- Introduction to Data Base Management System :

- Data :-

Data is the plural of datum, a single smallest piece. However, people use data as both the singular and plural forms of the word.
The terms data, information and knowledge are frequently used for overlapping concepts. The main difference is the level of abstraction being considered. Data is the lowest level of the abstraction, information is the next level, and finally, knowledge is the highest level among all three. Data on its own carries no meaning. For data to become information, it must be interpreted and take on a meaning.

- System :-

A set of detailed methods, procedures and routines created to carry out a specific activity, perform a duty, or solve a problem.
An organized, purposeful structure that consists of inter-related and inter-dependent elements (components, entities, factors, members, parts, etc.). These elements continually influence one another (directly or indirectly) to maintain their activity and the existance

of the system, in order to achieve the goal of system.

All systems have —

(a) inputs, outputs and feedback mechanisms,
(b) maintain an internal steady-state despite a changing external environment,
(c) display properties that are different than the whole but are not possessed by any of the individual elements, and
(d) have boundaries that are usually defined by the system observer.

- Data Base System :-

A data base system is a way of organizing information on a computer, implemented by a set of computer programs. This kind of organization should offer →

→ Simplicity : an easy way to collect, access-connect, and display information.

→ Stability : to prevent unnecessary loss of data.

→ Security : to protect against unauthorized access to private data.

→ Speed : for fast results in a short time, and for easy maintenance in case of changes in the organization that stores the information.

- A data base system is a term that is typically used to encapsulate the constructs of a data model, dbms and data base.

- A, DBMS, is a suit of computer software providing the interface between users and a database or databases.

- Overall, a DBMS, is a set of programs that enables us to store, modify and extract information from a database. It also provides users with tools to add, delete, access, modify and analyze data stored in one location.

- A group can access the data by using query & reporting tools that are part of the DBMS. It also provides the method for maintaining the integrity of stored data, running security and users access, and recovering information if the system fails.

- Need of DBMS :-

DBMS manages concurrent access to data in a predictable, repeatable and controlled manner for multiple end-users.

For ex - banking, airline systems, etc.

It is an important component of making data available on every device across the business.

A DBMS can make the same data available to multiple applications, and enables the sharing of customer data across order entry, invoicing & accounts receivable.

A DBMS creates back up data copies for disaster recovery. Data can be quickly recovered and operations restored, after a fire or a data management error.

- Real World Scenarios :-

To say that the databases are everywhere would be an understatement. They virtually permeate our lives: Online stores, health care providers, clubs, libraries, video stores, beauty salons, travel agencies, phone companies, government companies or agencies like FBI, INS, IRS and NASA - they all use databases. These databases can be very different in their nature and usually have to be specifically designed to cater to some special customer needs.

• Introduction to File System :

Traditional file processing system or simple file processing system, refers to the first computer based approach of handling the commercial or business applications. That is why, it is also called a replacement of the manual file system. Before we use computers, the data in the offices or business was maintained in the files, considered in the pre-computer age. It was laborious, time consuming, in-efficient, especially in case of large organizations. Computers initially designed for the engineering purposes, but they also helped efficient management but file processing environment simply transformed manual file work to computers.

• Drawbacks of File System :-

DBMS is needed because file system approach is having many drawbacks, like:

1. **Data Security :-**
   The data stored can be easily accessible and hence not secured.
2. **Data Redundancy :-**
   Information may get duplicated, and also may lead to data inconsistency.
3. **Data Isolation :-**
   It means that all the related data is not available in one file.

4. Data Dependence :- Application programs are closely dependent on the files in which data is stored, but there is no centralized execution of the data management functions.

5. Lack of flexibility :- Traditional systems are able to retrieve information for predetermined request for data.

6. Concurrent Access Anomalies :- Concurrent updates may result in inconsistant data.

- File System v/s DBMS :- A database is generally used for storing related, structured data, with well-defined data formats in an efficient manner for insert, update & retrieval. On the other hand, a file system is a more unstructured data store for storing arbitrary, probably unrelated data.

→ The file system is more general and databases are built on top of the general data storage services provided by the file system.

→ Files act locally whereas DBMS saves directly in a database.

→ File saves in temporary locations whereas DBMS saves in well arranged & permanent database locations.

→ Transactions are not possible in file system whereas in DBMS insert, update, delete, view, etc. are possible.
→ Data can be accessed through files, but in DBMS tables are used.
→ A file manager is used to store all relationships in file system and in DBMS structural tables are used.
→ Data in DBMS is more secure than in files.

Purpose of DBMS :

- Features of DBMS :

Various features are offered by DBMS, and some of them are :

1. Query ability →

A database query language & report writers allow users to interactively interrogate the database, analyze its data & update it according to users privileges on data.

2. Backup and replication →

A periodic copy of attributes and replication of data between sowers is done by DBMS.

3. Rule enforcement →

Rules can be applied by policies.

4. Security :→
Data is secured by defining users & their roles.

5. Computation :→
DBMS can supply calculations such as counting, summing, averaging, sorting, grouping, cross-referencing, etc.

6. Change & access logging :→
Logging services allows record of access occurrences.

7. Automated optimization :→
DBMS can adjust themselves to improve the speed of frequent requests.

- Applications of DBMS in Real World :

Some real world applications of DBMS are -

- Universities
- Banking & finance
- Airlines
- Telecommunications
- Enterprises
- Genetics
- Stock Market, etc.

- Goals of DBMS :

The primary goal of a DBMS is to provide an environment, that is, both convenient and efficient for people to use in retrieving and storing information.

- Advantages of DBMS:

DBMS has many advantages, like,

1. Data independence →

The DBMS can provide an abstract view of the data to insulate application code from such details.

2. Efficient data access →

A DBMS utilizes a variety of sophisticated techniques to store and retrieve data very efficiently.

3. Data integrity and security →

The DBMS can enforce integrity constraints on data and access controls that govern what data is visible to different classes of users.

4. Data administration →

When several users share the data, centralizing the administration of data can offer better organization of data representation to minimize redundancy and make retrieval efficient.

5. Concurrent access and crash recovery →

A DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time.

6. Reduced application development time →

The DBMS supports many important functions that are common to many applications accessing data stored in the DBMS.

Conclusion of the lecture →

In this lecture, we have concluded the introduction of data, system & DBMS along with need, features and advantages of DBMS over file system.

Introduction to lecture :-

In this lecture, we will discuss about the concept of physical & logical database,-

- Physical View of database
- Logical View of database
- Data Abstraction
  - Physical level
  - Logical level
  - View level
- Data Independence
  - Logical level
  - Physical level.

Concept of Physical and Logical Database :-

Models are created to visually represent the proposed database so that business requirements can easily be associated with database objects to ensure that all requirements have been completely and accurately gathered. Different types of diagrams are typically produced to illustrate the business processes, rules, entities and organizational units that have been identified. These diagrams often include ER diagrams, process flow diagrams and server model diagrams.

After all business requirements have been gathered for a proposed database, they must be modeled. Basically, data modeling serves as a link between business needs & system requirements.

Two types of modeling are as follows:-

1. — Logical Modeling
2. — Physical Modeling

1. Logical Modeling :-

It deals with gathering business requirements & converting those requirements into a model. The logical model revolves around the needs of the business, not the database, although the needs of the business are used to establish the needs of the database.

Logical modeling involves gathering information about business processes, business entities (categories of data) and organizational units. After this information is gathered, diagrams and reports are produced including ER diagrams, business process diagrams, and eventually process flow diagrams. The diagrams produced should show the processes and data that exists, as well as the relationships between business processes and data. Logical modeling should accurately render a visual representation of the activities and data relevant to a particular business.

* Logical modeling affects not only the direction of database design, but also indirectly affects the performance & administration of an implemented database.

The diagrams & documentation generated during logical modeling is used to determine whether the requirements of the business have been completely gathered.

Typical deliverables of logical modeling include-
→ Entity relationship diagrams
→ Business process diagrams
→ User feedback documentation.

2. Physical Modeling :-

Physical modeling involves the actual design of a database according to the requirements that were established during logical modeling.

Physical modeling deals with the conversion of the logical, or business model, into a relational database model. When physical modeling occurs, objects are being defined at the schema level.

A schema is a group of related objects in a database. A database design effort is normally associated with one schema.

During physical modeling, objects such as tables & columns are created based on entities and attributes that were defined during logical modeling. Physical modeling is when all the pieces come together to complete the process of defining a database for a business.

Physical modeling is database software specific, meaning that the objects defined during physical modeling can vary depending on the relational database software being used.

The implementation of the physical model is dependent on the h/w & s/w being used by the company. The h/w can determine what type of s/w can be used because s/w is normally developed according to common h/w & OS platforms.

Typical deliverables of physical modeling include the following:

1) Server model diagrams -

This shows tables, columns & relationships within a database.

2) User feedback documentation -

3) Database design documentation.

Data Abstraction :-

Data abstraction is -

→ When the DBMS hides certain details of how data is stored and maintained, it provides what is called as the abstract view of data.

→ This is to simplify user-interaction with the system.

→ Complexity (of data and data structure) is hidden from users through several levels of abstraction.

Purposes of Data Abstraction :-

The main purposes of data abstraction are -

1. → To provide abstract view of data.
2. → To hide complexity from user.
3. → To simplify user interaction with DBMS.

Levels of Data Abstraction :-

There are three levels of data abstraction -

1. <u>Physical level</u> :- It describes how a record is stored. The features are as follows -

(a) Lowest level of abstraction.
(b) It describes how data are actually stored.
(c) It describes low-level complex data structures in detail.
(d) At this level, efficient algorithms to access data are defined.

2. Logical level :-

It describes what data stored in database, and the relationships among the data. The features are as follows -

(a) It is next-higher level of abstraction. Here whole database is divided into small simple structures.
(b) Users at this level need not be aware of the physical-level complexity used to implement the simple structures.
(c) Here the aim is case of use.
(d) Generally, database administrators (DBAs) work at logical level of abstration.

3. View level :-

Application programs hide details of data types. Views can also hide information for security purposes. The features are as follows -

(a). It is the highest level of abstraction.
(b) It describes only a part of the whole database

(c) This view hides all complexity.
(d) It exists only to simplify user interaction with system.
(e) The system may provide many views for the whole system.

Data Independence :-

Data independence is a form of database management that keeps data separated from all programs that make use of data. Data independence ensures that data can not be redefined or reorganized by any of the programs that make use of data.

It is the type of data transparency that matters for a centralized DBMS. It refers to the immunity of user applications to make changes in the definition & organization of data.

Data independence is normally thought of in terms of two levels or types -

(1) Logical data independence :-

It makes it possible to change the structure of the data independently of modifying the applications or programs that make use of data. There is no need to rewrite

current applications as part of the process of adding to or removing data from the system.

