे उसका कोई relation नहीं होता है।



2) Physical Structure: - Database के physical structure की help से यह पता चलता है कि data कैसे store होता है और उसमें क्या-2 process perform होती है। जिससे software component की जानकारी मिलती है कि किस component का relation किससे है और हमें इसमें यह भी पता चलता है कि किस component का क्या कार्य है।



DML (Data Manipulation Language): - User की आवश्यकता के अनुसार database से data को updating, modification and data processing कराना होता है। इसमें select, form etc commond होते हैं। Language की दो classes होती हैं–

1) Procedural: - इसमें user यह specify करता है कि किस type के data की जरुरत है और कैसे उस data को प्राप्त किया जाए।

2) Non-Procedural: - इसमें user यह specify करता है कि किस type के data की जरुरत है लेकिन कैसे उस data को प्राप्त किया जाए। इसे specify करने की जरुरत नहीं होती है।

DDL (Data Definiation Language): - इसके help से database में new table बनाने से related सभी commond होत हैं। Create table etc.

Database Manager: - यह low level data, application program, query के बीच interface provide कराता है तथा साथ ही साथ डाटा की reiablility and security बनाए रखने का कार्य करता है।

File Manager: - यह harddisk में store data का management करता है। यह disk में data को store करने के लिए internal space create करता है।

Data Dictionary: - Contains metadata (Data about data) (एक या एक से ज्यादा) एक डाटा के अंदर दूसरा डाटा मतलब multiple data का collection होता है।

Database Users and Administrator

[Image: DBMS Users]

***Administrator*: -** यह complete database को management कराने का कार्य करता है। जैसे database से किसी record/table/database/user को delete/update/drop या security or permission etc कार्य को perform कराना। साथ-2 यह hardware and software से related maintenance का कार्य भी perform कराता है।

***Designers*: -** इसमें कई व्यक्तियों का समूह होता है जिनका कार्य होता है database को design करना अर्थात् format of data and database में use होने वाली entity, relation of table, constraints and view को design करने का कार्य करते हैं।

***End User*: -** End User का कार्य है database में store data/record को अपनी जरुरत तथा limitation के अनुसार उपयोग करना होता है तथा उसे database के internal में होने वाले कार्य से कोई मतलब नहीं होता है।

INSTANCE & SCHEMA: -

INSTANCE: - किसी particular point of time में database के actual data/record को manipulate कराना। Instance किसी एक time में particular record के लिए point होता है तथा यह duplicate नहीं हो सकता।

SCHEMA: - Database के logical structure का design Schema कहलाता है। इसके मदद से database में उपयोग होने वाले table के relationship को represent करते हैं। Schema को table view के नाम से जाना जाता है अर्थात् यह table structure view को दर्शाता है। Schema 3 type के होते हैं –

1. External
2. Conceptual
3. Internal

Schema में हर एक level के अनुसार database को design किया जाता है।

***Logical Data Independence*: -** External schema या application program को change किए बिना उसमें बिना effect के Conceptual Schema (logical data) में change किया जा सकता है अर्थात् database में नया field जोडना और हटाने पर उसके external Schema में कोई परिवर्तन नहीं होता है।

***Physical Data Independence*: -** Physical Schema के संरचना या storage में परिवर्तन करने पर उसके external structure में कोई परिवर्तन या असर नहीं होता है। इसमें परिवर्तन तब किया जाता है जब database की file का storage structure change हो गया हो या database को recognized किया गया हो।

Difference between file processing system and DBMS system: -

S. N.	File Processing system	S. N.	DBMS System
1.	यह small system के लिए होता है।	1.	यह large system के लिए होता है।
2.	इसका implimentation cost कम होता है।	2.	इसका implementation cost ज्यादा होता है।
3.	इसमें बहुत कम file used हो सकती है।	3.	इसमें multiple file used होते हैं।
4.	इसका structure simple होता है।	4.	इसका structure complex होता है।

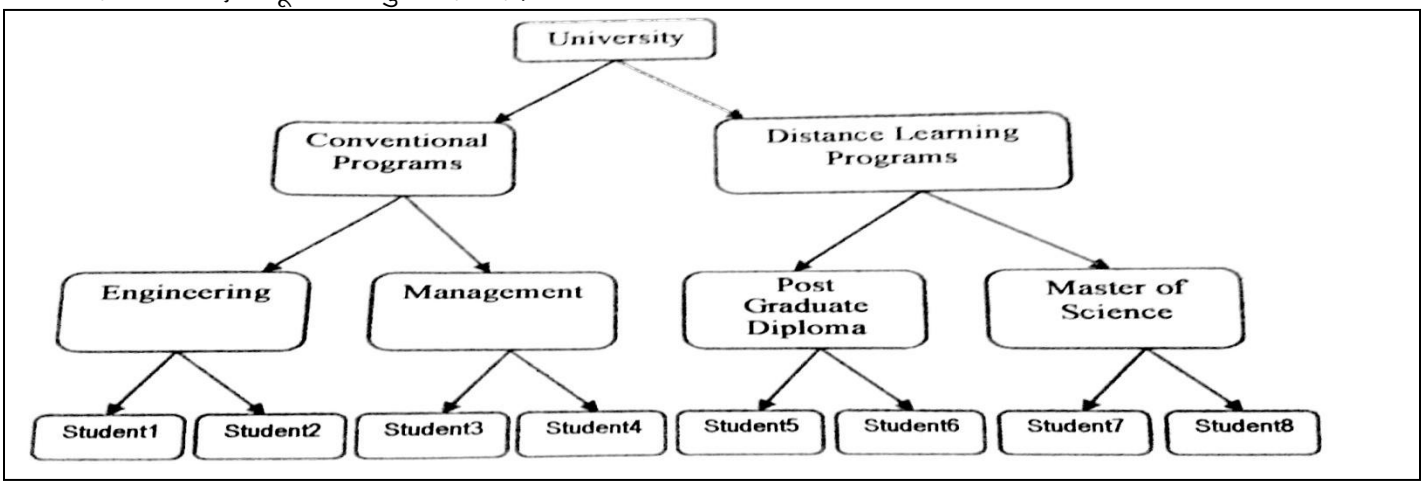
5.	इसमें security नहीं होती है।	**5.**	इसमें more security होती है।
6.	इसमें simple, primitive backup and recovery होता है।	**6.**	इसमें complex and sophisticated backup and recovery होता है।
7.	इसमें single user कार्य कर सकता है।	**7.**	इसमें multiple user कार्य कर सकता है।
8.	इसमें c, c++, cobal etc आते हैं।	**8.**	इसमें Orecle or Sbase, mogodb etc आते हैं।

Data Model Classification: -

Data Model	DBMS
Object Oriented Data Model	Object Store Versant
Hierarchical Data Model	IMS DBMS
Network Data Model	IDS, ID DBMS
Relational Data Model	MS-Access, Orecle, Sybase, Informix, Foxbase etc

Hierarchical Model: -

इस model में parent-child relationship होती है। इस model में प्रत्येक entity के पास केवल एक parent होता है और बहुत सारे children होते हैं। इस model में केवल एक entity होती है जिसे मि **Root** कहते हैं। इस model में data को Tree like structure में organized किया जाता है। इसमें data को record की तरह store किया जाता है जो कि एक दूसरे से जुडे रहते हैं।



Relational Model: -

इस model में data को relations अर्थात् table में store किया जाता है तथा प्रत्येक relation में Rows and Colums होते हैं। Relational Model, tables का समूह होता है जिसमें data and relationship को specify किया जाता है। Relational Model को 1969 में E. F. Codd द्वारा प्रस्तावित किया गया था।

Patient

PId	Pname	PAddress	PGender	Age
1.	Ram Kumar	M.P. Nagar, Bhopal	Male	67
2.	Laxman Garg	Subhas Gang Umaria	Male Child	7
3.	Sita Sharma	Station chauraha Rewa	Female Child	6

4.	Murli yadav	Bus stand Shahdol	Male	38
5.	Radha Verma	Kailash Nagar Katni	Female	41

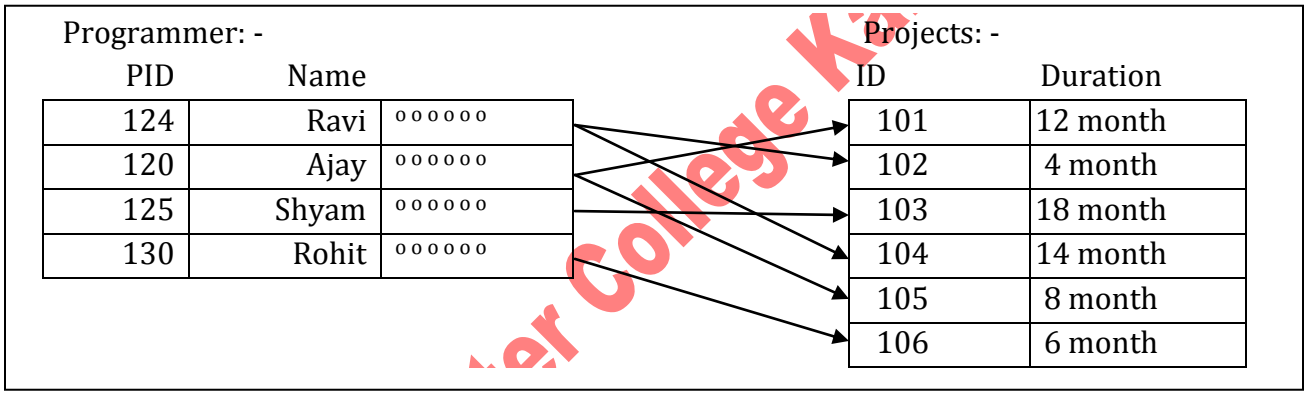
Doctor

Did	DName	Specialty	Mobile No.
101	A. Pathak	Heart	8517906324
102	P. Dwivedi	ENT	9098438606
103	H. Pandey	Eye	9179291220
104	A. Tiwari	Dentist	9039217212
105	Akash	Child	8982505087

Ward

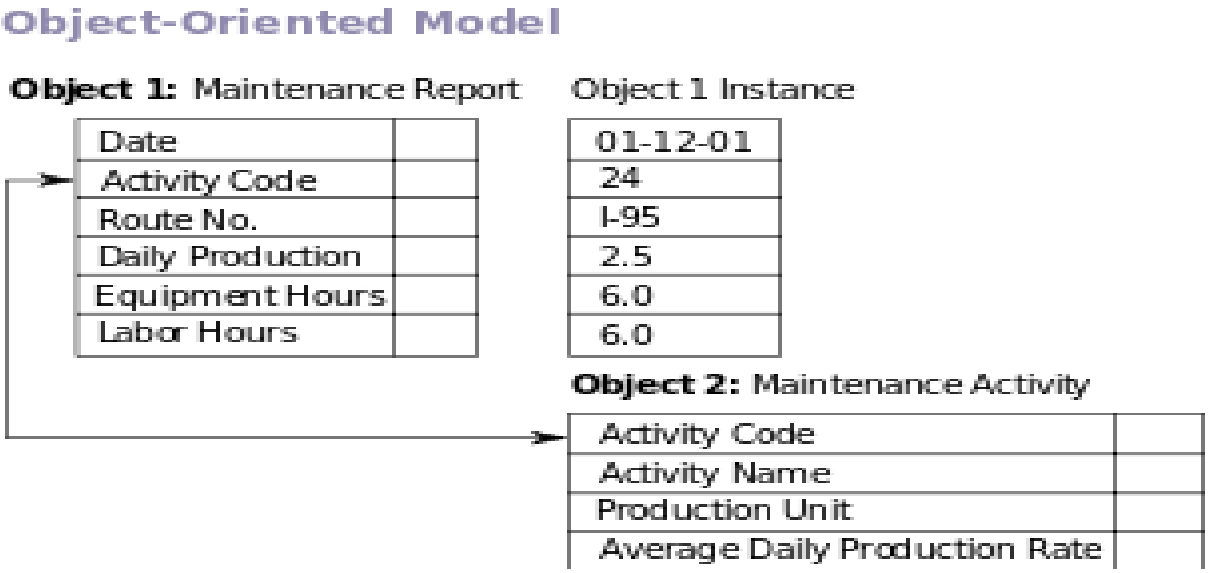
WNo	Type	Noofbeds	Cost
11	Special AC	10	450
22	Special Non AC	10	350
33	General	5	75
44	Student	50	50
55	Female	30	100

Network Model: - Network Model में entities को graph में organize किया जाता है और इनमें से कुछ entity अनेक path में से access कर सकती है, तो हम कह सकते हैं कि इस model में data को network के रुप में store and access करते हैं।



Object Oriented Model: -

Object Oriented Model में Information या data को object के रुप में प्रदर्शित किया जाता है तथा ये Objects instance variable में value को store किए रहते हैं। इस model में Object Oriented Programming capacity का use किया जाता है।



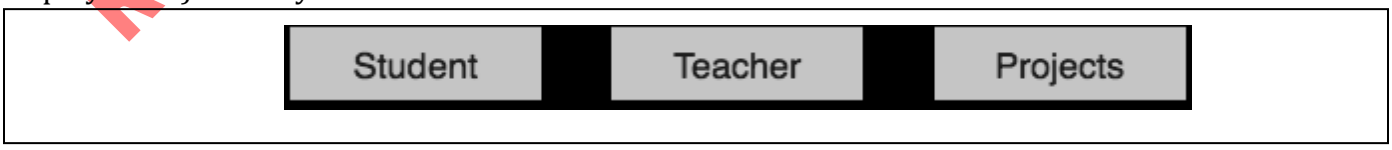
UNIT 2

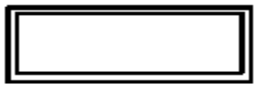
E-R MODEL& DIAGRAM: (ENTITY RELATIONAL MODEL): - Database में present data/Record के relationship को represent करने के लिए E-R Model का used किया जाता है। इसमें entity attribute and tuple का relation होता है।

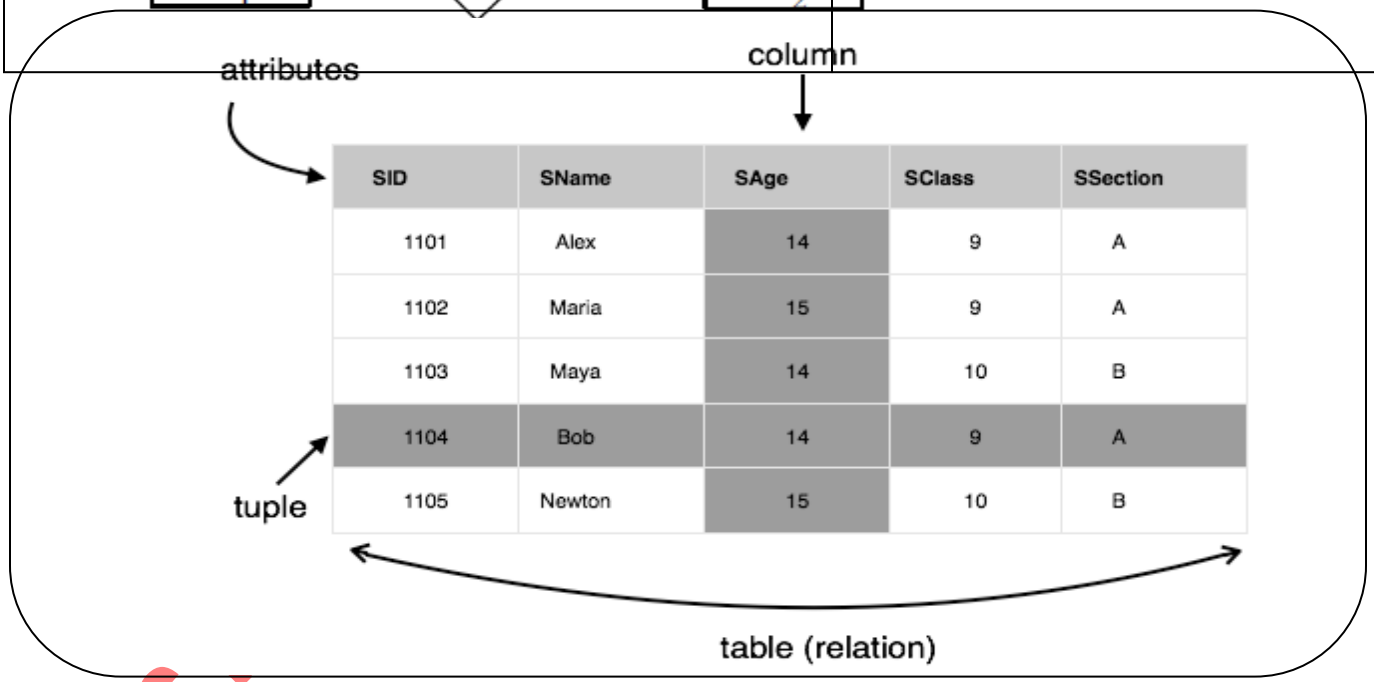
Tables: - Relational data model के अन्दर record को saved किया जाता है। Table के format में यह entities (table) को relation बना कर store करता है। एक table में Rows and Column होते हैं जहाँ Rows किसी record को represent करता है तथा column किसी attribute को represent करता है।

Tuple: - किसी table की एक single row जो कि contains करता है एक single record को इस प्रकार type के relation को एक tuple कहा जाता है।

Table Name/Entity: - Entity is a name of real world things (like- any person, customer, department, employee etc) ie entity is a collection of attribute.



Symbol	Meaning
	ENTITY TYPE
	WEAK ENTITY TYPE
	RELATIONSHIP TYPE
	ATTRIBUTE
	KEY ATTRIBUTE
	MULTIVALUED ATTRIBUTE
	COMPOSITE ATTRIBUTE
	DERIVED ATTRIBUTE
	CARDINALITY RATIO 1:N



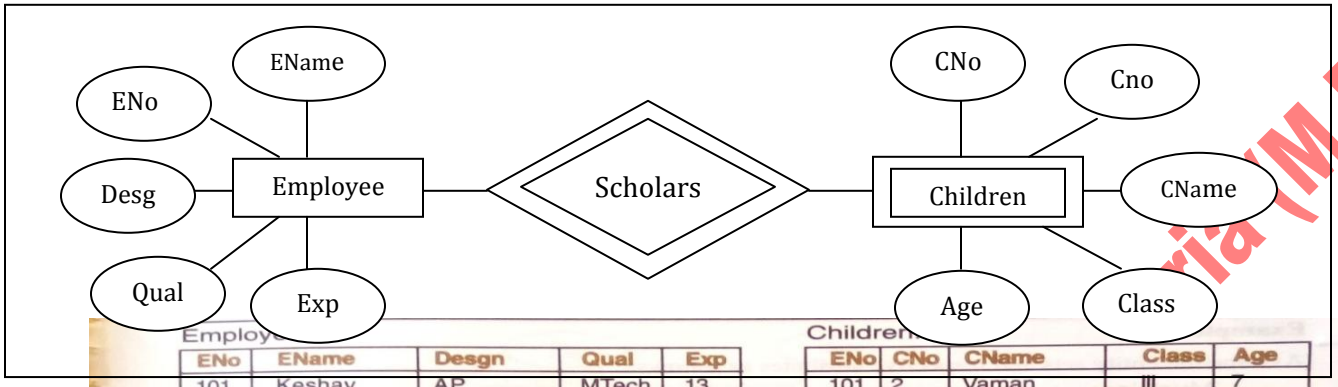
Attribute: - सभी attribute particular domain (entity) name के value के set को defind करते हैं। For example- School name का एक database है जिसमें एक student को as a entity considerd किया गया है। अब उस student के कई attribute हो सकते हैं like Name, Age, Class etc.

Week Entity: - एक ऐसी entity जिसको identify करने के लिए उसमें कोई भी "key" attribute न हो तथा वह अपनी identification के लिए किसी अन्य entity पर निर्भर हो सके, उसे week entity कहा जाता है।

Derived Attribute: - ऐसे attribute जिनकी value किसी अन्य attribute पर निर्भर हों derived attribute कहलाती है। Exapmle: -

MARKS 1	50
MARKS 2	100
MARKS 3	100
TOTAL MARKS	**250**

यहाँ marks 1,2,3 main attribute है तथा total marks derived attribute है, क्योंकि total marks, marks 1,2,3 पर निर्भर करता है।



Employee

ENo	EName	Desgn	Qual	Exp
101	Keshav	AP	MTech	13
102	Narayan	AP	ME	14
103	Madhav	Lecturer	BE	4
104	Govind	Professor	PhD	18
105	Vishnu	Director	PhD	24
106	Madhusudan	Lecturer	ME	4
107	Trivikram	Lecturer	BTech	3

Children

ENo	CNo	CName	Class	Age
101	2	Vaman	III	7
101	1	Shridhar	VII	11
102	1	Hrishikesh	V	9
103	1	Padmanabh	II	6
104	3	Damodar	VIII	12
104	2	Sankarshan	X	14
104	1	Vasudev	XII	16
105	2	Pradyumn	XI	15
105	1	Aniruddh	XII	16
106	1	Purushottam	III	7
107	1	Janardhan	III	7

Identified by

Composite Attribute: - ऐसे attribute जिनमें रखी गई जानकारी को और भी अन्य भाग में विभाजित किया जा सके उसे composite attribute कहते हैं।

Exapmle: - Student name एक attribute (main) है लेकिन इसें composite किया जा सकता है। जैसे first name, middle name and last name.

