ଓଡ଼ିଶା ରାଜ୍ୟ ମୁକ୍ତ ବିଶ୍ୱବିଦ୍ୟାଳୟ, ସମ୍ବଲପୁର, ଓଡ଼ିଶା
Odisha State Open University, Sambalpur, Odisha
Established by an Act of Government of Odisha.

DIPLOMA IN COMPUTER APPLICATION

DCA-05 DATABASE SYSTEMS

BLOCK

1 DATABASE SYSTEM CONCEPTS

EXPERT COMMITTEE

DIPLOMA IN COMPUTER APPLICATION

Course Writers

Chandrakant Mallick
Odisha State Open University, Sambalpur, Odisha
Bijay Kumar Paikaray
Centurion University of Technology and Management, Odisha

UNIT-1 INTRODUCTION TO DATABASE SYSTEMS

UNIT STRUCTURE

1.0 Introduction

Database Systems refer to the technology of storing and retrieving users' data with utmost efficiency along with appropriate security measures. Database systems are basically developed for large amount of data. When dealing with huge amount of data, there are two things that require optimization: Storage of data and retrieval of data**.** DBMS allows its users to create their own databases as per their requirement. These databases are highly configurable and offerplenty of options to the users. In this unit we will explain the fundamental concepts like data, information, file, record and a database, basic concepts of databases and Database Management Systems.

1.1 Learning Objectives

After learning this unit you should be able to

- Define Data, Information, File, Databases etc.
- Identify the limitations of File-Oriented System.
- Define a Database Systems
- Understand the Characteristics of Database System
- Know the Advantages and Disadvantages
- Differentiate between File-Oriented Systems, Database Systems.
- Define a Database Management System
- Know the Role of aDatabase Administrator (DBA)
- Know the types of Database Systems.
- Classify different Database Languages.

1.2 Data and Information

Data is the raw material that is to be processed to produce information. Data represents the facts and figures relating to an object, place, event etc. that are to be processed. Data is useless unless it is processed or has been made into something. Data has no meaning when it has not been interpreted.

1.2.1 Examples of Data

- **Student Data on Admission Forms:** When students get admission in a college. They fill admission form. This form contains raw facts (data of student) like name, father's name, address of student etc.
- **Data of Citizens:** During census, data of all citizens is collected.

- **Survey Data:** Different companies collect data by survey to know the opinion of people about their product.
- **Students Examination data:** In examination data about obtained marks of different subjects for all students is collected.

1.2.2 Information

Processed data is called Information. The data can be made useful by processing it to produce information. Information is basically the data plus the meaning. Data does not depend upon information but information depends upon data. Information is something that is being conveyed to the intended user for decision making.

1.2.3Examples of Information

- **Student Address Labels:** Stored data of students can be used to print address labels of students.
- **Census Report:** Census data is used to get report/information about total population of a country and literacy rate etc.
- **Survey Reports and Results:** Survey data is summarized into reports/information to present to management of the company.
- **Grade Cards of Individual Student:** In an examination system collected data (obtained marks in each subject) is processed to get total obtained marks of a student, grade and percentage and the class or division etc. Here, total obtained marks, Percentage and division etc. are Information.
- **Merit List:** After collecting admission forms from candidates, merit is calculated on the basis of marksobtainedby each candidate. Normally, percentage of marks obtained is calculated for each candidate. Now all the candidates' names are arranged in descending order by percentage. This makes a merit list. Merit list is used to decide whether a candidate will get admission in the college or not.

1.2.4Differences between Data and Information

- Data is the input for a computer and information is the output that is meaningful to the user.
- Data is unprocessed facts or mere figures but information is processed data which has some meaning.
- Data does not depend on information but information depends on data and without it, information cannot be processed.
- Data is not specific but information is specific enough to generate meaning.
- Data is the raw material that is collected but information is a detailed meaning generated from the data.

1.3What is Record?

In terms of computer data processing, a record is a collection of related data items calledfields. Each Field represents one item of information.

A set of recordsconstitutes a file. For example, a personnel file might contain records that have three fields: a name field, an address field, and a phone number field.

The organization of data in the record is usually prescribed by the programming language or a database system or a file system that defines the record's organization and/or by the application that processes it. Typically, records can be of fixed-length or be of variable length with the length information contained within the record.

In relational databases, a record is a group of related data held within the same structure. More specifically, a record is a grouping of fields within a table that reference one particular object. The term record is frequently used synonymously with row.

For example, a customer record may include items, such as first name, physical address, email address, date of birth and gender.A record is also known as a tuple. The tuples are entered into the database table called a relation. The related tables or relations are interlinked by means of a database management system.

1.4What is File?

As we know, files are used for storing the information permanently. The files can contain any type of information such as text, Images or Pictures or any data in any format.

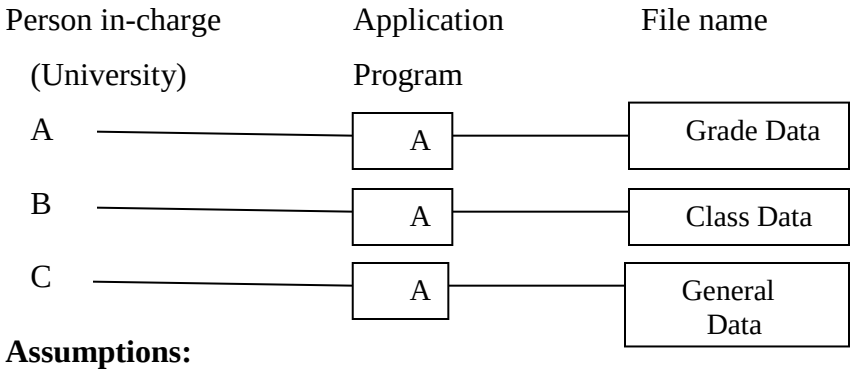
There are Many files which have their Owen Type and own names. When we Store a File in the System, then we must have to specify the Name and the Type of File. The Name of file will be any valid Name and Type means the application with the file has linked.

Thus a **file** is an object on a computer that stores data, information, settings, or commands used with a computer program. In a graphical user interface (GUI) such as Microsoft Windows, files display as icons that relate to the program that opens the file.

In the context of databases, a file is a collection of related records. For example, you might put the records of each of your customers in a file. In turn, each record would consist of *fields* for individual data items, such as customer name, customer number, customer address, and so forth. Depending on the operating system, files (and data sets) are contained within a catalog, directory, or folder.

1.5File-Oriented System

A time when there were no database system, file system was the only way to store, retrieve and manage data. Consider the scenario of a University administration using file oriented system.



Assumptions:

- A file named "General Data" may contain name, registration no, address and other information of the student.
- The file "Grade Data" includes the results of oral exam, written exams, seminars, projects etc.
- Class Data additionally comprises the attendances of the students.

While generating the student's progress report, the three files must be updated to give the consistent report.

1.5.1Advantages of File-Oriented System

1. **Backup:** It is possible to take faster and automatic back-up of database stored in files of computer-based systems. Computer systems provide functionalities to serve this purpose. It is also possible to develop specific application program for this purpose.
2. **Compactness:**It is possible to store data compactly.
3. **Data Retrieval:**Computer-based systems provide enhanced data retrieval techniques to retrieve data stored in files in easy and efficient way.
4. **Editing:**It is easy to edit any information stored in computers in form of files. Specific application programs or editing software can be used for this purpose.
5. **Remote Access:**In computer-based systems, it is possible to access data remotely. So, to access data it is not necessary for a user to remain present at location where these data are kept.

6. **Sharing:** Data stored in files of computer-based systems can be shared among multiple users at a same time.

1.5.2 Disadvantage of File-Oriented System

1. **Data Redundancy:** It is possible that the same information may be duplicated in different files. This leads to data redundancy results in memory wastage.
2. **Data Inconsistency:**Because of data redundancy, it is possible that data may not be in consistent state.
3. **Difficulty in Accessing Data:**Accessing data is not convenient and efficient in file processing system.
4. **Limited Data Sharing:**Data are scattered in various files. Also different files may have different formats and these files may be stored in different folders may be of different departments. So, due to this data isolation, it is difficult to share data among different applications.
5. **Integrity Problems:**Data integrity means that the data contained in the database in both correct and consistent. For this purpose the data stored in database must satisfy correct and constraints.
6. **Atomicity Problems:**Any operation on database must be atomic. This means, it must happen in it do **entirely** or not at all.
7. **Concurrent Access Anomalies:**Multiple users are allowed to access data simultaneously. This is for the sake of better performance and faster response.
8. **Security Problems:**Database should be accessible to users in limited way. Each user should be allowed to access data concerning his requirements only.

1.6Databases

A **database** is a collection of information that is organized so that it can easily be accessed, managed, and updated. In one view, databases can be classified according to types of content: bibliographic, full-text, numeric, and images.

In computing, databases are sometimes classified according to their organizational approaches. The most prevalent approach is the relational database, a tabular database in which data is defined so that it can be reorganized and accessed in a number of different ways. A distributed database is one that can be dispersed or replicated among different points in a network. An object-oriented programming database is one that is congruent with the data defined in object classes and subclasses.

For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks. A database management

system stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.

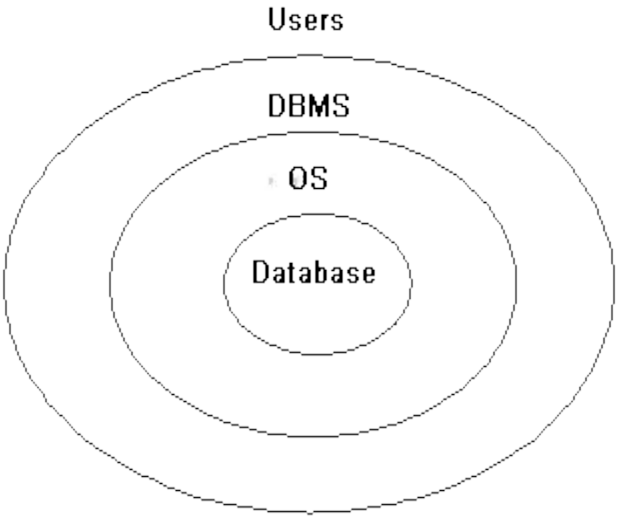


Fig: Layered Architecture of Database Systems

1.7Database System

Database system is a system to achieve an organized, store a large number of dynamical associated data, facilitate for multi-user accessing to computer hardware, software and data, that it is a computer system with database technology.

A database system aims to achieve a highly organized collection of data along with appropriate tools and applications that facilitate processing and access to that data.

Most people confuse a database system with a database management system, but the two are different. A DBMS is a suite of software tools used to store and manipulate data. Database management systems come in different types, such as document store, file store and relational database management systems.

1.7.1 Characteristics of Database System

Traditionally, data was organized in file oriented systems. DBMS concepts have been evolved to overcome the deficiencies in traditional file oriented systems. A modern DBMS has the following characteristics:

Real-world entity: A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.

Relation-based tables: DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.

Isolation of data and application: A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.

Less redundancy: DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.

Consistency: Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.

Query Language: DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.

ACID Properties: DBMS follows the concepts of Atomicity, Consistency, Isolation, and Durability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.

Multiuser and Concurrent Access: DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.

Multiple views: DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.

Security: Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot

see the data that belongs to the Purchase department. Additionally, it can also be managed how much data of the Sales department should be displayed to the user. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.

1.7.2 Requirements of Database Systems

Minimal data redundancy: Redundancy is carefully controlled by availed duplicate copies of the same data by an integrated view on date. For Example the controlled redundancy improves performance.

Data independence: Application programs are independent of representation of data and data storage (Abstract view)

Efficient data Access: DBMS uses a variety of techniques to store and retrieve data efficiently. For Example, Application of index structures to have faster access.

Concurrent data access: System to allow simultaneous access to the same data by different users. Each user gets the impression of exclusively accessing data. The concepts of transaction for the synchronization of concurrent data access.

Consistency of data: Caused by lack of redundancy. DBMS must ensure consistency of data after each transaction.

Integrity of data: correctness and completeness of data (Semantic aspects), formulation of integrity constraints and integrity rules.DBMS checks constraints for each insertion, change and deletion of data.

Data Security: Protection of the database against unauthorized access (e.gviews on data). Further the Access control with authentication and encoding as the possible protection mechanisms.

1.7.3 Advantage ofDatabase Systems

1. Improved data sharing:

- The DBMS helps create an environment in which end users have better access to more and better-managed data.
- Such access makes it possible for end users to respond quickly to changes in their environment.

2. Improved data security:

- The more users access the data, the greater the risks of data security breaches.Corporations invest considerable amounts of time, effort, and money to ensure that corporate data are used properly.
- A DBMS provides a framework for better enforcement of data privacy and security policies.

3. Better data integration:

- Wider access to well-managed data promotes an integrated view of the organization's operations and a clearer view of the big picture.
- It becomes much easier to see how actions in one segment of the company affect other segments.

4. Minimized data inconsistency:

- Data inconsistency exists when different versions of the same data appear in different places.
- For example, data inconsistency exists when a company's sales department stores a sales representative's name as "Bill Brown" and the company's personnel department stores that same person's name as "William G. Brown," or when the company's regional sales office shows the price of a product as $45.95 and its national sales office shows the same product's price as $43.95.
- The probability of data inconsistency is greatly reduced in a properly designed database.