(2) → Physical data independence :- This is the second type of data independence. This approach has to do with altering the organization or storage procedures related to the data, rather than modifying the data itself. This is present in most databases and file environment in which hardware storage of encoding, exact location of data on disk, merging of records, so on this are hidden from user.

Conclusion of lecture :-

In this lecture, we have concluded the concept of physical & logical database, data abstraction & its 3 levels and data independence & its types.

Introduction to the lecture :

In this lecture, we will discuss about the various data base architectures including -

→ 1-tier,
→ 2-tier,
→ 3-tier, &
→ client-server architecture.

- DBMS Architecture :-

Database architecture may be viewed as an extent of data modeling. It is used to implement different requirements of different end-users.

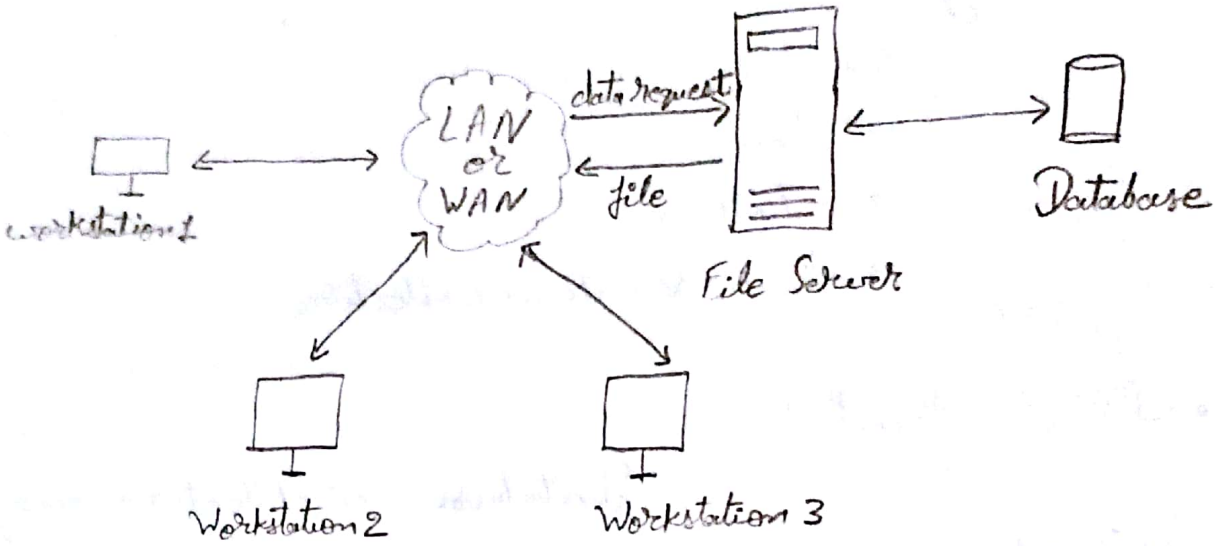
There is a wide variety of different DBMS architectures -

1. File - Server Architecture
2. Two-tier Client-Server Architecture
3. Three-tier Client-Server Architecture
4. N-tier Architecture
5. Peer-to-Peer Architecture
6. Distributed DBMS
7. Service-Oriented Architecture
8. Cloud Architecture.

1. File - Server Architecture :-

→ A file - server is a computer that is connected to a network and mainly serves as a shared storage.

→ In a file - server architecture the processing is distributed over the network.



File - Server Architecture

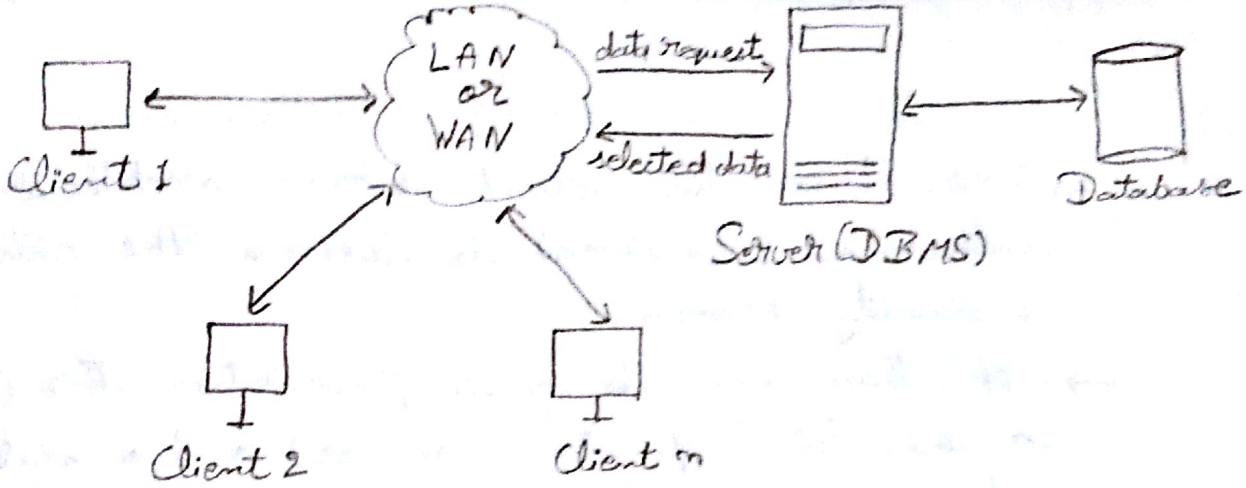
exc - ftp server

Disadvantages →

→ Heavey network traffic
→ High total cost of ownership (TCO)
→ Complex integrity, concurrency and recovery control

2. Two-tier Client - Server Architecture :-

→ Application consists of a client (first tier) and a server (second tier) that might run on different machines.
→ Supports decentralized business environments



Two-tier Client-Server Architecture

ex- mail

Client (first tier) tasks ⇒

- → presentation of data (user interface)
- → business and data application logic
- → send database requests to the server and process the results.

Server (second tier) tasks ⇒

- → manage (concurrent) database access (data servic
- → business logic (validation of data)

Advantages →

- → Increased Performance
- → Reduced communication costs
- → Reduced hardware costs
- → Increased consistency / security
- → Wider access to existing databases
- → Maps well to open system architectures

Disadvantages →
→ Limitations in terms of enterprise scalability with thousands of potential clients.

3. Three tier Client Server Architecture :—
→ The three tier client-server architecture was introduced (in 1990s) to address the enterprise scalability problem.
→ Application consists of a presentation tier (client), a logic tier (application server) and a data tier (database server) that might run on different platforms.

Presentation tier (first tier) tasks —
→ presentation of data (user interface)
→ basic input validation (thin client)
→ send requests to the server and visualize results

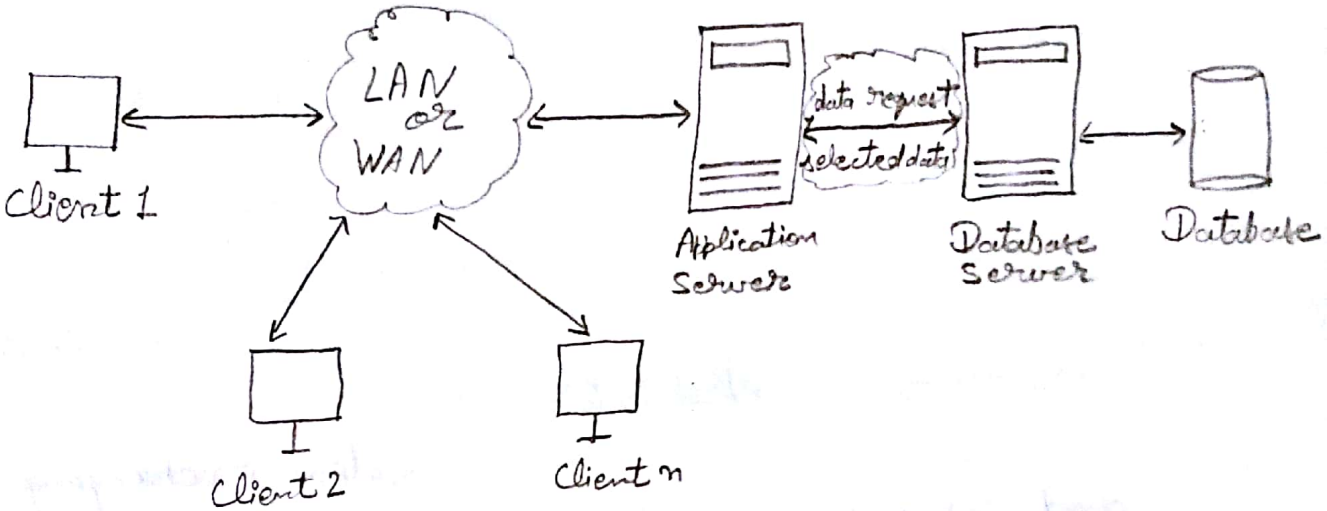
Logic tier (second tier / middle tier) tasks —
→ business logic
→ data processing logic

Data tier (third tier) tasks —
→ basic data validation
→ manage (concurrent) databases access (data services)

Advantages ⇒
→ reduced costs for thin clients due to lower resource requirements.
→ application logic is centralised in a single application server.

→ increased modularity
→ load balancing becomes easier with a clear separation between the core business logic and the database functionality

4.



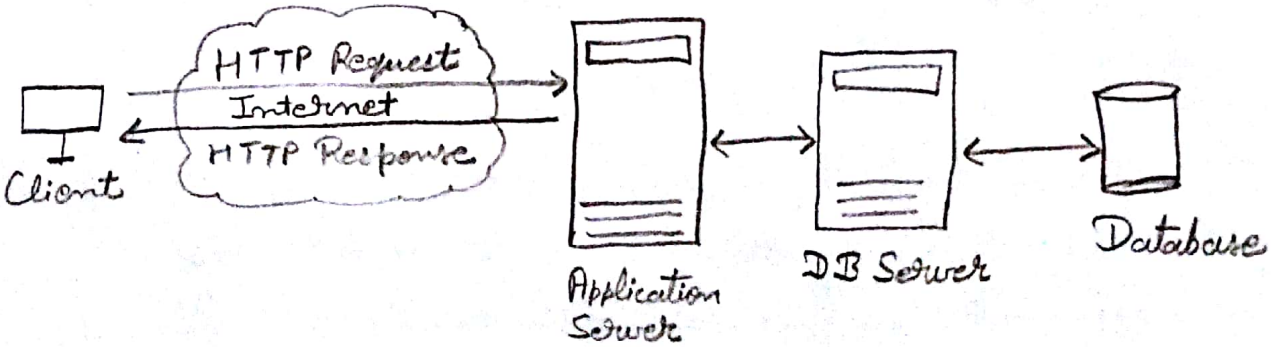
Three tier Client Server Architecture

ex-mail

4. Web Information System (WIS) :-

The three tier architecture maps very naturally to web environments.