Identification Relationship: - ऐसा relationship जो week entity का किसी अन्य entity के साथ relation established करती है जिसके मदद से week entity को identify किया जा सकता है उसे identification relationship कहते हैं।

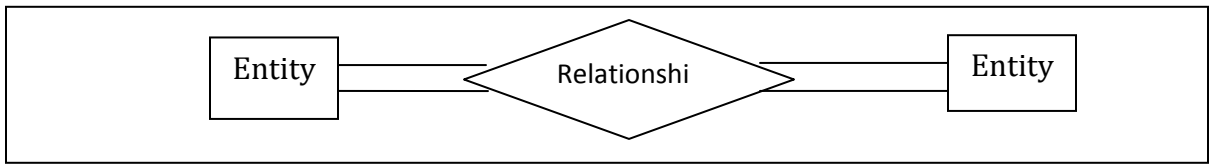
Constraints: - Constraints is a set of rules & regulations and restriction. जिसे follow किया जाता है।

Total Participation Constraints: - जब किसी entity set की सभी entity किसी relation में भाग ले तो वह total participation कहलाता है। Example: Student entity set जिसका relation cource से है क्योंकि यहाँ पर प्रत्येक student को किसी न किसी cource में admission लेना अनिवार्य है क्योंकि बिना admission के student entity का कोई existence नहीं है।

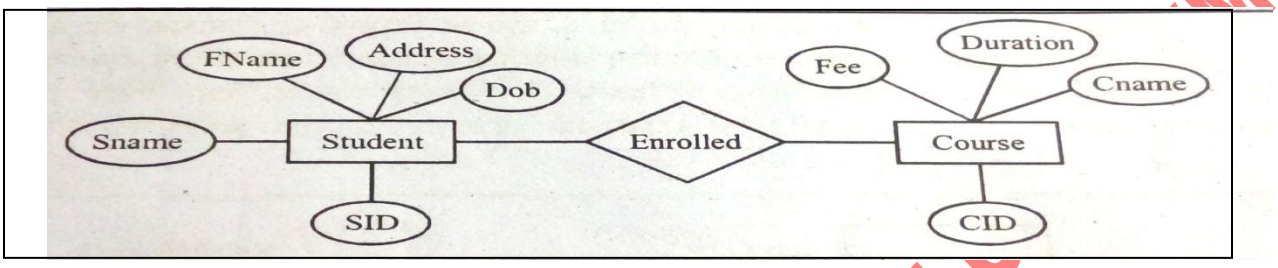
Partial Participation Constraints: - इस प्रकार के participation में प्रत्येक entity का relation में भाग लेना अनिवार्य नहीं होता है। Example: Student and library member के बीच में इसमें यह bound नहीं है कि प्रत्येक student, library के member हो।

Total Participation: - इसमें प्रत्येक entity, relation में भाग लेता है। Total participation को double line से represented किया जाता है।

Partial Participation: - इसमें entity set में present प्रत्येक entity relationship में भाग नहीं लेते है। Partial participation को single line से represented किया जाता है।



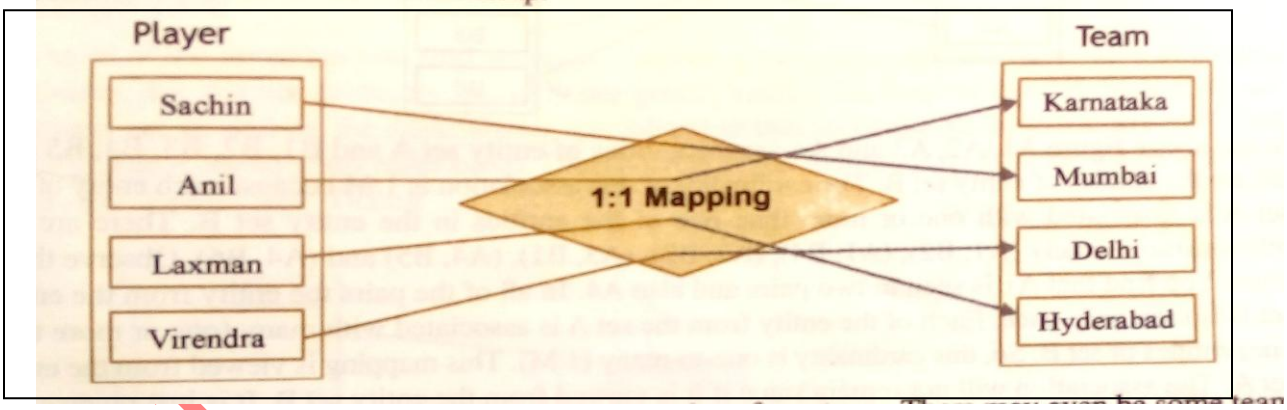
Relationship and Relationship Set: - Multiple entity के बीच के आपसी सम्बन्ध को relationship कहते हैं तथा एक से ज्यादा entity के set के बीच के relation को relationship Set कहा जाता है।



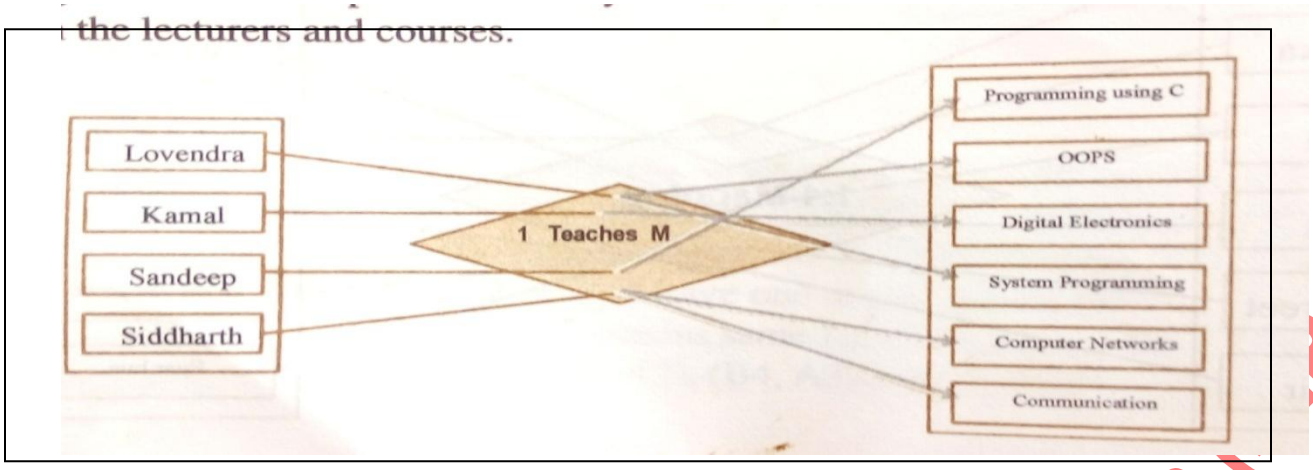
जैसे 100 student का admission हुए और 10 cource का registraction है जिनमें 100 student का group अपने आप में एक set of entity है। इसी तरह cource भी अपने आप में एक set of entity है, अब 100 student and 10 cources के आपस में relation को relationship set कहा जाता है।

Type of relationship: **-** Relationship के प्रकार इस बात पर निर्भर करते हैं कि किसी relationship में शामिल दो entity set में से प्रत्येक entity set के member (attribute) किसी अन्य entity set के एक ही member का relative हो सकता है या एक से ज्यादा member से इस आधार में relationship को अनेक भाग में विभाजित किया जा सकता है –

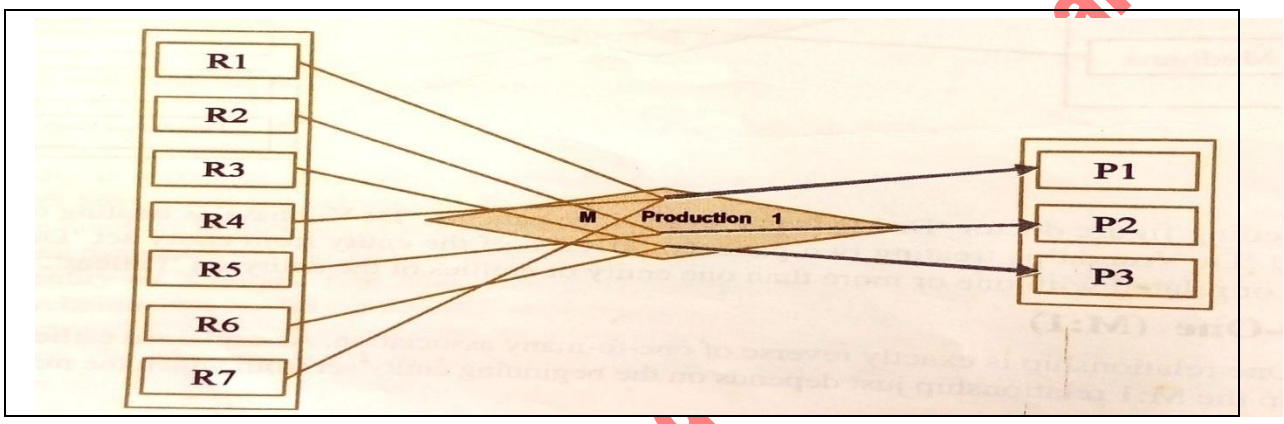
One–To–One (1:1): - जब किसी entity set A का member, entity set B के एक ही member से relative हो तो इस प्रकार set B के प्रत्येक members, set A के किसी एक member से relation हो, उसे 1:1 relationship कहते हैं।



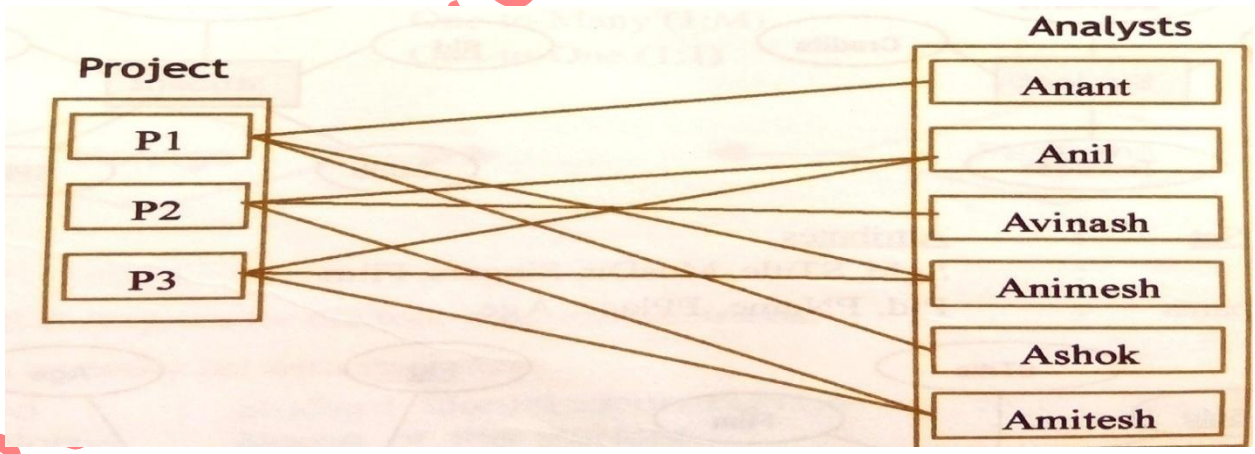
One-To-Many (1: M): - जब entity set A के प्रत्येक member में से कोई एक member, entity set B के multiple member से सम्बन्धित हो तो उसे 1:M relationship कहते हैं।



Many-To-One (M: 1):- Just opposite to one to many relationship.

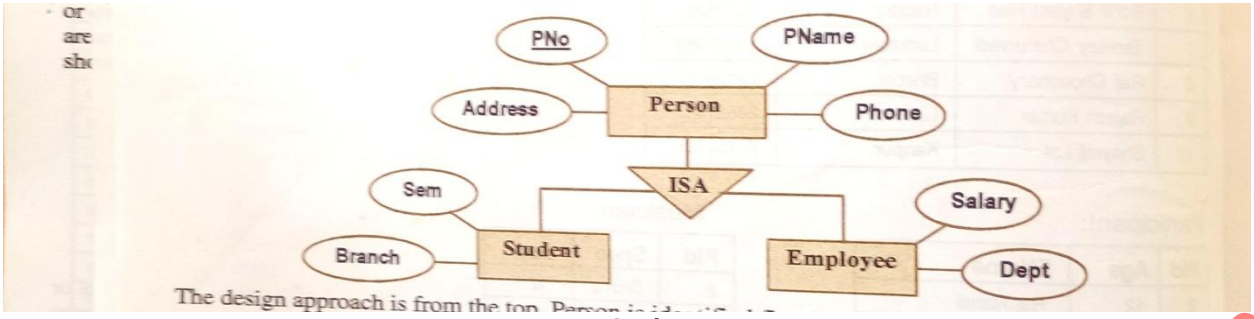


Many-To-Many (M: M):-- Set A के multiple members, Set-B के multiple members से relative हो सकते है तथा set-B के multiple members, Set-A के multiple members से relationship रख सके उसे M: M relationship कहा जता है।



***Feature of E-R Diagram*:-**

1) **Specialization:-** एक या एक से अधिक entity ऐसी हो सकती है जिसमे special type की functionality हो सकती है और उन्हे different level मे divide किया जा सकता है तथा नई बनी हुई entity child entity कहलाती है, लेकिन इस प्रकार के entity का अपनी अलग-2 functionality होती है, यह Top-Down approach कहलाती है ।



The design approach is from the top. Person is ...

Person

PNo	PName	Address	Phone No
11	Sachin	E-93 Vidya Nagar	222727
12	Sanjay	C-115 Ganesh colony	223272
13	Sunil	Ward No 5 Dev road	222781
101	Sateesh	B/5 Vikatgamj	222939
102	Ashish	P-25 Prakash Dwar	242358
103	Amit	D110 Vidya bahar	254306
104	Praveen	Ward No 12 Umaria	243567
14	Vivek	101 Kamp mohalla	222555
15	Akash	121 shahpura road	222275
16	Naman	105 vinyal town	234567
17	Neeraj	55 Nirmala nagar	242343

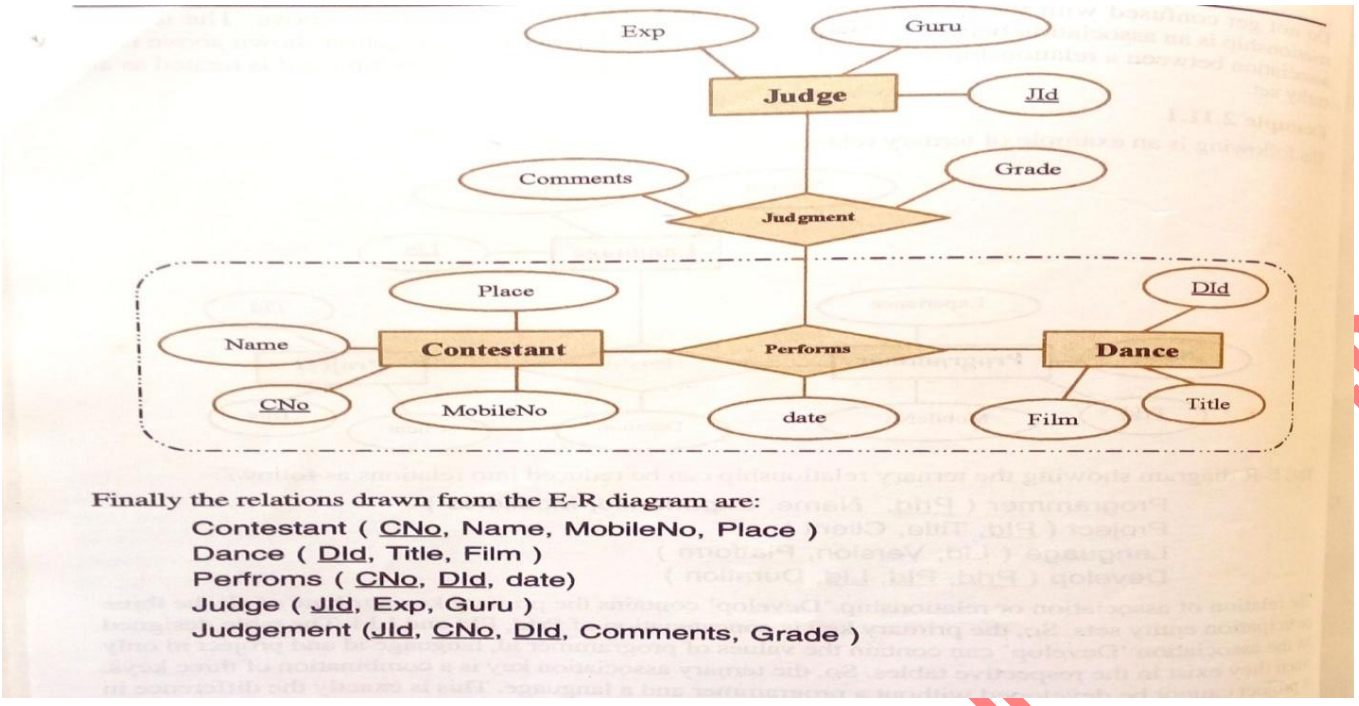
Student

PNo	Sem	Branch
101	2	Computer Science
102	2	Electronics
103	2	Electronics
104	2	Copmuter science

Employee

Pno	Dept	Salary
11	Computer science	22000
12	Electronics	21000
13	Computer science	22000
14	Electronics	15000
15	Computer science	27000
16	Electronics	35000
17	Computer science	25000

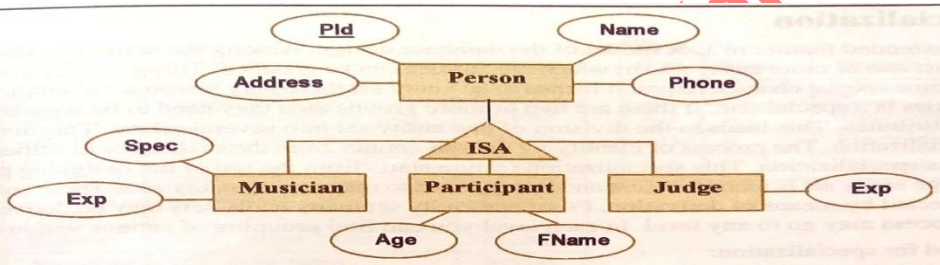
2) Aggregation: - इसका अर्थ है समूहीकरण अर्थात् अगल-2 Unit को उनके functionality के base पर combine कर एक बडा समूह बनाना। इसे 3 level में देखा जा सकता है।



Finally the relations drawn from the E-R diagram are:

Contestant (CNo, Name, MobileNo, Place)
Dance (DId, Title, Film)
Perfroms (CNo, DId, date)
Judge (JId, Exp, Guru)
Judgement (JId, CNo, DId, Comments, Grade)

3) Generalization:- It is just opposite specialization. इसमें botton-top में process होती है। इसमें child entity को combine करके नई entity बनती है जिसे parent entity कहते हैं।



The relations derived from the above E-R diagram are:

Person (PId, Name, Address, Phone)
Participant (PId, Age, FName)
Judge (PId, Exp)
Musician (PId, Spec, Exp)

The E-R diagram is reduced to the following tables:

Person:

PId	Name	Address	Phone
1	Suresh Wadkar	Juhu, Mumbai	22521425
2	Sonu Nigam	Kalyan, Mumbai	22415265
3	Smita Nandi	Tripura	244547
4	Snehashree	Mumbai	22154212
5	Vasundhara	New Delhi	11425421
6	Rohit Shyam Rao	Nagpur	14215421
7	Tanmay Chaturvedi	Lucknow	14521452
8	Raj Choudhary	Bhopal	22124451
9	Rajesh Kumar	Lucknow	24521523
10	Shaymji Lal	Kanpur	22450125

Judge:

PId	Exp
1	25
2	10

Participant:

PId	Age	FName
3	12	Raj Nandi
4	15	Rajeev Joshi
5	16	Harsh Modi
6	13	Raghvendra Rao
7	11	Sirish Chaturvedi

Musician:

PId	Spec	Exp
8	Drums	12
9	Sitar	8
10	Guitar	15

Code's of Laws:--

1) Representation of Information: - Relation database के logical format में से प्रत्येक information को केवल table में value के format में represent किया जा सकता है।

2) Accessing Information: - Database में यदि record present है तो access किया जा सकता है। यह access table के name, primary key value और दी गई परिस्थिति के सही होने पर उपलब्ध जानकारी के आधार पर होता है।

3) Arranging the proper NULL value: - जिस column में data/record present नहीं है तो उसे empty नहीं रखा जाता। उसमें NULL डाल कर दिया जाता है जिसका अर्थ यह हुआ कि data उपलब्ध नहीं है। NULL value को independent data type भी कहा जाता है।

4) Dynamic Online catalog based on relational Model: - User अपनी जरुरत के अनुसार data की खेज कर सके जिसके लिए query language का उपयोग किया जाता है।

5) Updating of View: - View constant नहीं होता जिसके लिए view को समय-2 पर update होते रहना चाहिए।

6) High level insertion, detection and modification time to time perform करना चाहिए।

7) Logical data, Physical data and integrity indecency होना चाहिए।

Domain Constraints: **-** किसी भी सम्भावित मान का set, Domain कहलाता है। अतः table के प्रत्येक attribute का एक विशेष domain होता है तथा उस attribute में उसके fixed range से बाहर value नहीं डाला जा सकता है। Domain सम्भावित value के प्रकार भी निर्धारित करता है। जैसे अगर किसी employee की age को set करना है, जो 18 से 60 के बीच हो जो कि integer type का हो सकता है अब उस table में सिर्फ integer type value ही insert कर सकते हैं और जो कि 18-60 के बीच की ही हो।

Entity Integrity and Reference Integrity & Key: **-** Entity integrity यह सुनिश्चित करता है कि किसी भी table के primary key का मान NULL नहीं हो सकता है।

Employee:

Emp_ID	Name	Age
XI	ABHILASH	25
NULL	PRAKASH	28

यहाँ पर entity integrity rule के अनुसार follow नहीं कर पाता।

Reference Integrity: **-** किसी एक entity का relation किसी दूसरे entity के record को referred करें, जिसे लागू करने के लिए foreign key का use किया जाता है।

Department:

Dept_ID	Name	Age
1	XI	IT
2	NULL	HR

यहाँ पर नियम के अनुसार reference integrity का उपयोग नहीं हो पा रहा है।

Key: - किसी entity set के particular record को एक single attribute की मदद से uniquely identify किया जा सके उसे **Key** कहते हैं।

Type of Key: -

1) Super key: - एक ऐसे attribute का set जिसका उपयोग table के record को unique रुप से identify करने के लिए किया जाता है, Super key कहलाता है।

Employee

Emp_Id	Name	Age
1001	ABHILASH	25
1002	PRAKASH	28

Emp_Id	Mobile No	Dept
1001	8517906324	IT
1002	8982505087	HR

Where {Emp_Id, Name, age}, {Emp_Id, Mobile, Dept} Emp_Id working as a super Key.