5. Improved data access:

- The DBMS makes it possible to produce quick answers to ad hoc queries.
- From a database perspective, a query is a specific request issued to the DBMS for data manipulation—for example, to read or update the data. Simply put, a query is a question, and an ad hoc query is a spur-of-the-moment question.
- The DBMS sends back an answer (called the query result set) to the application.
- For example, end users

6. Improved decision making:

- Better-managed data and improved data access make it possible to generate better-quality information, on which better decisions are based.
- The quality of the information generated depends on the quality of the underlying data.
- Data quality is a comprehensive approach to promoting the accuracy, validity, and timeliness of the data. While the DBMS does not guarantee data quality, it provides a framework to facilitate data quality initiatives.
- Increased end-user productivity

- The availability of data, combined with the tools that transform data into usable information, empowers end users to make quick, informed decisions that can make the difference between success and failure in the global economy.

1.7.4 Disadvantage of Database Systems

1. Increased costs:

- Database systems require sophisticated hardware and software and highly skilled personnel.
- The cost of maintaining the hardware, software, and personnel required to operate and manage a database system can be substantial. Training, licensing, and regulation compliance costs are often overlooked when database systems are implemented.

2. Management complexity:

- Database systems interface with many different technologies and have a significant impact on a company's resources and culture.
- The changes introduced by the adoption of a database system must be properly managed to ensure that they help advance the company's objectives. Given the fact that database systems hold crucial company data that are accessed from multiple sources, security issues must be assessed constantly.

3. Maintaining currency:

- To maximize the efficiency of the database system, you must keep your system current.
- Therefore, you must perform frequent updates and apply the latest patches and security measures to all components.
- Because database technology advances rapidly, personnel training costs tend to be significant. Vendor dependence.
- Given the heavy investment in technology and personnel training, companies might be reluctant to change database vendors.

4. Frequent upgrade/replacement cycles:

- DBMS vendors frequently upgrade their products by adding new functionality. Such new features often come bundled in new upgrade versions of the software.
- Some of these versions require hardware upgrades. Not only do the upgrades themselves cost money, but it also costs money to train database users and administrators to properly use and manage the new features.

1.8Database Administrator (DBA)

- A Person in the organization who controls the design and the use of the database is known as a database administrator.
- DBA provides necessary technical support for implementing a database.
- DBA works on such as design, development, testing, and operational phases.
- DBA is a technical person having knowledge of database technology.
- DBA does not need to be a business person. In short, DBA is a technically focused person but should understand about the business to administrator the database effectively.

A DBA is expected to stay updated about the emerging technologies and new design approaches. Typically, a DBA has either a degree in Computer Science or some on-the-job training with a particular database product or more extensive experience with a range of database products.
A DBA is usually expected to have experience with one or more of the major database management products, such as Structured Query Language,MySQL, PostgreSQL, Microsoft SQL ServerIngresand Oracle-based database management software.

1.8.1 DBA Responsibilities

- Maintaining all databases required for development, testing, training and production usage
- Maximizing uptime of databases
- Managing share resources used amongst applications
- Administrates all database objects, including tables, views, indexes, stored procedures, functions, packages, sequences and clusters
- Enforces and maintains database constraints to ensure integrity of the database
- Installation and configuration of DBMS server software and related products.
- Upgrading and patching/hot-fixing of DBMS server software and related products.
- Assists with impact analysis of any changes made to the database objects.
- Migrate database to another server.
- Evaluate DBMS features and DBMS related products.
- Ensure that the site is running the products that are most appropriate

- ensure that any new product usage or release upgrade takes place with minimal impact
- Establish and maintain sound backup and recovery policies and procedures.
- Take care of the Database design and implementation.
- Implementing HA Solution (Replication, Clustering, Mirroring and Log Shipping)
- Implement and maintain database security (create and maintain logins, users and roles, assign privileges).
- Performance tuning and health monitoring on DBMS, OS and application.
- Setting Up Server Level, Database Level and Operating System Alerts
- Implementation of robust maintenance plan (Index Defrag, Stats Update, and DBCC etc.)
- SQL Server T-SQL/ Oracle PL-SQL Tuning
- Setup and maintain documentation and standards.
- Perform reviews on the design and code frequently to ensure the standards are being adhered to
- Plan growth and changes (capacity planning).
- Do general technical troubleshooting and give consultation to development teams.
- Troubleshooting on DBMS and Operating System performance issue
- Documentation of any implementation and changes (database changes, reference data changes and application UI changes etc)
- Be able to provide a strategic database direction for the organization.
- Expert level knowledge of DBMS Architecture, all features in all versions and troubleshooting skill
    - Excellent knowledge of DMBS backup and recovery scenarios.
    - Good skills in all DMBS tools.
    - A good knowledge of DMBS security management.
    - A good knowledge of how DMBS acquires and manages resources.
    - Sound knowledge of the applications at your site.

A DBA should possess a sound understanding of the business.

- Knowledge on Operating System
- Work as part of a team and provide 7×24 supports when required.
- Interface with DBMS Corporation for technical support.

- A DBA should have sound communication skills with management, development teams, vendors, systems administrators and other related service providers.
- ITIL Skill set requirement (Problem Management/Incident Management/Chain Management etc)

1.8.2 Types of Database Administrator

1. **Administrative DBA** – Work on maintaining the server and keeping it running. Concerned with installation, backups, security, patches, replication, OS configuration and tuning, storage management etc. Things that concern the actual server software.
2. **Development DBA** - works on building queries, stored procedures, etc. that meet business needs. This is the equivalent of the programmer. You primarily write T-SQLor PL-SQL.
3. **Database Architect** – Design schemas. Build tables, FKs, PKs, etc. Work to build a structure that meets the business needs in general. The design is then used by developers and development DBAs to implement the actual application.
4. **Data Warehouse DBA** - responsible for merging data from multiple sources into a data warehouse. May have to design warehouse, but cleans, standardizes, and scrubs data before loading. In SQL Server, this DBA would use SSIS heavily.
5. **OLAP DBA** – Builds multi-dimensional cubes for decision support or OLAP systems. The primary language in SQL Server is MDX, not SQL here
6. **Application DBA-** Application DBAs straddle the fence between the DBMS and the application software and are responsible for ensuring that the application is fully optimized for the database and vice versa. They usually manage all the application components that interact with the database and carry out activities such as application installation and patching, application upgrades, database cloning, building and running data cleanup routines, data load process management, etc.

1.8.3 Steps to be Database Administrator

1. Understanding the underlying concepts of database and the DBA responsibilities.
2. A programming background is helpful, but knowledge of SQL programming (T-SQL/PL-SQL) is a must.
3. Read books, magazines and internet resources (User Blog, articles, tutorial etc), participates in DBA related forums and news groups.
4. Attend database courses

5. Visit relevant websites and talk with real DBAs. Go for social site and increase your network.
6. Look for opportunities to practice your DBA skills.
7. Get the Certification
8. Constant learning, practicing and gathering experience from work.

1.9 Other Users of the Database

Besides the Database Administrator, other database users are of 4 different types:

1. **Naive users:**
   These are the unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.

   E.g. consider a user who checks for account balance information over the World Wide Web. Such a user access a form, enters the account number and password etc. And the application program on the internet then retrieves the account balance using given account information which s passed to the user.

2. **Application programmers:**
   These are computer professionals who write application programs, used to develop user interfaces. The application programmer uses Rapid Application Development (RAD) toolkit or special type of programming languages which include special features to facilitate generation of forms and display of date on screen.

3. **Sophisticated users:**
   These users interact with the database using database query language. They submit their query to the query processor. Then Data Manipulation Language (DML) functions are performed on the database to retrieve the data. Tools used by these users are OLAP(Online Analytical Processing) and data mining tools.

4. **Specialized users:**
   These users write specialized database applications to retrieve data. These applications can be used to retrieve data with complex data types e.g. graphics data and audio.

1.10Database Languages

There are two main types of languages.

(i) Data definition language (DDL)

(ii) Data manipulation language (DML)

(i) Data Definition language (DDL)

- DBMS provides a facility known as data definition language (DDL)
- DDL is used by DBA to define conceptual and internal schema
- DDL describes the entities, attributes and their relationship.
- Ex. Create table account. (Account no char (10), balance integer).
- DDL generates a set of tables stored in a data dictionary
- Data dictionary contains metadata (data about data) such as
- Data base schema
- Data storage and definition language (SDL) which describes storage structure and access methods used.
- Integrity constraints such as
- Domain constraints
- Referential integrity
- Assertions
- Authorization.
- DDL compiler processes the DDL statements in order to identity the description of schema objects / construct.
- DDL statements are used to create / define, update/alter and drop schema objects.

(ii) Data manipulation language (DML)

- The language for accessing and manipulating data
- Data access involves retrieval of a set of data items.
- Data manipulation involves user queries for insertion, deletion and update of records.
- Two types of DML statements.
- Procedural- uses specify what data is required and how to get those date
- Non procedural: user only specifies what data is required.
- DML is a part of the query language SQL
- The DML commands of SQL are
- Insert: to insert a new record
- Delete: to delete an existing record
- Update: to modify an existing record

But in modern database architecture, the other languages that are used by the database are:

(i) Storage Definition language (SDL):

- Used to specify by the internal schema
- It describes the storage structure and access methods of the database.

(ii) Transaction control language (TCL)

- A transaction is a collection of operations that performs a single logical function in a database application.
- TCL provides the transaction management commands such as commit and Roll back.

(iii) View Definition language(VDL)

- Used to specify user views and their mapping to conceptual schema.
- Ex. View point.

(iv) Data control languages (DCL)

- Allows a user to grant privileges or right to others to access the database.
- Ex. Grant revoke.

(v) Fourth Generation language(4GL)

- Designed to reduce the overall time, effort and cost of software development.
- The main domains and families of 4GLs are: database queries, report generators, data manipulation, analysis and reporting, screen painters and generators, GUI creators, mathematical optimization, web development and general purpose languages.
- In most of the database, TCL, SDL, VDL and DCL are all part of the DBMS's DDL component. Where DDL is used to define both conceptual and external schema.

1.11Let us Sum Up

The data must be organized in some logical manner so that it can be processed effectively and efficiently.Generally data can be organized in the form of files, records, fields, characters, bytes and bits.

A File is a group of related records. A **record**in turn is a set of related data items called fields. A **Field**represents an individual data item within a record. For example, Roll, name, age, sex, class of a student.**A Database is an** integrated collection of related files.

A **Database System**is a structured collection of data that is managed to meet the needs of a community of users. A computer database relies upon

software to organize the storage of data. This software is known as a database management system (DBMS).

A **Database Management System** (**DBMS**) is computer software designed for the purpose of managing databases based on a variety of data models.

A **DBMS** can also be defined as a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database.There are two main types of languages.Data definition language (DDL), Data manipulation language (DML).But in modern database architecture, the other languages that are used by the database are:Storage Definition language (SDL), Transaction control language (TCL), View Definition language (VDL), Data control languages (DCL), Fourth Generation language (4GL).

1.12Self Assessment Questions

1. Differentiate between Data and Information.

   ……
   ……
   ……
   ……
   ……

2. List the limitations of file oriented Systems.

   ……
   ……
   ……
   ……
   ……

3. Define a database system. Write the characteristics of a databaseSystem.

   ……
   ……
   ……
   ……
   ……

4. Write the benefits of a database system.

   ……
   ……
   ……
   ……
   ……

5. Write the role and Responsibilities of a DBA.

   ……
   ……
   ……
   ……
   ……

6. What are different types of users in the database systems?

   ……
   ……
   ……
   ……
   ……

7. What are different types of database languages?

   ……
   ……
   ……
   ……
   ……

1.13 Model Questions

1. Discuss the requirements of using a database system.
2. Discuss the advantages and disadvantages of database systems.
3. What are different types of DBA? Discuss their role.
4. Differentiate between the File Oriented Systems and Database Systems.
5. Who is a DBA? How can you be a DBA?

1.14References and Further Readings

1. Abraham Silberschatz, Henry Korth, and S. Sudarshan, "Database System Concepts", Mc Grow Hill
2. S.K Singh, Database Systems, Concepts, Design and Applications, Pearson Educationhttp://dbastudynotes.blogspot.in/2012/04/database-administrator-dba.html
3. http://searchsqlserver.techtarget.com/definition/database-administrator
4. http://tutorialink.com/dbms/advantage-and-disadvantages-of-file-oriented-system.dbms

UNIT-2 DATABASE SYSTEM ARCHITECTURE

UNIT STRUCTURE

2.0 Introduction

The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. n-tier architecture divides the whole system into related but independent **n** modules, which can be independently modified, altered, changed, or replaced.

In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming. So the application tier is normally accessible to the users.

2.1 Learning Objectives

After learning this unit you should be able to

- Define a Schema, Sub schema and Instances
- Understand the use of three-tier architecture of Database Systems
- Know the advantages of three -tier architecture
- Identify the characteristics of three-tier architecture
- Know the meaning of data independence
- Understand the mapping of different schema in the three-tier architecture
- Classify different types of Database Systems

2.2 Database System Architecture

In a standard terminology and general architecture for database systems was produced in 1971 by the DBTG (Data Base Task Group) appointed by the Conference on Data Systems and Languages (CODASYL, 1971). The DBTG recognized the need for a two level approach with a system view called the schema and user views called subschema. The American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC) produced a similar terminology mid architecture in 1975 (ANSI 1975). ANSI-SPARC recognized the need for a three level approach with a system catalog.

2.3 Schema

As we know the data values in the database changes frequently, while the plans or schemes remain the same over long periods of time. The database plans consist of types of entities that a database deals with, the relationship among these entities and the ways in which the entities and relationships are expressed from one level of abstraction to the next level for the users' view. The users' view of the data (also called logical organization of data) should be in a form that is most convenient for the users and they should not be concerned about the way data is physically organized. Therefore, a DBMS should do the translation between the logical (users' view) organization and the physical organization of the data in the database.