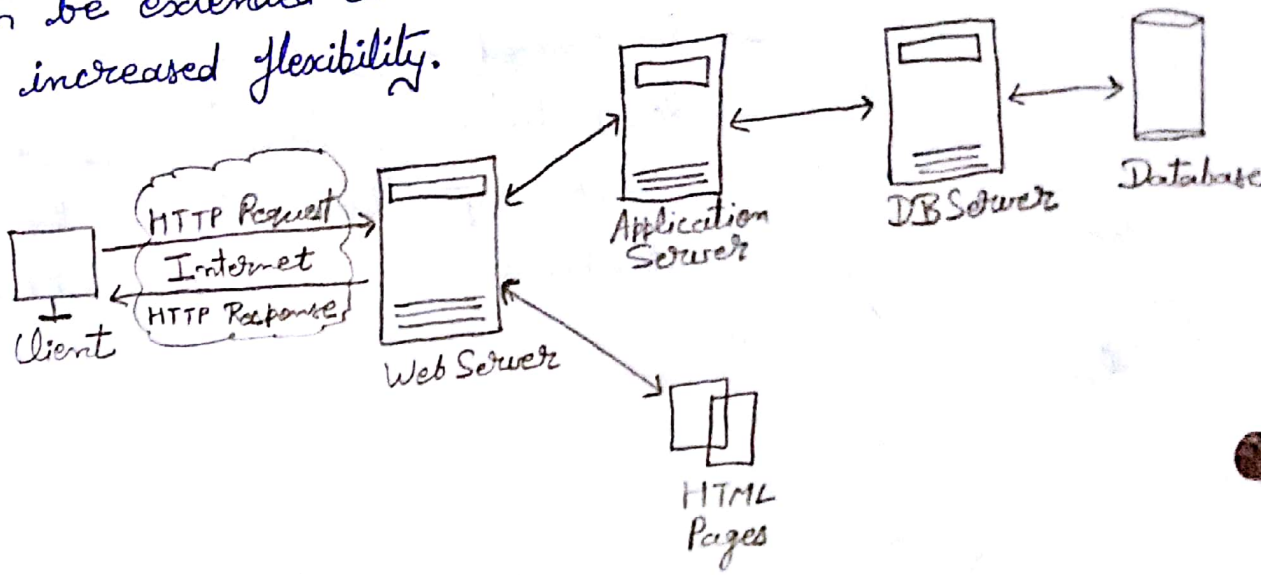
→ The move towards thin browser clients has dramatically reduced the costs for software deployment.



Web Information System

ex-mail

5. N Tier Architecture :- The three-tier architecture can be extended with additional intermediary tiers for increased flexibility.



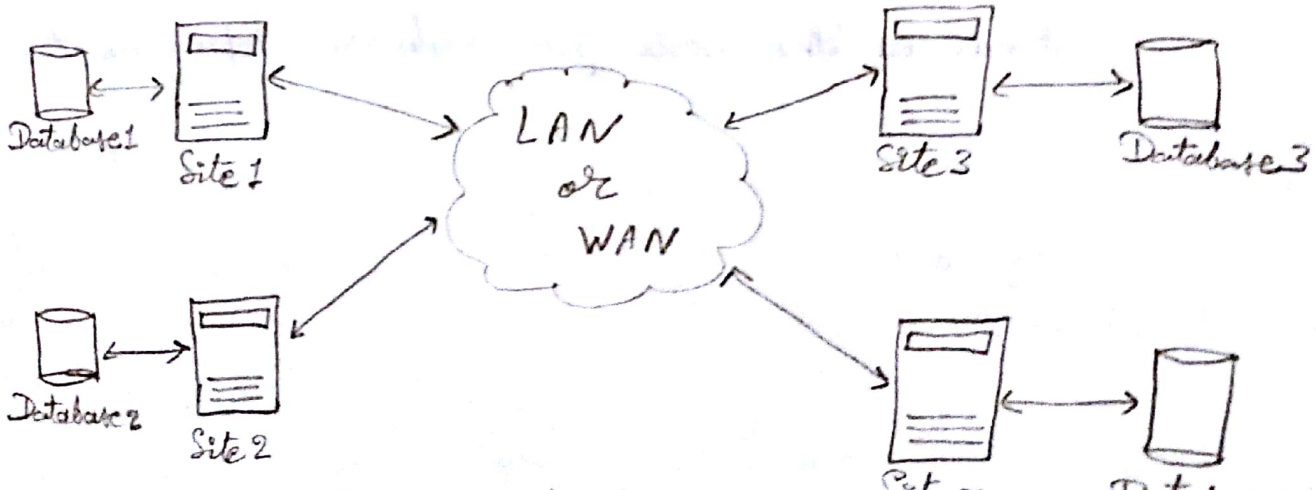
N Tier Architecture

ex - Google.

6. Peer-to-Peer Architecture :-

System exchanging information and services in a peer-to-peer like manner without a central authority (no global schema).

→ Data and service sharing
- no dedicated client & servers.
- sites may dynamically form new client/server relationships.



Peer-to-Peer Architecture

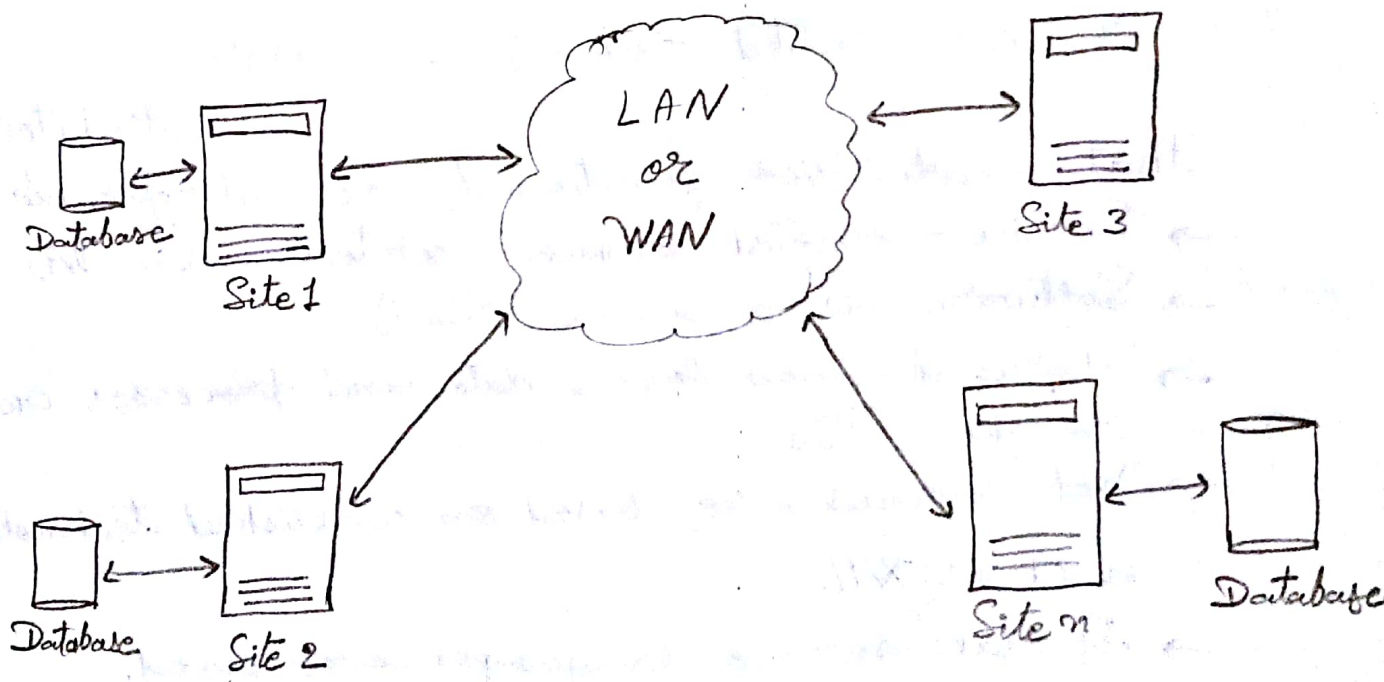
7. Distributed DBMS (DDBMS) :-

Distributed database ⇒

logically related collection of shared data and metadata that is distributed over a network.

Distributed DBMS ⇒

software system to manage the distributed database in a transparent way.



Distributed DBMS

→ Distinction between local and global transactions —

⇒ Local transaction : accesses only data from the site from which the transaction was initiated.

⇒ global transaction : accesses data from several different sites.

Advantages of Distributed DBMS ⇒

→ Data sharing
→ Autonomy
→ Availability
→ Costs and scalability
→ Integration of existing DBMS
→ Dynamic organisational structure

8. Service - Oriented Architectures (SOA) :-

Architecture that modularises functionality as interoperable services.

→ Service - oriented database architecture (SODA)
→ Software as a Service (SaaS)
→ Share business logic, data and processes via web service APIs.
→ Web services are based on established technologies such as XML.
→ Special service languages are used.

9. Cloud Computing :-

Internet based computing with on-demand and pay-per-use access to shared resources, data and software.

Advantages of Cloud Computing :-

→ web-based access (Web Service API or browser)
→ pay only for the services that are actually used (pay-per-use)
→ no initial investment required (for resources)

Cloud Services ⇒

→ Infrastructure as a Service (IaaS)
  - OS virtualization
  - Amazon EC2
→ Platform as a Service (PaaS)
  - provide platform to run applications
  - Google App Engine
→ Software as a Service (SaaS)
  - provide software as a service over the Internet
  - Google Docs.

Conclusion of the lecture ⇒

In this lecture, we have concluded the various types of DBMS architectures along with their advantages & uses.

GC

DETAILED LECTURE NOTES

DATE: 21-01-13

Name of Faculty: Richa Mehra College: PGI Dept: C.S

Name of Subject with Code: DBMS 4EC6.2 Branch.: EC Class: B.Tech - IV sem

Introduction to the lecture :-

In this lecture, we will firstly discuss about the views of data; secondly we will discuss the Data Base System structure.

Views –

- External
- Conceptual
- Internal

Data Base System Structure –

- Users
- Query Processor
- Storage Manager
- Database Administrator

Views ⇒

Views are divided into three levels, usually referred to as –

- the internal level
- the external level
- the conceptual level

The Internal Level ⇒

It is also known as the storage level. It is the one closest to physical storage, i.e., the one concerned with the way the

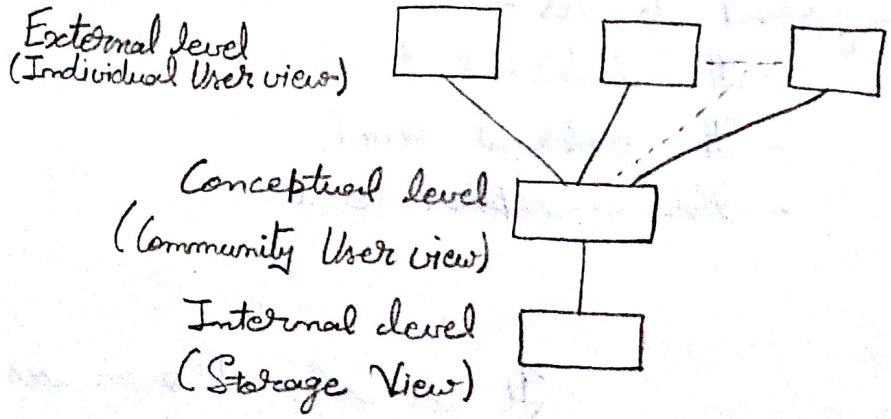
data is stored inside the system. It is a low level representation of entire database.