2) Candidate Key: - Super key के set में से जो primary key बनने की eligibility रखते हैं उन्हें candidate key dgrs gSaA ;gk¡ ij emp_id, name, age attribute शामिल हैं जिनमें कि name, age external attribute की category में आते हैं जिसे delete भी कर दिया जाए तब भी emp_id candidate key की तरह कार्य करती है।

3) Primary Key: - Available candidate key की मदद से database designer जिसका selection record को identify करने में करता है उसे primary key कहते हैं। अगर किसी relation में एक ही candidate key हो तो उसमें 100% primary kry भी होगी, लेकिन अगर उसमें एक से ज्यादा candidate key हैं तो उसमें किसी एक का selection primary key के रुप में किया जाता है।

Employee

Emp_Id (primary key)	Name	Age
1001	ABHILASH	25
1002	PRAKASH	28

I. Primary key unique रुप से record को identify करने में capable होता है।
II. Primary key NULL value को contain नहीं करता है, अर्थात् उसमें कुछ न कुछ unique value होना चाहिए।

4) Alternative Key (Second key): - Candidate key में से select किए गए primary key के रुप के attritube के अतिरिक्त शेष बचे key को alternative key कहा जाता है।

Customer Details

Cust_Id (Primery key or alternative key)	Bank_A/c_No (Primary kry/alternative key)	Name	Age
1001	484321	ABHILASH	25
1002	484322	PRAKASH	28

5) Foreign Key: - किसी दो table (entity) के बीच के reference intrgrity established करने के लिए उपयोग किए जाने वाले common attribute जो कि दोनो में से किसी एक table में primary key हो तथा दूसरे table में वह primary kry न हो तब दूसरे table में इस attribute को Foreign key कहते हैं।

Employee

EMP_Id (Foreign key)	Name	Age
1001	ABHI	25
1002	PRAKASH	28

→

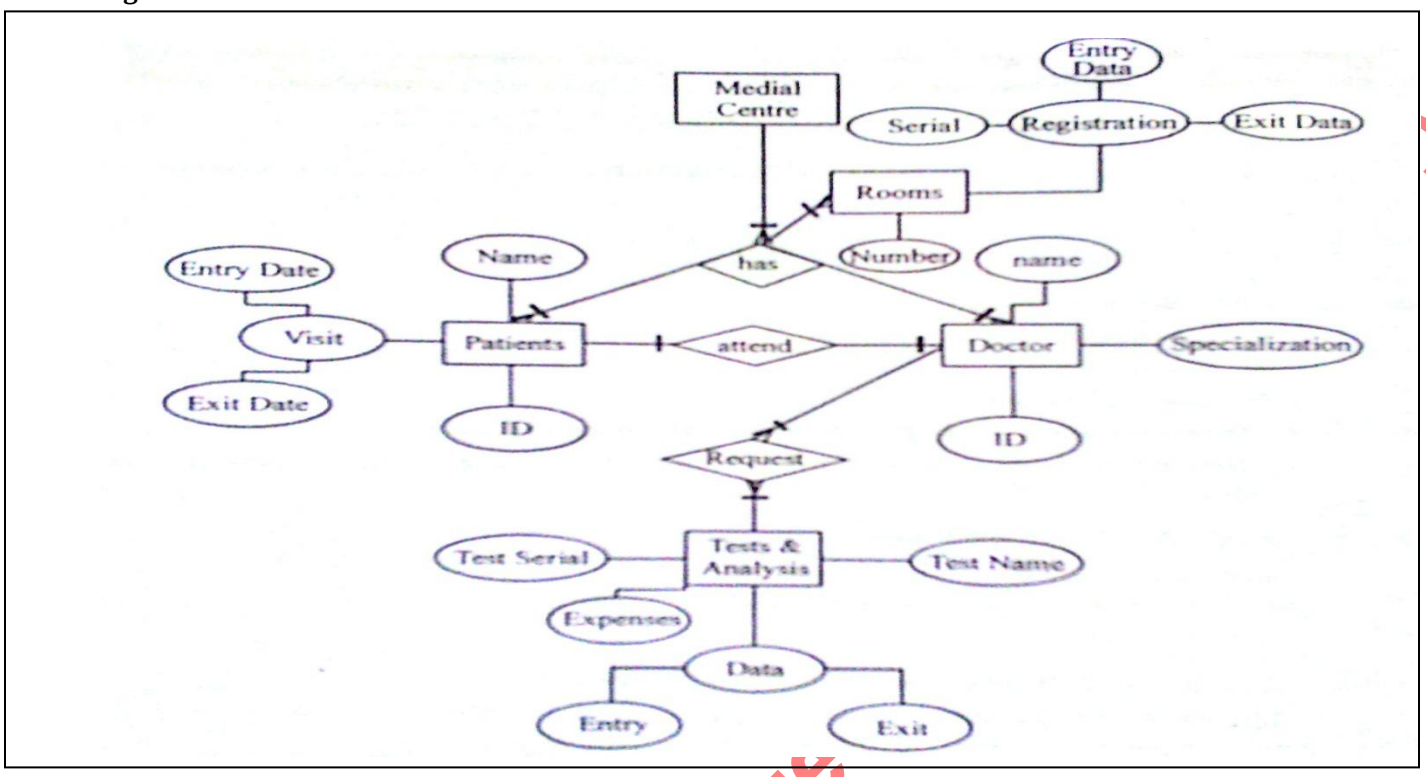
Depertment

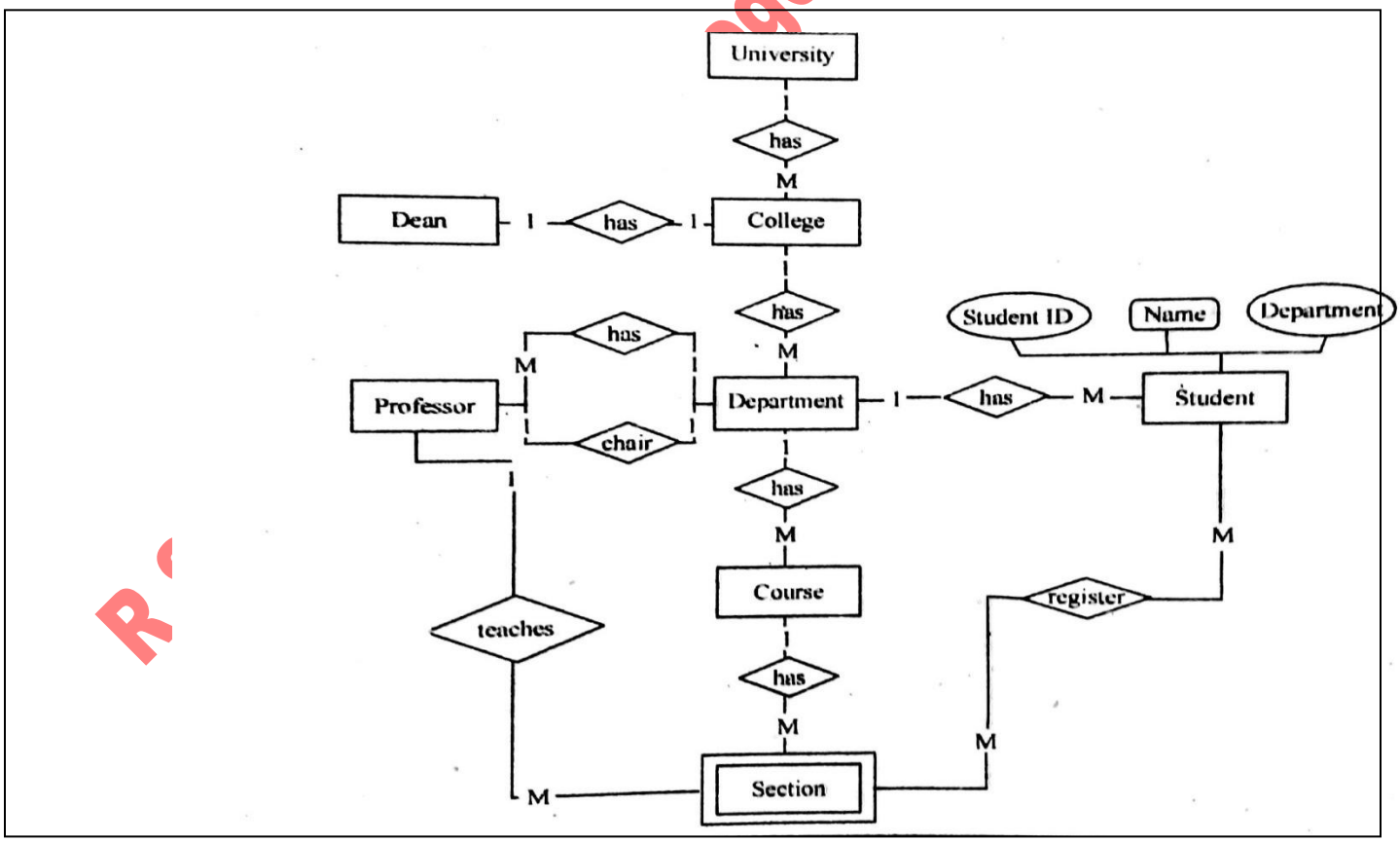
Dep_Id (Primary key)	EMP_Id (Foreign key)	Name	Age
10	1001	ABHI	25
11	1002	PRAKASH	28

Extension and Intension: -
Set के सभी members की जानकारी extension कहलाती है।
Set A = {Orange, Green, Blue, White}

And Set के सभी member की list represent न करके केवल उस condition को दर्शाना जो उस Set के बारे में represent करता हो जैसे Set A = {A color of National flag}

E-R Diagaram:





UNIT-3

➢ *Feature of Good Database Design*: -

1. Data को store करने के लिए storage area sufficient होना चाहिए।
2. End user को data को readily करने के लिए उपलब्ध होना चाहिए।
3. Database की design ऐसी होनी चाहिए जिसे कोई अनाधिकारिक व्यक्ति access न कर सके।
4. Database design ऐसी होनी चाहिए जिसे आसानी एवं सटीक तरीके से manage किया जा सके।
5. Overall database design ऐसा होना चाहिए जिससे उसकी performance को accept किया जा सके।
6. Design ऐसी होनी चाहिए कि उसमें duplicate data store न हो।

Universal Relation: - किसी database की design करने के लिए multiple problem को face करना पड़ता हैं एक entity set में multiple attribute हो सकते हैं। इन set of attribute को universal schema के तहत प्रदर्शित किया जाता है और उसे **Universal Relation** कहते हैं।

Relation A

Code	Professor	Department
1001	A.K. Pathak	……
1002	P.K. Dwivedi	……
1003	S. Khan	……
……	A.K. Pathak	C. S.
……	Ankur Tiwari	EEE
……	P.K. dwivedi	Math

Extended Universal Relation: -

Relation 1

Code	Department
1001	C. S.
1002	EEE
1003	MATH

Relation 2

Professor	Department
A.K. Pathak	C. S.
S. Khan	EEE
P.K. Dwivedi	MATH

यहाँ पर universal relation का concept follow नहीं हो पा रहा है क्योंकि NULL entry हो रही है जो कि data base के नियम में नहीं है। अतः दस समस्या से बचने के लिए relation को extend कर दिया गया है जिससे Extended Universal Relation का उपयोग किया जाता है।

Functional Dependency: - Database file के किसी attribute को अपनी identity के लिए किसी अन्य attribute पर निर्भर होना functional dependency कहलाता है।

Student

Enroll_No (Primary key)	Name	Age
1001	ABHI	25
1002	PRAKASH	28
1003	ANUJ	30

जैसे student table के सभी attribute enrollment number में निर्भर करता है। अगर एक ही enrolled number एक से ज्यादा student से जुडा होगा तब उस condition में attribute को identify कराने में समस्या उत्पन्न होगी। इसलिए एक enrolled number किसी एक ही student को assigned होगा।

Armstrong's Axioms: -

1) Reflexive Rules: - यदि Alpha, Attribute का एक set है और Bita, alpha का sub-set है तो alpha holds bita attribute (subset) का मान।

2) Augmentation Rules: - If $a \rightarrow b$ holds and y attribute का set है तो $ay \rightarrow by$ also holds. इस प्रकार के adding attributes dependencies को show करते हैं, जिसमें basic dependencies को change नहीं किया जा सकता है।

3) Transitivity Rules: - Transitive rule, algebra की ही तरह है। यदि $a \rightarrow b$ को holds करता है और $b \rightarrow c$ को holds करता है तो $a \rightarrow c$ also holds $a \rightarrow b$. जहाँ a एक functionality है जिसे b determines करता है।

Trivial Functional Dependency: -

1) **Trivial: -** यदि functional dependency (FD) X → Y को holds करता है जहाँ Y, X का एक Subset हो तो उसे trivial FD के नाम से जाना जाता है। Trivial FD's हमेशा Subset को holds करता है।

2) **Non-Trivial: -** यदि functional dependency (FD) X → Y को holds करता है जहाँ Y, X का Subset न हो तो उसें Non-Trivial FD के नाम से जाना जाता है।

3) **Completely Non-Trivial: -** If an FD X → Y को holds करे जहाँ X intersect Y = Φ हो उसे एक completely Non-Trivial FD कहा जाता है।

What is normalization? Explain with details.

Anomalies: - Database के अच्छे design का न होना जिसके कारण database administrator को database में proper value के insertion, deletion, updating इत्यादि उचित तरीके से कार्य perform न हो उसे anomalies problem कहते हैं।

Normalization: - Relational database design के लिए E-R model and Normalization दो मुख्य पद्धति है। Relational database design के सभी attribute को एक ही जगह पर एकत्र कर लिया जाता है तथा आगे फिर इन्हें functional dependency and multi-value dependency etc के आधार पर decompose करके छोटे-2 relation में विभाजित करना decompose कहलाता है तथा इन छोटे-2 relation का storage relation database कहलाता है। यह अनियमितता की समस्या को हल करता है।

Normalization एक ऐसी तकनीक है जो complex unrelational database के संरचना को small relation format में change कर देती है। जिसके मदद से small relational database को manage करना तथा एक अच्छे database को create कर सकते हैं।

इस प्रकार उचित व्यवस्थित एवं तकनीकी मदद तथा सही decompose से create किए गए erlational database में सफलतापूर्वक database operation perform कर सकते हैं।

Normalization decomposition process में information loss होने से बच जाता है। यह bottom up approach में बहुत महत्वपूर्ण है तथा normalization की मदद से अच्छे database की design प्राप्त हो जाती है।

Type of Normilization Form: - First Normalization form, normalization की process intilization करने के लिए database के सभी attribute को एक ही जगह में एकत्र कर लिया जाता है। कोई भी relation, first normalization form तभी होगा जब –

1. Relation में कोई भी Row/Tuple duplicate न हो।

2. प्रत्येक column (attribute) में होने वाली सभी entry एक ही type की होनी चाहिए।

3. Relation में store की गर्द प्रत्येक data value, single value होनी चाहिए अर्थात् किसी भी row में एक ही attribute में एक से ज्यादा value नहीं आने चाहिए।

NOTE: - Each relation में एक key attribute होना आवश्यक होता है। यह attribute single/composite attribute हो सकता है।

Unorganized Relation: -

Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

Re-arranged Relation (1 N, F): -

Each attribute must contain only a single value from its predefined domain.

Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

Second Normal Form (2nd N.F):- एक relation second N.F तब होता है जब वह first N.F हो तथा उसके सभी Non-Key attribute पूरी तरह से relation के किसी एक candidate key attribute पर निर्भर हो।

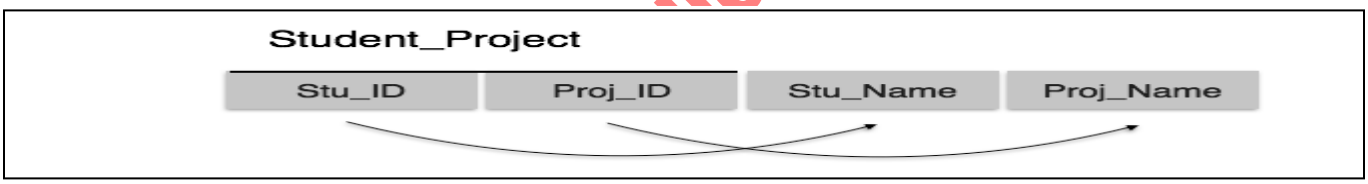
Prime attribute: An attribute, which is a part of the prime-key, is known as a prime attribute.

Non-prime attribute: An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

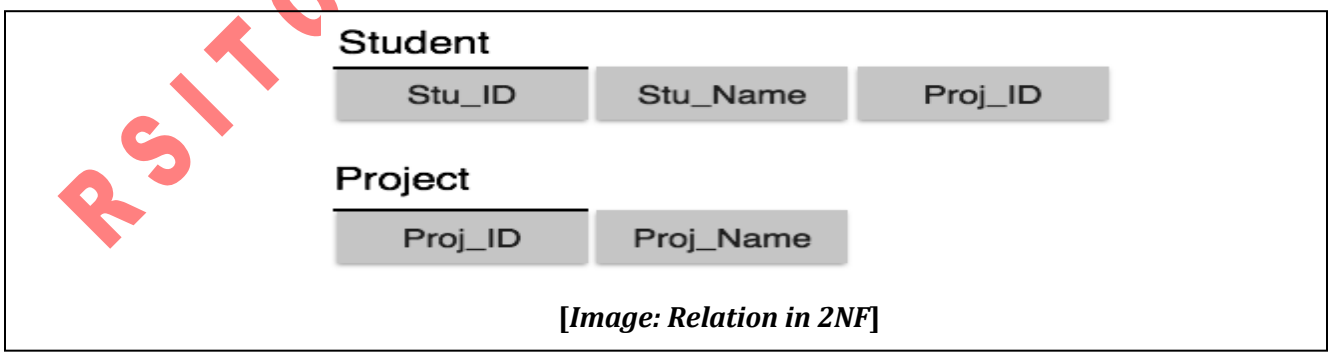
1. Relation में यह define होना चाहिए कि एक relation में किसी एक ही item (attribute) के बारे में ही जानकारी को प्रदर्शित करें।

2. जिस relation में सिर्फ एक ही key attribute हो तो already second N.F में होता है अतः relation को second N.F में decompose करते समय यह सम्भावना होनी चाहिए कि उत्पन्न होने वाली सभी new relation में एक ही key attribute हो।

3. Key attribute के रुप में composite key होने की condition में किसी भी attribute की key attribute में शामिल किसी एक attribute पर partial dependency को consider नहीं किया जा सकता।



यहाँ पर rule के अनुसार stu_Id and proj_Id, prime_key attribute है पर stu_name and proj_name दोनो non-prime attribute है, लेकिन दोनो अलग-2 primary key attribute defined कर रहे है। अलग-2 Non-key attribute को जिसे हम partial dependency कहते हैं, इसे second normal form की category में नहीं रख सकते। अतः second N.F बनाने के लिए partial dependency को remove करना पडेगा जो इस प्रकार का होगा–

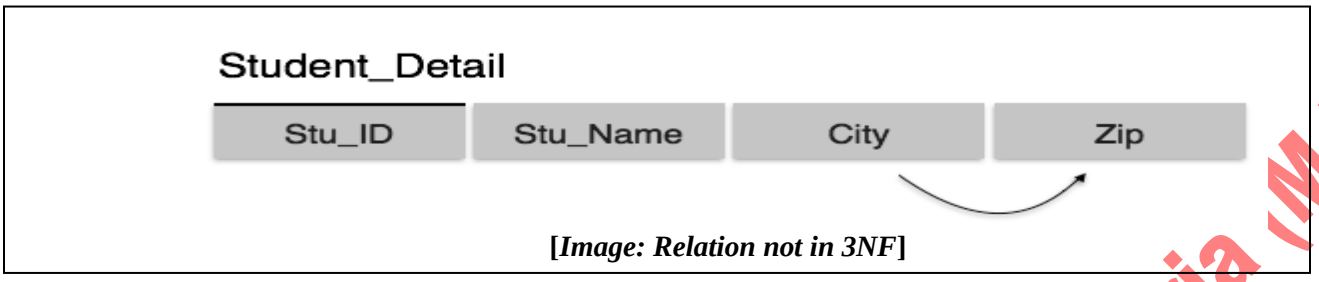


[*Image: Relation in 2NF*]

हम relation को दो भाग में तोड़ सकते हैं जो above picture में है जो कि इसमें कोई भी partial dependency नहीं है।

3rd Normalization Form: - कोई भी relation 3rd Normal form तभी consider किया जाता है जब वह 2nd Normal form में हो और following condition में satisfy करे।

1. No non-prime attributes को prime key attribute में transitively निर्भर होना चाहिए।
2. कोई भी Non-trivial functional dependency X→A है तो, या तो:
   X एक super key हो अथवा A एक prime attribuitr हो।



[*Image: Relation not in 3NF*]

यदि हमे find करना हो ऊपर दिए गए Student_detail relation से, तो Stu_ID एक key है और सिर्फ prime key attribute. हमने find किया कि City को identified किया जा रहा है by Stu_ID as well as Zip itself से जहाँ पर Zip न ही super key है और न ही City prime attribute है। Additionally, Stu_ID→ Zip → City, जो कि transitive dependency को show करता है।
यदि relation को third normal form मे create करना है तो relation को two part मे तोडना होगा।

Student_Detail

Stu_ID	Stu_Name	Zip

Zip Codes

Zip	City

[*Image: Relation in 3NF*]

Boyce-Codd Normal Form: - Boyce-codd normal form(BCNF) के होने के लिए relation को third normal form में होना चाहिए। कोई भी Non-trivial functional dependence X→A, X को super key होना चाहिए। ऊपर दिए गए चित्र में Stu_ID, Student_Detail relation के लिए super key है और Zip, zip codes relation के लिए super key है। अर्थात् Stu_ID→Stu_Name, Zip और Zip → City. जो कि conform है कि दोनो relation BCNF के अंदर है।

4TH Normalization Form: - एक relation या table तब 4NF में होती है जब वह निम्नलिखित condition में satisfy करते है। एक relation या table तब 4NF में होती यदि वह 3 normal form (3NF) में हो तथा उसके पास कोई multivalued dependency न हो।

Multi valued Dependencey क्या होती है?

Multivalued dependencey तब होती है जब एक table में एक से ज्यादा independent multivalued attributes होते है। Multivalued dependencey को —» चिन्ह से प्रदर्शित किया जाता है।
उदाहरण के लिए– Student किसी एक subject को read करने के लिए multiple book का used करता है, अर्थात् student book multi value dependency है।

Book

Subject	Professer	Book_Name
DBMS	A.K. PATHAK	A, B
DBMS	S. KHAN	A, B

DIGITAL	P.K. DWIVEDI	C, D
NETWORKING	A. PRAJAPATI	E, F

Book table को 4.NF में change करने के लिए इसे 1.NF में change करना होगा, क्योंकि professor कि multivalue dependency है।

SUBJECT	PROFESSER	BOOK_NAME
DBMS	A.K. PATHAK	A
DBMS	A.K. PATHAK	B
DBMS	S. KHAN	A
DBMS	S. KHAN	B
DIGITAL	P.K. DWIVEDI	C
DIGITAL	P.K. DWIVEDI	D
NETWORKING	A. PRAJAPATI	E
NETWORKING	A. PRAJAPATI	F

इस table को 4 NF में परिवर्तित करके multi value dependency को Remove किया जाता है। अब यहाँ पर professor and Book_Name एक दूसरे से independence हैं तथा वे Subject पर निर्भर है, तो हम इस प्रकार की dependency को निम्न प्रकार से प्रदर्शित करते हैं–

Subject –≫ Professor

Subject –≫ Book_Name

Book

SUBJECT	PROFESSER
DBMS	A.K. PATHAK
DBMS	A.K. PATHAK
DBMS	S. KHAN
DBMS	S. KHAN
DIGITAL	P.K. DWIVEDI
DIGITAL	P.K. DWIVEDI
NETWORKING	A. PRAJAPATI
NETWORKING	A. PRAJAPATI

Subject_book_name

SUBJECT	Book_Name
DBMS	A
DBMS	B
DIGITAL	C
DIGITAL	D
NETWORKING	E
NETWORKING	F

थ्कसी table में multi value dependenct होने के कारण record को update, insert, delete इत्यादि query perform करने में समस्या होती है इसलिए table को 4NF में change किया जाता है।

5th Normal Form:

एक relation या table तब 5NF में होती है जब वह निम्नलिखित condition को satisfy करती है– "एक table या relation तब 5NF में होती है जब वह 4NF में हो तथा table में कोई Non-loss decomposition न हो।"

Non loss decomposition:-

जब कभी table या relation में data normalization किया जाता है यदि उसमें सूचना (information) का loss (हानि) नही होता है तब उसे Non-loss decomposition कहते है।

कभी-कभी हम इसे loss less decomposition भी कहते है।

आसान भाषा में कहें तो loss less decomposition एक ऐसी प्रक्रिया है जिसमें duplicate data को हटा दिया जाता है और यह भी देखा जाता है कि इसमें original data की हानि न हो।

UNIT -4

***Relation Algebra and Operation*: -** किसी भी relation या table में विभिन्न प्रकार के algebraic operations के collection को perform कराना relation algebra and operation कहलाता है। Relation के मुख्यतः दो भाग होते हैं–

1) Set Oriented Operation

2) Relation Oriented Operation

1) Set Oriented Relation:- इसमें tradional set operationआते हैं। जैसे– Union, Cartesian product, intersection, difference etc.

Union: - इसमें दो table के बीच relation में operation perform होता है जिसमें दोनो table के record को combine कर दिया जाता है और फिर उन record को filter कर एक single table format में record को प्रदर्शित किया जाता है तथा duplicate value को remove करके single value के रुप में उपयोग किया जाता है।

A

Roll_No	Name
101	Abhi
102	Prakash
103	Akash

B

Roll_No	Name
101	Abhi
104	Hemant
105	Ankur

A∪B

Roll_No	Name
101	Abhi
102	Prakash
103	Akash
104	Hemant
105	Ankur

Interjection: - Just opposite of union इसमें output में table common tuple को show किया जाता है।

A ñ B

Roll_No	Name
101	Abhi

Diffrence operation: - इसमें Left side and right side table के record के बीच में operation perform किया जाता है और left side के table के record को लिया जाता है, लेकिन जो common tuple न हो right side की table में।

A−B

Roll_No	Name
102	Prakash
103	Anuj

Cartesian product: - इसमें सभी possible combination value output के रुप में show होता है।

A

Roll_No	Name
101	Abhi
102	Prakash
103	Akash

B

Roll_No	Name
104	Hemant
105	Ankur
106	Anuj

A * B

A (Roll_No)	A (Name)	B (Roll_No)	B (Name)
101	Abhi	104	Hemant
101	Abhi	104	Hemant
102	Prakash	105	Ankur
102	Prakash	105	Ankur
103	Akash	106	Anuj
103	Akash	106	Anuj

2) Relation Oriented operation: - Set Operation की मदद से data को multiple करने के लिए limited facility होती है लेकिन इसमें बहुत कुशल तरीके से operation को perform कर सकते हैं।

A) Projection: - इसके मदद से किसी particular attribute को चुन सकते हैं, अर्थात् पूरे Attribute को चुनने की जरुरत नहीं होती है। इसमें record vertical show होते हैं।

Natation: - ΠA1, A2, An (r)

Where A1 and A2 are attribute names of relation r.