The plan or scheme of the database is known as Schema. Schema gives the names of the entities and attributes. It specifies the relationship among them. It is a framework into which the values of the data items (or fields) are fitted. The plans or the format of schema remains the same. But the values fitted into this format changes from instance to instance. In other terms, schema means overall plans of all the data item (field) types and record types stored in a database. Schema includes the definition of the database name, the record type and the components that make up those records

2.3.1 Types of Schema

A schema is defined as an outline or a plan that describes the records and relationships existing at the particular level.

There are three different types of schema in the database corresponding to each data view of database. In other words, the data views at each of three levels are described by schema. These are explained as follows.

External Schema:

- The level closest to the user and highest level of abstraction.
- Concerned with the way in which data are required by the user / viewed by the user.
- Describes only part of the data base called view level.
- Each external schema is used by a data model.
- Described the part of database, which is relevant to the user.

Conceptual Schema:

- It described the logical structure of the data base.
- It described the entities, data types, relationships, user operations and constraints.
- It defines a schema based on a data model.

- It gives information about existing data and relationships in database.

Internal Schema:

- The lowest level of abstraction. It describes the physical storage structure of the database.
- The internal schema uses physical data model.
- It describes complete details of the storage and access paths.
- It describes the detailed data structure of the data items.

Database Schema and State:

- A schema describes the structural design of the DB.
- A State describes a concrete instance of the DB.

2.3.2 Three Views of Data

- The view at each level is described by a schema
- The Schema describes the way in which the entities of one level of abstraction may be mapped in to the next level.
- The overall design of a database is called database schema.
- The database schema includes information such as
  - Characteristics of data items such as entities & attributes.
  - Logical structure and relationship among data items.
  - Format for storage representations.
  - Integrity parameters such as physically authorization and backup policies.

External view: The External view is described by means of a schema called external schema that correspond to different views of the data. **Conceptual view**: The Conceptual view is defined by conceptual schema, which describes all the entities, attributes, and relationships together with integrity constraints.

Internal View: Internal View is defined by internal schema, which is a complete description of the internal model, containing definition of stored records, the methods of representation, the data fields, and the indexes used.

There is only one conceptual schema and one internal schema per database. The schema also describes the way in which data elements at one level can be mapped to the corresponding data elements in the next level.

Thus, we can say that schema establishes correspondence between the records and relationships in the two levels. In a relational database, the schema defines the tables, the fields in each table, and the relationships between fields and tables. Schemas are generally stored in a data dictionary.

The data in the database at any particular point in time is called a database instance. Therefore, many database instances can correspond to the same database schema. The schema is sometimes called the **intension** of the database, while an instance is called an **extension** (or state) of the database.

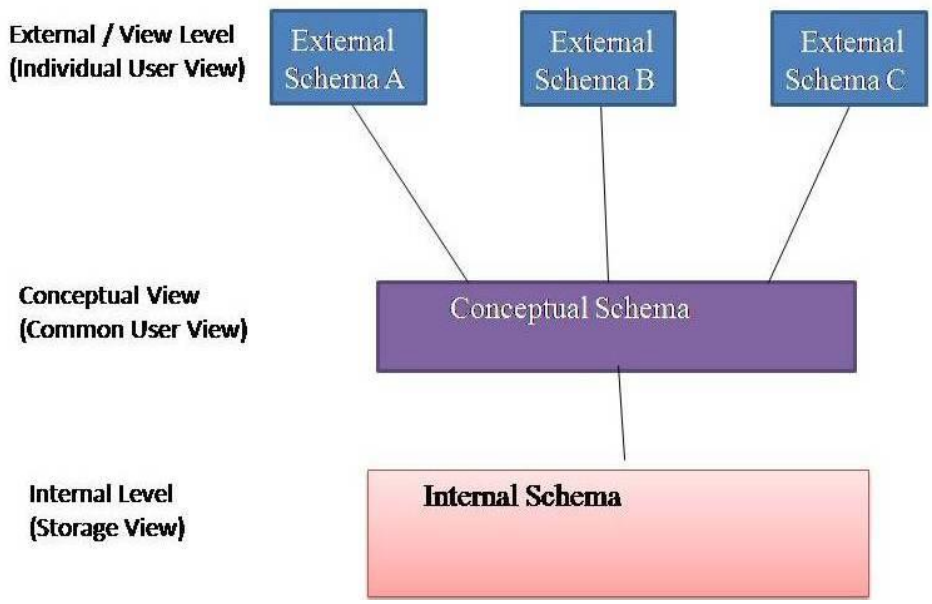


Fig: Three Views of Data

Example: To understand the difference between the three levels, consider again the database schema that describes College Database system. If User1 is a Library clerk, the external view would contain only the student and book information. If User2 is an account office clerk then he/she may be interested in students detail and fee detail. It shows specific information actually available at each level regarding a particular user.

The external view would depend upon the user who is accessing the database. The conceptual level contain the logical view of the whole database, it represents the data type of each required field. The internal view represents the physical location of each element on the disk of the servers well as how many bytes of storage each element needs.

2.3.3 Subschema

A subschema is a subset of the schema and inherits the same property that a schema has. The plan (or scheme) of a view is known as a subschema. Subschema refers to an application programmer's (user's) view of the data item types and record types, which he or she uses. It gives the users a window through which he or she can view only that part of the database, which is of interest to him. Therefore, different application programs can have different view of data.

2.3.4 Database Instance

It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

You can better understand its workings with the help of an example. The example is of an organization that has an employee database.

This database will have three instances, which are Production that is used for storing live data pre-production which is used to test new functionality prior to release for production development which is used by database developers in order to create new functionality.

2.4. Three Tier Architecture of Databases

Three-Tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.

The objectives of three-tier architecture are to separate each user's view of the database from the Way the database is physically represented. There are several reasons why this separation is desirable:

- Each user should be able to access the same data, but have a different customized view of the data. Each user should be able to change the way he or she views the data, and this change should not affect other users.
- Users should not have to deal directly with physical database storage details, such as indexing or hashing. In other words a user's interaction with the database should be independent of storage considerations.
- The Database Administrator (DBA) should be able to change the database storage structures without affecting the user's views.
- The internal structure of the database should be unaffected by changes to the physical aspects of storage, such as the changeover to a new storage device.
- The DBA should be able to change the conceptual structure of the database without affecting all users.

There are following three levels or layers of DBMS architecture:

- External Level
- Conceptual Level
- Internal Level

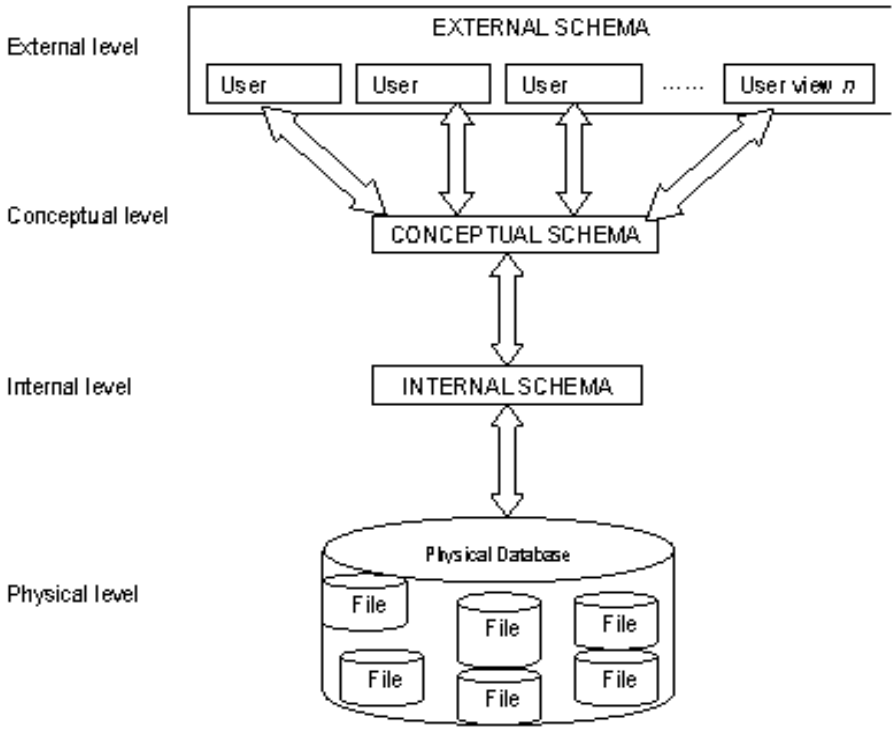


Fig. Three-tier database architecture

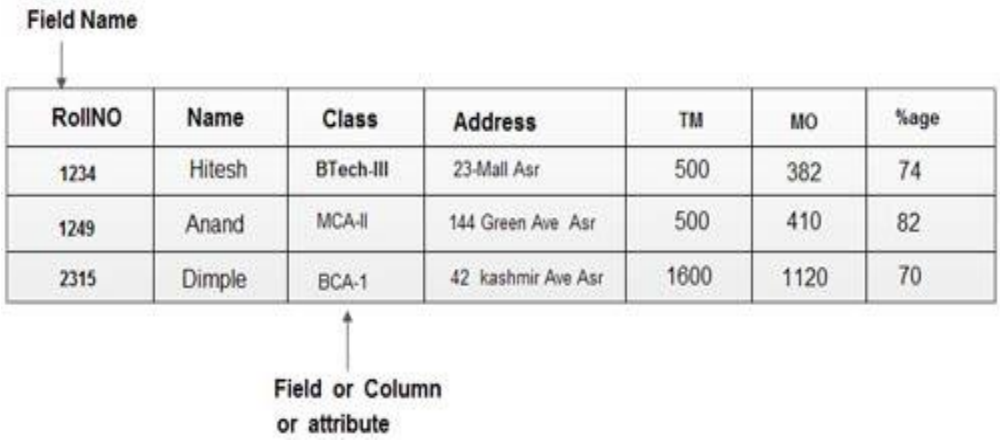
2.4.1. External Level or View level

It is the users' view of the database. This level describes that part of the database that is relevant to each user. External level is the one which is closest to the end users. This level deals with the way in which individual users vie\v data. Individual users are given different views according to the user's requirement.

A view involves only those portions of a database which are of concern to a user. Therefore same database can have different views for different users. The external view insulates users from the details of the internal and conceptual levels. External level is also known as the view level. In addition different views may have different representations of the same data. For example, one user may view dates in the form (day, month, year), while another may view dates as (year, month, day).

2.4.2 Conceptual Level or Logical level

It is the community view of the database. This level describes what data is stored in the database and the relationships among the data. The middle level in the three-tier architecture is the conceptual level. This level contains the logical structure of the entire database as seen by the DBA. It is a complete view of the data requirements of the organization that is independent of any storage considerations. The conceptual level represents:
All entities, their attributes, and their relationships;
An Entity is an object whose information is stored in the database. For example, in student database the entity is student. An attribute is a characteristic of interest about an entity.
For example, in case of student database Roll No, Name, Class, Address etc. are attributes of entity student.



- The constraints on the data;
- Semantic information about the data;
- Security and integrity information.

The conceptual level supports each external view, in that any data available to a user must be contained in or derivable from, the conceptual level. However, this level must not contain any storage dependent details. For instance, the description of an entity should contain only data types of attributes (for example, integer, real, character) and their length (such as the maximum number of digits or characters), but not any storage considerations, such as the number of bytes occupied. Conceptual level is also known as the, logical level.

2.4.3 Internal level or Storage level

It is the physical representation of the database on the computer. This level describes how the data is stored in the database. The internal level is the one that concerns the way the data are physically stored on the hardware.

The internal level covers the physical\ implementation of the database to achieve optimal runtime performance and storage space utilization. It covers the data structures and file organizations used to store data on storage devices. It interfaces with the operating system access methods to place the data on the storage devices, build the indexes, retrieve the data, and so on.

The internal level is concerned with such things as:

- Storage space allocation for data and indexes;
- Record descriptions for storage (with stored sizes for data items);
- Record placement;
- Data compression and data encryption techniques.

There will be only one conceptual view, consisting of the abstract representation of the database in it's entirely. Similarly there will be only one internal or physical view, representing the total database, as it is physically stored.

2.4.4 Advantages of Using Three-Tier Architecture

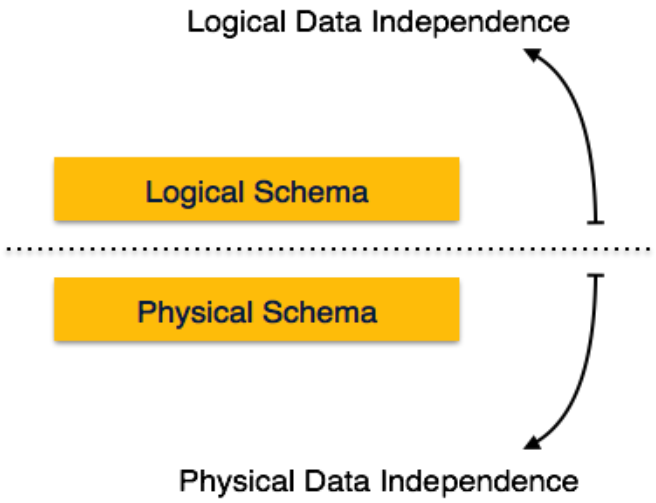
- It makes the logical separation between business layer and presentation layer and database layer.
- Migration to new graphical environments is faster.
- As each tier is independent it is possible to enable parallel development of each tier by using different sets of developers.
- Easy to maintain and understand large project and complex project.
- Since application layer is between the database layer and presentation layer so the database layer will be more secured and client will not have direct access to the database.
- Posted data from presentation layer can be verified or validated at application layer before updating it to the database.
- Database Security can be provided at application layer.
- Application layer or middle layer or business layer can be a protection shield to the database.
- New rules or new validation rules can be defined any time and changes made to middle layer will not affect presentation layer.
- We can hide unnecessary methods from business layer in the presentation layer.
- Easy to apply object oriented concept
- Easy to update data provider queries.