The External Level ⇒

It is also known as the user logical level. It is the one closest to the users, i.e., the one concerned with the way the data is seen by individual users, so the level is the individual user level. A user can be an application programmer or an end user of any degree of sophistication.

The Conceptual Level ⇒

It is also known as the community logical level or just the logical level. It is a level of indirection between the other two. The conceptual view is a representation of the entire information content of the database in a form that is somehow or somewhat abstract in comparision with the way in which the data is physically stored.



Three Levels of Views

Example –

(in C)

External –

```
Struct employee {
    char empno(6);
    int salary;
}
```

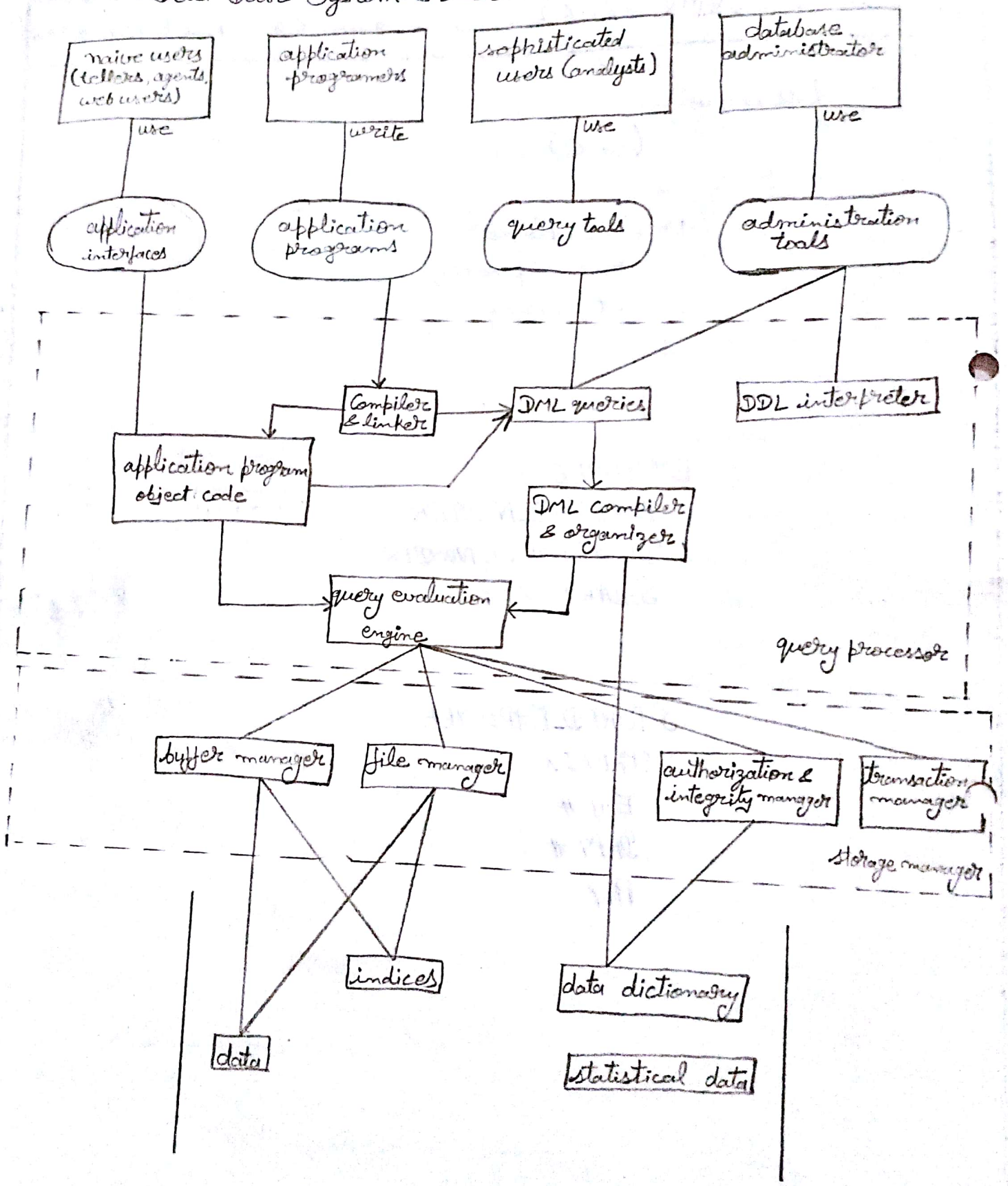
Conceptual –

EMPLOYEE
EMPLOYEE_NUMBER
DEPARTMENT_NUMBER
SALARY

Internal –

STORED_EMPLOYEE
PREFIX
EMP#
DEPT#
PAY

Data Base System Structure :



Data Base System Structure

1. Users :

There are 4 different types of database - system users, differentiated by the way they expect to interact with the system. Different types of user interfaces have been designed for the different types of users.

(1) Naive Users :-

Naive users are unsophisticated users who interact with the system by invoking one of the application programs that have been specified for the users.

For ex - a bank teller who needs to transfer $50 from account A to account B invokes a program called transfer. This program asks the teller for the amount of money to be transferred, the account from which the money is to be transferred, and the account to which the money is to be transferred.

The typical user interface for naive users is a forms interface, where the user can fill in appropriate fields of the form. They also can simply read reports generated from the database.

(2) Application Programmers :-

Application programmers are computer professionals who write application programs.

Application programmers can choose from many tools to develop user interfaces.
For Ex - Rapid Application Development (RAD) tools are tools that enable an application programmer to construct forms & reports without writing a program.

(3) Sophisticated Users :-

They interact with the system without writing programs. Instead, they form their requests in a database query language.

Analysts who submit queries to explore data in the database fall in this category.

For ex - Online analytical processing (OLAP) tools simplify analysts task by letting them view summaries of data in different ways.

(4) Specialized Users :-

These are those sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework.

For ex - computer-aided design systems, knowledge base & expert systems, etc.

2. Database Administrator :

A person who has central control, of both the data & the programs, is called a Data Base Administrator (DBA).

The functions of a DBA include:

(1) Schema definition: The DBA creates the original database schema by executing a set of definition statements in the DDL.

(2) Storage structure and access method definition

(3) Schema and physical-organization modification: The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.

(4) Granting of authorization for data access: By granting different types of authorization, the database administrator can regulate which parts of the database various users can access.

(5) Routine Maintenance: Database administrator's routine maintenance activities are -

- Periodically backing up the database
- Upgrading disk space as required
- Monitoring jobs running on the database

3. Query Processor : The query processor components include –

(1) DDL interpreter – It interprets DDL statements and records the definitions in the data dictionary.

(2) DML compiler – It translates DML statements in a query language into an evalution plan consisting of low-level instructions. The DML compiler also performs query optimization.

(3) Query evaluation engine – It executes low-level instructions generated by the DML compiler.

4. Storage Manager :

A storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

The storage manager is responsible for the interaction with the file manager.

The storage manager translates the various DML statements into low-level file system commands. So, the storage manager is responsible for storing, retrieving, and updating data in the database.

The storage manager components include –

(1) Authorization and integrity manager – It tests for the

satisfaction of integrity constraints and checks the authority of users to access data.

(2) Transaction Manager – It ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.

(4) File Manager – It manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

(5) Buffer Manager – It is responsible for fetching data from disk storage into main memory and deciding what data to cache in main memory. It enables the database to handle data sizes that are much larger than the size of main memory.

The storage manager implements several data structures as part of the physical system implementation :

(1) Data files which store the database itself
(2) Data dictionary which stores metadata about the

structure of the database (or schema).

(3) Indices which provides fast access to data items that hold particular values.

Conclusion of the lecture:

In this lecture, we have concluded the three types of views, i.e, internal, conceptual, & external along with the database system structure comprises of types of users, administrator (DBA), query processor and storage manager.

Atomicity. In a transaction involving two or more discrete pieces of information, either all of the pieces are committed or none are.

Consistency. A transaction either creates a new and valid state of data, or, if any failure occurs, returns all data to its state before the transaction was started.

Isolation. A transaction in process and not yet committed must remain isolated from any other transaction.

Durability. Committed data is saved by the system such that, even in the event of a failure and system restart, the data is available in its correct state.

Query Processor:

As query is very much necessary to find out only the data user need from tons of data of the database, query processor is very important to process these query requests. Query processors come with the following components,

1. **DDL Interpreter:** It interprets the DDL statements and records the definitions in data dictionary.
2. **DML Compiler:** It translates the DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation understands. It also performs query optimization which actually picks up the lowest cost evaluation plan from various alternatives.
3. **Query Evaluation Engine:** It executes the low level instruction compiled by the DML compiler.

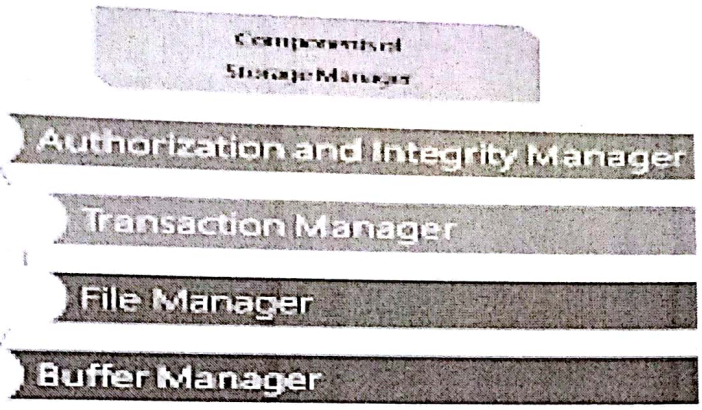
DML Precompiler

All the DBMS have two basic sets of Languages - Data Definition Language (DDL) that have the set of commands needed to define the format of the data that is being stored and Data Manipulation Language (DML) which tells the set of commands that modify, process data to make user definable output. The DML statements can as well be written in an application program. The DML precompiler changes DML statements (such as SELECT...FROM in Structured Query Language (SQL)) embedded in an application program to normal procedural calls in the host language. The precompiler relate with the query processor in order to produce the appropriate code.

Storage Manager:

A storage manager is a program module which is responsible for storing, retrieving and updating data in the database.