Duplicate rows are automatically eliminated, as relation is a set.

For example:

Πsubject, author (Books)

Selects and projects columns named as subject and author from the relation Books.

Select name from A

Name
Abhi
Prakash
Akash

Selection: - इसके मदद से किसी भी relation से conditioin base में single or multiple record को show कर सकते हैं।

Notation: $\sigma p(r)$

Σsubject="database" and price="450"(Book)

Output: Selects tuples from books where subject is 'database' and 'price' is 450.

Or

Example: select * from A where name=”abhi”;

Name

Abhi

Join: - किन्हीं दो relation को आपस में combine करके new relation creates कर सकते है with the help of join operation.

Type of join operation: -

1) Natural Join: - इसमें दो table (relation) के सिर्फ matched pair को ही new table में उपयोग करते हैं।

Employee

EMP_ID	EMP_NAME	H. No
101	ABHI	H1
102	PRAKASH	H2
103	ANUJ	H3
104	SHADAT	H4
105	MINTU	H5
106	AKASH	H6

Department

EMP_ID	Dept_Name	Salary
101	IT	18000
102	HR	20000
103	SALE	16000
104	IT	18000
105	ACCOUNT	13000
106	LECTURER	15000

EMPLOYEE |×| DEPARTMENT

EMP_ID	EMP_NAME	H. No	Dept_Name	Salary
101	ABHI	H1	IT	18000
102	PRAKASH	H2	HR	20000
103	ANUJ	H3	SALE	16000
104	SHADAT	H4	IT	18000
105	MINTU	H5	ACCOUNT	13000

2) EQUI JOIN: - इसमें एक realtion के tuple दूसरे relation के tuple से common attribute के मान को join करते हैं और new table में रखते हैं।

Example: -

EMPLOYEE |×| DEPARTMENT

EMP_ID	EMP_NAME	H. No	Dept_Name	Salary
101	ABHI	H1	IT	18000
102	PRAKASH	H2	HR	20000
103	ANUJ	H3	SALE	16000
104	SHADAT	H4	IT	18000
105	MINTU	H5	ACCOUNT	13000

3) Outer Join: - इन्य सभी join में दोनलो relation में सिर्फ वे ही attribute select किए जाते हैं जिनमें joining attribute की value common हो, जिससे data का loss हो जाता है। Data loss को बचाने के लिए outer join का उपयोग किया जाता है। इसमें joining operation में शामिल हो रहे दोनो relation में से किसी एक या दोनो ही relation के पूरे tuple का उपयोग किया जा सकता है। इसमें जहाँ जानकारी नहीं होती है उसके जगह पर NULL उपयोग करते हैं।

Type of outer join: -

1) Left Outer Join: R1 |×| R2

यह भी natural join की तरह होता है, लेकिन इसमें सिर्फ left side के tuple के record ही शामिल होते हैं, चाहे common हो या न हो, अगर नहीं होते है तो उनके जगह पर NULL उपयोग करते हैं।

Employee

EMP_ID	NAME	H. NO
101	ABHI	H1
102	PRAKASH	H2
103	ANUJ	H3
104	SHADAT	H4
105	MINTU	H5
106	AKASH	H6

Department

EMP_ID	Dept_Name	Salary
101	IT	18000
102	HR	20000
103	SALE	16000
104	IT	18000
105	ACCOUNT	13000
106	LACTURER	15000

R1] ×| R2

EMP_ID	EMP_NAME	H. No	Dept_Name	Salary
101	ABHI	H1	IT	18000
102	PRAKASH	H2	HR	20000
103	ANUJ	H3	SALE	16000
104	SHADAT	H4	IT	18000
105	MINTU	H5	ACCOUNT	13000
106	AKASH	H6	LECTURER	15000

2) <u>Right Outer Join</u>: R1 |× [R2

यह left outer join का Just Opposite होता है। यहाँ पर right side के tuple को consider किया जाता है।

EMP_ID	EMP_NAME	H. No	Dept_Name	Salary
101	ABHI	H1	IT	18000
102	PRAKASH	H2	HR	20000
103	ANUJ	H3	SALE	16000
104	SHADAT	H4	IT	18000
105	MINTU	H5	ACCOUNT	13000
106	AKASH	H6	LECTURER	15000

3) Full Outer Join: इसमे दोनो join को combine कर दिया जाता है, left and right तथा जहाँ पर value match नही होते उसके जगह पर NULL used किया जाता है।

EMP_ID	EMP_NAME	H. No	Dept_Name	Salary
101	ABHI	H1	IT	18000
102	PRAKASH	H2	HR	20000
103	ANUJ	H3	SALE	16000
104	SHADAT	H4	IT	18000
105	MINTU	H5	ACCOUNT	13000
106	AKASH	H6	NULL	NULL
107	NULL	NULL	ACCOUNT	13000

Division: R1/R2 – इसमें division operation को किन्ही भी दो table के बीच perform करते है, तो result में प्राप्त new relation में R1के वे मान शामिल किए जाते हैं जो R2 के सभी मान से relation रखते हैं।

Let R1 is a relation and R2 is also a relation

A	B
A1	B1
A1	B2

B
B1

A2	B1
A3	B1
A5	B1
A4	B2
A2	B2

Than R3 = R1/R2

A
A1
A2
A3
A5

यहाँ पर R3 में हमें {A1, A2, A3, A3} मिल रहा है क्योंकि R1 केवल {A1, A2, A3, A5} ही R2 के {B1} से relationship रखता है।

***Relational Calculation*:** यह भी relational algebra के समान query लिखने के लिए उपयोग किया जाता है। दोनो के बीच अंतर यह है कि relational algebra procedure based होता है, जबकि यह procdure based नहीं होता है। यह calculation को निश्चित रुप से identify नहीं करता है।

There are two types:

1.Tuple relational calculaus: - इसमें tuple को input के रुप में लिया जाता है। TRC में बनाई जाने वाली query के लिए जाने वाले tuple variable का नाम English के lower case में लिया जाता है तथा query में variable के नाम के बाद लिया जाने वाला "I" such that पढ़ा जाता है।

TRC में निम्नलिखित Operator का उपयोग किया जाता है–

>Greater then, <less then,>=greater then equal, <= less then equal, = equal, V for all etc.

Department

EMP_ID	Dept_Name	Salary
101	IT	18000
102	HR	20000
103	SALE	16000
104	IT	18000
105	ACCOUNT	13000
106	LACTURER	15000

Department

EMP_ID	Dept_Name	Salary
101	IT	18000
102	HR	20000
103	SALE	16000
104	IT	18000

$\{t/t \in department^{t}.emp_{id} < 105\}$

Domain Relational Calculaun: - यह domain पर आधारित होता है, इसमें domain को consider किया जाता है।

{<a1, a2, a3>|<a1, a2, a3> ∈ departemnt , a < 103}

यहाँ पर a1, a2, a3 variable department table की range है। EMP_ID, DEPT_NAME, SALARY variable को show कर रहे हैं।

Result:

Department

EMP_ID	Dept_Name	Salary
101	IT	18000
102	HR	20000

SQL: (*Structur query languagge*): -

SQL एक programming language है जिसका उपयोग Relational Databases में किया जाता है। इसको relational algebra and tuple relational calculus के लिए designed किया गया है।

SQLदोनो data definition and data manipulation languages को comprises करता है। SQL के data definition properties का उपयोग करके कि data को defined, designee and DML के द्वारा manipulate किया जा सकता है। जैसे उसे modify करना हो इत्यादि।

Database languages: - किसी system में database को create and maintain करने के लिए database languages का प्रयोग किया जाता है। Database में निम्नलिखित languages का प्रयोग किया जाता है।

DDL: DDL का पूरा नाम data definition language है जो कि conceptual schema को define करने के लिए use किया जाता है तथा यह इस बात की जानकारी भी देता है कि physical devices में इस schema को कैसे implement किया जाता है। SQL में जो सबसे महत्वपूर्ण DDL statements है, वो निम्न हैं:

1. **CREATE**: - Database में objects को create करने के लिये।
2. **ALTER:** - Database के structure में करने के लिए।
3. **DROP**: - Database में से objects को delete करने के लिए।
4. **COMMENT**: - Data dictionary में comments को add करने के लिए।
5. **RENAME**: - Object का rename करने के लिए।

DML: DML का पूरा नाम data manipulation language है और वह language जो database में data manipulate करने के काम आती है| वह language DML (data manipulation language) कहलाती है।
इसके कुछ उदाहरण निम्नलिखित है:-

1. **SELECT:** - एक Database में से Data को retrieve करना।
2. **INSERT:** - Table में Data को insert करना।
3. **UPDATE:** - Table में मौजूदा Data को update करना।
4. **DELETE:** - Table में से सभी records को delete करना।
5. **CALL:** - Java subprogram को call करने के लिए।
6. **LOCK TABLE**: - Concurrency को Control करने के लिए।

DCL: DCLका पूरा नाम data control language है। Database में stored data के access को control करने के लिए DCL का प्रयोग किया जाता है।

इसके कुछ उदाहरण निम्नलिखित हैं–

1. **GRANT:** - Database के लिए user's को privilege प्रदान करने के लिए।

2. **REVOKE:** - GRANT command द्वारा दिए गए विशेषाधिकार को वापस लेने के लिए REVOKE commond का प्रयोग किया जाता है।

Some SQL Commands:

CREATE DATABASE: -

इसके मदद से database को create किया जा सकता है।

1) CREATE DATABASE employee;

Then activate database:

2) Use <database_file_name>

Use employee;

3) Show database strucutre :

Sql> show databases;

4) Create Table:

```
CREATE TABLE Persons
(
PersonID int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255)
);
```

5) Describe structure of table :

Sql> desc <table_name>;

Sql> desc Persons;

6) Insert values inside table:

Syntex: Insert into <table_name> values (value);

Sql> insert into Persons values('101',"pathak","abhi","hn90","katni");

7) Display Records:

Select *from persons;

8) update records:

```
UPDATE Persons
SET lastname='pathak', City='umaria'
WHERE FirstName='abhilash';
```

9) Delete :

```
DELETE FROM persons
WHERE lastname='pathak';
```

Delete All Data

इस command की help से किसी एक table के सभी rows को बिना deleting table के delete किया जा सकता है। This means that the table structure, attributes, and indexes में कोई change नहीं होता है।

DELETE FROM table_name;

or

DELETE * FROM table_name**;**

10) BETWEEN Operator Example

इसके मदद से किसी attribute में present records के बीच Range का मान निकाल सकते हैं।

The following SQL statement selects all products with a price BETWEEN 10 and 20:

Example: -

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

11) SQL Aliases: SQL aliases का प्रयोग करके temporarily किसी एक table or एक column की heading को rename कर सकते हैं।

```
SELECT CustomerName AS Customer, ContactName AS [Contact Person]
FROM Customers;
```

12) The SQL ORDER BY Keyword

The ORDER BY keyword is used to sort the result-set by one or more columns.
The ORDER BY keyword sorts the records in ascending order by default. To sort the records in a descending order, you can use the DESC keyword.

```
SELECT * FROM Customers
ORDER BY Country;
```

13) The SQL WHERE Clause

The WHERE clause का प्रयोग table ये records को filter करने के लिए किया जाता है।

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

14) SQL SELECT DISTINCT Statement

यह किसी एक records को केवल एक बार ही दर्शाता है।

```
SELECT DISTINCT City FROM Customers;
```

15) ALTER TABLE Statement

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

```
1>ALTER TABLE table_name
ADD column_name datatype
2> ALTER TABLE table_name
DROP COLUMN column_name
3> ALTER TABLE table_name
MODIFY COLUMN column_name  data type
```

16) SQL DROP INDEX, DROP TABLE, and DROP DATABASE

DROP statement के मदद से Indexes, tables, and databases को easily deleted/removed किया जा सकता है।

DROP INDEX Statement: -

The DROP INDEX statement is used to delete an index in a table.
example:ALTER TABLE table_name DROP INDEX index_name
The DROP TABLE Statement
The DROP TABLE statement is used to delete a table.
example:DROP TABLE table_name

The DROP DATABASE Statement: The DROP DATABASE statement is used to delete a database.
example: DROP DATABASE database_name

The TRUNCATE TABLE Statement
What if we only want to delete the data inside the table, and not the table itself?
Then, use the TRUNCATE TABLE statement:
TRUNCATE TABLE table_name

***SQL Functions*: -**

SQL में कई built-in functions होते हैं, जिनके मदद से data calculation को perform किया जाता है। SQL aggregate functions का collection होता है जो एक single value को retrun करते है, जो कि Column के अंदर की value को calculate करता है।

UCASE & LCASE ( ): - String or character को Upper/Lower case में change करने के लिए।

Example: select UCASE ("Abhilash") from table name;

Select LCASE ("abhilash") from table name;

1) Length ( ): - इस function के मदद से किसी character की length find कर सकते हैं।
Select count ("abhilash") from tabke_name;

2) Ltrim ( ): - यह function string के left side से space remove करता है।
Select ltrim ("abhilash") from table_name;

3) Rtrim ( ): - यह function string के right side से space removes करता है।
Select rtrim ("abhilash") from table_name;

4) Substr ( ): - यह function दिए गए किसी string में से substring को display करता है।

5) Abs ( ): - इस function को absolute function कहते हैं, जिसके मदद से absoulute value find कर सकते हैं।
Select abs (12) from EMP;

6) इस function से किसी भी number का power apply करके o/p display करता है।
Select power (number, power);
Select power (5, 2);

7) Mod ( ): इसके मदद से एक number दूसरे number को divide करता है और remainder को वापस करता है।
Select mod (13, 12);

8) Sqrt ( ): - इसके मदद से square root find कर सकते हैं।
Select sqrt (144);

9) Sign ( ): - Signed function यह show करता है कि वह positive है या कि negative.
Select sign (12);

10) Sysdate ( ): - show the current date.
Select sysdate ( );

11) Sun ( ): - किसी table के column को sum करता है।
Select sum (fields_name) from table_name;
Select sum (id) from EMP;

12) Max, Min & Avg ( ): -: Attribute के मान से maximum, minimum and average value find कर सकते हैं।
Example: select max (id) from EMP;
Select min (id) from EMP;
Select avg (id) from EMP;

13) Desc ( ): - किसी table/records को display करने के लिए इस function का प्रयोग करते हैं।
Sql> Desc tablename;

14) Order by: - Fields को order by display करने के लिए प्रयोग किया जाता है।
Sql> select *from EMP order by name;

What is indexing and hasing explain with example:

- **Index:** किसी भी table में record proper arrange order में होता है। वह assending या desecending order में set किया जाता है। यह कार्य memory में होता है। Index create करने से CPU की performance बढ़ जाती है।
  Indexing करने से database table/file की एक indexing create हो जाती है, जिससे CPU को data को find/search करने में ज्यादा time consume नहीं होता है। Indexing की मदद से data आसानी से मिल जाता है। Indexing में दो field's होते है– एक index_No और दूसरा जिसमें hardware का address store होता है।
  किसी भी table का index create होने के बाद index fike primary file बन जाती है और original file secondary पिसम बन जाती है।

Index file/Primary file की मदद से secondary file से data/information को fetch/find/search किया जाता है।

Syntex: Create index <index_name> on <table_name> (field's name);
Example: create index emp1 on EMP (id);
sql> show index from emp;
Indexing को indexing attribute के base पर defined किया गया है।

Indexing can be of the following types: -

- **Primary Index**: Primary index को ordered data file के रुप में एक key field के अनुसार defined किया गया है ओर key field को generally primary key कहते हैं।
- **Secondary Index**: Secondary index एक candidate key की तरह होते है और प्रत्येक record के अंदर unique value होती है और एक non-key साथ में duplicate values होते हैं।
- **Clustering Index**: Clustering index को ordered data file की तरह defined किया गया है। ये data file के ordered एक non-key field की तरह होते है।

Ordered Indexing is of two types:

- Dense Index
- Sparse Index

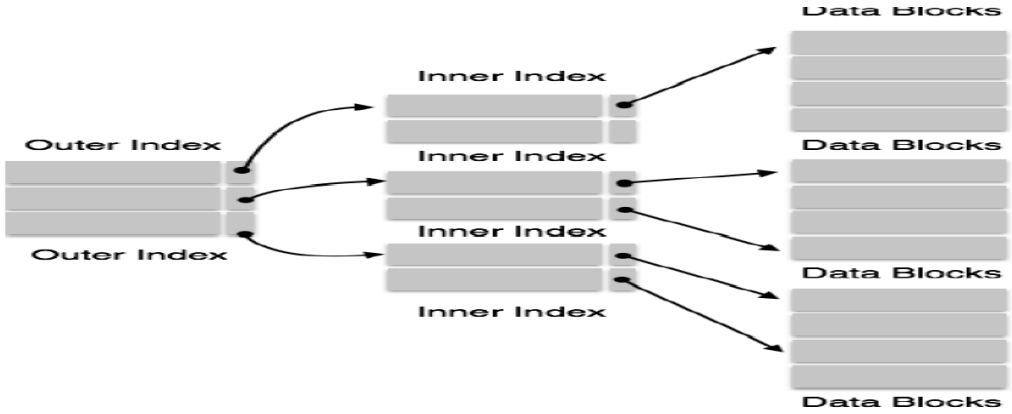
Dense Index: **-** Dense index में सभी database record के अंदर एक search key value create होती है जिससे searching fast हो जाती है पर index format में record को store करने के लिए अधिक space की requirement होती है। Index, search key value record contain करता है और pointer जो कि disk में auctual record को point's करता है।

China	→	China	Beijing	3,705,386
Canada	→	Canada	Ottawa	3,855,081
Russia	→	Russia	Moscow	6,592,735
USA	→	USA	Washington	3,718,691

[Image: Dense Index]

Sparse Index: **-** Sparse index में प्रत्येक search key के लिए index records को create नहीं किया जाता है। Index record इसमें एक search key contains करती है और एक actual pointer जो कि disk में present data को point करता है जिससे record को search किया जाता है। सबसे पहले हम index record के द्वारा data को actual location के पास पहुँचाते है। यदि दिए गए index के अनुसार हम data को सीधे जहाँ पहुँचाना चाहते हैं वह वहाँ पर दिखाई नहीं देता है तब वह system को start करते हैं और तब तक करता है जब तक कि वह इच्छानुसार data को found नहीं कर लेता है।

Multilevel Index**: -** Index record, search key value and data pointer's का प्रयोग करता है जबकि multi level index, actual database file को disk में store करता है, जिससे database का आकार बढ़ जाता है अतः index record विशाल हो जाता है। इससे main memory के अंदर search operation करने में अधिक समय waste होता है तथा speed धीरे हो जाती है। यदि single level index का प्रयोग किया जाता है तो बड़े आकार के indexing को manage करना संभव नहीं हो पाता जिसके लिए multiple disk accesses का प्रयोग किया जाता है।

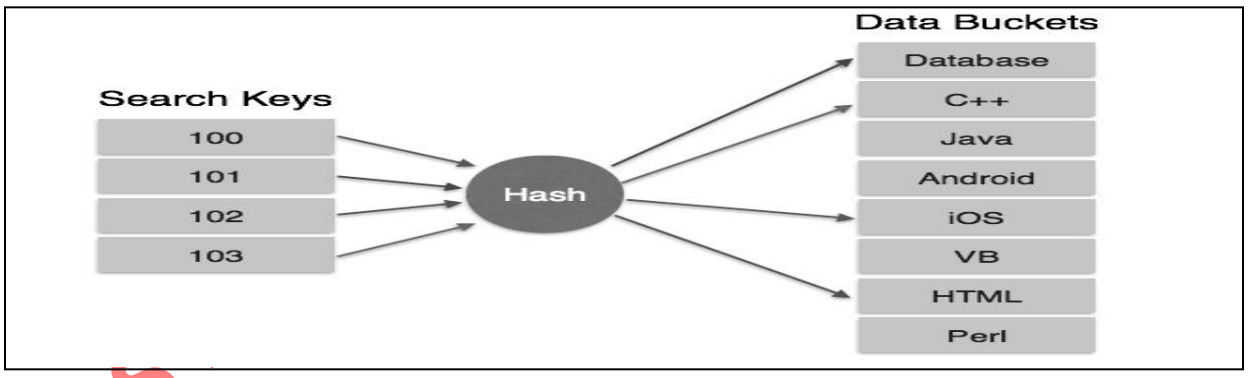


[*Image: Multi-level Index*]

Multi level index की मदद से index को कई छोटे-2 भागों में तोडा जाता है जिसमें memory में अलग-2 जगह के एक single block में data बिखरा हुआ होता है जिससे data searching के लिए outer index, inner index से matched करता है। फिर inner index data block से data को memory के किसी भी location से search करता है।

Hasing: - Squential file organization में किसी data को locate करने के लिए मे पूरी file को access करना होता है जिसमें input-output operation में काफी ज्यादा समय लगता है। इस disadvantage को देखते हुए किसी file की index करने के लिए **Hasing process** का प्रयोग किया जाता है। Hasing process दो प्रकार की होती है–

1) Static Hasing: - Hash file organization में जिस block में record present होता है, पहले उस block को search किया जाता है। इस block को **bucket** कहा जाता है। इन bucket में एक या एक से अधिक record present रहते हैं और इन bucket का आकार अलग-अलग होता है। Search operation में किसी भी record को इन bucket में ही static hashing की help से search किया जाता है। Index file को create करने के लिए hash function का प्रयोग किया जाता है।

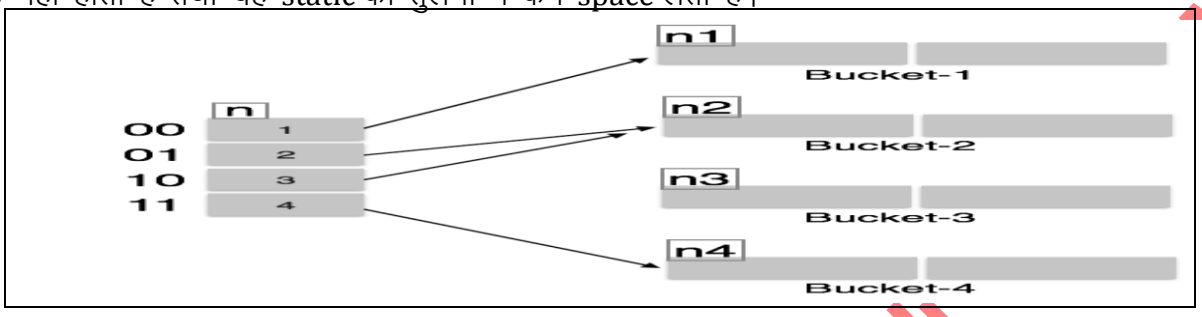


इस function की मदद से record bucket के रुप में विभाजित होकर storage device में store हो जाता है और इन bucket की information pointer के रुप में store रहती है। Bucket में store records की information दो तरह की होती है –

A) Distribution in Uniform: - इसमें प्रत्येक record को number allot किए जाते हैं जिसकी value field से मिलती है। अगर कोई record bucket में present है तो कई number allot किए जाते हैं। से record number के अनुसार व्यवस्थित होते हैं।

B) Distribution in Random: - इसमें record के number को Random order में ser किए जाते है। किसी भी record को insert किया जाता है तो situation के अनुसार किसी bucket में space न होने पर जो उस मान को store कर सके तब उसे condition को bucket overflow कहा जाता है।

2) <u>Dynamic hashing</u>: - यह static hashing का advantage होता है जिसमें bucket की संख्या fixed होती है जिससे bucket overflow की समस्या का सामना करना पड़ता है, जबकि dynamic में size fixed नहीं होती है। अगर data insert हुआ तो new bucket create हो जाती है तथा अगर data delete हुआ तो bucket भी delete हो जाती है जिससे data/bucket overflow की problem create नहीं होती है। इसमें file के size increased होने पर उसकी performance में गिरावट नहीं होती है तथा यह static की तुलना में कम space लेता है।



Que: - What are the charecteristics of Ststic and Dynamic Hashing? What are the advantages and disadvantages of Hashing?