2.4.5 Disadvantage of Using Three-Tier Architecture

- To implement even small part of application it will consume lots of time.
- Need good expertise in object oriented concept (classes and objects).
- It is more complex to build.

2.5 Data Independence

A database system normally contains a lot of data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.



- Denotes the property that higher levels of abstraction are not influenced by changes in the lower levels.
- The ability to change the schema at one level of a database system without affecting a schema definition in the next higher level is called data independence.
- There are three levels of abstractions so there are two types of data independence.

There are two kinds of data independence:

- Logical data independence
- Physical data independence

2.5.1 Logical data independence

It is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), or to reduce the database (by removing a record type or data item). In the latter case, external schemas that refer only to the remaining data should not be affected. Only the view definition and the mappings need be changed in a DBMS that supports logical data independence. Application programs that reference the external schema constructs must work as before, after the conceptual schema undergoes a logical reorganization. Changes to constraints can be applied also to the conceptual schema without affecting the external schemas or application programs.

2.5.2 Physical Data Independence

Physical data independence is the capacity to change the internal schema without having to change the conceptual (or external) schemas. Changes to the internal schema may be needed because some physical files had to be reorganized—for example, by creating additional access structures—to improve the performance of retrieval or update. If the same data as before remains in the database, we should not have to change the conceptual schema. Whenever we have a multiple- level DBMS, its catalog must be expanded to include information on how to map requests and data among the various levels. The DBMS uses additional software to accomplish these mappings by referring to the mapping information in the catalog. Data independence is accomplished because, when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between the two levels is changed. Hence, application programs referring to the higher-level schema need not be changed.

2.5.3 Data Abstraction

Major purpose of DBMS is to provide users with abstract viewof data i.e. the system hides certain details of how the data arestored and maintained.

Since database system users are not computer trained,developers hide the complexity from users through *3 levels ofabstraction*, to simplify user's interaction with the system.

1) Physical level of data abstraction:

This is the lowest level of abstraction which describes howdata are actually stored.

2) Logical level of data abstraction:

This level hides what data are actually stored in thedatabase and what relationship exists among them.

3) View Level of data abstraction:

View provides security mechanism to prevent user fromaccessing certain parts of database.

2.6 Mappings

Data mapping is a process used in data warehousing by which different data models are linked to each other using a defined set of methods to characterize the data in a specific definition. This definition can be any atomic unit, such as a unit of metadata or any other semantic. This data linking follows a set of standards, which depends on the domain value of the data model used.

Data mapping serves as the initial step in data integration.

2.6.1 Mapping between Views

The DBMS is responsible for mapping between these three types of schema. Two mappings are required in a database system with three different views.

2.6.2 External/Conceptual Mapping

Each external schema is related to the conceptual schema by the external/conceptual mapping. A mapping between the external and conceptual views gives the correspondence among the records and the relationships of the external and conceptual views the external view is an abstraction of the conceptual view, which in its turn is an abstraction of the internal view. It describes the contents of the database as perceived by the user or application program of that view. The user of the external view sees and manipulates a record corresponding to the external view. There is a mapping from0 a particular logical record in the external view to one (or more) conceptual record(s) in the conceptual view.

2.6.3 Differences between External/Conceptual Views

Following could be differences that exist between the two:

Names of the field's and. records, for instance, may be different. A number of conceptual fields can be combined into a single external field, for example, Last Name and First Name at the conceptual level but Name at the external level. A given external record could be derived from a number of conceptual .records.

2.6.4 Conceptual/Internal Mapping

Conceptual schema is related to the internal schema by the conceptual/internal mapping. This enables the DBMS to find the actual record or combination of records in physical storage that constitute a logical record in conceptual schema. Mapping between the conceptual and the internal levels specifies the method of deriving the conceptual record from the physical database.

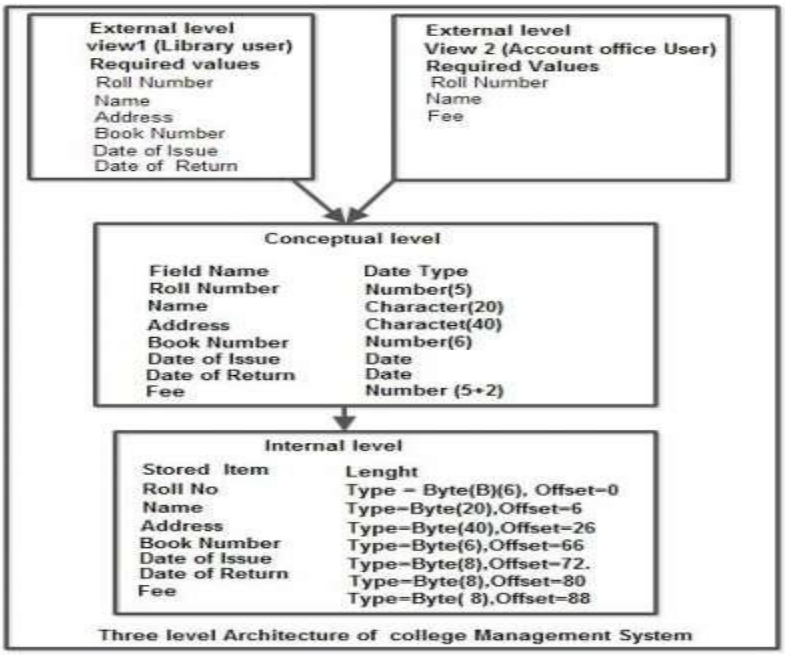


Fig: Mapping between Views

2.7 Types of Databases

Several criteria are normally used to classify DBMSs. The *first* is the data model on which the DBMS is based. The structure of a database is defined by the **data model. A data model is** a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

The main data model used in many current commercial DBMSs is the relational data model. The object data model was implemented in some commercial systems but has not had widespread use. Many legacy (older) applications still run on database systems based on the hierarchical and network data models. The relational DBMSs are evolving continuously, and, in particular, have been incorporating many of the concepts that were

developed in object databases. This has led to a new class of DBMSs called object-relational DBMSs.

We can hence categorize the databases based on the data model as below:

- Hierarchical databases.
- Network databases.
- Relational databases.
- Object-oriented databases

The *second* criterion used to classify DBMSs is the number of users supported by the system.

Single-user systems support only one user at a time and are mostly used with personal computers.

Multiuser systems, which include the majority of DBMSs, support multiple users concurrently.

A *third* criterion is the number of sites over which thedatabase is distributed. They are centralized and distributed databases.

Centralized Databases: We can say a DBMS is centralized if the data is stored at a single computer site. A centralized DBMS can support multiple users, but the DBMS and the database themselves reside totally at a single computer site.

Distributed Database:A distributed DBMS (DDBMS) can have the actual database and DBMS software distributed over many sites, connected by a computer network. Homogeneous DDBMSs use the same DBMS software at multiple sites.

A recent trend is to develop software to access several autonomous preexisting databases stored under heterogeneous DBMSs. This leads to a federated DBMS (or multi-database system), in which the participating DBMSs are loosely coupled and have a degree of local autonomy. Many DBMSs use client-server architecture.

2.7.1 Hierarchical Databases

The database designed based on the Hierarchical Database Model is called Hierarchical Databases. It is very fast and simple. In a hierarchical

database, records contain information about their groups of parent/child relationships, just like as a tree structure. The structure implies that a record can have also repeating information. In this structure data follows a series of records; it is a set of field values attached to it. It collects all records together as a record type. These record types are the equivalent of tables in the relational model, and with the individual records being the equivalent of rows. To create links between these record types, the hierarchical model uses these types of relationships.

Advantages

Hierarchical database can be accessed and updated rapidly because in this model structure is like as a tree and the relationships between records are defined in advance. This feature is a two-edged.

Disadvantage

This type of database structure is that each child in the tree may have only one parent, and relationships or linkages between children are not permitted, even if they make sense from a logical standpoint. Hierarchical databases are so in their design. It can add a new field or record requires that the entire database be redefined.

2.7.2 Network Databases

A network databases are mainly used on large digital computers. If more connections can be made between different types of data, network databases are considered more efficient. Its limitations must be considered when we have to use this kind of database.

Advantages:

Flexibility: Multiple parent/child relationships allowed a network database to represent data that did not have a simple hierarchical structure.

Standardization: The CODASYL standard boosted the popularity of the network model, and minicomputer vendors such as Digital Equipment Corporation and Data General implemented network databases.

Performance: Despite their greater complexity, network databases boasted performance approaching that of hierarchical databases. Sets were represented by pointers to physical data records, and on some systems, the database administrator could specify data clustering based on a set relationship.

Disadvantages:

Like hierarchical databases, they were very rigid. The set relationships and the structure of the records had to be specified in advance.

2.7.3 Relational Databases

In relational databases, the relationship between data files is relational. Hierarchical and network databases require the user to pass a hierarchy in order to access needed data. These databases connect to the data in different files by using common data numbers or a key field. Data in relational databases is stored in different access control tables, each having a key field that mainly identifies each row. In the relational databases are more reliable than either the hierarchical or network database structures. In relational databases, tables or files filled up with data are called relations (tuples) designates a row or record, and columns are referred to as attributes or fields.

Relational databases work on each table has a key field that uniquely indicates each row, and that these key fields can be used to connect one table of data to another.

The relational database has two major reasons:

1. Relational databases can be used with little or no training.
2. Database entries can be modified without specify the entire body.

Properties of Relational Tables:

In the relational database we have to follow some properties which are given below.

- The column Values are Atomic
- In Each Row is independent and unique.
- Column Values are of the same kind.
- The Sequence of Columns is Insignificant.

- Sequence of Rows is Insignificant.
- Each Column Has a Unique Name.

2.7.4 Object-Oriented databases

It adds the database functionality to object programming languages. This approach is the analogical of the application and database development into a constant data model and language environment. Applications require less code, use more natural data modeling, and code bases are easier to maintain. Object developers can write complete database applications with a decent amount of additional effort.

The object-oriented database derivation is the integrity of object-oriented programming language systems and consistent systems. The power of the object-oriented databases comes from the cyclical treatment of both consistent data, as found in databases, and transient data, as found in executing programs.

Object-oriented databases use small, recyclable separated of software called objects. The objects themselves are stored in the object-oriented database. Each object contains of two elements:

1. Piece of data
2. Instructions or software programs called methods, for what to do with the data.

The benefits to object-oriented databases are compelling. However the disadvantage of Object-Oriented Databases is that it is more expensive to develop.

2.7.5 New Database Systems

Apart from these database systems, some of the database systems for new database applications include the following.

1. **Design databases**: Designing CAD/CAM/CASE Systems.
2. **Multimedia databases**: Designing systems to support data storage and access which are in the form of text, image, video, voice, graphs, figures etc.
3. **Knowledge bases**: AI and Expert systems represent information as facts and rules that can be collectively viewed as knowledge base.

4. **Mobile Databases:** A mobile database is a database that can be connected to by a mobile computing device over a wireless mobile network. Mobile DBMSs are needed to support these applications data processing capabilities. Mobile data-driven applications enable us to access any data from anywhere, anytime.

2.8 Let Us Sum Up

In this unit we have discussed the general architecture, different schema of a database system and different types of database systems. We summarise the discussion below:

The plan or scheme of the database is known as Schema. Schema gives the names of the entities and attributes. It specifies the relationship among them. In other words, the overall design of the database is called the database **schema**.

The collection of information stored in the database at a particular moment is called an **instance** of the database. The **physical schema** describes the database design at the physical level, while the **logical schema** describes the database design at the logical level. A database may also have several schemas at the view level, sometimes called **sub-schemas** that describe different views of the database.

Application programs are said to exhibit **physical data independence** if they do not depend on the physical schema, and thus need not be rewritten if the physical schema changes.

Underlying the structure of a database is the **data model**: a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

We can hence categorize DBMSs based on the *data model*: relational, object oriented, hierarchical, network etc.

2.9 Self Assessment Questions

1. Define a Schema. What are different types of Schema?

……
……
……
……
……
……
……
……

2. What do you mean by intension and extension of the database?

……
……
……
……
……
……
……

3. What are the disadvantages of using Three-Tier Architecture?

……
……
……
……
……
……
……
……

4. What do you mean by data abstraction? What are the three levels of data abstraction in databases?

……
……
……
……
……
……
……
……

5. What are the advantages of network databases?

……
……
……
……
……
……
……
……

2.10 Model Questions

1. What do you mean by a Schema? Discuss the characteristics of different types of Schema.
2. Discuss the three-tier architecture of databases.
3. What are the advantages of using Three-Tier Architecture?
4. Differentiate between Physical and Logical Data Independence.
5. Discuss how mapping takes place in different views of three tier databases architecture.
6. Discuss different types of databases with their relative merits and demerits.