Following are the components of the storage manager;



1. **Authorization and Integrity Manager:** It tests the integrity constraints and checks the authorization of users to access data.
2. **Transaction Manager:** It ensures that no kind of change will be brought to the database until a transaction has been completed totally.
3. **File Manager:** It manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
4. **Buffer Manager:**

   It decides which data is in need to be cached in main memory and then fetch it up in main memory. This is very important as it defines the speed in which the database can be used.

Introduction to the Lecture :

In this lecture, we will discuss about E-R Model. This will cover -

- E-R Diagram
  - Entity
  - Attribute
  - Relationship
  - Symbols
  - Mapping Cardinalities
    - One : One
    - One : Many
    - Many : One
    - Many : Many

Entity :

An entity is an object that exists and is distinguishable from other objects.
For ex - specific person, company, etc.

Entity Set :

An entity set is a set of (attributes) entities of the same type that share the same properties.
For ex - set of all persons, companies

Attributes :

An entity is represented by a set of attributes,

That is descriptive properties possessed by all members of an entity set.

For Ex- customer = (customer-id, customer-name, customer-street, customer-city)

For each attribute, there is a set of permitted values, called the domain, or value set of that attribute.

The various types of attributes are -

(1) → Simple and composite attributes ⇒ They are not divided into subparts while composite one can be divided into subparts.

(2) → Single valued and multi-valued attributes ⇒ Single valued attributes can have a single value for a particular entity. While when an entity can has multiple attributes, it is known as the multi-valued attribute, like - a person can have 0, 1, or several phone numbers.

(3) → Derived attributes ⇒ The value for this type of attribute can be derived from the values of other related attributes or entities.

For ex- customer's age, which can be derived from date-of-birth attribute.

Relationship :

A relationship is an association among several entities.

Relationship Set :

A relationship set is a set of relationships of the same type. Formally, it is a mathematical relation on $n \geq 2$ (possibly non-distinct) entity sets.

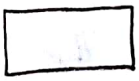
If $E_1, E_2, \ldots, E_n$ are entity sets, then a relationship set R is a subset of

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

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

Symbols :

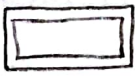
The symbols commonly used in E-R diagrams are –

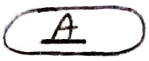
1. Rectangle  represents entity sets
2. Ellipse  represents attributes
3. Diamond  represents relationship sets
4. Lines — link attributes to entity sets & entity sets to relationship sets.

(5) Double ellipse  represent multivalued attributes

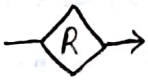
(6) Double lines  indicates total participation of an entity in a relationship set

(7) Dashed ellipse  denotes derived attributes

(8) Double rectangle  represents weak entity sets

(9) Primary key  represents primary key

(10) One-to-one relationship  denotes 1:1 relationship

(11) Many-to-one relationship  denotes M:1 relationship

(12) Many-to-many relationship  denotes M:M relationship

Mapping Cardinalities :

It express the number of entities to which another entity can be associated via a relationship. For binary relationship sets between entity sets A and B, the mapping cardinality must be one of :

(1) One-to-One →

An entity in A is associated with at most one entity in B, and an entity in B is associated with

at most one entity in A.

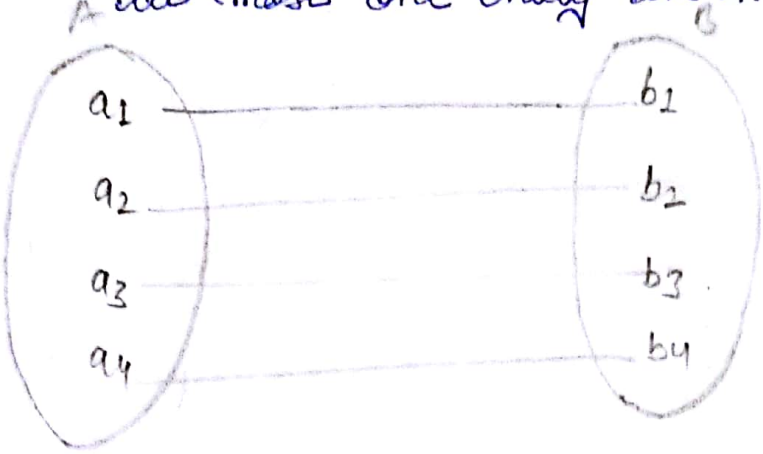


Fig: One-to-One

(2) One-to-Many →

An entity in A is associated with any number in B. An entity in B is associated with at most one entity in A.

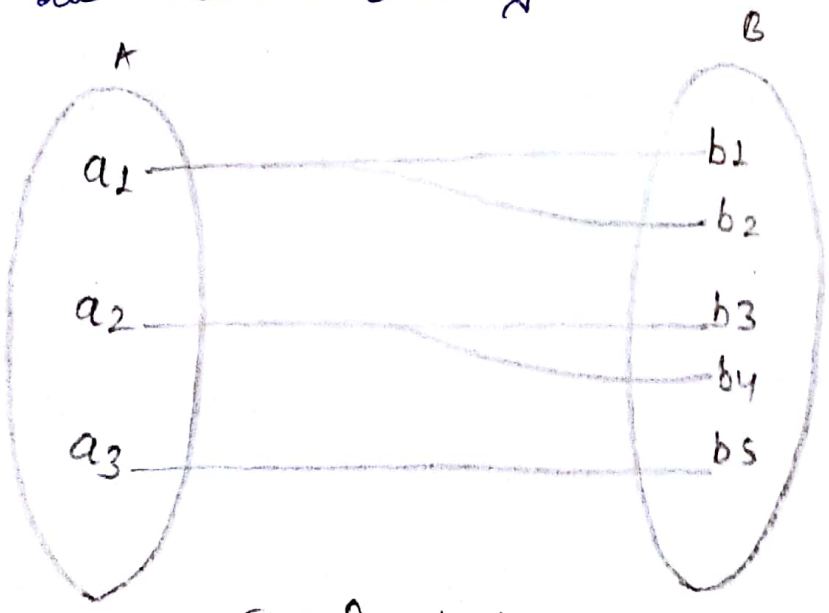


Fig: One-to-Many

(3) Many-to-One →

An entity in A is associated with at most one entity in B. An entity in B is associated

with any number in A.

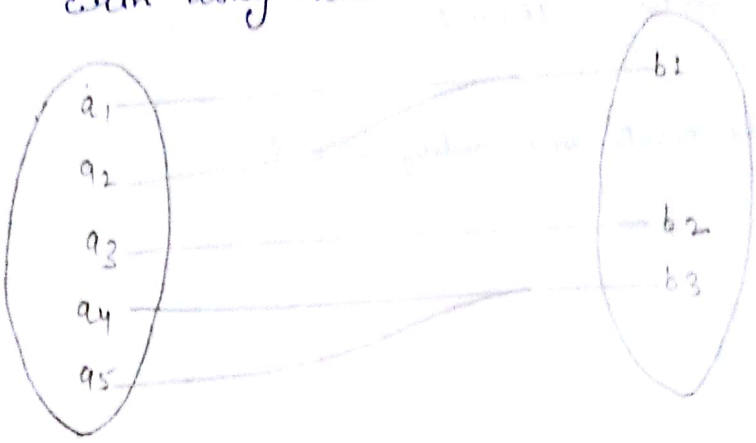


Fig: Many-to-one

(4) Many-to-Many → Entities in A and B are associated with any number from each other.

Fig: Many-to-Many

Conclusion of lecture :

In this lecture, we have concluded about Entity, entity sets, attributes, relationship, relationship sets, symbols, and mapping cardinalities like – 1:1, 1:M, M:1, M:M.

Introduction to lecture :- In this lecture, we will discuss about

- Use of entity sets or attributes
- Use of entity set & relationship set
- Binary v/s n-ary relationship set

Entity :- An entity is a real-world object that exists and is distinguishable from other objects. An entity may be concrete (a person, a book, etc) or abstract (holiday). A prg. lang. variable can be correspondent to an entity in E-R Model.

Entity Set :- An entity set is a set of entities of the same type.
Entity sets need not to be disjoint. For ex - the entity set employee (all employees of a bank) and

The entity set customer (all customers of the bank may have members in common.

Attributes :-

An attribute is a function which maps an entity set into a domain.

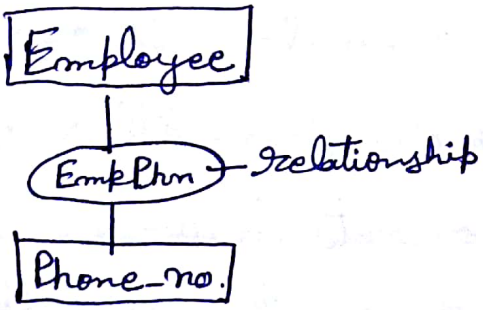
Relationships among entity sets can be defined in many ways.

Ex -

Employee (entity)
emp_name, phone_no. (attributes)

↓

Phone_no (entity)
country_code / Std Code, Phone_no. (attributes)



An employee can have several phone numbers.

DETAILED LECTURE NOTES

Degree of Relationship :- The degree of a relationship is the number of entity types that participates in the relationship.

It is called as the arity of relationship.

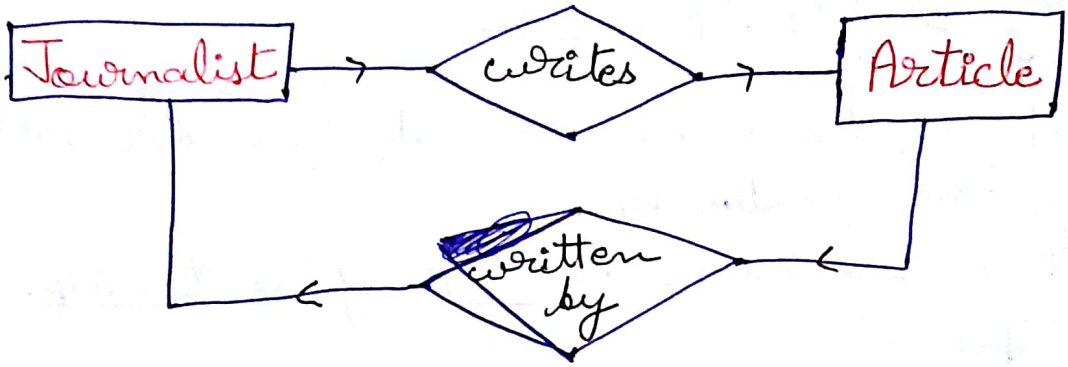
Types :-

1) Unary (one entity involved in relationship)
2) Binary (two
3) Ternary (three
4) N-ary (n

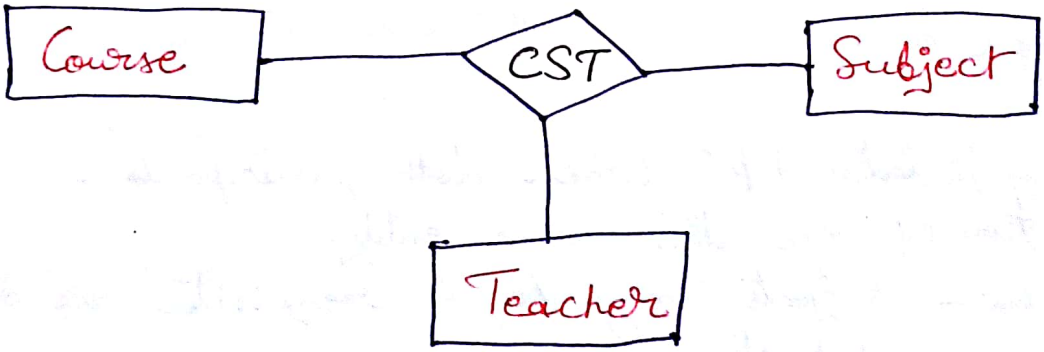
1) **Unary Relationship :** When both participants in the relationship are the same entity.