Ans: -

<u>Features of Static Hashing</u>: -

1. यह Hashing structure किसी record को search करते समय data file के सभी record पर जाने पर जोर नहीं डालती।

2. Static Hashing system Address B के सभी bucket के set के लिए search "Key" value K की खोज करती है।

3. यह system search "Key" value पर एक function के माध्यम से गणना करके data item के address का पता लगाती है।

<u>Features of Dynamic Hashing</u>: -

1. यह Hashing system file के आकार के अनुसार searching का कार्य करती है।

2. जैसे-जैसे file का आकार बढ़ता जाता है यह system स्वयं को पुनः व्यवस्थित करती जाती है।

<u>Advantages of Hashing</u>: -

1. File का आकार बढ़ने पर hashing process slow नहीं होती। File का आकार बढ़ने के साथ-2 hashing का प्रदर्शन भी विस्तृत होता जाता है।

2. भविष्य के उपयोग के लिए bucket reserve करके रखने की आवश्यकता नहीं होती।

<u>Disadvantages of Hashing</u>: -

1. यह एक जटिल प्रणाली है।

2. Bucket Address table में indirection में level बहुत अधिक होते हैं।

Que: - What is the difference between Static and Dyanmic Hash function?

Ans: - Static and Dynamic Hash Function में नि0 लि0 अन्तर है –

1. Static Hash system एक सांकेतिक hash system है, जबकि Dyanmic hash system एक परम्परागत hash system है।

2. Static hash function केवल एक input parameter लेता है जो एक निश्चित लम्बाई का string होता है। जबकि Dynamic hash function दो input parameter लेता है जहाँ पहला parameter एक string होता है तथा दूसरा parameter एक सुरक्षा पैरामीटर है जो सुरक्षा आवश्यकताओं को स्पष्ट करता है।

3. Static hash function अतिरिक्त सुरक्षा सुविधाएं उपलब्ध करवाता है। Dynamic hash function अतिरिक्त सुरक्षा उपलब्ध नहीं कराता। इसमें केवल एक सुरक्षा पैरामीटर दिया जाता है जिसके अनुसार यह मांगी गई सुरक्षा सुविधा उपलब्ध कराता है।

Que: - Explain the B-tree and describe charecteristics of B-tree.

Ans: - <u>B-Tree</u>: -

B-Tree की मूल संरचना सन् 1970 में R. Bayer तथा E. McCreight द्वारा develop की गई। जिसने धीरे-2 स्वयं को Index structure तैयार करने की सबसे लोकप्रिय तकनीक के रुप में स्थापित किया। B-Tree मुख्य रुप से एक संतुलित balanced short tree के रुप में जानी जाती है, जो outer shorting के लिए बहुत उपयोगी है। Database systems में भी B-Tree का शसक्त उपयोग है। Data base system में file के index का संरचना तैयार करने के लिए B-Tree का ही उपयोग किया जाता है। B-tree को database systems की index structure तैयार करने के लिए एक standerd technique के रुप में मान्यता प्राप्त है।

प्रत्येक data file, records का collection होता है, जिसमें index एक unique "Key" को दर्शाता है जिससे record की पहचान सम्भव हो पाती है। इस संरचना को B-Tree के माध्यम से लागू करने की तकनीक IBM की virtual storage access method में present की गई है। जिसे short form में VSAM कहा जाता है। इसे हम निम्न उदाहरण से समझ सकते हैं–

सामान्यतः data में परिवर्तन करने सम्बंधी कार्यों के लिए data को main memory में store करना आवश्यक होता है, लेकिन ऐसी applications में जहाँ database का आकार बहुत बड़ा हो और इसकी तुलना में main memory कम हो, सम्पूर्ण data main memory में नहीं रखा जा सकता तो इस condition में data को hard disk में ही रखना पड़ता है और उसे खण्ड़ों में main memory में लाया जाता है अर्थात् एक बार में कुछ data record main memory में लाए जाते हैं और processing के समय hard disk में स्थित अन्य भाग की आवश्यकता पड़ने पर पहले वाला भाग main memory से हटाकर उसके स्थान पर अन्य भाग load कर दिया जाता है। यह प्रकिया **swaping** कहलाती है। इस प्रक्रिया को सम्पन्न कराने के लिए file की record structure में main memory pointer के साथ-2 hard disk memory address pointer भी रखना आवश्यक होता है ताकि आवश्यकतानुसार main memory तथा hard disk space, दोनो को access किया जा सके। इसे हम निम्न record संरचना के द्वारा समझ सकते हैं–

```
        Struct   Employee
{
    ent Eid;
    Char E_name [90];
    Char E-address [90];
    Char Dept [70];

    Struct Employee * left;
    Struct Employee * right;

    D_block d_left;
    D_block d_right;
    }
```

जहां Employee, Structure में pointers के दो set लिए गए हैं जिनमें से *left and *right main memory pointer है तथा d_left and d_right hard disk pointer है। जब किसी node के children main memory में ही store हो तो उन्हें

access करने के लिए *left and *right pointer का उपयोग किया जाता है और अगर node children disk में हो तो d_left and d_right pointer का उपयोग किया जाता है। जहां left node के left child तथा right node के right child को show कर रहा है।

यहां data block हस्तान्तरण के द्वारा main memory में भेजा जाता है, अर्थात् एक बार में data का एक block main memory में आता है। चूंकि disk की data पढ़ने की गति कम होती है इसलिए यह प्रयास किया जाता है कि disk access की आवश्यकता कम से कम पड़े।

उपरोक्त उदाहरण में वर्णित संपूर्ण संरचना B-Tree पर ही आधारित है, जिसमें प्रत्येक 'Key-Board' के दो children node हैं अर्थात् left pointer B-Tree के left child को show करता है तथा right pointer B-Tree के right child को show करता है।

हम ऐसी Tree बना सकते हैं जिसके node के एक से अधिक children हों तथा एक से अधिक 'Key' हों। इस प्रकार की Tree की एक property order होती है जो यह दर्शाती है कि Tree में maximum कितने node हो सकते हैं। अगर Tree में node के maximum children की क्षमता N है तो इसका अर्थ होगा कि Tree में Tree का order N है। Disk access को कम से कम रखने के लिए यह आवश्यक है कि Tree की height कम से कम हो, सभी leave node एक ही स्तर पर हों तथा leave को छोड़कर सभी nodes में कम से कम अधिकतम् क्षमता से आधे node (N/2) उपलब्ध हों। अगर Tree में एक ही node हो तो इस स्थिति में Root को कोई child नहीं होता अन्यथा Root में कम से कम दो या अधिक से अधिक N children होते ही हैं। इस प्रकार की Tree जिसमें ये विशेषताएं हों **balanced short Tree** कहलाती हैं।

Features of B + Tree: -

1. प्रत्येक node में maximum N या minimum N/2 children हों।
2. Root का या तो कोई child नहीं हो या फिर कम से कम दो या अधिक से अधिक N children हों।
3. प्रत्येक board में maximum children होने की स्थिति में अधिकतम से एक कम अर्थात् N-1 'Key' हो।
4. सभी 'Key' निर्धारित क्रम में जमी हों। प्रत्येक 'Key' की left sub-Tree की सभी 'Key' उससे छोटी हों तथा right sub-Tree की सभी 'key' उससे बड़ी हों।
5. जब एक नई 'Key' पहले से पूरे भरे हुए node में insert की जाए तो यह node दो भागों में टूट जाए तथा बीच का मान parent node में आ जाए।

Que: - What is a view? How to create view? Explain.

View: - View, एक table की तरह होता है, जिसके contents दूसरी tables से लिखे जाते हैं। View tables, data को नहीं रखते हैं। ये एक window की तरह कार्य करते हैं, जिसके माध्यम से एक table के contents को display किया जा सकता है।

जैसे यदि हम चाहते हैं कि हमारे office के क्लर्क को दूसरे Employee की salary के बारे में पता नहीं चलना चाहिए, लेकिन उसे Emp_Name, Post तथा dept_Name के बारे में information की आवश्यकता है, तो इस case में हम employee table का एक view create कर सकते हैं जिसमें क्लर्क की आवश्यकतानुसार information को store कर सकते हैं। इस view में केवल आवश्यक information को रखा जाता है। View को create command के द्वारा create किया जा सकता है। इसका Syntax नि0 लि0 रुप में होता है–

Create View < view name > as< select statement >;

जैसे – यदि हम क्लर्क (clerk) के अनुसार information को display करना चाहते हैं तो इसके लिए view को create करने के लिए निम्नलिखित command का प्रयोग करते हैं–

Create view clerk as Emp_name, dept_name from EMP, dept
Where Emp.dept_code=dept.dept_code;

इस बार view create करने के पश्चात् इसे दूसरी Table की तरह ही treat कर सकते हैं। इस view को create करने के पश्चात् इसकी information को देखने के लिए create command को उसी तरह प्रयोग करते हैं जैसे Table की information को देखने के लिए command को प्रयोग करते हैं।

```
Select*from clerk;
```

QUERY PROCESSING: -

Cost Components in Query Execution (क्वेरी के पालन में लागत के अवयव): -

किसी query को पालित (perform) करने में आने वाली लागत में निम्नलिखित अवयव होते हैं–

1. Access Cost to Secondary Storage: यह लागत द्वितीयक भंडारण माध्यम, जैसे– disk से data blocks को खोजने, पढ़ने और लिखने में आती है। खोजने में आने वाली लागत की access structure के प्रकार, जैसे– Ordering, Hashing and Primary या secondary index, पर निर्भर करती है। इसके अलावा file disk पर एक ही जगह है या अलग-2 भागों में बिखरी हुई है इस बात का भी प्रभाव access cost पर पड़ता है।

2. Storage Cost: यह किसी query को perform करने की रणनीति द्वारा उत्पन्न की जाने वाली माध्यमिक फाइलों (intermediate file) को store करने में आने वाली लागत होती है।

3. Computation Cost: यह query के performance के समय memory में data buffers पर की जाने वाली क्रियाओं में आने वाली लागत है। इन क्रियाओं में records की searching, sorting, join के लिए records को merge करना और field value पर गणनाएँ करना शामिल है।

4. Memory Usages Cost: यह query पालन के समय memory buffers की आवश्यक संख्या लेने में आने वाली लागत होती है।

5. Communication cost: यह query को उस terminal से, जहां वह उत्पन्न हुई है, database site पर भेजने और वहां से result प्राप्त करने में आने वाली लागत होती है।

बडे databases के लिए मुख्य जोर द्वितायक भंडारण के access की लागत को न्यूनतम करने पर होता है। साधारण लागत function अन्य factors को छोड देते हैं और विभिन्न query पालन रणनीतियों की तुलना disk and memory के बीच होने वाले block transfer की संख्या के आधार पर करते है। छोटे databases के लिए जहाँ files का अधिकांश data पूरी तरह memory में store किया जा सकता है, मुख्य जोर गणना लागत को न्यूनतम करने पर दिया जाता है। Distributed databases के communication cost का भी न्यूनतम किया जाता चाहिए।

Query Optimization: **-** Query optimization बहुत से relational database management system का function है। Query optimizer possible Query plans में सबसे efficient तरीके से दी गई query को exeute करने का प्रयास करता है।

सामान्यतः Query optimizer, users के द्वारा सीधे प्रयोग में नहीं लाया जाता है। एक बार Query को जब database server में submit किया जाता है तो इसे parsec (दिव्यकला) में parse किया जाता है। जहां Optimization perform होता है। यद्यपि कुछ database engines, query optimizer को hints के साथ guide करना allow करते हैं।

एक query, database की information के लिए request होती है। ये किसी व्यक्ति के address जैसे- SS# 123-45-6789 को खोजने के लिए हो सकती है या दिल्ली में रहने वाले 20 से 29 वर्ष के उन मनुष्यों को खोजने के लिए भी हो सकती है जिनकी वार्षिक आय 2 लाख रुपयों से ऊपर हो जो कि Queries, related data को access करके

final information को दिखाती है। यद्यपि database structure जटिल होते है और ये different data structure से different orders में database से information को access करती है। ये Query की processing कई तरीकों से कर सकती है। प्रत्येक तरीका अलग-2 processing time लेता है। किसी Query के अलग-2 processing time query को execute करने में काफी कम या ज्यादा समय ले सकते हैं। जैसे कोई तरीका कुछ seconds में result दिखा दे और कोई तरीका उसी कार्य में कई minute लगा दे। Query optimization जो कि एकautomated process है, ऐसा Query को execute करने का तरीका ढ़ूंढ़ता है जोकि सबसे कम ले। चूँकि ये एक जटिल प्रक्रिया है और साथ ही इसमें स्वयं अच्छा खासा समय लग जाता है। अतः Query optimization कई बार खर्चीला और असम्भव भी साबित होती है। अतः Query optimization के द्वारा सबसे अच्छे तरीके को ढूंढ़ने के बजाय एक सामान्य अच्छा तरीका ढूंढ़ने पर अधिक जोर दिया जाता है। यह तरीका सामान्यतः सबसे अच्छे तरीके के काफी करीब होती है।

General Considerations and Implementation – समय के दृष्टिकोण best query plan को sort करने और अपनी choise से query plan को sort करने में काफी अन्तर होता है। Optimizer स्वयं यह निश्चित नहीं कर सकता कि वह स्वयं best query ढूंढ़े या user को select करने की choise दे। Cost based Query optimizer, used resource के base पर उस Query execution के तरीके को छांटता है जिसकी estimated सबसे कम है। Optimization के दो प्रकार होते हैं– पहले logical optimization के द्वारा relational algebra के sequence को query को हल करने के लिए generate किया जाता है। इसके अतिरिक्त physical optimization भी होता है जिसमें प्रत्येक operation को perform करने के माध्यम को निश्चित किया जाता है।

Query Estimation - भिन्न-2 execution plans, भिन्न-2 cost optimization estimation पर same result को produce करते हैं। इसके लिए नि0 लि0 points को निर्धारित किया गया है–

(i) इसे low cost query execution plan के लिए अवश्य search करना चाहिए।
(ii) Statistics
(iii) Candidates
(iv) Histograms
(v) Input output Vs Computation Costs
(vi) Time to first rupee Vs Completion time

बहुत सारे database operations के लिए tuple Vs previous tuple या tuple Vs tuple another table को read करने की आवश्यकता होती है। इसके लिए सामान्यतः नि0 लि0 techniques का use किया जाता है–

(i) Iteration
(ii) Buffered I/O
(iii) Index
(iv) Sort
(v) Hash

➢ **OLTP (Online Transcation Processing): -**

OLTP को Online transaction processing के नाम से जाना जाता है। OLTP information system की एक class है जो transcation oriented application को manage करती है। इसके माध्यम से data की entry तथा आवश्यकतानुसार data का Online retrieve करके transcation processing के कार्य को सम्पन्न किया जाता है। OLTP का प्रयोग processing को refer करने के लिए प्रयोग किया जाता है, जिसमें system, user की requests को शीघ्रता से response करता है। एक BANK के लिए ATM business transcation processing application का example है। OLTP के simplicity तथा efficiency दो main benefit है। SAPERP business application, OLTP के प्रमुख उदाहरण के रुप में organization में use किया जाता है। इस application में SAPDB database के रुप में, SQL server database application के रुप में database server में उपस्थित रहता है। इसे database tier के नाम से जाना जाता है। इससे application server connect होते हैं जिन्हें application tier के नाम से

जाना जाता है। Client user, client application के माध्यम से application server का use करके SAPDB database में से अपनी आवश्यकतानुसार Online transcation को perform करता है। Client application, GUI, Web GUI, HTML/SOAP, Web services के रुप में होते हैं। इसे client tier के नाम से जाना जाता है।

Requirements of OLTP –

OLTP के लिए किसी ऐसे network या company की आवश्यकता होती है जो लगातार transcation processing को support उपलब्ध कराता है। इसी वजह से नए OLTP ऐसे client server processing और brockering software का use करते हैं जो transcation को network में अलग computer plateform में run कराये। Efficient OLTP बड़े application में किसी ऐसे transcation management software जैसे CICS और database optimization tactics पर depend होते हैं, जो OLTP based database में बहुत ज्यादा concurrent update की processing को facilitate को divide कर सके। OLTP brockering programs बडे decentralized database system में network के बहुत से computers में processing को विभाजित कर सकते हैं। प्रायः OLTP, services oriented architecture ओर web services में integrated होते है। OLTP input information को process and update करते हैं और processed information को collect करते हैं इसलिए OLTP को support करने के लिए किसी database management system का use किया जाता है। OLTP को किसी Client server system पर carry किया जा सकता है।

Benefit- OLTP के दो important benefit हैं– सरलता और दक्षता। इसके द्वारा कम paper का प्रयोग किया जाता है और साथ ही तेजी से revenue and expences की forecasting की जा सकती है। अतः OLTP किसी भी business को आसान बना सकता है।

Disadvantages- इसमें कभी-2 offline mainteneance की भी आवश्यकता होती है जो information technology solution की need होती है। इससे cost-benefit analysis पर प्रभाव पड़ता है या हम कह सकते हैं कि इससे reoccurring cost बढ़ जाती है।

OLTP Database-

OLTP database real time transcation को handle करते है। जिनके द्वारा specific requirements की पूर्ति होती है। जैसे हम एक departmental store चला रहें हैं तो OLTP database को यह निश्चित करना चाहिए कि inventory table, purchasing table and customer table से related transcations के according update हो रही है या नहीं। OLTP database automatic होने चाहिए। ये database constraint होना चाहिए अर्थात् प्रत्येक transcation database को सही अवस्था में छोडे और इसके साथ ही ये database isolated होना चाहिए यानि एक transcation से दूसरे transcation की स्थिति प्रभावित नहीं होनी चाहिए। OLTP database को durable होना चाहिए अर्थात् commited transcation से database की स्थिति resultant state में आनी चाहिए। OLTP database की ये शर्तें user को कुछ लम्बी लग सकती हैं लेकिन OLTP database को successfully run करने के लिए ये सब आवश्यक OLTP database को front end पर use किया जाता है जिसकी वजह से यदि number of transcation बढ़ते हैं तो database को इसको properly tackle करने के लिए मजबूत और अनुपातिक होना चाहिए। यदि हम किसी Online book store को OLTP की मदद से चला रहें हैं तो OLTP database को 15 seconds में updation को पूर्ण करता हुआ होना चाहिए।

किसी भी OLTP database में row level locking काफी important होती है जिसमें किसी भी record को दूसरे transcation के लिए तब तक lock में रखा जाता है जब तक कि उसका current transcation complete न हो जाए। OLTP में concurrent programming जैसी problems आने की काफी आशंका रहती है। जब कोई user एक साथ किसी record को process करने की कोशिश करते हैं तो प्रत्येक user द्वारा जल्दी-2 record को process करने की कोशिश की जाती है ताकि दूसरे users के लिए waiting time अधिक न जाए। OLTP से related सामान्य समस्याओं का समाधान oracle 10g के documention में उपलब्ध कराई गई है।

OLTP के Exapmle-

1) Online Banking- जब bank से money withdraw करते हैं तो उसी समय हमारा account update हो जाता है और next transaction के लिए update value available हो जाती है। Money withdraw या तो ATM से की जाए या bank

से, वह transaction OLTP से ही process होता है। इसी प्रकार जैसे ही कोई amount हमारे account में check या cash से जमा होती है तो amount जमा राशि उतना बढ़कर reflect होता है।

2) POS (Post of sale)- POS किसी भी mail, market या city को refer करता है जहां किसी वस्तु की sale होती है। वस्तु एक से अधिक भी हो सकती है। POS के द्वारा किसी भी क्षण में की गई sale को Online update किया जाता है। POS के लिए Customized software एवं hardware का प्रयोग किया जाता है। POS के द्वारा sale return भी की जा सकती है। POS के अन्य कार्यों में inventory management, CRM, financial व warehousing भी perform होते हैं। POS में सभी Units independent तरीके से कार्य करती है अतः वे error prone भी होते हैं।

3) On Line Reservattion- Railway या Airlines में reservation OLTP से होते हैं। इस प्रकार यदि कोई व्यक्ति Railway के लिए reservation करा रहा है तो वह तत्क्षण यह जान सकता है कि क्या particular category के लिए seates available हैं। OLTP से reservation होने के कारण किसी भी व्यक्ति को Railway station या Airport जाने की जरुरत नहीं है और वह reservation घर बैठे ही internet के द्वारा करा सकता है। यदि कोई व्यक्ति reservation cancel कराता है तो वह भी OLTP के द्वारा perform हो जाता है। इससे जो सीटें उपलब्ध होती हैं, वह कम समय में पुनः reserve हो जाती है।

Recovery Concet:-

Transcation failure से recovery का अर्थ सामान्यतया यह होता है– database को फिर से उस असफलता के समय से ठीक पहले की सबसे नवीनतम सुसंगत अवस्था (Consistent state) में ला देना। ऐसा करने के लिए system को उन परिवर्तनों के बारे में information रखनी चाहिए जो विभिन्न transcations के द्वारा data items पर लागू किए गए थे। इन informations को सामान्यतया system log में रखा जाता है। Recovery की प्रचलित रणनीति को हम निम्न प्रकार संक्षेप में व्यक्त कर सकते हैं–

1. यदि किसी दुर्घटनावश असफलता (Catastrophic failure), जैसे– disk crash के कारण database के एक बड़े भाग को हानि पहुँची है तो recovery method में database की किसी भूतकाल की प्रति (Past copy) को उसके backup archival storage जैसे– maganatic tap से restore कर दिया जाता है और उस पर सभी committed किए हुए transcations की actions को Backup log का help से फिर से लागू किया जाता है।

2. यदि database को कोई भौतिक हानि नहीं पहुँची है, परन्तु कुछ database inconsistent (विसंगत) हो गया है तो रणनीति कुछ actions को Undo करके उन changes को reversing की होती है जिनके कारण विसंगति उत्पन्न हुई है। Database को किसी सुसंगत अवस्था की पूरी अभिलेखागार प्रति की आवश्यकता नही होती, बल्कि recovery के समय केवल system log में रखी गई Online entries की ही सहायता ली जाती है।

सैद्धांतिक रुप से हम दुर्घटनारहित असफलताओं (Non-catastrophic failures) से recovery करने की मुख्य तकनीकों को दो अलग-2 वर्गों में रखते हैं– स्थगित सुधार (Deffered update) और तत्काल सुधार (Immediate update)। स्थगित सुधार तकनीकों में database को disk पर भौतिक रुप से तब तक नहीं सुधारा जा सकता जब तक कि कोई transcation अपने commit point तक नहीं पहुँच जाता। Commit point तक पहुँचने के बाद ही transcation द्वारा किए गए सुधारों को database पर record या लागू किया जाता है। Commit point पर पहुँचने से पहले सभी transcations updates को पहले local buffers में किया जाता है। Commit के समय updetes को पहले system log में पूरी तरह लिखा जाता है और फिर database में record किया जाता है। यदि कोई transcation अपने commit point तक पहुँचने से पहले ही असफल हो जाता है, तो इसके द्वारा database में कोई परिवर्तन नहीं किया गया होता, इसलिए Undo या उलटने की कोई आवश्यकता नहीं होती है। लेकिन किसी commit किए गए transcations के effect को फिर से करने (Redo) की आवश्यकता हो सकती है, क्योंकि सम्भव है कि उसके प्रभावों को database पर record न किया गया हो। इस कारण स्थगित सुधार तकनीकों को NO-UNDFO/REDO algorithm के नाम से भी जाना जाता है।

तत्काल सुधार तकनीकों में database को किसी transcation की कुछ actions द्वारा उस transcations के Commit point पर पहुँचने से पहले ही सुधारा जा सकता है। हालांकि इन actions को database पर लागू

करने से पहले सामान्यतया disk log पर बलपूर्वक record किया जाता है। जिससे recovery करना सम्भव हो जाता है। यदि कोई transcation database पर कुछ सुधार करने के बाद किन्तु अपने commit point पर पहुँचने से पहले ही असफल हो जाता है तो database पर पड़े हुए इसके प्रभाव को समाप्त या Undo किया जाना चाहिए अर्थात् Transcation को Rollback किया जाना चाहिए। इस प्रकार तत्काल सुधार के सामान्य मामलों में Undo और Redo करने की आवश्यकता होती है। इस कारण तत्काल सुधार तकनीकों को UNDO/REDO algorithm के नाम से भी जाना जाता है। अधिकांश मामलों में हम इन्हीं तकनीकों का सहारा लेते हैं।

इस Algorithm के एक संस्करण में, जहाँ किसी transcation के commit point तक पहुँचने से पहले ही सभी सुधारों को database में record कर लिया जाता है, केवल UNDO करने की आवश्यकता होती है, इसलिए इसको UNDO/NON-REDO algorithm के नाम से जाना जाता है।

UNIT-5

Transaction Model: -

एक transcation एक group of task को define कर सकता है। कोई एक single task minimum processing unit होती है जिसे further divide किया जा सकता है। Database system में रखे गए data पर processing करने के लिए अपनाए जाने वाले विभिन्न तकनीक transcation model कहलाते हैं।

Database system में transcation systemका होना आवश्यक है जिससे data dh relibility and error free transcation का प्रयोग कराए। Transcation Model में ACID properties को follow किया जा सके।

सभी database के अंदर ऐसी facility होनी चाहिए जिससे एक ही समय में data की processing एक बार में ही हो जाए। जैसा हम Bank का एक उदाहरण लेते है जिसमें transcation होता है अर्थात् अगर transcation होगा तो वह पूरा होगा अन्यथा वह शुरु ही नही होगा।

Atomicity: - इस property states में किसी एक transaction को आवश्यक है कि वह atomic unit की तरह treat करे। अर्थात् या तो सभी operations को execut करायें अथवा कोई भी नहीं। Database के अंदर transaction के partially completed होने की जगह नहीं है (आधा अधूरा नहीं होगा)। Transaction States को defined करना आवश्यक होता है या तो before the execution of the transaction या तो after the execution/ abortion/ failure of the transaction.