2.11 References and Further Readings

1. S.K Singh, Database Systems, Concepts, Design and Applications, Pearson Education.
2. Abraham Silberschatz, Henry Korth, and S. Sudarshan, "Database System Concepts", Mc Grow Hill
3. R. Elmasri and S. Navathe, "Fundamentals of Database Systems" Pearson Education.
4. https://www.tutorialspoint.com/dbms/dbms_data_models.htm
5. http://asp-net-by-parijat.blogspot.in/2014/12/advantages-and-disadvantages-of-using-3.html
6. http://ecomputernotes.com/fundamental/what-is-a-database/what-is-a-database-architecture
7. http://www.c-sharpcorner.com/uploadfile/65fc13/types-of-database-management-systems/

UNIT-3 DATA MODELS

UNIT STRUCTURE

3.0 Introduction

Planning the structure of database is called data models. It involves planning about tables, their columns, mapping between the tables, how they are structured in the physical memory etc. A data model helps to put the real world requirement into a design. This makes the developer to understand the relationship between various objects in the database. It helps to highlight any drawbacks of the plan and correct it at the design stage itself.

Enterprise Architects supports comprehensive functionality for modeling database structures. In this unit we will cover the core features for data modeling over the full lifecycle of a database. We will discuss the basic modeling process – that is outlining a conceptual model and then working through the other models.

3.1 Learning Objectives

After learning this unit you should be able to

- Define a Data Model
- Know the components of a Data Model.
- Know different types of Data Models.
- Compare the advantages and disadvantages of Data Models.

3.2 Data Models

A database management system is an information system typically consists of a database (contained stored data) together with programs that capture, store, manipulate, and retrieve the data.

A model is a representation of reality, 'real world' objects and events, associations. It is an abstraction that concentrates on the essential, inherent aspects an organization and ignores the accidental properties. A data model represents the organization itself. It should provide the basic concepts and notations that will allow database designers and end users unambiguously and accurately to communicate their understanding of the organizational data.

Data Model can be defined as an integrated collection of concepts for describing and manipulating data, relationships between data, and constraints on the data in an organization.

The taxonomy of classification of data models is given in the figure below.

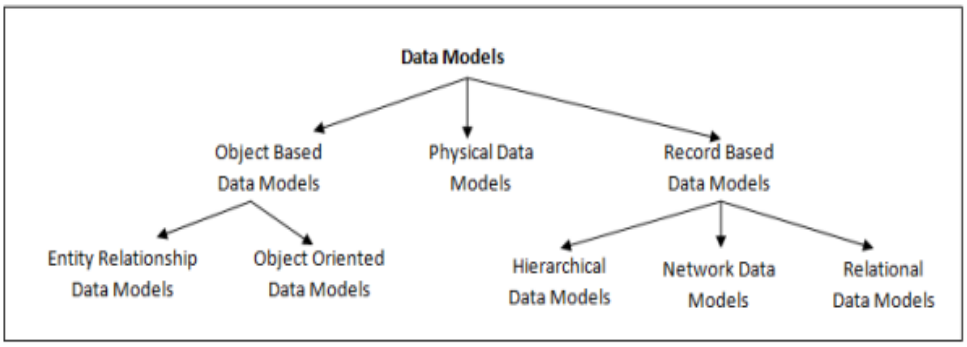


Fig: Taxonomy of Data Models

3.2.1 A data model comprises of three components

- A structural part, consisting of a set of rules according to which databases can be constructed.
- A manipulative part, defining the types of operation that are allowed on the data (this includes the operations that are used for updating or retrieving data from the database and for changing the structure of the database).
- Possibly a set of integrity rules, which ensures that the data is accurate.
- The purpose of a data model is to represent data and to make the data understandable. There have been many data models proposed in the literature. They fall into three broad categories:
- Object Based Data Models
- Physical Data Models
- Record Based Data Models

The object based and record based data models are used to describe data at the conceptual and external levels, the physical data model is used to· describe data at the internal level.

3.3 Object Based Data Models

Imagine we have to design database for a college. What is the real world entities involved with a college? They are college, Students, Lecturer, Courses, Subject, and Marks etc. Once all the entities are listed, we find out the relationship between them and try to map all of them. Also we list what are the attributes related to each entity like student id, name, lecturer name, course that he is teaching, different subjects, pass mark, grade levels etc. Here we are not bothered about what data value is stored, what is the size of each data etc. We know only entities involved, their attributes and mapping at this stage.

Object based Data Models are based on above concept. It is designed using the entities in the real world, attributes of each entity and their relationship. It picks up each thing/object in the real world which is involved in the requirement.

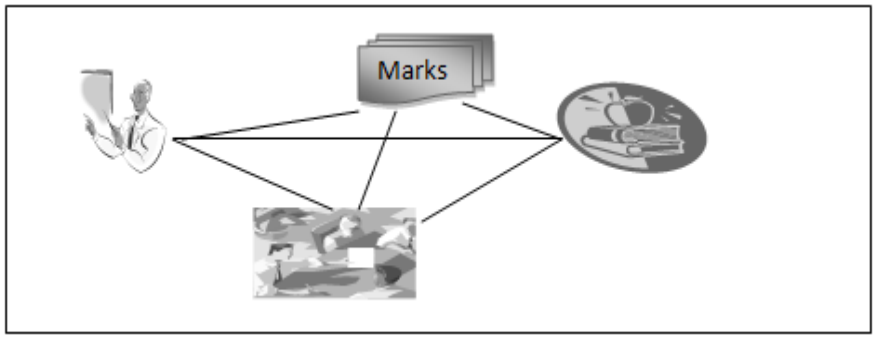


Fig: Object based Data Models

There are two types of object based data Models – Entity Relationship Model and Object oriented data model. ER data model is one of the important data model which forms the basis for the all the designs in the database world. It defines the mapping between the entities in the database. Object oriented data model, along with the mapping between the entities, describes the state of each entity and the tasks performed by them.

3.3.1 Entity Relationship Data Models

This model is introduced by the scientist Peter Chen in 1976.E.R model is a generalization of hierarchical and network data model. The relationship between the entity set is represented by named E-R relationships from one entity set to another.

Consider the example above. It maps entities like Student, Lecturer, Subjects, and Marks with each other to form the relation among them. It also list attributes of each objects. ER model represents the all these entities, attributes and their relationship in the form of picture to make the developer understand the system better. A simple ER diagram for above example can be drawn as below. Are you able to understand what are the entities involved, what are its attributes and their relations that we were discussing better here? Yes, it is clean and clear what a STUDENT database look like. It gives the clear understanding of how they are scattered and mapped. If we have missed any entities or attribute or the mapping, we can easily identify here. If we represent it in some tables, it would be difficult to identify this gap.

In the below diagram, Entities or real world objects are represented in a rectangular box. Their attributes are represented in ovals. Primary keys of entities are underlined. All the entities are mapped using diamonds. This is one of the methods of representing ER model. There are many different forms of representation. More details of this model are described in ER data model article.

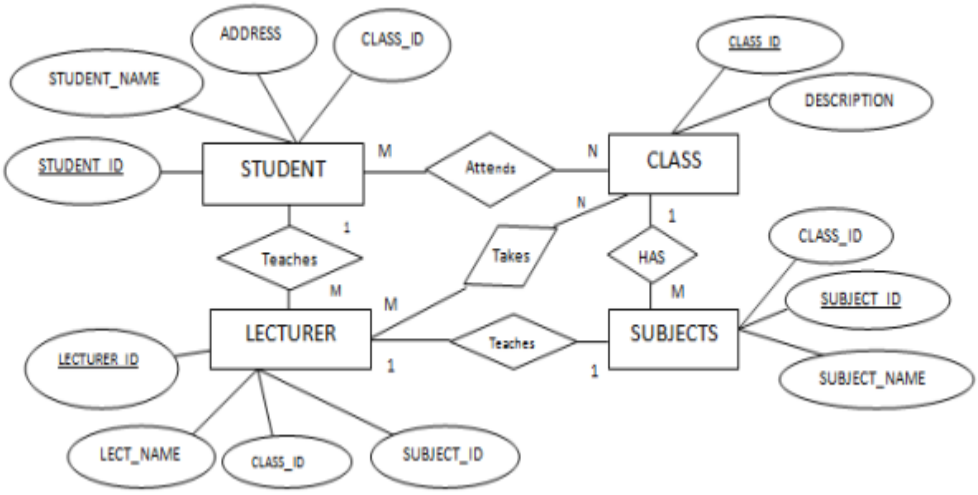


Fig: Example of E-R Diagram

Basically, ER model is a graphical representation of real world objects with their attributes and relationship. It makes the system easily understandable. This model is considered as a top down approach of designing a requirement.

Advantages

- It makes the requirement simple and easily understandable by representing simple diagrams.
- One can covert ER diagrams into record based data model easily.
- Easy to understand ER diagrams

Disadvantages

- No standard notations are available for ER diagram. There is great flexibility in the notation. It's all depends upon the designer, how he draws it.
- It is meant for high level designs. We cannot simplify for low level design like coding.

3.3.2 Object Oriented Data Model

This data model is another method of representing real world objects. It considers each object in the world as objects and isolates

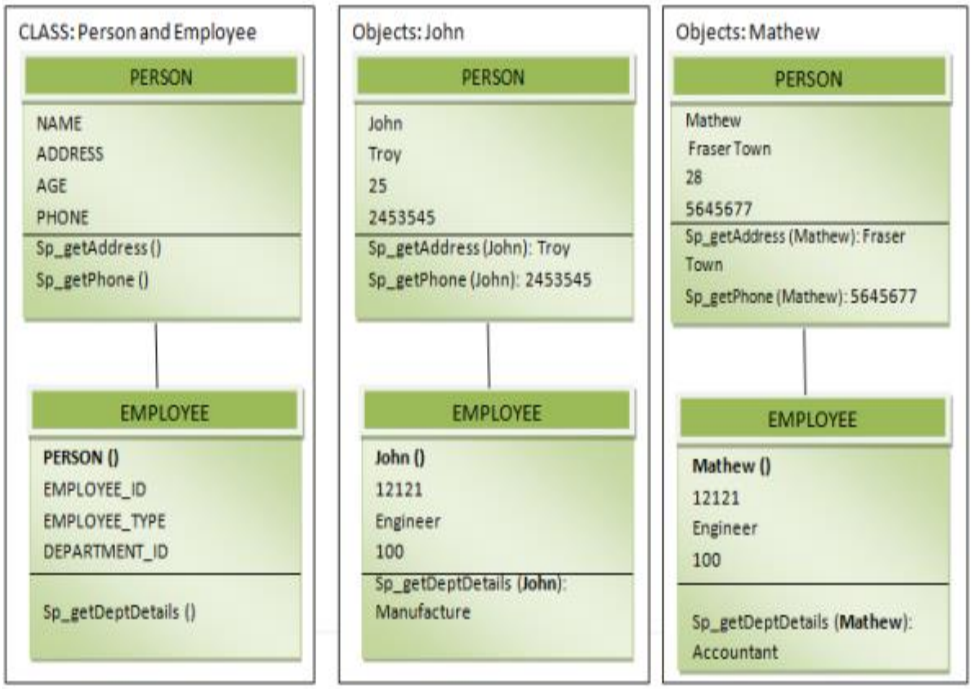
it from each other. It groups its related functionalities together and allows inheriting its functionality to other related sub-groups.

Let us consider an Employee database to understand this model better. In this database we have different types of employees – Engineer, Accountant, Manager, Clark. But all these employees belong to Person group. Person can have different attributes like name, address, age and phone. What do we do if we want to get a person's address and phone number? We write two separate procedure sp_getAddress and sp_getPhone.

What about all the employees above? They too have all the attributes what a person has. In addition, they have their EMPLOYEE_ID, EMPLOYEE_TYPE and DEPARTMENT_ID attributes to identify them in the organization and their department. We have to retrieve their department details, and hence we sp_getDeptDetails procedure. Currently, say we need to have only these attributes and functionality.

Since all employees inherit the attributes and functionalities of Person, we can re-use those features in Employee. But do we do that? We group the features of person together into class. Hence a class has all the attributes and functionalities. For example, we would create a person class and it will have name, address, age and phone as its attribute, and sp_getAddress and sp_getPhone as procedures in it. The values for these attributes at any instance of time are object. i.e. ; {John, Troy, 25, 2453545 : sp_getAddress (John), sp_getPhone (John)} forms on person object. {Mathew, Fraser Town, 28, 5645677: sp_getAddress (Mathew), sp_getPhone (Mathew} forms another person object.

Now, we will create another class called Employee which will inherit all the functionalities of Person class. In addition it will have attributes EMPLOYEE_ID, EMPLOYEE_TYPE and DEPARTMENT_ID, and sp_getDeptDetails procedure. Different objects of Employee class are Engineer, Accountant, Manager and Clerk.



Here we can observe that the features of Person are available only if other class is inherited from it. It would be a black box to any other classes. This feature of this model is called encapsulation. It binds the features in one class and hides it from other classes. It is only visible to its objects and any inherited classes.

Advantages

- Because of its inheritance property, we can re-use the attributes and functionalities. It reduces the cost of maintaining the same data multiple times. Also, these information are encapsulated and, there is no fear being misused by other objects. If we need any new feature we can easily add new class inherited from parent class and adds new features. Hence it reduces the overhead and maintenance costs.
- Because of the above feature, it becomes more flexible in the case of any changes.
- Codes are re-used because of inheritance.
- Since each class binds its attributes and its functionality, it is same as representing the real world object. We can see each object as a real entity. Hence it is more understandable.

Disadvantages

- It is not widely developed and complete to use it in the database systems. Hence it is not accepted by the users.

- It is an approach for solving the requirement. It is not a technology. Hence it fails to put it in the database management systems.