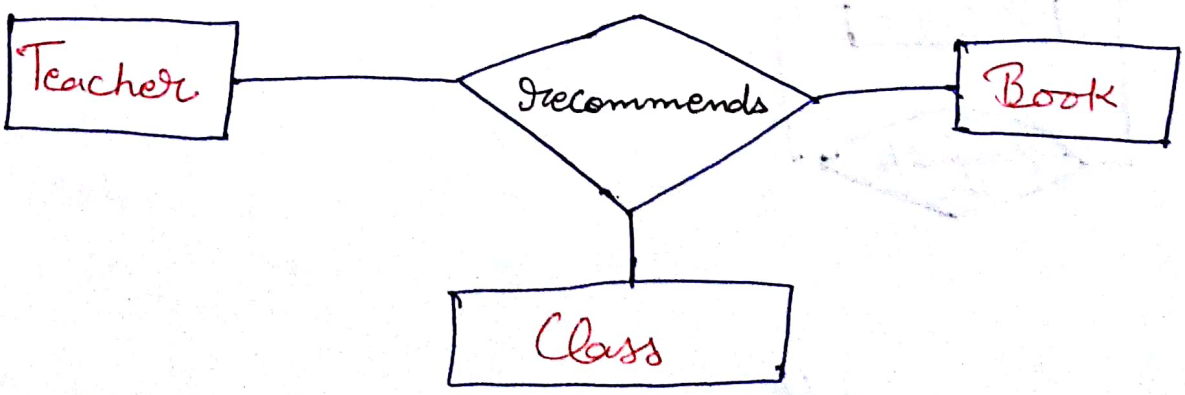
For ex - Subjects may be prerequisites for other subjects.

2) Binary Relationship : when two entities participate in the relationship.



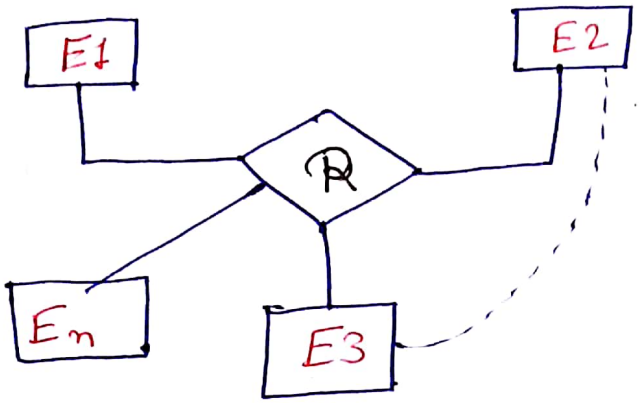
3) Ternary Relationships : when three entities participates in the relationship.





DETAILED LECTURE NOTES

4) N-ary Relationship : N no. of entities participated in the relationship.



Cardinality ⇒ It is the number of instances of entity B that can be associated with entity A.

Conclusion to lecture : In this lecture, we have discussed about use of entity sets, attributes & relationship sets, and degree of relationship.

Introduction to the lecture:

In this lecture, we will discuss about the concept of keys & the various types of keys, like,

- Primary key
- Secondary key
- Candidate key
- Composite key
- Super key
- Foreign key, etc.

Keys:

A key is a single or combination of multiple fields. Its purpose is to access or retrieve data rows from table according to the requirement. The keys are defined in tables to access or sequence the stored data quickly & smoothly. They are also used to create links between different tables.

Types of keys:-

(1) Primary key →

The attribute (or combination of attributes) that uniquely identifies a row or record in a relation is known as primary key. A primary key is the candidate key which is selected as the principle unique identifier.

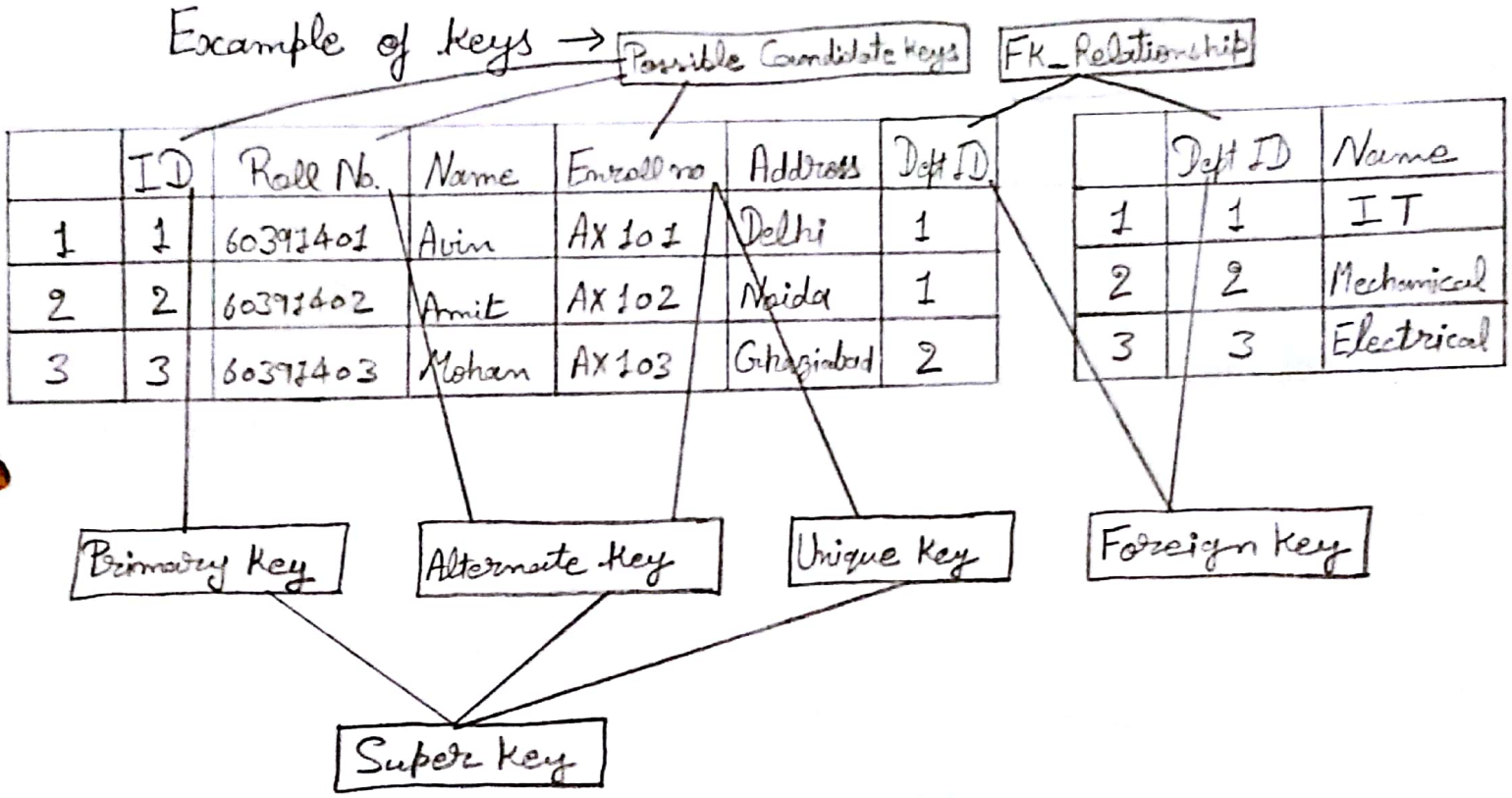
(2) Secondary key : A field that is basis for retrieval is known as secondary key. It is a non-unique field. One secondary key value may refer to many records.

(3) Candidate key : It is also termed as Alternate key. The fields or combination of fields that are not used as primary key are known as candidate key.
It is of two types, as,

(1) Composite key – It is also termed as Concatenate key. A primary key that consists of two or more attributes is known as composite key.

(2) Super key – A super key is any set of attributes that uniquely identifies a row. It can also be defined as a set of attributes of a relation (table) upon which all attributes of the relation are functionally dependent.

(4) Foreign key : A foreign key is an attribute or combination of attribute in a relation whose value match a primary key in another relation. The table in which foreign key is created is called as dependent table. The table to which foreign key is refers is known as parent table.



(5) Unique Key :- A column or combination of columns which can be used to uniquely identify a record in a table, it can have one NULL value.

Primary key can be considered a special case of unique key with a NOT NULL constraint.

Conclusion of the lecture →

In this lecture, we have concluded about keys & their various types, as, primary, secondary, candidate, composite, super & foreign key.

Introduction to lecture :- In this lecture, we will discuss about :
- weak entity set
- concept of discriminator
- Relationship

Weak Entity Set :-

The entity set that does not have sufficient attributes to form a primary key is cla weak entity set. While an entity set that has a primary key is cla Strong entity set.

Ex - In IT dept. -

* R1 : Class {Division Name, Classroom No., Class Teacher}
→ It is a strong entity, as by Division Name, any member can be uniquely identified, so it is Primary key.

* R2 : Student {Roll No, Name, Attendance}
→ It is a weak entity set as roll no of students in different divisions can be same.

Discriminator :

To resolve the problem of not having primary key in weak entity set, concept of the discriminator is introduced.

A <u>discriminator</u> of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.

It is also c/a <u>Partial Key</u>.

- The primary key of a weak entity set is formed by the primary key of the strong entity set (on which the weak entity set is existence depends) and the weak entity set's discriminator.

In previous example, if we combine "Roll No." with "Division Name", then any member of Student relation can be identified uniquely.

Hence "Roll No" is the Discriminator for Student relation.

Ex -

Payment {payment_number, payment_date, payment_amount}

- payment for various loan accounts may share same payment_number, so {loan_number, payment_number} acts as primary key.

Relationship:- The association among entities is cla Relationship.

For ex- an employee works_at a department, a student enrolls in a course

Relationship Set :- A set of relationships of similar type is cla Relationship set.