Consistency: - यह property यह निर्धारित करती है कि transcation की execution पूरा होने के बाद system valid state में हो। अगर transcation पूरा हो तो उसके द्वारा किए गए सभी changing system में proper arranging order में record हो। अगर transcation के समय में कोई error आती है या किसी वजह से वह task पूरा नही हो पाता है तो उसके माध्यम से किए जा रहे transcation पूरी तरह से roll back हो जाए।

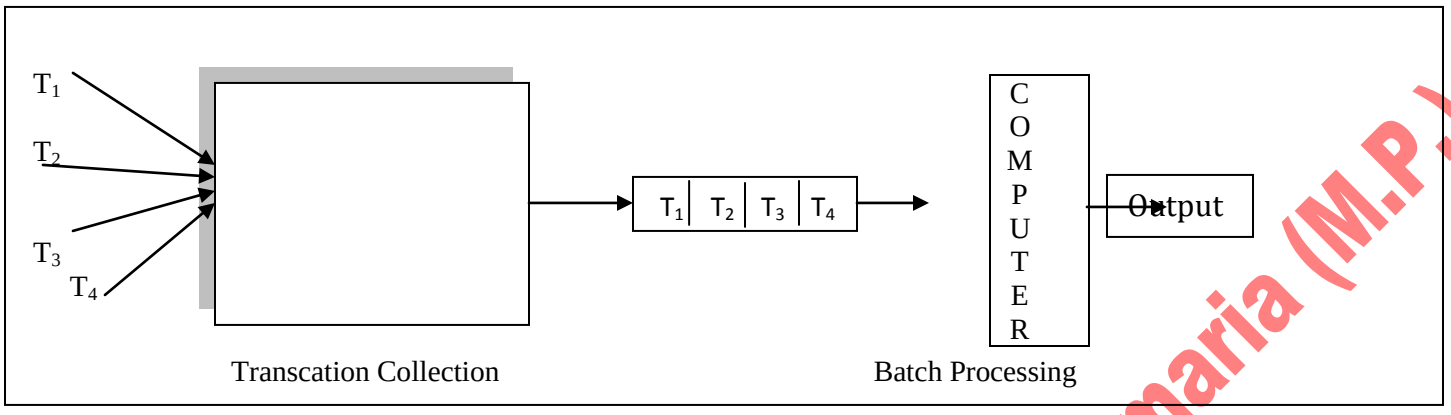
Durability: - Each trasaction में durability (टिकाऊपन) होना चाहिए कि उसके माध्यम से किए गए सभी परिवर्तन changing compleet and permanent हैं।

Isolation: - इसमें सभी transcation अपने आप में पूरी तरह अलग-अलग रुप में execute होता है। अगर system पर एक ही समय में दो या दो से अधिक transcation चल रहें हो और वे एक ही कार्य कर रहें हो तो भी सभी transcation इस प्रकार से execute हो जैसे उसे लगे कि वह अकेला ही system में Run हो रहा है अर्थात् सभी transcation की हर process एक दूसरे से अलग होनी चाहिए।

Types of transaction processing: -

1) BatchProcessing: - यह computer पर आधारित processing का एक तरीका है जिसमें वह data की processing करता है, उसे एक fixed time तक collect किया जाता है तथा collected data को एक unit के रुप में bach बनाकर उसकी एक बार में processing करा दी जाती है।

Example: - Electricity bill, student result बनाना bach processing है।

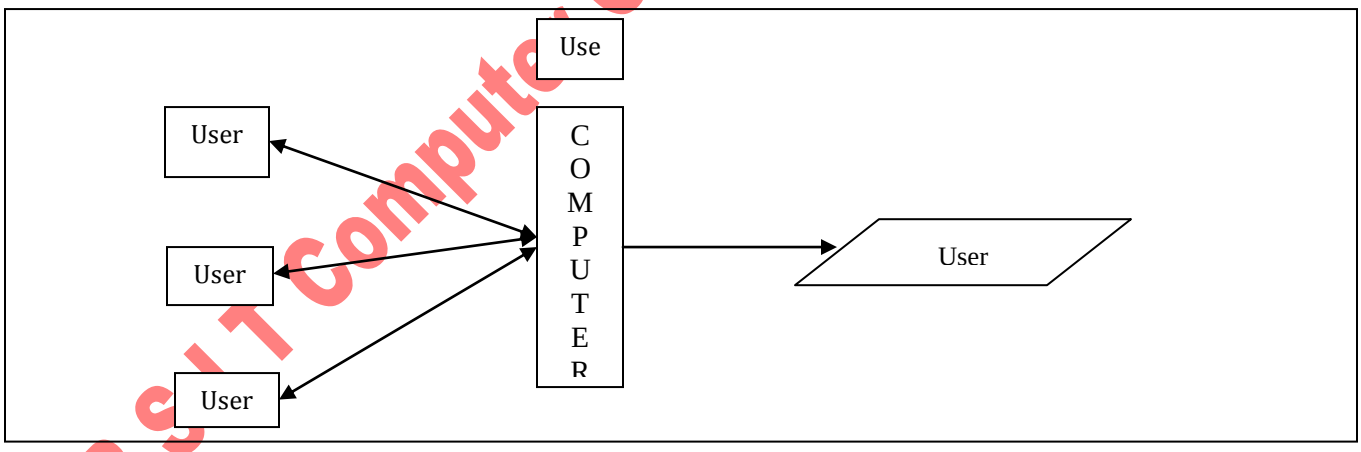


Batch Processing

2) Online Transcation Processing: - इसमें computer based trasaction complete होते ही तुरंत उसकी processing की जाती है तथा records को update कर दिया जाता है। Bank account में होने वाली trasaction में online processing होती है। इसमें तुरंत update हो जाता है।

Trasaction processing activity में नि0 लि0 process का प्रयोग किया जाता है।

- **Storage of data**: Trasaction के लिए data store करना।
- **Verification of data:** इसमें यह जाँच की जाती है कि collect किया गया data सही और पूर्ण है कि नहीं।
- **Correction of data**: Data में पाई जाने वाली error को correct करना।
- **Processing of data:** Data पर important calculation कर output को प्राप्त करना।
- **Generating the Report:** प्राप्त output को एक proper format में report generate करना।



Online Processing

3) Advance Transaction Processing: - आज के समय में tradation trasaction processes system की तुलना में advance and more flexible system उपलब्ध है, जैसे: -

Remote backup System:-

Remote backup system data की बहुत ही high level की उपलब्धता को provide कराता है, जहाँ तक कि यह primary site में available data के destroy होने के बाद भी transaction processing के process करने

में help करता है। यह एक computer के data को दूसरे computer में रखता है। यह system data के loss होने की स्थिति में भी data को recover करके उसे primary condition में लाने तक में help करता है।

High performance Transaction System: -

इन system में काफी high processing वाले hardware होते हैं जो transcation processing की दर को काफी अधिक रखते हैं, अर्थात् इन system में general system की तुलना में बहुत ज्यदा data होते हैं।

Real Time Transaction: -

इस प्रकार के system में database की पूरी reliability बनाए रखते हुए भी fixed time period में transcation को पूरा करने की क्षमता होती है। ये दो प्रकार के होते हैं –

1. **Hard Real Time: -** इस प्रकार के system में अगर fixed time period में कार्य पूरा न हो तो उस कार्य का कोई value नहीं होता है।
2. **Soft Real Time: -** इसमें fixed time period के बाद कार्य पूरा होने पर ही वह accept होता है लेकिन उसकी value कम हो जाती है।

***Transaction processing monitor*: -**

Starting time से ये एक ही processor द्वारा अनेक terminal को process कराने की क्षमता रखने वाले multi threaded server के रुप में विकसित किया गया था।

ये system अनेक server की मदद सेबहुत सारे clients पर बहुत complex transaction processing system develop करने तथा उसका management करने के लिए आवश्यक structure provide कराता है।

***Transaction workflow*:**

इसमें multiple processing unit के माध्यम से एक ही समय में अनेक कार्यों के execution और उनके बीच में coordination establish करने में मदद मिलती है। Suppose एक simple transcation, bank employee Rs 500 A's के account से B's के account में transfers करता है। यह बहुत आसान और small transaction processing है पर अगर यदि data का transcation करना है तो बहुत सारे low-level tasks involve होते हैं।

A's Account
Open_Account (A)
Old_Balance = A.balance
New_Balance = Old_Balance - 500
A.balance = New_Balance
Close_Account (A)

B's Account
Open_Account (B)
Old_Balance = B.balance
New_Balance = Old_Balance + 500
B.balance = New_Balance
Close_Account (B)

❖ ***Serializability*: -**

जब multiple transactions को operating system के द्वारा किसी एक multi programming environment के अंदर execute किया जाता है तो यह possibilities होती है कि वह किसी एक transcation से तथा साथ के कुछ दूसरे transcation से भी instructions inter leaved होता है।

Schedule: - किसी transaction के execution की sequenceing process को schedule कहा जाता है। एक schedule में बहुत सारे transactions हो सकते हैं, जैसे कि कई instructions/tasksमिलकर बने होते हैं।

Serial Schedule: यह एक प्रकार की scheduling है जो कि transaction को एक sequencing way से प्रयोग करती है। इसमें एक ही transaction सबसे पहले execute होता है ओर तब तक second execute नहीं होता है, जब तक कि पहला complete न हो जाए। जैसे ही पहला transaction complete होता है तब तुरंत दूसरा transaction start हो जाता है। Transaction key process order होता है। एक के बाद एक इस प्रकार की scheduling में transaction serial manner में होता है।

इस प्रकार की scheduling सिर्फ अलग-अलग multi environment transcation के लिए सही है लेकिन अगर same environment transcation की scheduling किया जाए तो यह सही प्रक्रिया नहीं है। उसके लिए parallel scheduling का प्रयोग किया जाता है। उसमें या तो equvilence अथवा serializability होगा।

Equivalence Schedules: - Equivalent schedule निम्न प्रकार के हो सकते हैं–

Result Equivalence: - यदि दो same result को execution के बादे schedule produce करते है तो उसे result equivalent कहा जाता है, जो कि ये शायद दूसरे value के set से कुछ value के same result और कुछ विभिन्न result देता है। View equivalent में यदि transcation के अंदर similer manner में दोनो schedule एक ही तरह के action को perform करे तो यह दो schedule रहेगा।

For example –

- यदि T, initial data S1 के अंदर read करता है तो यह initial data S2 के अंदर भी read करेगा।
- यदि T, S2 की value को read करता है जो कि J के द्वारा लिखी गई है तो यह S2 की value को भी read करेगा जो कि J के द्वारा लिखी गई है।
- यदि T, final written operation data value S1 में perform करता है तो यह final written operation data value S2 में भी perform करेगा।

Conflict Equivalence: - यदि हम निम्न properties को perform करते हैं तो confliting में दो schedule रहेगा–

1) यदि दोनो result separate transcation को belong करते हैं।

2) यदि दोनो result same data item को access करते हैं।

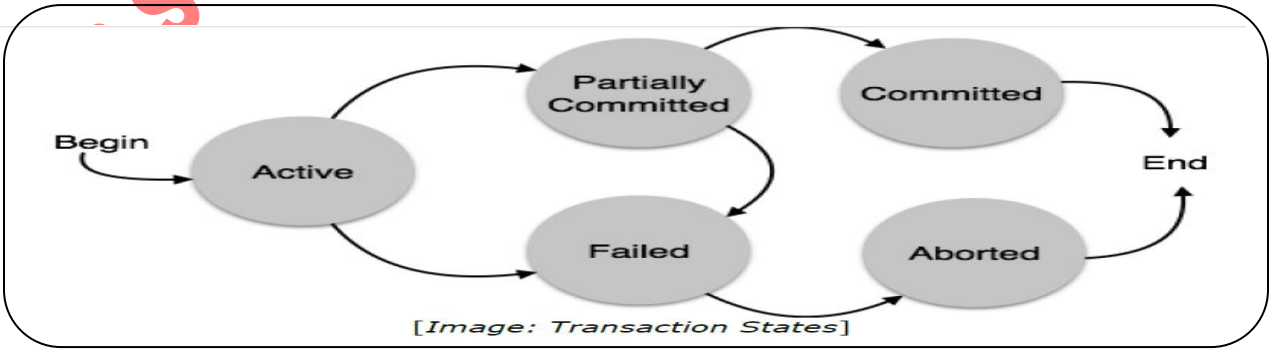
कम से कम इनमें से कोई भी एक write operation को perform करता है।

Two schedules के पास multiple transactions हो और साथ में conflicting operations perform हो तो उसमें conflict equivalent होता है। यदि सिर्फ और सिर्फ इन condtion को follow करे–

दोनो schedule एक ही तरह के transcation के set में contain करे।

Note– View equivalent schedules के view serializable and conflict equivalent schedules को conflict serializable कहते हैं। सभी conflict serializable schedules में बहुत view serializable होते हैं।

States of Transactions: - A transaction in a database can be in one of the following states:



[Image: Transaction States]

Active: इस state में transaction, execute होना शुरु होता है। यह प्रत्येक transcation process की initial state होती है।

Partially Committed: जब एक transaction final operational के लिए executes होता है तो उसे partially committed state कहा जाता है।

Failed: एक transaction failed state कहलाता है यदि किसी भी कारण से database recovery system fails हो जाए, तथा तब उस failed transaction को और आंगे precede न किया जा सके।

Aborted: यदि कोई भी checks fails हो जाता है और transaction एक failed state में पहुँचता है तो recovery manager उस trasaction को rolls back करता है और सभी write operations database के ऊपर perform करता है जिससे database again अपने original state में आ जाए, जहाँ पर कोई भी transaction का exection हुआ होता है में और फिर से होता है। Transactions अगर rollback नहीं होता हो पाता है तो वह उस trasaction को kill कर देता है। Database recovery module, transcation के aborts होने पर दो operations को select करता है–
Re-start the transaction
Kill the transaction

Committed: यदि एक transaction सभी operations को successfully executes करता है तो वह committed कहलाता है तथा database system में permanently updated हो जाता है।

Concurrency Control:

Multiprogramming environment के अंदर जहाँ पर एक ही समय में multiple transactions को execute किया जाता है, जिससे वह control करना बहुत ही important होता है। इस प्रकार के transaction को concurrency transaction के नाम से जाना जाता है। इसमें concurrency control protocols का use किया जाता है जो कि concurrent transactions में आने वाले atomicity, isolation, and serializability problem को control करता है, जिसमें multi-user एक ही समय में access कर सके। Concurrency control protocols को दो भाग में विभाजित किया गया है –

1) Lock-based protocols
2) Timestamp-based protocols

Lock-based Protocols: -

Locke based protocol में एक lock एक mechanism होती है जो Database systems को यह बताता है कि एक particular data को read/write operation को किस trasaction में apply करना है, जिन्हें दो तरीकों से किया जा सकता है–

- **Binary Locks: -** इसमें data item को दो states में lock किया जाता है, या तो यह या तो locked होगा या तो unlocked.

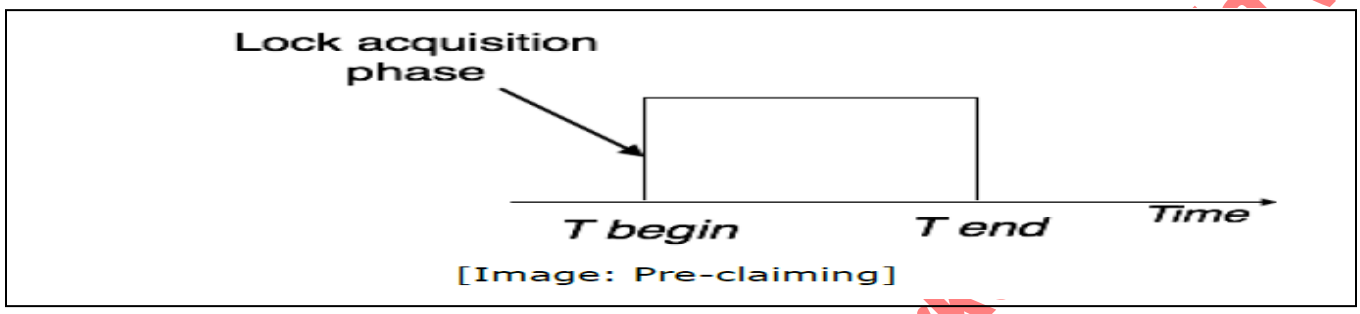
- **Shared/exclusive Locks:** - इस प्रकार के lock में एक data item को कोई दूसरी trasaction भी reading के purpose से प्रयोग कर सकती है।

Exclusive lock: इसमें एक transaction में किसी एक data item में exclusive lock को read/write दोनो को उपयोग कर सकती है तथा कोई दूसरा data item/trasaction same data item पर नहीं कर सकती है। उपयोग करने से पहले सबसे पहले उसे किसी particular transcation के लिए lock किया जाता है ताकि जब तक वह lock है तब तक सिर्फ एक ही process एक समय में उपयोग कर सके।
There are four types of lock protocols available:

SimplisticLock Protocol: - Simplistic lock – based protocol प्रत्येक write operation performed होने के पहले transcation को lock allow कराता है तथा बाद में data item में write operation पूरा होने के बाद transcation unlock हो जाता है।

Pre-claimingLock Protocol: -

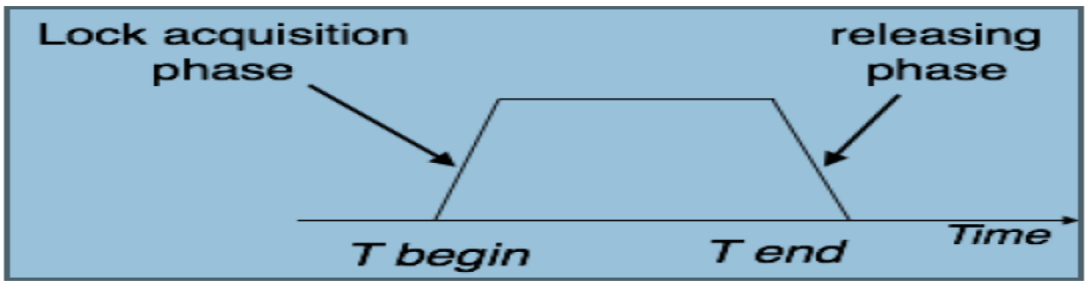
Pre-claiming protocols operations को evaluate करता है तथा data items की एक list create करता है जोकि lock के लिए जिसे आवश्यक होता है। Execution के पहले transaction system से request करता है कि सभी operation को उस particular transcation के लिए lock किया जाए। यदि सभी locks को permission granted हो जाती है तो transaction executes कराता है और operations के पूरा होते ही use होने वाले सभी locks को releases कर दिया जाता है ताकि दूसरा trasaction उसे उपयोग कर सके। यदि locks को granted नहीं किया गया तो transaction rolls back हो जाएगा और waits करेगा तब तक, जब तक उसे locks granted न कर दिया जाए।



[Image: Pre-claiming]

Two-Phase Locking –2PL: -

यह locking protocol के किसी एक transcation के execution phase को three parts में विभाजित करता है। First part में जब transaction starts executing, तब यह permission के लिए locks को requiest करताा है। Second part में जहाँ transaction acquires (सभी locks के उपयोग के लिए अधिकार प्राप्त करता है) तथा third phase starts होता है इस phase में transaction किसी new locks के लिए demand नहीं कर सकता है। यहाँ पर केवल acquired locks को releases किया जाता है।

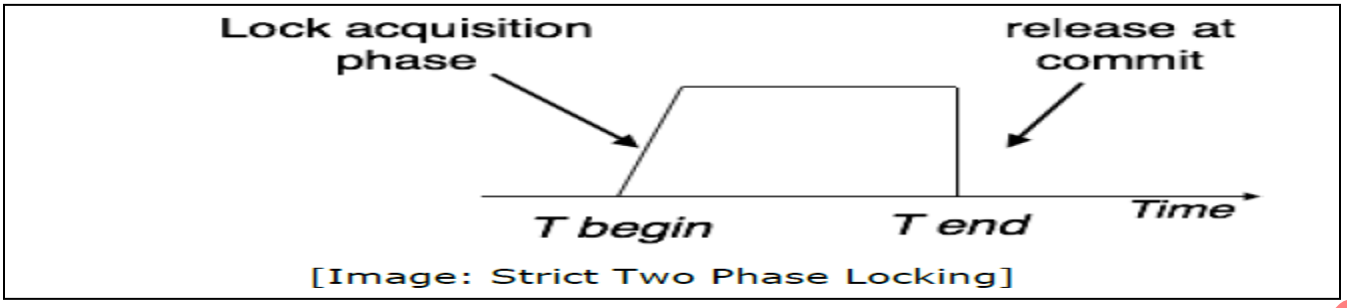


[Image: Two Phase Locking]

Two-phase locking में two phases होते है। First phase **growing** कहलाता है जहाँ सभी locks acquired transcation के द्वारा शुरु होते हैं और second phase **shrinking** कहलाता है जहाँ सभी used हो रहे locks को transaction से released करने के लिए प्रयोग किया जाता है जो कि exclusive (write) lock कहलाता है और shared (read) lock कहलाता है।

Strict Two-Phase Locking: -

इसका पहला phase 2PL की तरह होता है जो कि सभी locks के acquiring करने के बाद transaction फिर continues normally execute करता है पर 2PL से थोडा अलग Strict-2PL उपयोग करने के ठीक बाद उसे release नहीं करता है। Strict-2PL सभी locks को holds करता है जब तक कि commit point और सभी locks at a time एक साथ releases न कर दे।



[Image: Strict Two Phase Locking]

Timestamp-based Protocols:

यह सबसे ज्यादा उपयोग होने वाला concurrency protocol है जो कि timestamp based protocol कहलाता है। इस protocol का प्रयोग system time or logical counter मे count करने में timestamp की तरह किया जाता है।

Lock-based protocols का उपयोग एक ही समय में होने वाले execution में conflicting pair's के order और transactions को manage करने के लिए किया जाता है, जहाँ timestamp-based protocols transaction के create होते ही बहुत जल्दी working करना start हो जाता है।

Every transaction का एक times tamp के साथ associated होता और उसके order से जो कि determined किया जाता है किसी transaction के total execution time मे जो कि Prioty based पर कार्य किया जाता है जिसमे कम time लगता है और जो पहले entering करता है वह सबसे पहले execute होगा।

Timestamp Ordering Protocol:

Timestamp-ordering protocol निश्चित कराता है serializability को जो कि transactions के process मे आने वाली conflicting read and write operations को जो कि यह responsibility होती है कि timebased order के अनुसार transaction को execute कराए।

- The timestamp of transaction Ti को denoted किया जा रहा है TS (Ti).
- Read timestampमे data-item X है जिसे R-timestamp(X) के द्वारा denoted किया जा रहा है।
- Write timestampमे data-item X है जिसे W-timestamp(X) के द्वारा denoted किया जा रहा है ।

Timestamp ordering protocol works as follows:

- यदि एक transaction Ti एक read(X) operation को issues करता है।

If TS (Ti) < W-timestamp(X)

- Operation rejected.