3.4 Physical Data Model

Physical data model represent the model where it describes how data are stored in computer memory, how they are scattered and ordered in the memory, and how they would be retrieved from memory. Basically physical data model represents the data at data layer or internal layer. It represents each table, their columns and specifications, constraints like primary key, foreign key etc. It basically represents how each tables are built and related to each other in the database.

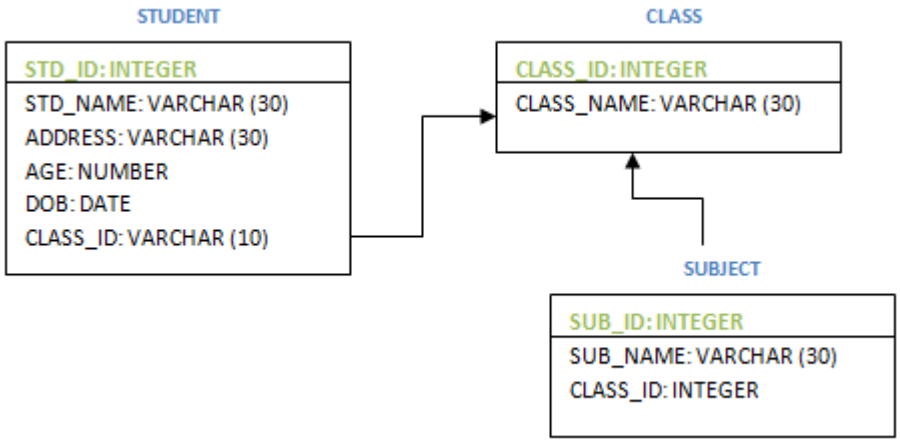


Fig: Physical data model

Above diagram shows how physical data model is designed. It is represented as UML diagram along with table and its columns. Primary key is represented at the top. The relationship between the tables is represented by interconnected arrows from table to table. Above STUDENT table is related to CLASS and SUBJECT is related to CLASS. The above diagram depicts CLASS as the parent table and it has 2 child tables – STUDENT and SUBJECT.

In short we can say a physical data model has

- Tables and its specifications – table names and their columns. Columns are represented along with their data types and size. In addition primary key of each table is shown at the top of the column list.

- Foreign keys are used to represent the relationship between the tables. Mapping between the tables are represented using arrows between them.
- Physical data model can have de-normalized structure based on the user requirement. The tables might not be in normalized forms.

Physical data model is dependent on the RDBMS i.e.; it varies based on the RDBMS used. This means data type notation varies depending on the RDBMS. For example, we have different data types in SQL server and oracle server. In addition, the representation of physical data model diagram may be different, though it contains same information as described above – some may represent primary key and foreign keys separately at the end of the column list. This data model depends on the user / designer how he specifies the diagram and the RDBMS servers. Below diagram shows different ways of representing a table.

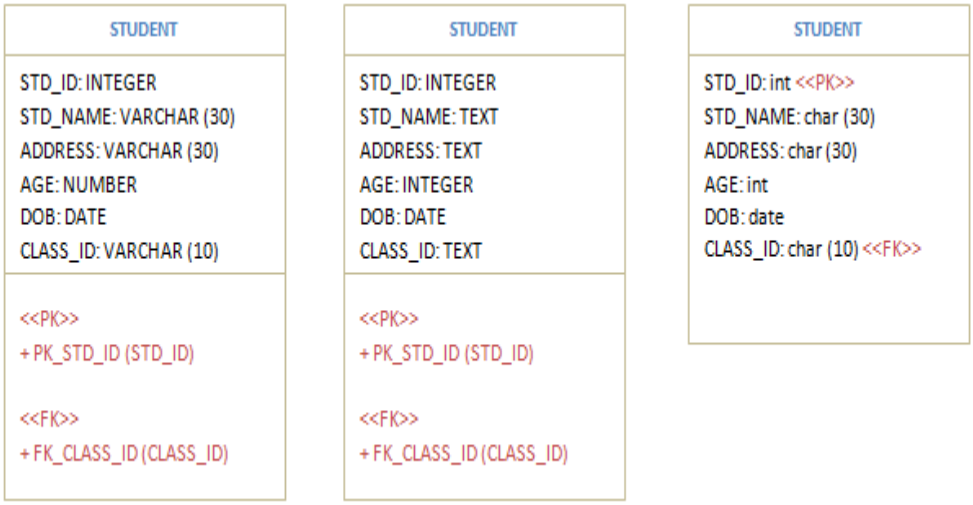


Fig: Different ways of representing a table

Hence object based data model is based on the real requirement from the user, whereas record based data model is based on the actual relationships and data in DB. The Physical data model is based on the table structure in the database.

3.5 Record Based Data Models

These data models are based on application and user levels of data. They are modeled considering the logical structure of the objects in the database. This data models defines the actual relationship between the data in the entities.

For example, employee and department entities are related to each other by means of department. This minute level of relationship is defined in the record based data models. We will not be able get the mapping at this level in the object based data models. There it simply defines two entities are related. But the actual relationship between any two entities can be observed in record based data models.

There are 3 types of record based data models defined so far- Hierarchical, Network and Relational data models. Most widely used record based data model is relational data model. Other two are not widely used. Let us understand how they are different from each other.

3.5.1 Hierarchical Data Model

Imagine we have to create a database for a company. What are the entities involved in it? Company, its department, its supplier, its employees, different projects of the company etc are the different entities we need to take care of. If we observe each of the entity they have parent –child relationship. We can design them like we do ancestral hierarchy. In our case, Company is the parent and rests of them are its children. Department has employees and project as its children and so on. This type of data modeling is called hierarchical data model.

In this data model, the entities are represented in a hierarchical fashion. Here we identify a parent entity, and its child entity. Again we drill down to identify next level of child entity and so on. This model can be imagined as folders inside a folder!

In our example above, it is diagrammatically represented as below:

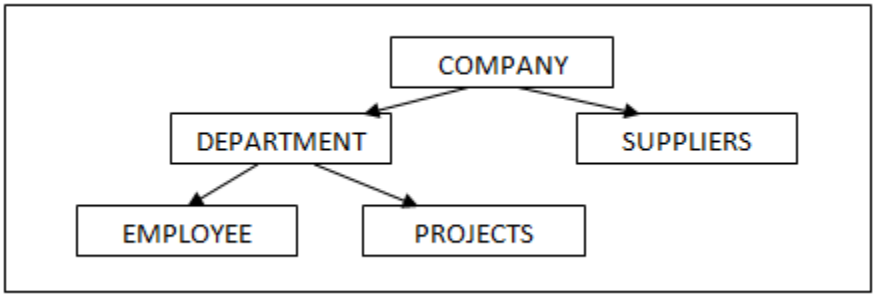


Fig: Example of Hierarchical Data Model

It can also be imagined as root like structure. This model will have only one main root. It then branches into sub-roots, each of which will branch again. This type of relationship is best defined for 1: N type of relationships. E.g.; One company has multiple departments

(1:N), one company has multiple suppliers (1:N),one department has multiple employees (1:N), each department has multiple projects(1:N) . If we have M: N relationships, then we have to duplicate the entities and show it in the diagram. For example, if a project in the company involves multiple departments, then our hierarchical representation changes as below:

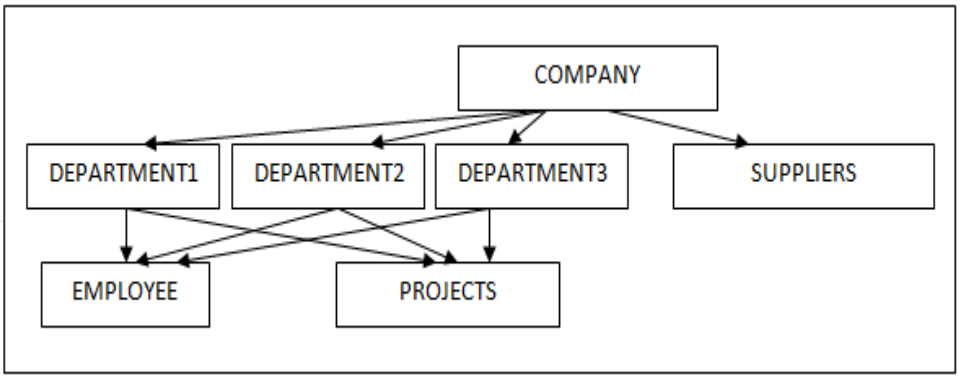


Fig: An Example of Modified Hierarchical Data Model

Advantages

- Simple and easy to use
- Data with hierarchical relationship can be mapped on this model.
- Suitable for application such as: Employee Dept**.**

Disadvantages:

- Search for (an element or) a record is difficult.
- Insertion, deletion and updating is difficult.
- Many – Many relationships cannot be established.
- Data with non hierarchical relationship cannot be mapped.
- The hierarchical relationship is maintained using points which require extra storage.
- Changes in relationship require changes in entire structure of the database.
- Processing is sequential among branches of the tree so access time is high.

3.5.2 Network Data Model

This is the extended version of hierarchical data model. It is designed to address the drawbacks of the hierarchical model. It helps to address M: N relationship. This data model is also represented as hierarchical, but this model will not have single parent concept. Any child in the tree can have multiple parents here.

A company has different projects and departments in the company own those projects. Even suppliers of the company give input for the project. Here Project has multiple parents and each department and supplier have multiple projects. This is represented as shown below. Basically, it forms a network like structure between the entities, hence the name.

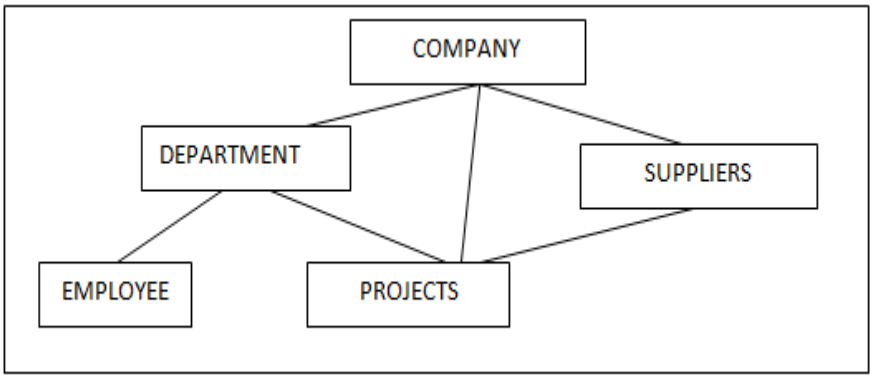


Fig: Network Data Model

Advantages

- Accessing the records in the tables is easy since it addresses many to many relationships. Because of this kind of relationship, any records can be easily pulled by using any tables. For example, if we want to know the department of project X and if we know SUPPLIER table, we can pull this information. i.e.; SUPPLIER has the information about project X which includes the departments involved in the projects too. Hence makes the accessibility to any data easier, and even any complex data can be retrieved easily and quickly.
- Because of the same feature above, one can easily navigate among the tables and get any data.
- It is designed based on database standards – ANSI/SPARC.

Disadvantages

- If there is any requirement for the changes to the entities, it requires entire changes to the database. There is no independence between any objects. Hence any changes to the any of the object will need changes to the whole model. Hence difficult to manage.
- It would be little difficult to design the relationship between the entities, since all the entities are related in some way. It

requires thorough practice and knowledge about the designing.

3.5.3 Relational Data Models

This model is designed to overcome the drawbacks of hierarchical and network models. It is designed completely different from those two models. Those models define how they are structured in the database physically and how they are inter-related. But in the relational model, we are least bothered about how they are structured. It purely based on how the records in each table are related. It purely isolates physical structure from the logical structure. Logical structure is defines records are grouped and distributed.

Let us try to understand it by an example. Let us consider department and employee from our previous examples above. In this model we look at employee with its data. When we say an employee what all comes into our mind? His employee id, name, address, age, salary, department that he is working etc. are attributes of employee. That means these details about the employee forms columns in employee table and value set of each employee for these attribute forms a row/record for an employee. Similarly, department has its id, name.

Now, in the employee table, we have column which uniquely identifies each employee – that is employee Id column. This column has unique value and we are able to differentiate each employee from each other by using this column. Such column is called as primary key of the table. Similarly department table has DEPT_ID as primary key. In the employee table, instead of storing whole information about his department, we have DEPT_ID from department table stored. i.e.; by using the data from the department table, we have established the relation between employee and department tables.

EMPLOYEE

EMP_ID	EMP_NAME	ADDRESS	DEPT_ID
100	Joseph	Clinton Town	10
101	Rose	Fraser Town	20
102	Mathew	Lakeside Village	10
103	Stewart	Troy	30
104	William	Holland	30

DEPARTMENT

DEPT_ID	DEPT_NAME
10	Accounting
20	Quality
30	Design

Fig: Relational Data Models

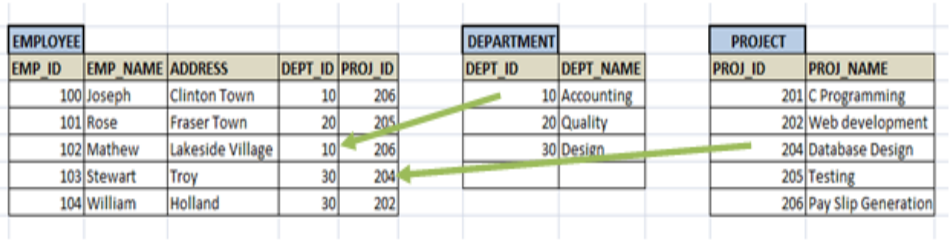
Observe the table structures above. They are very simple to understand. There is no redundant data as well. It addressed major drawback of earlier data models. This type of data model is called relational data model.