- Relationships can have attributes (similar to the entities) and are cla Descriptive attributes.

Conclusion to lecture :-

In this lecture, we have concluded about weak entity set, dominant & subordinate attributes, discriminator, relationship and relationship sets.

Extended Features :

Some of the extended E-R features are -

1. Generalization :-

The refinement from an initial entity set into successive levels of entity subgroupings represents a top-down design process in which distinctions are made explicit. The design process may also proceed in a bottom-up manner, in which multiple entity sets are synthesized into a higher-level entity set on the basis of common features.

For ex - The database designer may have first identified a customer entity set and the employee entity set with the attributes name, street, city, customer-id and name, street, city, emp-id, salary respectively.

There are Higher and lower level entity sets which also may be designated by the terms superclass & subclass. Here person entity set is the superclass of the customer and employee subclasses.

Generalization is a simple inversion of the specialization. It proceeds from the recognition that a number of entity sets share some common features. On the basis of their commonalities, generalization synthesizes these entity sets into a

single, higher-level entity set. Generalization is used to emphasize the similarities among lower-level entity sets and ~~the~~ to hide the differences; it also permits an economy of representation in that shared attributes are not repeated.

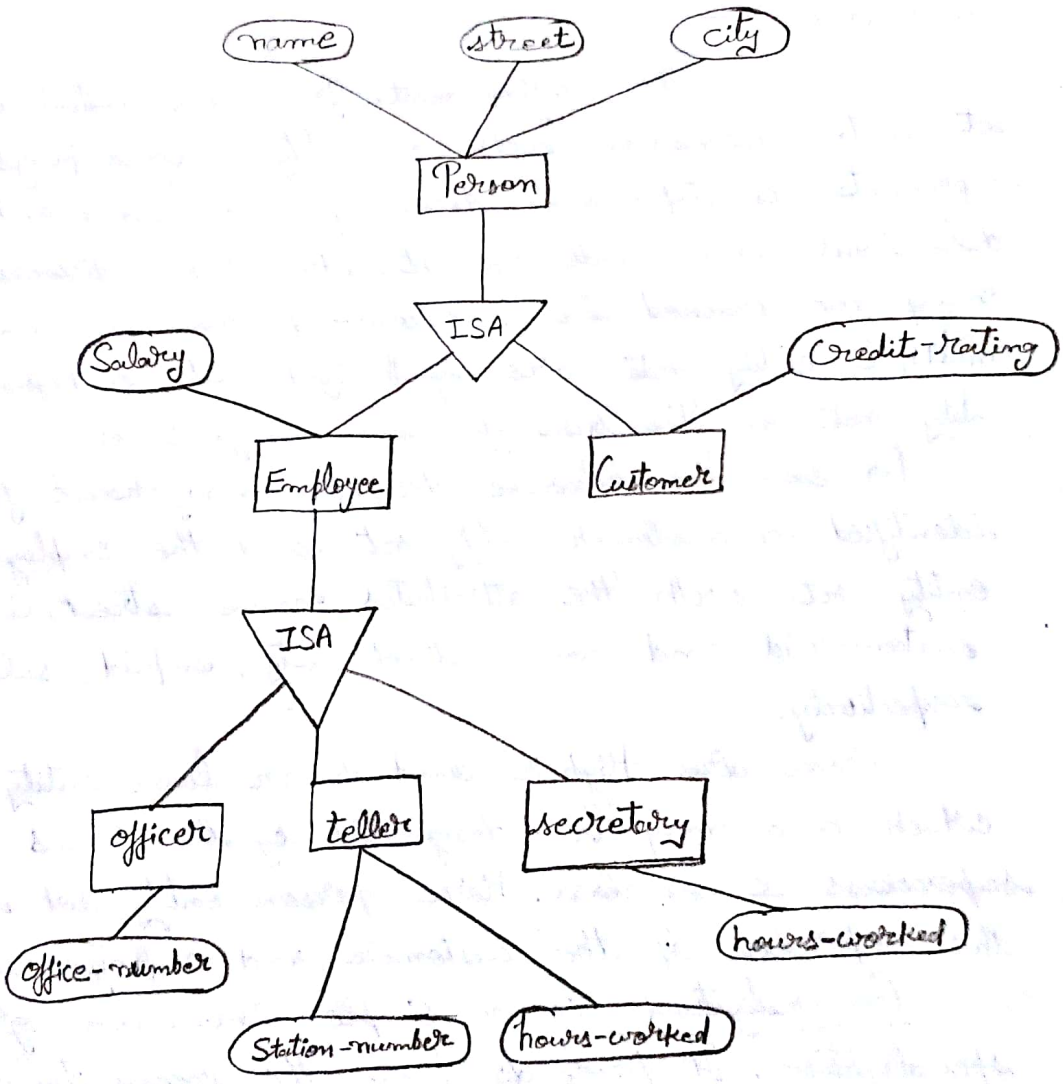


Fig. Specialization and Generalization

2 Specialization :—

An entity set may include sub-groupings of entities that are distinct in some way from other entities in the set. For instance, a subset of entities within an entity set may have attributes that are not shared by all the entities in the entity set. The process of designating subgroupings within an entity set is called Specialization. An entity set may be specialized by more than one distinguishing feature. When more than one specialization is formed on an entity set, a particular entity may belong to multiple specializations.

In example, the label ISA stands for "is a" and specialization is depicted by this.

3. Inheritance :—

The attributes of the higher-level entity sets are said to be inherited by the lower-level entity set.

A lower-level entity set (or subclass) also inherits participation in the relationship sets in which its higher-level entity (or superclass) participates.

→ A higher-level entity set with attributes and relationships that apply to all of its lower-level entity sets.

→ Lower-level entity sets with distinctive features that apply only within a particular lower-level entity set.

If an entity set is a lower-level entity set in more than one ISA relationship, then the entity set has multiple inheritance, & the resulting structure is said to be a lattice.

4. Aggregation :—

One limitation of the E-R model is that it can not express relationships among relationships. For ex- Consider the ternary relationship works-on, between a employee, branch and job. Suppose we want to record managers for combinations

The best way to model a situation like this is to use Aggregation.

* Aggregation is an abstraction through which relationships are treated as higher-level entities.

Thus, in example, we regard the relationship set works-on (relating the entity sets employee, branch, and job) as a higher-level entity set called works-on.

We can then create a binary relationship manages between works-on and manager to represent who manages what tasks.

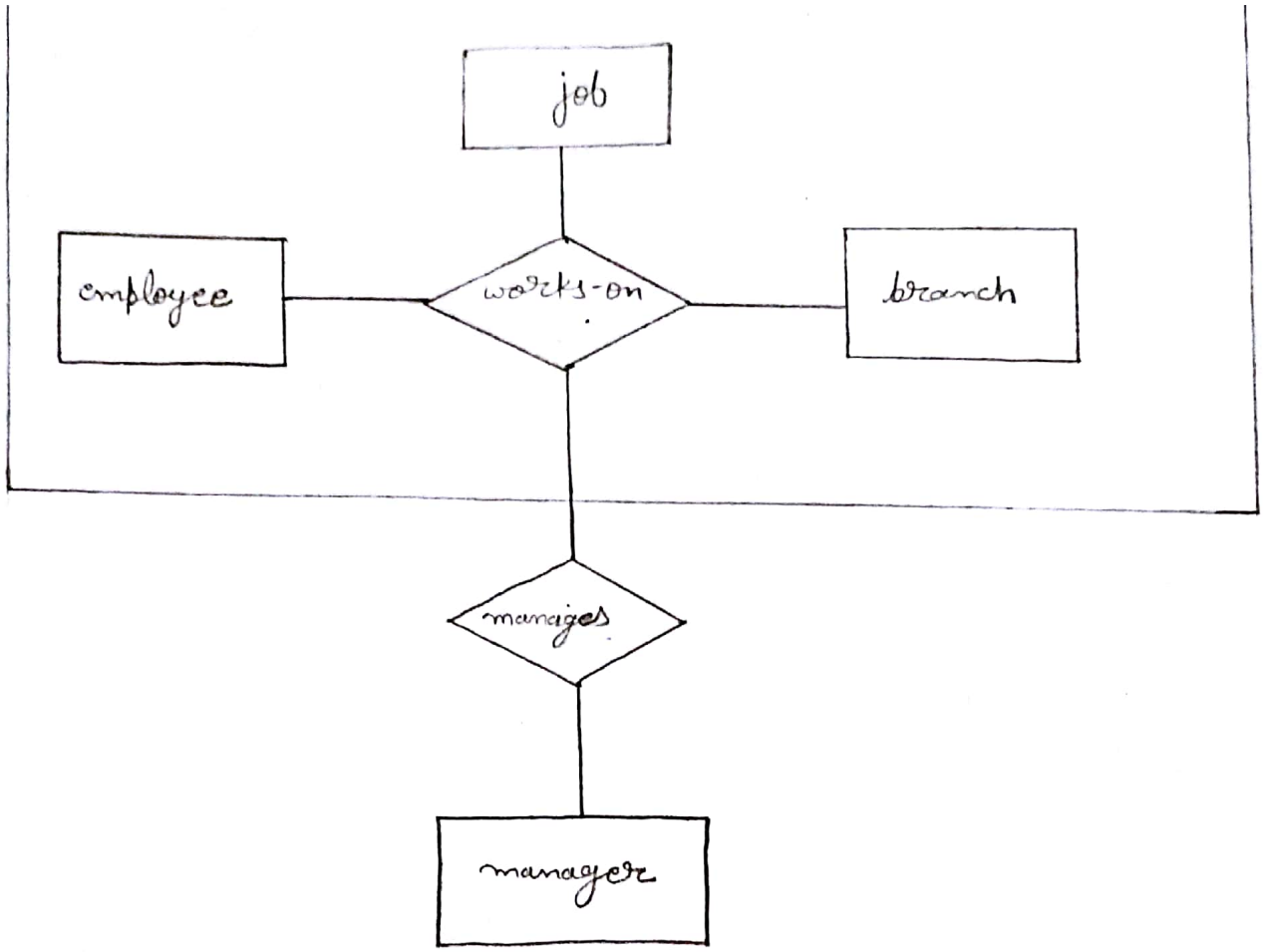


Fig: E-R diagram with aggregation

Conclusion of the lecture:

In this lecture, we have concluded about the types of entity set, like, weak & strong entity sets, along with some of the extended features, like, generalization, specialization, inheritance & aggregation.

Introduction to lecture :- In this lecture, we will discuss about -

- Existence Dependencies
  - Dominant Entity
  - Subordinate Entity
- Total Participation
- Partial Participation
- Weak & Strong Entity sets

Existence Dependencies :- If the existence of an entity A depends on the existence of entity B, then A is said to be existence dependent on B.

Dominant Entity :- Here B is c/a Dominant entity.
A member of a strong entity set is c/a Dominant entity.