If TS (Ti) >= W-timestamp(X)

- Operation executed.

All data-item timestamps updated.

- यदि एक transaction Ti issues करता है एक write(X) operation:

If TS (Ti) < R-timestamp(X)

- Operation rejected.

If TS (Ti) < W-timestamp(X)

- Operation rejected and Ti rolled back.

Otherwise, operation executed.

Thomas' Write Rule: - यह rule states यदि TS(Ti) <W-itemstamp((X) है तो operation को reject किया जा सकता है और Ti rolled back हो जाता है।

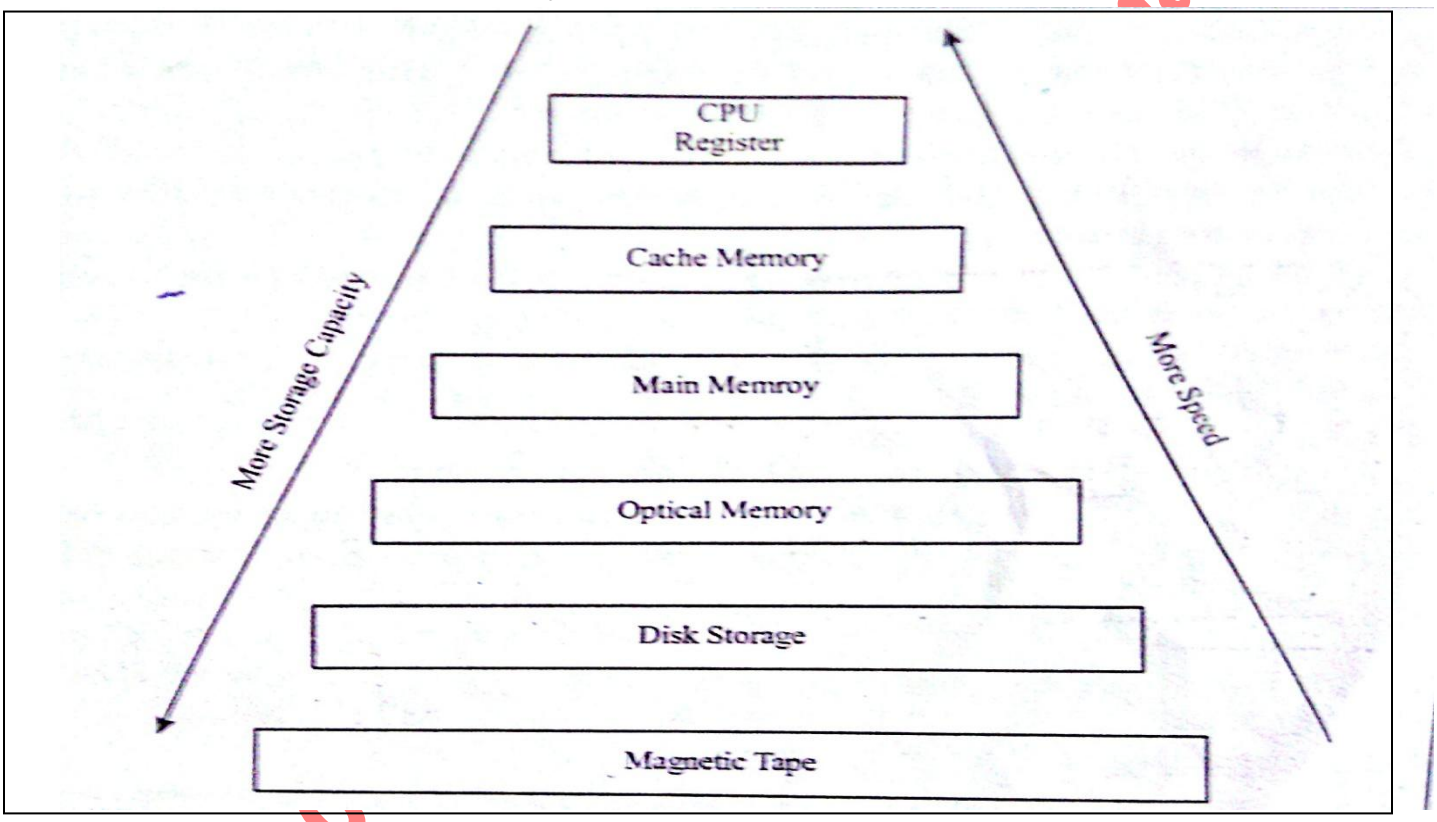
Timestamp ordering rules में बनाए गए view serializable के schedule को modified किया जा सकता है। Ti rolled back के अलावा 'write' operation itself ignored हो जाता है.

Que: - What is Memory Hierarchy Explain.

Ans: - ***MemoryHierarchy: -***

Storage devices; जैसे register cash memory, Main memory, Disk device, Magnatic tap, Optical disk etc सभी के द्वारा हम उनकी speed and storage capacity के base पर सुसंगठित पदानुक्रम में रख सकते हैं। यह क्रम हम निम्न चित्र के द्वारा समझ सकते हैं–

Memory का उपरोक्त पदानुक्रम memory ***Hierarchy*** कहलाता है। उपरोक्त Hierarchy में दर्शाई गई अलग-2 memory का use तथा system में उनकी भूमिका भी अलग-2 होती है। जैसे–



Memory Hierarchy

Cash Memory: - Cash Memory C.P.U. तथा main memory के बीच medium का कार्य करती है। यह सबसे महंगी लेकिन अत्यंत तीव्र गति की memory होती है जो अभी चल रही processing से सम्बन्धित आवश्यक data block main memory से लेकर तेजी से C.P.U. को उपलब्ध करवाती है। इसका आकार बहुत छोटा होता है।

C.P.U. Register: - Register memory का सबसे आंतरिक भाग है जो ALU द्वारा सीधे access की जा सकती है।

Main Memory: - यह एक अस्थायी संग्रह मेमोरी होती है जिसमें वे program and user data रहते हैं जिन पर हम अभी कार्य कर रहे हैं। इसे हम RAM (Randam Access Memory) के नाम से जानते हैं। यह Megabyte के आकार में आती है लेकिन वर्तमान में RAM gigabyte के आकार में भी उपलब्ध है।

Optical Memory: - Optical memory के अंतर्गत C.D., D.V.D. आदि storage media आते है। इसका उपयोग अधिक मात्रा में data स्थायी रुप से store करने के लिए किया जाता है। ये offline storage medium है; अर्थात् इनमें हम वह data तुरन्त save करते हुए नहीं चल सकते जिस पर हम सभी कार्य कर रहें हैं। इसमें केवल वे program and data store किए जाते हैं जिन्हें हमें एक बार store करने के बाद भविष्य में केवल बार-बार read करना है।

Disk Storage: - इसे हम Harddisk के नाम से जानते हैं जिसकी storage capacity बहुत अधिक होती है, वर्तमान में personal computers में सामान्यतः 80 gigabyte या 160 gigabyte hard disk का use किया जा रहा है जो कि एक बहुत विशाल storage capacity है। यह एक online storage medium है; अर्थात् हम computer पर जो भी कार्य करते हैं वह तुरन्त Hard disk में store होता है। इसमें stored data randam रुप से access किया जा सकता है; अर्थात् हम सीधे किसी भी file या किसी भी data item को read कर सकते हैं।

***Magnatic Tap*: -**

यह सबसे सस्ता storage medium है। इसका उपयोग ऐतिहासिक स्वभाव का data रखने के लिए किया जाता है जिसे एक बार store करने के बाद बार-बार read करने की आवश्यकता ना हो। यह एक sequential storage medium है इसलिए यह अधिक प्रचलन में नहीं है।

उपरोक्त Memory Hierarchy का मुख्य उद्देश्य computer की तीव्र processing capacity तथा हमारी विशाल storage आवश्यकता के बीच तालमेल स्थापित करना है जिसमें memory की लागत (cost) सबसे महत्वपूर्ण पक्ष है।

इस memory Hierarchy को हम accessing system के base पर वर्गीकृत करके भी देख सकते हैं, जो निम्नानुसार होगा–

- Random Access Memory (RAM)
- Daynamic Access Storage Memory (DASD)
- Sequential Access Memory (SAM)

Computer की Read/Write Memory RAM, Random Access Memory की श्रेणी में आती है। Computer की Online स्थायी storage device, hard disk, direct access storage device के अंतर्गत आती है, जबकि magnetic tap , sequential access memory के अंतर्गत आती है।

Memory Hierarchy के वर्गीकरण का एक और महत्वपूर्ण आधार memory का access time भी है। ऊपर चित्र में दिखाए गए memory Hierarchy classification के पीछे वास्तव में access time अधिक महत्वपूर्ण तथ्य है। जहां तीव्रतम गति की मैमोरी को सबसे ऊपर रखा जाता है, जबकि न्यूनतम् गति की मैमोरी को सबसे नीचे रखा जाता है।

Forword and Backword Recovery: -

Recovery system यह सुनिश्चित करती है कि किसी भी condition में recover होने के बाद प्राप्त data विश्वसनीय हो। System फेल्योर की स्थिति में यह बहुत ही गंभीर मुद्दा होता है, क्योंकि जब system स्वयं ही फेल हो चुका हो तब वह data पर से भी अपना control खो देता है।

System फेल्योर या Database error से data की recovery data को पिछली विश्वसनीय में या अगली विश्वसनीय अवस्था में लाकर की जा सकती है। Data को पिछली विश्वसनीय अवस्था में लाना backword recovery कहलाता है। जबकि अगली विश्वसनीय अवस्था प्राप्त करना forword recovery कहलाता है।

Backword Recovery (UNDO): - इसमें फेल्योर के समय चल रहे transcation के द्वारा किए गए अविश्वसनीय तथा आधे अधूरे परिवर्तनों को निरस्त (UNDO) कर दिया जाता है, ताकि database वापस error आने से पूर्व की अवस्था में लाया सके।

Forword Recovery (REDO): - इसमें transcation द्वारा error आने के ठीक पहले तक पूरे किए जा चुके परिवर्तनों को database की पिछली copy पर पुनः apply किया जा सके।

जब system में किसी विशेष error का पता चलता है तो recovery system की वर्तमान स्थिति का सटीक मल्यांकन करती है तथा system में आवश्यकतानुसार समायोजन करती है।

System में UNDO and REDO operation एक ही बार किए जाने चाहिए, क्योंकि बार-बार किए गए UNDO and REDO और अधिक अनिश्चित स्थिति उत्पन्न कर सकते हैं।

What is data overload? Describe problems related with it.

ऐसी organizations जहां central database नहीं रखा जाता तथा data sharing की facilities का use नहीं किया जाता सामान्यतः file based system पर ही depend होती है। इस व्यवस्था में सभी विभाग अपनी-2 आवश्यकता का data अपनी सुविधानुसार रखते हैं जिसके कारण प्रायः एक ही जानकारी एक से अधिक स्थानों पर रखी जाती है। जैसे किसी company में उसके customers का विवरण sales department में भी रखेगा तथा account department में भी। इस condition में जब पहले से ही organization की समस्त जानकारियों से related data की मात्रा बहुत अधिक हो अगर एक ही data की नकल भी एक से अधिक स्थानों पर रखी जाए तो समस्त data की कुल मात्रा बहुत अधिक हो जाती है, जिससे system पर data का बोझ हो जाता है। यह स्थिति data की अधिकता या overload कहलाती है, जिससे अनेक व्यवहारिक समस्याएं उत्पन्न हो जाती है। जिनमें से प्रमुख समस्याएं निम्नानुसार हैं–

Data Inconsistencies (डाटा असंगति)

एक ही data एक से अधिक स्थानों पर रखे जाने पर data को update करते समय कई बार कुछ स्थानों पर तो dataupdate कर दिया जाता है लेकिन गलती से इसे कुछ स्थानों पर update करना रह जाता है, जिसके कारण एक ही data को अलग-2 स्थानों पर अलग-2 मान मिलते हैं जिससे data की संगति समाप्त हो जाती है।

Data Reliability of Data (डाटा की वि वसनीयता में कमी

असंगति के कारण data की विश्वसनीयता समाप्त हो जाती है।

Accessibility Problem (डाटा प्राप्ति की समस्या)

परंपरिक file system में data की पुनः प्राप्ति एक कठिन कार्य है, क्योंकि इसमें file को access करने के लिए भी अलग से code लिखना पड़ता है।

Slow Access Speed (धीमी एक्सेस गति)

Data अधिक होने पर आवश्यक data को search and access करने में समय अधिक लगता है।

Integrity Problem (इंटिग्रिटि की समस्या)

Central database में असानी से एक ही स्थान पर आवश्यक कन्सट्रेन्ट लगाकर data input में होने वाले errors को control किया जा सकता है, लेकिन जब एक ही data एक से अकि स्थानों पर रखा जाए तो data की एकरुपताबनाए रखना अधिक कठिन कार्य है, जिससे integrity की problem आती है।

Security Problem (सुरक्षा की समस्या)

File based system में अनाधिकृत व्यक्तियों द्वारा data को access किए जाने से बचाने के लिए पर्याप्त व्यवस्था नहीं होती है, जबकि database based system data की सुरक्षा के लिए पर्याप्त सुविधाएं उपलब्ध होती है।

उपरोक्त समस्याओं से बचने के लिए centrl database व्यवस्था अपनाई जाती है जिसमें आवश्यक data की एक ही copy centrl form में रखी जाती है। जिसमें से सभी user अपनी आवश्यकता का data share कर सकते हैं। इससे system पर data के दोहराव का बोझ नहीं रहता तथा data की विश्वसनीयता भी अधिक होती है। यद्यपि centrl database को अनेक user share करते हैं लेकिन DBMS password द्वारा user arithmetic and access अधिकार आबंटन के माध्यम से काफी कठोर सुरक्षा व्यवस्था देता है जिसके कारण central database में data और अधिक सुरक्षित रहता है।

Shadow Paging: -

अगर transcation क्रमानुसार perform हों तो shadow paging technique useful है। यह data recovery की प्रमुख techniques में से एक है। इस तकनीक में transcation के समय दो page table रखे जाते हैं–

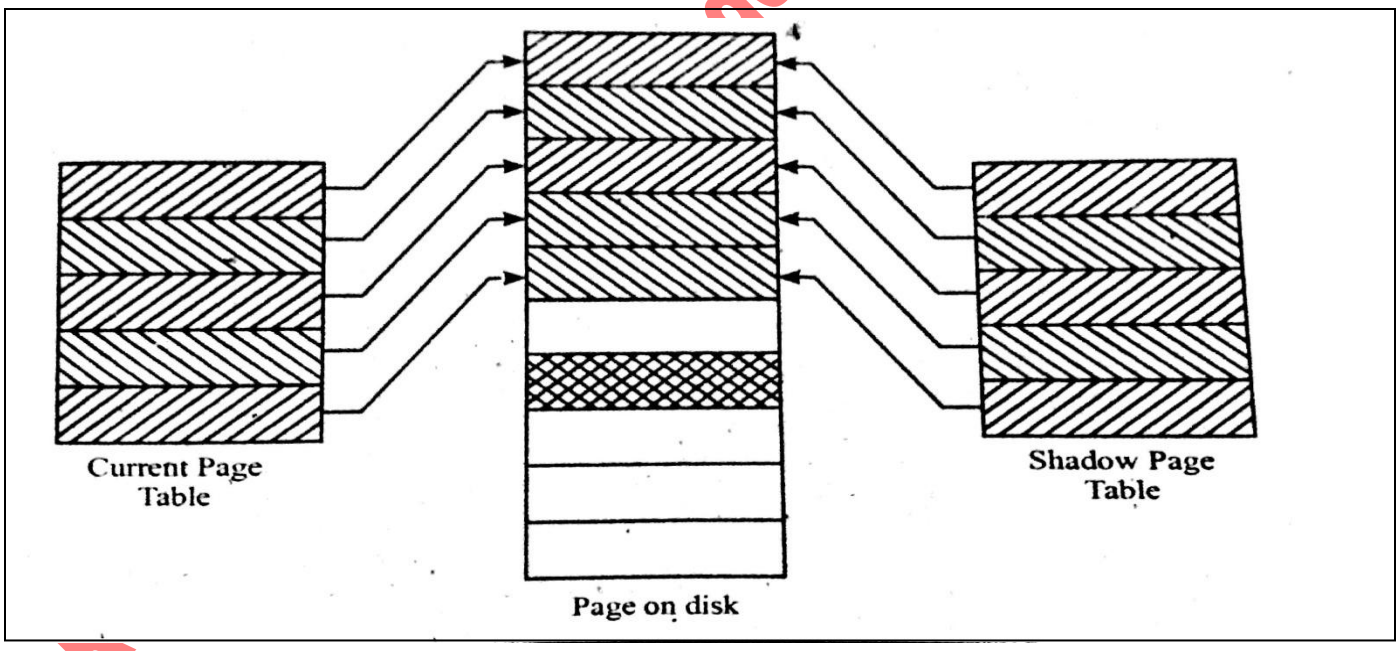
a) Current page table
b) Shadow page table

Shadow page table में वे सभी information store की जाती है जिनकी सहायता से आवश्यकता पड़ने पर database की transcation से पूर्व की अवस्था प्राप्त की जा सकती है, अर्थात् किसी कारण से transcation पुराना हो पाए तो हू file का पुराना data जैसा या ठीक उसी रुप में प्राप्त कर सके। सरल शब्दों में कहें तो किसी भी data file पर transcation आरम्भ करने से पूर्व उसके पंतों की एक copy बनाकर रख ली जाती है ताकि अगर transcation किसी कारणवश बीच में रुक जाए और उसके द्वारा किए गए परिवर्तन आधे-अधूरे हों तो पेजों की copy से मूल data पुनः प्राप्त किया जा सके। इस प्रकार रखी गई data files के पेजों की copy shadow page कहलाती है। एक तरह से हम इसें data की छाया प्रति भी कह सकते हैं। Transcation के दौरान data items को access करने के लिए केवल current page table का उपयोग किया जाता है। Shadow page table को सिर्फ आवंछित स्थितियों के लिए सुरक्षित रखा जाता है।

इस तकनीक में transcation पूरा होने के बाद उसे अन्तिम रुप देने के लिए निम्न प्रकिया अपनाई जाती है–

1) परिवर्तन हुए सभी पेज hard disk में भेज दिए जाते हैं।
2) Current page table को hard disk में store कर दिया जाता है।
3) Current page table को new shadow page table बना दिया जाता है, इसकी प्रकिया निम्नानुसार होती है–

- Shadow page table का pointer एक निश्चित location पर स्थिर रखा जाता है।
- Current page table को new shadow page table बनाने के लिए shadow page table पर pointer, new current page table पर set कर दिया जाता है, जिससे नवीनतम page shadow table के रुप में set हो जाता है।



Advantages of shadow paging: -

- इसमें long record लिखने के समान अतिरिक्त कार्य नहीं करना पड़ता।
- Data की recovery बहुत आसान होती है।

Disadvantages of Shadow paging: -

- पूरे page table को copy करने में memory काफी अधिक खर्च होती है।

- Data disk में अनेक स्थानों पर बांटा जाता है।
- प्रत्येक transcation के बाद बहुत सारे पुराने data को delete करना पड़ता है।
- Transcation के बाद परिवर्तित पेज तथा current page table को hard disk में store करना, new page को पुनः shadow page बनाना पुनः pointers set करना आदि अनेक अतिरिक्त कार्य बढ़ जाते हैं और प्रत्येक transcations के बाद यह सब कार्स करना एक बड़ा बोझ हो जाता है।

Que: -Operational data से आप क्या समझते हैं?

Ans: - किसी भी संस्था के समस्त उपयोगी आँकडें जिन्हें एकत्रित कर संस्था में डाटा के रुप में system में input किया जाता है, operational data कहलाता है। जिस पर विभिन्न processing work कर संस्था processing के output के रुप में useful information प्राप्त करती है। सरल शब्दों में हम कह सकते हैं कि वह समस्त Row Data जिस पर संस्था द्वारा कुछ न कुछ operation किया जाता है operational data कहलाता है।

प्रत्येक संस्था के लिए उसकी आवश्यकता के अनुसार समस्त operational data की पहचान करना तथा उसे एकत्रित करना बहुत महत्वपूर्ण कार्य है क्योंकि बिना डाटा के संस्था को information प्राप्त नहीं हो सकती और information के बिना किसी भी तरह के decision नहीं लिए जा सकते और संस्था आंगे नहीं बढ़ सकती। किसी भी संस्था के लिए data एकत्रित करना एवं उसे computer में input करना एक बडा कार्य होंता है जिसमें data के sources का पता लगाना, sources से data प्राप्त करना, data की sorting करना, उसे क्रम में जमाना, data की सत्यता की पहचान करना और उसे computer में सही रुप से input कराना और भविष्य के उपयोग के लिए सुरक्षित रखना आदि सभी कार्य आते हैं। Data input होने के बाद उस पर उचित operation कर उपयोगी information प्राप्त करना भी अधिक महत्वपूर्ण कार्य है।

संस्था के लिए आवश्यक समस्त operational data की पहचान बेहतर database management नीति के लिए आवश्यक है। Operational data की यह पहचान problem defination से आती है। किसी भी संस्था में database management system की पहली प्राथमिकता operational data पर ही केन्द्रित होता है।

उदाहरण– किसी भी संस्था में कर्मचारियों से related information को हम operational data के उदाहरण केरुप में देख सकते हैं।

जैसे –

Operational Data	विवरण
1, 2, 3, 4, 5	Serial number
Abhi, Prakash, Akash, Rupesh, Rohit	Employee Name
1001, 1002, 1003, 1004, 1005	Employee ID number

Que: - Write a note on log based recovery.