This model is based on the mathematical concepts of set theory. It considers the tables as a two dimensional table with rows and columns. It is least bothered about the physical storage of structure and data in the memory. It considers only the data and how it can be represented in the form of rows and columns, and the way it can establish the relation between other tables.

3.5.3.1 Five important rules relational data model

1. Order of rows / records in the table is not important. For example, displaying the records for Joseph is independent of displaying the records for Rose or Mathew in Employee table. It does not change the meaning or level of them. Each record in the table is independent of other. Similarly, order of columns in the table is not important. That means, the value in each column for a record is independent of other. For example, representing DEPT_ID at the end or at the beginning in the employee table does not have any affect.
2. Each record in the table is unique. That is there is no duplicate record exists in the table. This is achieved by the use of primary key or unique constraint.
3. Each column/attribute will have single value in a row. For example, in Department table, DEPT_NAME column cannot have 'Accounting' and 'Quality' together in a single cell. Both have to be in two different rows as shown above.
4. All attributes should be from same domain. That means each column should have meaningful value. For example, Age column cannot have dates in it. It should contain only valid numbers to represent individual's age. Similarly, name columns should have valid names; Date columns should have proper dates.
5. Table names in the database should be unique. In the database, same schema cannot contain two or more tables with same name. But two tables with different names can have same column names. But same column name is not allowed in the same table.

   Examine below table structure for Employee, Department and Project and see if it satisfies relational data model rules.



EMPLOYEE

EMP_ID	EMP_NAME	ADDRESS	DEPT_ID	PROJ_ID
100	Joseph	Clinton Town	10	206
101	Rose	Fraser Town	20	205
102	Mathew	Lakeside Village	10	206
103	Stewart	Troy	30	204
104	William	Holland	30	202

DEPARTMENT

DEPT_ID	DEPT_NAME
10	Accounting
20	Quality
30	Design

PROJECT

PROJ_ID	PROJ_NAME
201	C Programming
202	Web development
204	Database Design
205	Testing
206	Pay Slip Generation

Fig: Example of Relational Data Models

3.5.3.2 Properties of a Relation/Table:

1. The data represented in this model is in the form of two dimensional tables called relation.
2. An entity is represented by a tuple in a row.
3. The rows of a table are distinct.
4. Ordering of rows is immaterial.
5. Each column of the table is assigned distinct heading called name of the attribute.
6. In each column data item are of similar type.
7. The ordering of the columns is immaterial.
8. If there are M. columns, it is said to be of degree m.
9. If there are N rows, it is called an N tuple table or cardinality of the table is N.
10. Both the rows and columns can be viewed in any sequence at any time without affecting the information.

3.5.3.3 Properties of Relational Data Base Management System

- A relational database management is represented by relational data models.
- It uses a collection of tables to represent the data and the relationship among them.
- Each table has multiple no of rows and columns
- Supports the concept of null values.
- Does not require the user to understand its physical implementation.
- Provides information about its contents and structure.

3.5.3.4 Domain and Integrity Constraints:

Domain Constraints:

- Limit the range of domain an attribute values on.
- Specify uniqueness and nullness of attributes.
- Specify default value of an attribute.

Integrity constraints:

Entity integrity: Every tuple is uniquely identified by a unique non null attribute, primary key i.e. the primary key values cannot be null.

Referential integrity: Rows in different tables are correctly related by valid key values (Foreign keys refers to primary keys)

3.5.3.5 Basic terminologies of relational data model

1. **Entity:** A real world object with some properties which can be easily identified is called an entity.
2. **Attributes:** An attribute is a descriptive property or a characteristic of an entity. Ex name, roll marks, etc.
3. **Degree:** The no of columns or attributes associated with a table (or a relation among them) is called degree of the relation.
4. **Cardinality:** The no of rows in a table is called cordiality.
5. **Tuples:** A relation consisting of a number of records represented in row wise information called tupelos.
6. **Domain:** The set of possible values that can allot to an attribute is called domain of that attribute. Domain d is a set of atomic values of an attribute.
7. **Entity set:** A set of entities representing a real world object. Ex. Student, teacher, depositor, player etc.
8. **Weak entity:** An entity set which may not have sufficient attribute to form a primary key then it is called an weak entity.
9. **Strong entity:** An entity set which have a key attribute.
10. **Key:** An attribute that allows us to uniquely identity a record or an entity type.
11. **Super key:** A set of attributes that collectively allows us to identify an entity is an entity set.
12. **Candidate key:** A candidate key is the minimal super key. The super key for which no proper subset is a super key.
13. **Primary key:** Primary key is the one of the candidate key to identify the entity uniquely. The primary key is the principal means of identifying an entity.
14. **Entity type:** The occurrences of an entity set are called entities
15. **Composite key:** A composite key is a candidate key that consists of two or more attributes.
16. **Foreign key:** A foreign key is an attribute (a group of attributes) that is primary key to another relation. A ford on key represents a relationship between two tables.

Advantages of relational data model

- **Structural independence**: Any changes to the database structure, does not the way we are accessing the data. For example, Age is added to Employee table. But it does not change the relationship between the other tables nor changes the existing data. Hence it provides the total independence from its structure.
- **Simplicity**: This model is designed based on the logical data. It does not consider how data are stored physically in the memory. Hence when the designer designs the database, he concentrates on how he sees the data. This reduces the burden on the designer.
  Because of simplicity and data independence, this kind of data model is easy to maintain and access. This model supports structured query language – SQL.
  Hence it helps the user to retrieve and modify the data in the database. By the use of SQL, user can get any specific information from the database.

Disadvantages of relational data model

- Compared to the advantages above, the disadvantages of this model can be ignored.
- High hardware cost: - In order to separate the physical data information from the logical data, more powerful system hardware – memory is required. This makes the cost of database high.
- Sometimes, design will be designed till the minute level, which will lead to complexity in the database.
- Below table provides the comparison among the three models as follows:

Table: comparison among the three models

Hierarchical Data Model	**Network Data Models**	**Relational Data Models**
Supports One-to-Many Relationship	Supports both One-to-Many and Many-to-Many relationship	Supports both One- to-Many and Many to Many relationship
Because of single parent-child relationship, difficult to navigate through the child	It establishes the relationship between most of the objects, hence easy to access compared to hierarchical model	It provides SQL, which makes the access to the data simpler and quicker.

Flexibility among the different object is restricted to the child.	Because of the mapping among the sub level tables, flexibility is more	Primary key and foreign key constraint makes the flexibility much simpler than other models.
Based on the physical storage details	Based on the physical storage details	Based on the logical data view

3.6 Let Us Sum Up

A data model is a plan for building a database. The data model is a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

To be effective, it must be simple enough to communicate to the end user the data structure required by the database yet detailed enough for the database design to use to create the physical structure.

The conventional data models like Network data Models and hierarchical databases had their disadvantages, they were very rigid. The set relationships and the structure of the records had to be specified in advance. Changing the database structure typically required rebuilding the entire database. But Relational data models are more popular and widely used because of its simplicity, Structural independence. The Entity-Relation Model (ER) is the most common method used to transform the relational model to relational databases. In the next unit we will provides a detailed discussion on the concepts or ER Model and Relational Models.

3.7 Self Assessment Questions

1. Define data model. Name different types of record based data models.

   ……
   ……
   ……
   ……

2. What is a Physical Data Model? What is the purpose of a Physical Data Model?

   ……
   ……
   ……
   ……

3. Discuss the advantages and disadvantages of hierarchical data model.

……
……
……
……

4. What is an Entity Relationship Data Model? Write its advantages and disadvantages.

……
……
……
……

5. Explain the merits and demerits of relational data model.

……
……
……
……

3.8 Model Questions

1. What is an Object Oriented Data Model? Write its advantages and Disadvantages.
2. What are different records based data models? Discuss their relative merits and demerits.
3. Explain the five basic rules of relational databases?
4. Explain the properties of a relational data model.
5. Describe the merits and demerits of Network Data Model.
6. Distinguish between logical and physical database design.
7. Explain the difference between external, internal, and conceptual schemas.

3.9 References and Further Reading

1. "Database System Concepts" by Abraham Silberschatz, Henry Korth, and S. Sudarshan, Mc Grow Hill
2. "Fundamentals of Database Systems" by R. Elmasri and S. Navathe, Pearson Education.
3. S.K Singh, Database Systems, Concepts, Design and Applications, Pearson Education.
4. http://ecomputernotes.com/fundamental/what-is-a-database/type-of-data-models
5. http://ecomputernotes.com/fundamental/what-is-a-database/database-model
6. https://www.tutorialcup.com/dbms/object-based-data-models.htm

UNIT-4 RELATIONAL DATA MODELING USING ENTITY-RELATIONSHIP MODEL

UNIT STRUCTURE

4.0 Introduction

ER Data Model is based on the real world objects and their relationship. In other words, each and everything, either living or non-living things in this world forms the object in database world. We identify all the required objects for our database requirement and give the shape of database objects. Before putting them into database, it is very much essential to understand the requirement properly and design them efficiently. It is like a foundation of the building. For this purpose, we use ER diagrams where we plan the database pictorially. ER diagram basically breaks requirement into entities, attributes and relationship. In this unit we will discuss in detail about the E-R Data model and how to convert the E-R model to relational tables.

4.1 Learning Objectives

After learning this unit you should be able to

- Define an entity and identify different types of entities
- Define an Attribute and A relationship
- Identify different types of Keys and Constraints
- Know the symbols used to represent the ER Diagram Symbols
- Understand the principles of Conversion of ER diagrams to Relations

4.2 Entity

In a database, we would be grouping only related data called attributes together and storing them under one group name called Entity. In relational form an entity is represented in a Table. This helps in identifying which data is stored where and under what name. It reduces the time to search for a particular data in a whole database. Say, we are creating a School database. Then what all things come into our mind? It is school, students, teachers, subjects, class etc. If you observe what we have listed are items/names which are part of school, but each of them having their own group of related information? i.e.; Student has ID, name, address, DOB, class in which he is studying etc. for a Student. Similarly, TEACHERS will have their ID, name, address subjects that they are teaching, class which they are teaching etc. Hence each entity forms a group of related information called attributes. In real world each and every object forms an entity. In short, all the living and non-living things

in the world form entity. One can consider all nouns as entities, if that makes it simpler.

- **Strong Entity**: Entities having its own attribute(s) as primary key(s) are called strong entity. For example, STUDENT has STUDENT_ID as primary key. Hence it is a strong entity.
- **Weak Entity**: Entities which cannot form their own attribute as primary key are known as weak entities. These entities will derive their primary keys from the combination of its attribute and primary key from its mapping entity.

  Consider CLASS and SECTION entity. The SECTION has SECTION _ID and NAME as its attribute. But SECTION _ID alone cannot be a primary key, since it fails to tell for which course it is related to. We will not be uniquely identifying the course section by this attribute alone. But if this attribute along with CLASS_ID gives the meaning for each section and we can uniquely identify the sections.

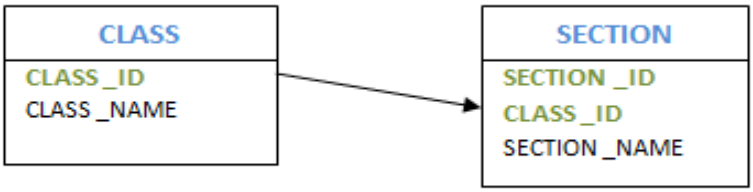


Fig: Strong and Weak Entity

- **Composite Entity**: Entities participating in the many to many relationships are called composite entity. In this case, apart from two entities that are part of relation, we will one more hidden entity in the relation. We will be creating a new entity with the relation, and create a primary key by using the primary keys of other two entities.

  Consider the example, multiple students enrolled for multiple courses. In this case, we create STUDENT and COURSE. Then we create one more table for the relation 'Enrolment' and name it as STUD_COURSE. Add the primary keys of COURSE and STUDENT into it, which forms the composite primary key of the new table.

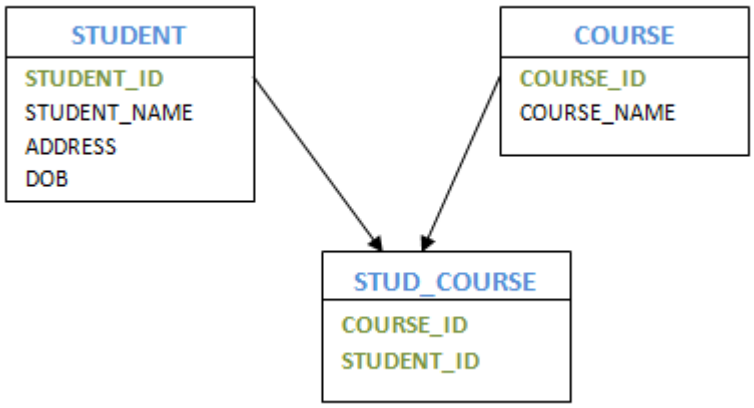


Fig: Composite Entity

- **Recursive Entity**: If a relation exists between the same entities, then such entities are called as recursive entity. For example, mapping between manager and employee is recursive entity. Here manager is mapped to the same entity Employee. HOD of the department is another example of having recursive entity.

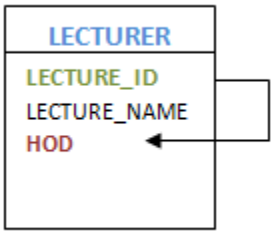


Fig: Recursive Entity

4.3 Attribute

- The characteristics of an entity are called attributes. Ex. Roll no, name, marks, etc are the attributes of an entity student.
- An attribute is a descriptive property of an entity
- An attribute can have single value or multiple value or range of values.
- In addition, each attribute can contain certain type of data like only numeric value, or only alphabets, or combination of both, or date or negative or positive values etc.