Subordinate Entity :- Here 'A' is c/a Subordinate entity. The member of a weak entity set is c/a subordinate entity.

Total Participation :- The participation of an entity set E in a relationship R is said to be total, if every entity in E participates in at least one relationship in R.

Partial Participation :- If only some entities in E participate in relationships in R, the participation on entity set E in relationship R is said to be partial.

* In E-R diagram, double lines indicate total participation.

Conclusion to lecture: In this lecture, we have discussed about the types of entities and their participation types in relationships

Introduction to lecture :-

In this lecture, we will discuss about the various constraints of SQL, as, -

- NOT NULL
- Unique key
- Primary key
- Foreign key

along with the Assertions, like, -

- Domain constraints
- ~~Functional Dependency~~ Check Constraint
- Referential Integrity

Constraints :-

Constraints lets us define the way the database engine automatically enforces the integrity of a database. Constraints define rules regarding the values allowed in columns and are the standard mechanism for enforcing integrity.

SQL server supports the following classes of constraints :

(1) NOT NULL Constraint :-

The NOT NULL constraint enforces a column to NOT accept NULL values. The NOT NULL constraint enforces a field to always contain a value. This means that we can not insert a new record, or update a record without adding a value to this field.

For ex-

```
Create table dept
(
    dept_name varchar(20) NOT NULL,
    building varchar(20)
)
```

(2) Unique key Constraint :-

Unique key constraint enforce the uniqueness of the values in a set of columns. In a unique constraint, no two rows in the table can have the same value for the columns. Primary key also enforce uniqueness, but it does not allow for NULL as one of the unique values.

For ex-

```
Create table dept
(
    dept_name varchar(20) NOT NULL UNIQUE,
    building varchar(20)
)
```

(3) Primary key constraint :-

Primary key constraints identify the column or set of columns that have values that uniquely identify a row in a table.

No two rows in a table can have the same primary key value. We can not enter NULL for any column in a primary key.

Overall - Each table should have a primary key, and each table can have only one primary key.

For ex-

```
Create table dept
(
    dept_name varchar(20) NOT NULL,
    building varchar(20),
    PRIMARY KEY (dept_name)
)
```

(4) Foreign Key constraint :-

Foreign key constraints identify and enforce the relationships between tables. A foreign key in one table points to a candidate key in another table. The foreign key constraint also prevents that invalid data

from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

For ex –

```
Create table department_details
(
    dept-head varchar(20) NOT NULL PRIMARY KEY,
    dept_name varchar(20) FOREIGN KEY
        REFERENCES dept(deptname)
)
```

Assertions :–

An assertion is a predicate expressing a condition that we wish the database always to satisfy.

→ An assertion in SQL takes the form

Create assertion <assertion-name> Check <predicate>

→ When an assertion is made, the system tests it for validity, and tests it again on every update that may violate the assertion

→ Asserting for all X, P(X) is achieved in a round about fashion using ~~not~~

not exists X such that not P(X)

For example –

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

```
Create assertion sum-constraint check
    (not exists (select * from branch
                 where loan.branch-name =
                       branch.branch-name)
     >= (select sum(amount) from
         account where loan.branch-
         name = branch.branch-name)))
```

Domain Constraints :–

Integrity constraints guard against accidental damage to the database, by ensuring that authorized changes to the database do not result in a loss of data consistency.

→ Domain constraints are the most elementary form of integrity constraint.

→ They test values inserted in the database, and test queries to ensure that the comparisons make sense.

→ New domains can be created from existing data types.

→ For ex – create domain Dollars numeric (12, 2)

Check Constraint :- A check is a piece of SQL which makes sure a condition is satisfied before action can be taken on a record.

→ The check constraint is used to limit the value range that can be placed in a column.

→ If you define a CHECK constraint on a single column it allows only certain values for this column.

→ If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

→ Use check clause to ensure that an hourly-wage domain allows only values greater than a specified value.

```
Create domain hourly-wage numeric (5,2)
        constraint value-test check (value >=4.00)
```

Referential Integrity :-

It ensures that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation.

For ex -

If "Perryridge" is a branch name appearing in one of the tuples in the account relation, then there exists a tuple in the branch relation for branch "Perryridge".

Referential Integrity in SQL :-

Primary and candidate keys & foreign keys can be specified as part of the Referential integrity in SQL, create table statement:

- → The primary key clause lists attributes that comprise the primary key.
- → The unique key clause lists attributes that comprise a candidate key.
- → The foreign key clause lists the attributes that comprise the foreign key and the name of the relation referenced by the foreign key.

By default, a foreign key references the primary key attributes of the referenced table.

foreign key (account-number) references account

Example –

```
create table account
    (account-number char(10),
     branch-name char(15),
     balance integer,
     primary key (account-number),
     foreign key (branch-name) references branch)
```

- Entity Integrity :- It is concern the concept of the primary key. It is an integrity rule which states that every table must have a primary key (unique and not null).

Conclusion of lecture :-

In this lecture, we have concluded about the various types of constraints, like, NOT NULL, Unique key, Primary key, Foreign key ; and assertions, like, domain constraint, Check constraint & referential integrity.

Introduction to lecture:

In this lecture, we will discuss about the various types of database design along with some of the analysis.

Design of Database

- Requirement Analysis
- Conceptual database design
- logical database design
- Physical database design

Design of Database:

The systematic process of designing a database is known as design methodology. Database design involves understanding operational & business needs of an organization, modeling the specified requirements and realizing the requirements using a database. The goal of designing a database is to produce efficient, high quality and minimum cost database.

The overall database design and implementation process consists of several phases. like -

1. Requirement Analysis :-

It is the process of knowing and analyzing the expectation of the

users for the new database application in as much detail as possible.

They review the current file processing system or DBMS system, and interact with the users extensively to analyze the nature of business area to be supported and to justify the need for data and databases. The initial requirements may be informal, incomplete, inconsistent and partially incorrect. The requirement specification techniques such as object-oriented analysis, data flow diagram, etc. are used to transform these requirements into better structured form. This phase can be quite time consuming; however it plays the most crucial and important role in the success of the database system. The result of this phase is the document containing the specification of user requirement.

2. Conceptual Database Design :- In this phase, the database designer selects a suitable data model and translate the data requirements resulting from previous phase into a conceptual database schema by applying the concepts of chosen data model.

The conceptual schema is independent of any specific DBMS. The main objective of conceptual schema is to provide a detailed

overview of the organization. In this phase, a high level description of the data and constraints are developed. The E-R diagram is generally used to represent the conceptual database design. The conceptual schema should be expensive, simple, understandable, minimal and formal.

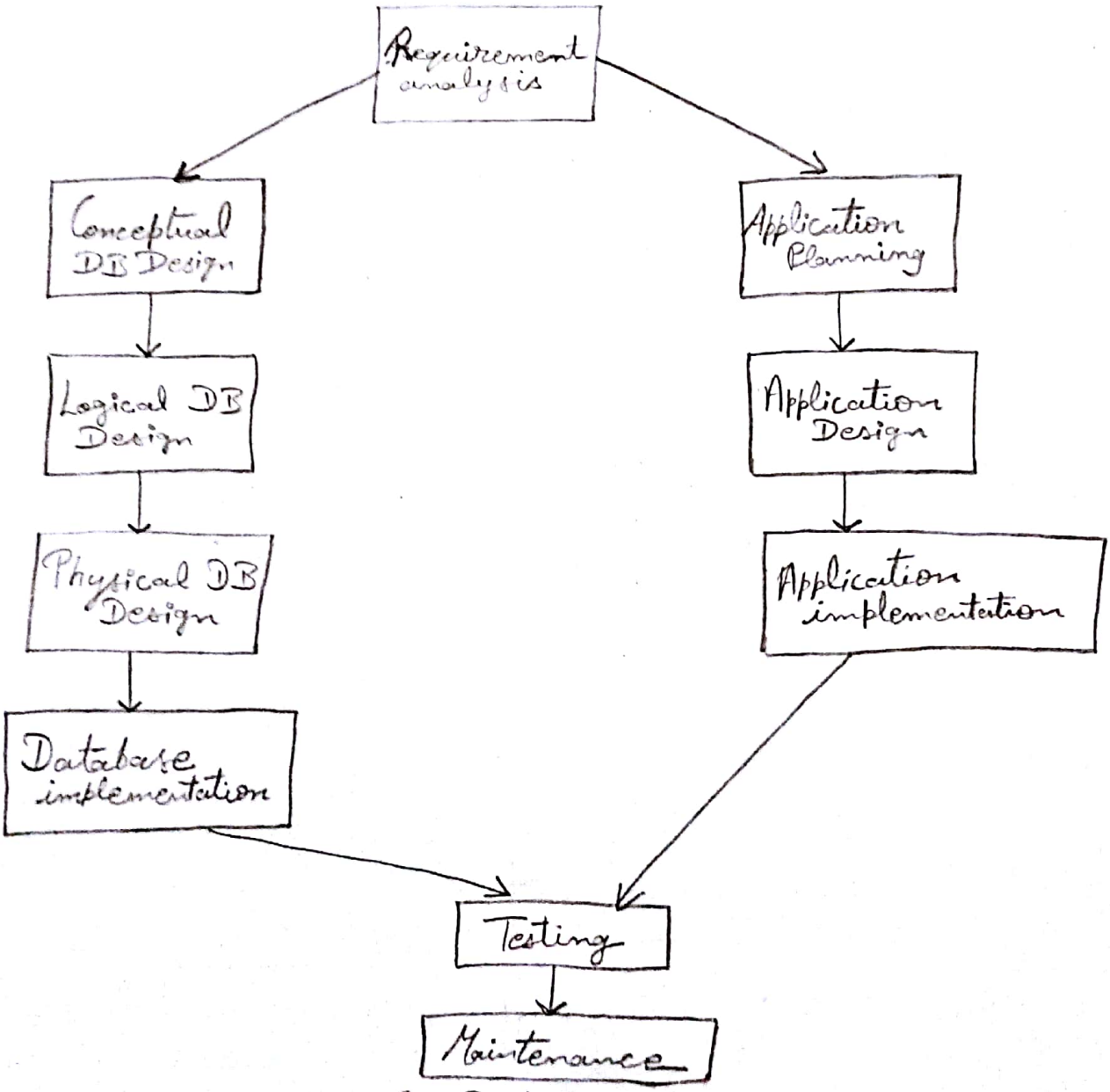


Fig: Database Design

3. Logical Database Design :- In this phase, the database designer moves from an abstract data model to the implementation of the database. In case of relational model, this phase generally consists of mapping the E-R model into a relational schema.

4. Physical Database Design :- In this phase, the physical features such as storage structure, file organization and access paths for the database files are specified to achieve good performance. The various option for file organization and access paths include various types of indexing, clustering of records, hashing techniques, etc.

Conclusion of lecture :

In this lecture, we have concluded about the various database design phases, like, requirement analysis, conceptual, logical & physical database design.