Ans: - <u>Log-Based Recovery</u>: -

इस तकनीक के अनुसार storage media में एक log रखा जाता है। यह log, log records का group होता है, जो database पर की जाने वाली प्रत्येक update activities का record रखता है। जब भी कोई transcation start होता है तो वह पहले log record में अपनी रजिस्ट्रि कराता है। मान लीजिए एक transcation Ti start हो रहा है, यह निम्नानुसार पहले log record में अपनी रजिस्ट्री कराएगा।

$< Ti\ Start > Log\ Record$

Transcation Ti write (X) function perform करने से पहले log record में इसकी जानकारी $<Ti, X_1, V_1, V_2>$ लिखेगा, जहां V_1 X का write ( ) function चलाने से पहले का मान है तथा V_2 वह मान है जो अब X में लिखा जाना है।

- Log record अपनी information में इस विवरण को लिखकर रख लेगा कि Ti transcation ने data items X पर write ( ) function perform किया तथा write ( ) function चलाए जाने से पहले X का मान V_1 and write ( ) function द्वारा अब X का मान V_2 कर दिया गया है।
- जब Ti अपना last statement समाप्त कर लेगा तब log record <Ti commit> लिखा जाएगा।

आवश्यकता पडने पर इस log record की सहायता से data items के मानों की recovery की जा सकती है। Log record से यह data recovery निम्न दो तरीकों से की जा सकती है–

a) Differed database modification
b) Immediate database Modification.

- ***Buffer Management*: -**

Database में परिवर्तन के समय उसके records को तथा log record को अनेक buffer आबंटित होते है तथा इन buffer में store values में बडे पैमाने पर परिवर्तन होते हैं। यद्यपि buffer management operating system के द्वारा किया जाता है लेकिन अनेक बार buffer management के कार्य में भी DBMS अपनी भूमिका निभाता है ताकि data की विश्वसनीयता को यह स्वयं सुनिश्चित कर सके। इससे related प्रमुख कार्य निम्न लिखित है–

1. Log Record Buffering: - Log record buffer main memeory में store किए जाते हैं तथा इन्हें स्थायी memory में भी भेजा सकता है। जब buffer में log record के block पमरी तरह भर गए हों या Log force operation perform हुआ हो। Log force operation transcation के पूरा होने पर perform होता है जो transcation के सभी लोग records को स्थयी मेमोरी में store करता है।

Log record buffering के संबन्ध में निम्न नियमों का पालन आवश्यक है–

- कोई भी transaction तभी पूर्ण माना जाता है जब उसका log record <Ti commit> स्थायी memory में store हो चुका है।
- Log record स्थायी memory में उसी क्रम में store होते हैं जिस क्रम में वे create हुए हैं।
- जब main memory से data का कोई block database में store हो तब यह आवश्यक है कि उससे related समस्त log records भी स्थायी memory में जाए।

2. <u>Database Buffering</u>: - Database main memory में अपने data block के buffer भी रखता है। Database buffering management के अन्तर्गत main memory में data block के लिए buffer आबंटित करना, कार्य पूरा होने पर उन्हें database में store करना तथा कार्य के दौरान database से buffer में data लाना आदि कार्य आते हैं। इसमें जब database को आबंटित buffer पूरी तरह भरे हों, लेकिन database से और data लाने की आवश्यकता हो तब स्वेपिंग का कार्य भी buffer management के अन्तर्गत आता है। Database buffering में सामान्य प्रबन्धन सम्बंधी बातों का विशेष ध्यान रखना आवश्यक होता है। जैसे डाटा बुर के डिस्क में store होने के बाद उसमें कोई updation कार्य नहीं होना चाहिए। Data buffer का व्यवस्थित प्रबन्धन निम्न तरीकों से सुनिश्चित किया जाता है–

- Data items को buffer block में load करने से पहले transcation related buffer blocks पर exclusive block प्राप्त होता है ताकि इन buffer blocks को उस समय कोई अन्य transcation access ना कर सके।
- कार्य पूरा होने के बाद ही buffer block पर log मुक्त किए जाते हैं।
- Block को अंतिम रुप से disk में store करने से पहले system यह सुनिश्चित करता है कि blocks में कोई updation कार्य ना चल रहा हो।

Transcation के दौरान database buffers main memory में भी रखे जा सकते हैं और virtual memory में भी। इन्हें main memory में रखने में सबसे बड़ी असुविधा यह है कि main memory database and application program के बीच divide होती है। अतः main memory में database को एक सीमित मात्रा में ही buffer block मिल सकते

हैं जो उसके लिए पर्याप्त नहीं होते। अतः इस असुविधा से बचने के लिए database buffer virtual memory में ही रखे जाते हैं।

RAID: - RAID अनेक छोटी तथा सस्ती hard disks का समूह होता है जहां data इन अनेक disks के बीच बोटा जाता है। RAID का अर्थ Redundant Array of Independent Disks होता है। इस व्यवस्था की निम्न विशेषताएं होती हैं–

- Operating system सभी disk के समूह को एक logical drive के रुप में देखता है।
- Data इन physical disks के बीच बंटा होता है।
- Data की एक से अधिक copy इन अलग-2 disks में रखी जाती है।
- किसी disk के फेल होने पर उसके data की अन्य disk में रखी गई copy से data recover कर लिया जाता है।

एक array के रुप में रखी गई ये disk system को विरूवसनीयता प्रदान करती है तथा data को अधिक सुरक्षित रखती है। इस व्यवसथा का सबसे बड़ा लाभ यह है कि किसी disk के फेल होने पर भी इसमें कार्य में रूकावट नहीं आती और system चलता रहता है। RAID के अंतर्गत आने वाली प्रमुख अवधारणाएं निम्नलिखित हैं–

Stripping: - Data को एक से अधिक डिस्कों के बीच बांटने की प्रक्रिया stripping कहलाती है।

Mirroring: - Data को एक से अधिक डिस्कों में copy करना Mirroring कहलाता है। Mirroring का अर्थ किसी भी वस्तु की same to same copy दूसरे जगह बनान होता है। अर्थात् यहां mirroring से तात्पर्य एक disk के data की copy दूसरे disk में बनाने से है।

RAID सुविधा के भी अलग-2 level उपलब्ध है। जहां भिन्न-2 level के RAID में हमें अलग-2 तरह की सुविधाएं मिलती हैं। इसके 76 level उपलब्ध हैं जिनमें मिलने वाली सुविधाएं निम्नानुसार हैं–

RAID – Level 0: - RAID का यह स्तर Non-Redundant होता है अर्थात् इस स्तर पर data की अलग–अलग disk में copy नहीं होती। इस स्तर के RAID में डाटा केवल अलग–अलग डिस्कों में बंटा होता है। इस स्तर के RAID का कार्य प्रदर्शन बहुत अच्छा होता है, क्योंकि इसमें डाटा का दोहराव नहीं होता लेकिन इसमें किसी disk के फेल होन पर data के नष्ट होने का खतरा रहता है।

Non – Redundant
RAID – Level 0
Mirrored
RAID – Level 1
Redundency Through Hamming code
RAID – Level 2
Bit – interleaved Parity
RAID – Level 3
Block – Parity
RAID – Level 4
Block- Level Distributed Parity
RAID – Level 5
Dual Redundency

RAID – Level 6

RAID – Level 1: - ऊपर दर्शाए गए चित्र से यह स्पष्ट है कि Mirrored level है; अर्थात् Level-1 में सम्पूर्ण data की दो या दो से अधिक disk में copy होती है। इस स्तर में किसी डिस्क के फेल होने पर भी डाटा का नुकसान नहीं होता, क्योंकि डाटा की एक से अधिक copy उपलब्ध होती है। इस स्तर पर data reading process बहुत तेज होती है लेकिन data writing process धीमी होती है। Data के दोहराव के कारण यह स्तर अधिक महंगा होता है।

RAID – Level 2: - इस स्तर पर data के दोहराव के साथ–साथ error सुधार के लिए Hamming Code का use भी किया जाता है। इसमें डिस्कों में error से बचाव के कारण सुरक्षा अधिक रहती है। विशेष रुप से जिन डिस्कों में आंतरिक रुप से error को देखने की सुविधा नहीं होती वहां यह स्तर बहुत उपयोगी है।

RAID – Level 3: - इस स्तर पर डाटा अनेक डिस्कों में बंटा होता है तथा इसमें Parity bit का use किया जाता है। यह Parity एक ही डिस्क में store रहती है। इस स्तर पर Data Bits के रुप में होता है अर्थात् Data bits के Set (byte) के रुप में रखा जाता है।

RAID – Level 4: - इस स्तर में Data blocks में बंटा होता है। इस स्तर में भी Parity का उपयोग किया जाता है। इसमें भी Parity एक ही disk में store रहती है।

RAID – Level 5: - यह स्तर भी स्तर-4 के समान ही है, अंतर केवल यह है कि इसमें Parity अनेक डिस्कों के बीच बटी रहती है। Parity अलग–अलग डिस्कों में बंटी होने के कारण इस स्तर में write operation तेजी से होता है, लेकिन read operation तुलनात्मक रुप से धीमा होता है क्योंकि Data read करते समय इन Parity bits को छोडते हुए data read करना पडता है।

RAID – Level 6: - यह Dual-Redundency level है अर्थात् इसमें प्रत्येक डाटा के मूल Copy के अतिरिक्त कम से कम दो copy और रखी जाती है। इस स्तर पर data की विश्वसनीयता सबसे अधिक होती है क्योंकि एक से अधिक बार डाटा को copy करते समय प्रत्येक copy की एक-दूसरे से तुलना होती है जिसके कारण error की सम्भावना नहीं रहती। यह स्तर डिस्क फेल्योर से भी अधिक सुरक्षित स्थिति में रहता है क्योंकि डाटा की दो से अधिक copy उपलब्ध रहती है।

Database Adminsistrator: - किसी भी बडी सेस्था में जिसका कार्य केंद्रित है, Database adminsisrator एक बहुत महत्वपूर्ण तथा जिम्मेदारी वाला पद होता है। Database adminsisrator एवं Database manager में दक्ष व्यक्ति होता है जिसका कार्य database design करना,ए उसका management करना, database की सुरक्षा करना, data base के सभी users के access अधिकार निर्धारित करना, उन्हें लागू करना तथा अवांछित व्यक्तियों द्वारा database को नुकसान पहुँचाने के प्रयासों से database की रक्षा करना होता है। इस प्रकार हम कह सकते हैं कि Database adminsisrator की भूमिका उस केंद्रित व्यक्ति की होती है जो database को पूरी तरह नियंत्रित करता है।

बहुत बडी तथा हजारों लोगों के जीवन से जुडी संस्थाओं जैसे Indian Railway में जहां प्रमुख कार्य सभी गाडियों तथा यात्रियों के आरक्षण के data पर केन्द्रित है और उसमें भी जहां data सभी स्टेशनों द्वारा Share किया जाता है, हम समझ सकते हैं कि Database adminsisrator का कार्य कितना महत्वपूर्ण और जोखिम तथा तनाव से भरा होता है। इस तरह की संस्थाओं में Database adminsisrator की भूमिका इतनी तनावपूर्ण होती है कि वे कई-कई रात ठीक से सो नहीं पाते। इस तरह की संस्थाओं में database management का कार्य किसी एक व्यक्ति के उत्तरदायित्व में ना होकर एक से अधिक विशेषज्ञ व्यक्तियों के समूह द्वारा किया जाता है।

Database adminsisrator की भूमिका application programs and database की structure को control करने की भी होती है। संगठन के व्यक्तियों से विचारविमर्श करके DBA databse का logical view तैयार करता है तथा database की physical structure भी लागू करता है। इस प्रकार DBA, Database Management के तीनों स्तरों पर महत्वपूर्ण भूमिका निभाता है–

- Database की logical and physical structure तैयार करना।
- Database tables की संरचना करना।
- Database की secondary storage में सही रुप से store करना तथा उचित access control सुनिश्चित करना।

इसी के साथ DBA का एक और बहुत महत्वपूर्ण कार्य सुरक्षा सीमाएँ, integrity सीमाएँ तथा back up and recovery process को design करना भी होता है। इस प्रकार DBA की सभी भूमिकाओं को हम निम्न बिन्दुओं में समेट सकते हैं–

- सभी आवश्यकताएं निर्धारित करना।
- Database का निर्माण एवं प्रबन्धन।
- Database का centralized control.
- DBMS के तीनों स्तरों को विकसित करना।
- सभी आवश्यक मानदंडों को अपनाकर database integrate बनाए रखना।
- User password and access authority तय करना।
- फेलियर की स्थिति में database recovery की process को सफलतापूर्वक संचालित करना।

1) Schema को परिभाशित करना: - इस कार्य में DBA Database की structure design करता है। इसके लिए users की सभी आवश्यकताओं को परिभाषित किया जा सकता है, Database में रखे जाने वाले सभी items निर्धारित किये जाते है तथा database की logical definiation द्वारा स्कीमा तथा योजनाओं का संकलन किया जाता है। यह स्कीमा डाटाबेस की DDL statement को लिखकर design किया जाता है। ये statement DDL compiler द्वारा अनुवादित किए जाते हैं जिससे table बनते हैं। इसके अलावा DBA data डिक्शनरी तैयार करके इन tables के विषय में जानकारी तथा परिभाषिक शब्दामें का विवरण प्रस्तुत करता है। जो आगे सदैव Database management में important role play करती है।

2) इनपुट कन्सट्रेक्ट तथा नियम निर्धारित करना: - Database design के समय सभी आवश्यक input construct, data input के नियम तथा data entry operator के लिए सभी आवश्यक निर्देश निर्धारित करना काफी महत्वपूर्ण होता है। ताकि database गलत data input को स्वीकार ना करे।

3)Database की भैतिक संरचना में बदलाव: - समय-2 पर संस्था नियम बदलते रहते हैं जिनके अनंसार database की संरचना में भी आवश्यक परिवर्तन करना अनिवार्य होता है। Database की संरचना में ये परिवर्तन भी DBA ही करता है।

4) Data की उपलब्धता को सुनि चत करना- DBA का एक अन्य कार्य यह सुनिश्चित करने का भी है कि सभी अधिकारिक users को आवश्यक data and information सरलता से उपलब्ध रहे।

5) Access method निर्धारित करना: - DBA का कार्य सही data accessing technique and process निर्धारित करना भी है ताकि data कम से कम समय में access किया जा सके तथा resources का भी पूरा उपयोग हो सके।

6) Users के access अधिकार निर्धारित करना: - DBA का सबसे महत्वपूर्ण कार्य यह निर्धारित करना होता है कि किस यूजर को कौन से data item access करने की अनुमति होगी और कौन से नहीं। ये Access Authority "Grant" कहलाते हैं। DBA, DCL भाषा की सहायता सेये access अधिकार निर्धारित करता है। इस कार्य के अन्तर्गत access के प्रकारों का निर्धारण भी आता है अर्थात् एक यूजर डाटा को केवल पढ़ सकता है या उसमें नया डाटा जोडने और पुराने डाटा में परिवर्तन का भी कार्य कर सकता है। यह निर्धारण Access प्रकार कहलाता है।

7) Data की सुरक्षा सुनि चत करना- DBA का सबसे महत्वपूर्ण कार्य database को अनाधिकारिक users से बचाना होता है, इसके लिए password निर्धारण तथा DCL commands का उपयोग किया जाता है।

8) Databse Recovery: - समस्त सावधानियों के बावजूद database में error आने पर या किसी तकनीकी खराबी के कारण system फेलियर की स्थिति में DBA का मुख्य कार्य डाटा की पुनः प्राप्ति करना होता है।

Que: - Write the Advantages and Dis-Advantages of Distributed Database Syatem.

Ans: - <u>Advantages</u>: -

1) Sharing: - एक यूजर अन्य यूजर का डाटा शेयर कर सकता है।

2) Reliability and Ability: - Distributed System में हम प्रत्येक site का data back-up के रुप में अन्य site पर भी रख सकते हैं, इसका मुख्य लाभ यह है कि अगर कोई site किसी तकनीकी खराबी के कारण बंद भी हो जाती है तो भी उसका data अन्य site पर उपलब्ध copy से प्राप्त किया जा सकता है। इससे system की विश्वसनीयता बढ़ती है तथा डाटा हमेशा उपलब्ध रहता है।

3) Incremental Growth: - जैसे-2 संस्था विकास करती है हम आसानी से केन्द्रित system में नई sites जोड सकते हैं।

4) Parallel Evaluation: - अगर किसी Query में एक से अधि sites से data की आवश्यकता है, तो हम Query को अलग-2 Sub-Query में बांटकर अलग-2 sites के data प्राप्त कर सकते हैं और प्राप्त data का समानान्तर मूल्यांकन कर सकते हैं।

<u>Dis-advantages</u>: -

1) Complexity in Control and Management: - जैसे-2 distributed database system में साइट्स की संख्या बढ़ती जाती है, नियंत्रण एवं प्रबंधन गतिविधियां अधिक जटिल होती जाती हैं।

2) Software Cost: - Distributed Database System को लागू करना अधिक मुश्किल कार्य है, इसमें centralized database system की तुलना में software लागत अधिक आती है।

3) Higher Possibility of Error: - जब अनेक sites distributed system का समानान्तर रुप से उपयोग करती हैं, तो गलतियों तथा system में error आने की सम्भावना अधिक होती है।

4) More Overhead: - Data and Messgaes के आदान-प्रदान तथा साइट्स के बीच आपसी तालमेल स्थापित करने में System का कार्य बहुत बढ़ जाता है, जो system को केन्द्रीय सिस्टम में नहीं करना पडता।

5) Security: - Distributed System में सुरक्षा सम्बन्धी जटिलताएं और आवश्यकताएं बहुत अधिक होती हैं।

<u>ODBC:</u> - ODBS का अर्थ Open Database Connectivity होता है। यह एक विशेष application program interface होता है, जो एक ही application द्वारा एक से अधिक DBMS access करने की सुविधा देता है, अर्थात् ODBC के माध्यम से हम किसी application में अलग-अलग DBMS के data access कर सकते हैं। मान लीजिए हम एक banking application software develop कर रहें हैं जिनमें कुछ data files ऑरेकल से तथा कुछ files SQL 2000 से access करने की आवश्यकता है तो यह कार्य हम ODBC के माध्यम से कर सकते हैं। यह एक open database connectivity सुविधा है, अर्थात् हम इसके द्वारा खुले रुप से किसी भी बाहरी database को अपनी application से जोड़ सकते हैं। इसके लिए ODBC आवश्यक interface प्रदान करता है।

ODBC के निम्नलिखित चार main component होते हैं–

- Application
- Driver Manager
- Driver

- Data Source

1) Application के माध्यम से user आवश्यक command देता है। ये command SQL statement के माध्यम से दिए जाते हैं।

2) Driver manager application के माध्यम से दिए गए command को perform करवाने के लिए उपयुक्त driver component को application में load करना।

3) Driver Component data source का application से संपर्क स्थापित करता है। यह application द्वारा दिए गए निर्देश को DBMS तक पहुँचाता है।

4) Data source data files का संग्रह होता है जो driver के माध्यम से प्राप्त command के अनुसार data खोज कर उसे driver के माध्यम से return करता है।

DSN: - DSN का अर्थ database source names होता है। ये database source के नाम होते हैं जिनका उपयोग database से connection स्थापित करने के लिए किया जाता है।

जैसा कि हम जानते हैं ODBC के माध्यम से हम किसी भी outer databse को application से जोड़ सकते है, इसके लिए हमें application में उस data source का नाम देना आवश्यक है जिसे हम application से जोडना चाहते हैं। यह source का नाम ही DSN कहलाता है।

DSN तीन प्रकार के होते हैं–

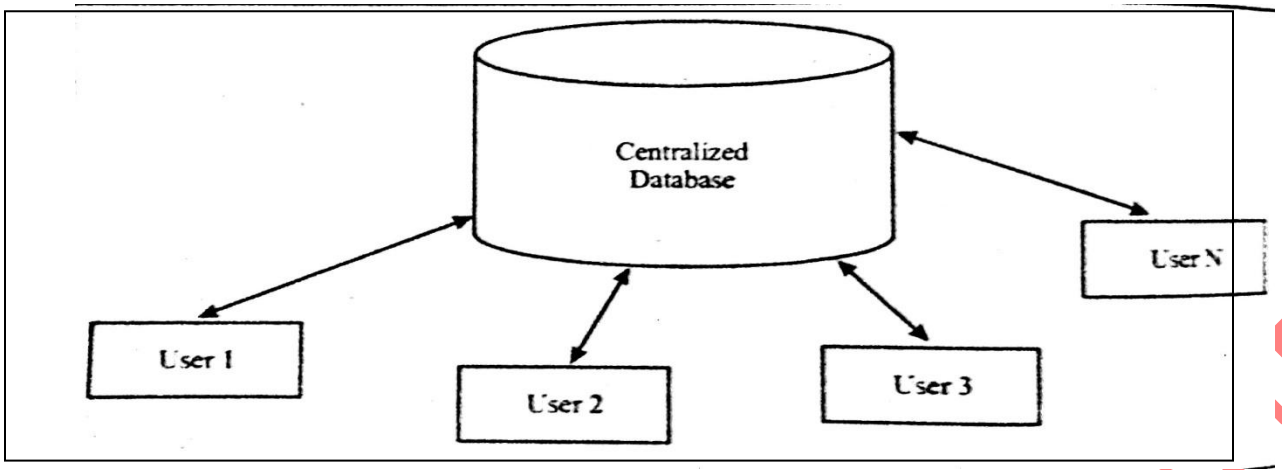
- System DSN
- User DSN
- File DSN

System DSN and User DSN उस system and user का नाम बतलाते हैं जहां से file प्राप्त करना है। File DSN उस file का नाम बतलाता है जिसमें data store है।

Centralized Database Syatem: - किसी भी संस्था में जहां अनेक विभाग हों तथा विभाग computerized रुप से अपना data management and processing का कार्य करते हों वहां अगर सभी विभाग स्वतंत्र रुप से अपने-2 data का प्रबन्धन करे तो इसमें अनेक कठिनाइयां हैं; जैसे–

- प्रत्येक विभाग को अपना अलग data entry operator रखना होगा।
- प्रत्येक विभाग को data store करने के लिए अलग-2 storage medium, database software आदि hardware तथा software की आवश्यकता होगी।
- स्वभाविक रुप से एक ही तरह की जानकारी एक से अधिक स्थानों पर उपलब्ध होगी जिसके कारण अनावश्यक रुप से सूचनाओं का दोहराव होगा।
- किसी data को update करते समय अगर वह एक स्थान पर हो तो update कर दिया जाता है लेकिन गलती से अन्य स्थानों पर update नहीं हो पाता तो एक ही विषय में अलग-2 department भिन्न-2 information देगें जिससे data की विश्वसनीयता समाप्त हो जाएगी।

इसकी अपेक्षा अगर संस्था का समस्त data केन्द्रीय रुप से एक ही स्थान पर रखा जाए जिसका DBMS software द्वारा व्यवस्थित प्रबन्धन हो तथा network के माध्यम से सभी user उसे share कर सके। तो यह व्यवस्था centralized database system कहलाती है। इसे हम निम्न चित्र के माध्यम से समझ सकते हैं–



Importance of Centralized database system: -

1) डाटा की पुनरावृति ना होना– समस्त data centralized form से एक ही सथान पर रखे जाने के कारण data का अनावश्यक दोहराव नहीं होता जिससे समय और संसाधनों की काफी बचत होती है।

2) भोयरिंग की सुविधा (Sharing facility): - Centralized database system नियंत्रित रुप से data को share करने की सुविधा प्रदान करता है।

3) Data independency: - Database application program से स्वतंत्र होता है अर्थात् हमें database पर परिवर्तन करने पर application program में या application program में परिवर्तन करने पर database में बदलाव की आवश्यकता नहीं होती।

3) उन्नत इन्टेग्रिटी (Improved Integrity): - इसमें data की विश्वसनीयता तथा एकरुपता काफी अधिक होती है।

4) उन्नत सुरक्षा (Improved Security): - इसमें हर स्तर पर password and user access अधिकारियों के कारण data की सुरक्षा काफी उच्च स्तर की होती है।

5) उन्नत बैकअप एवं रिकवरी (Imporooved Backup and Recovery): - Centralized database में नियमित रुप से backup process अपनाई जाती है जिसके कारण किसी भी आवंछित स्थिति, जहां तक कि software फेल्योर की स्थिति में भी data recovery सम्भव है।

उपरोक्त समस्त सुविधाओं एवं लाभों को देखते हुए database एक महत्वपूर्ण सुविधा है।