Depending on the values that an attribute can take, it is divided into different types.

1. **Simple Attribute**
   These kinds of attributes have values which cannot be divided further. For example, STUDENT_ID attribute which cannot be divided further. Passport Number is unique value and it cannot be divided.
2. **Composite Attribute**
   This kind of attribute can be divided further to more than one simple attribute. For example, address of a person. Here address can be further divided as Door#, street, city, state and pin which are simple attributes.
3. **Derived Attribute**
   Derived attributes are the one whose value can be obtained from other attributes of entities in the database. For example, Age of a person can be obtained from date of birth and current date. Average salary, annual salary, total marks of a student etc are few examples of derived attribute.
4. **Stored Attribute**
   The attribute which gives the value to get the derived attribute are called Stored Attribute. In example above, age is derived using Date of Birth. Hence Date of Birth is a stored attribute.
5. **Single Valued Attribute**
   These attributes will have only one value. For example, EMPLOYEE_ID, passport#, driving license#, SSN etc have only single value for a person.
6. **Multi-Valued Attribute**
   These attribute can have more than one value at any point of time. Manager can have more than one employee working for him, a person can have more than one email address, and more than one house etc is the examples.
7. **Simple Single Valued Attribute**
   This is the combination of above four types of attributes. An attribute can have single value at any point of time, which cannot be divided further. For example, EMPLOYEE_ID – it is single value as well as it cannot be divided further.
8. **Simple Multi-Valued Attribute**
   Phone number of a person, which is simple as well as he can have multiple phone numbers is an example of this attribute.

9. **Composite Single Valued Attribute**
   Date of Birth can be a composite single valued attribute. Any person can have only one DOB and it can be further divided into date, month and year attributes.
10. **Composite Multi-Valued Attribute**
    Shop address which is located two different locations can be considered as example of this attribute.
11. **Descriptive Attribute**
    Attributes of the relationship is called descriptive attribute. For example, employee works for department. Here 'works for' is the relation between employee and department entities. The relation 'works for' can have attribute DATE_OF_JOIN which is a descriptive attribute.

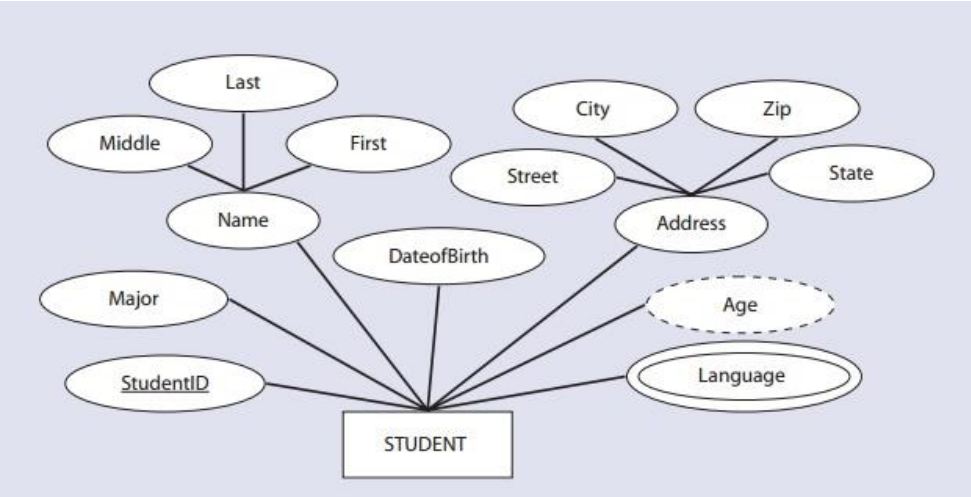


Fig: Attributes of the STUDENT entity type

4.4 Keys

- An important constraint on the entities of an entity type is the key **or** uniqueness constraint **on attributes.**
- A key is an attribute (also known as column or field) or a combination of attribute that is used to identify records.
- Sometimes we might have to retrieve data from more than one table, in those cases we require to join tables with the help of keys.
- The purpose of the key is to bind data together across tables without repeating all of the data in every table
- Such an attribute is called a **key attribute**, and its values can be used to identify each entity uniquely.

- For example, the Name attribute is a key of the COMPANY entity type because no two companies are allowed to have the same name.
- For the PERSON entity type, a typical key attribute is Social Security Number.
- Sometimes, several attributes together form a key, meaning that the combination of the attribute values must be distinct for each entity.
- If a set of attributes possesses this property, we can define a composite attribute that becomes a key attribute of the entity type.

The various types of key with e.g. in SQL are mentioned below,
(For examples let suppose we have an Employee Table with attributes 'ID' , 'Name' ,'Address' , 'Department_ID' ,'Salary')

4.1 Types of Keys

(a) Super Key - An attribute or a combination of attribute that is used to identify the records uniquely is known as Super Key. A table can have many Super Keys.

Examples of Super Keys:

- ID
- ID, Name
- ID, Address
- ID, Department_ID
- ID, Salary
- Name, Address
- Name, Address, Department_ID and so on.

As any combination which can identify the records uniquely will be a Super Key.

(b) Candidate Key - It can be defined as minimal Super Key or irreducible Super Key. In other words an attribute or a combination of attribute that identifies the record uniquely but none of its proper subsets can identify the records uniquely.

Examples of Candidate Key:

- Code
- Name, Address

For above table we have only two Candidate Keys (i.e. Irreducible Super Key) used to identify the records from the table uniquely. Code Key can identify the record uniquely and similarly combination of Name and Address can identify the record uniquely, but neither Name nor Address can be used to identify the records

uniquely as it might be possible that we have two employees with similar name or two employees from the same house.

(c)Primary Key - A Candidate Key that is used by the database designer for unique identification of each row in a table is known as Primary Key. A Primary Key can consist of one or more attributes of a table.

Example of Primary Key: Database designer can use one of the Candidate Key as a Primary Key. In this case we have "Code" and "Name, Address" as Candidate Key, we will consider "Code" Key as a Primary Key as the other key is the combination of more than one attributes.

(d)Foreign Key - A foreign key is an attribute or combination of attribute in one base table that points to the candidate key (generally it is the primary key) of another table. The purpose of the foreign key is to ensure referential integrity of the data i.e. only values that are supposed to appear in the database are permitted.

Examples of Foreign Key –

Let consider we have another table i.e. Department Table with Attributes "Department_ID", "Department_Name", "Manager_ID", "Location_ID" with Department_ID as an Primary Key. Now the Department_ID attribute of Employee Table (dependent or child table) can be defined as the Foreign Key as it can reference to the Department_ID attribute of the Departments table (the referenced or parent table), a Foreign Key value must match an existing value in the parent table or be NULL.

(e)Composite Key - If we use multiple attributes to create a Primary Key then that Primary Key is called Composite Key (also called a Compound Key or Concatenated Key).

Examples of Composite Key: if we have used "Name, Address" as a Primary Key then it will be our Composite Key.

(f) Alternate Key - Alternate Key can be any of the Candidate Keys except for the Primary Key.

Examples of Alternate Key is "Name, Address" as it is the only other Candidate Key which is not a Primary Key.

(g) Secondary Key - The attributes that are not even the Super Key but can be still used for identification of records (not unique) are known as Secondary Key.

Examples of Secondary Key can be Name, Address, Salary, Department_ID etc. as they can identify the records but they might not be unique.

4.5 Relationship

- There are several implicit relationships among the various entity types.
- In fact, whenever an attribute of one entity type refers to another entity type, some relationship exists.
- For example, the attribute Manager of department refers to an employee who manages the department.
- In the ER model, these references should not be represented as **relationships** or **relation**. There is a relation "borrower" in the entities customer and account which can be shown as follows:

4.5.1 Degrees of Relationship

In a relationship two or more number of entities can participate. The number of entities who are part of a particular relationship is called degrees of relationship. If only two entities participate in the mapping, then degree of relation is 2 or binary. If three entities are involved, then degree of relation is 3 or ternary. If more than 3 entities are involved then the degree of relation is called n-degree or n-nary.

4.5.2 Cardinality of Relationship

How many number of instances of one entity is mapped to how many number of instances of another entity is known as cardinality of a relationship. In a 'studies' relationship above, what we observe is only one Student X is studying in on Class Y. i.e.; single instance of entity student mapped to a single instance of entity Class. This means the cardinality between Student and Class is 1:1.

4.5.3 Based On the Cardinality

One-to-One (1:1): As we saw in above example, one instance of the entity is mapped to only one instance of another entity.

Consider, HOD of the Department. There is only one HOD in one department. That is there is 1:1 relationship between the entity HOD and Department.

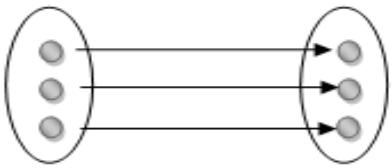


Fig: One-to-One (1:1) relationship

One-to-Many (1: M): As we can guess now, one -to- many relationship has one instance of entity related to multiple instances

of another entity. One manager manages multiple employees in his department. Here Manager and Employee are entities, and the relationship is one-to-many. Similarly, one teacher teaches multiple classes are also a 1: M relationship.

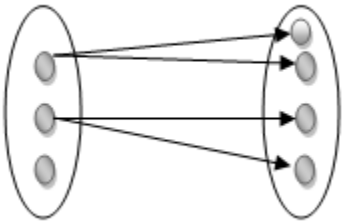


Fig: One-to-Many (1: M) relationship

Many-to-Many (M: N): This is a relationship where multiple instances of entities are related to multiple instances of another entity. A relationship between TEACHER and STUDENT is many-to- many. How? Multiple Teachers teach multiple numbers of Students.

Similarly, above example of 1:1 can be M: N!! Surprised? Yes, it can be M: N relationship, provided, how we relate these two entities. Multiple Students enroll for multiple classes/courses makes this relationship M: N. The relationship 'studies' and 'enroll' made the difference here. That means, it all depends on the requirement and how we are relating the entities.

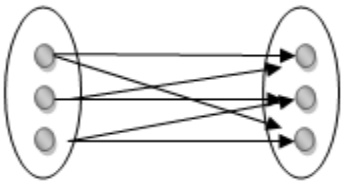


Fig: Many-to-Many (M: N) Relationship

4.6 Constraints

This represents how an entity is involved in the relation. That means, if all the entity values are participating in any relation, then it is called total participation. If only few values of an entity are part of relation, then it is a partial participation.

For example, 'Employee Works for a Department'. Here all the employees work for one or the other department. No employees are left without department. Hence all the employees are participating

in this relation. Thus, participation of employee is total participation.

But individual Department will not have all the employees. Each department will have only few groups of employees working. Hence partial participation of department is seen here.

4.7 ER Diagram Symbols

Since ER diagram is the pictorial representation of real world objects, it involves various symbols and notation to draw the diagrams. Let us see one by one below.

Entity: Rectangles are used to represent the entity in the diagram. Name of the Entity is written inside the rectangle.

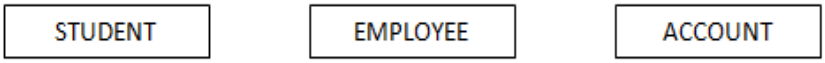


Fig: Entity Types

A strong entity is represented by simple rectangle as shown above.

A weak entity is represented by two rectangles as shown below.

Fig: Weak Entity Sets

Attribute: An oval shape is used to represent an attribute. Name of the attribute is written inside the oval shape and is connected to its entity by a line.

Multi-value attributes are those which can take up more than one values are represented by double oval shapes.

Derived attributes are those which can be derived from other attributes are represented by oval shapes with dashed lines.

A composite attribute is an attribute that has two or more simple attributes as its components. Each simple attribute is represented by an oval shape, but these attributes will be connected to its parent attribute called composite attribute forming a tree structure.

The below mentioned figure, gives the representation of composite attribute, here address of a student.

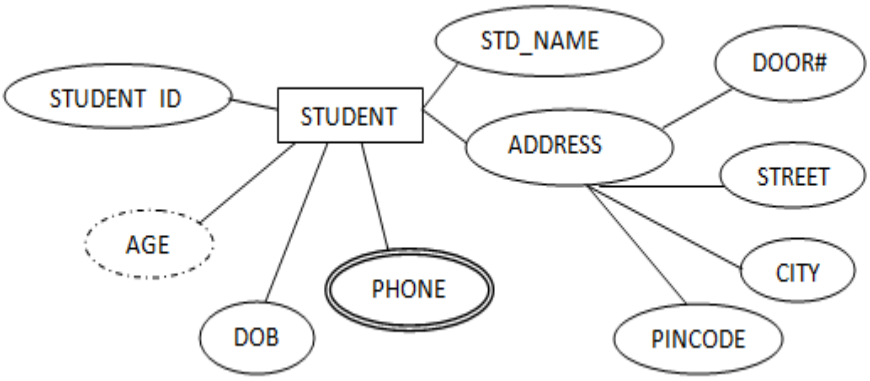


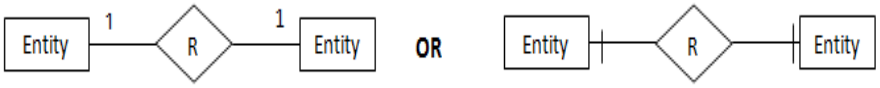
Fig: Example of attributes

Primary Key: An underline to the attribute name is put to represent the primary key. The key attribute of the weak entity is represented by dashed underline.

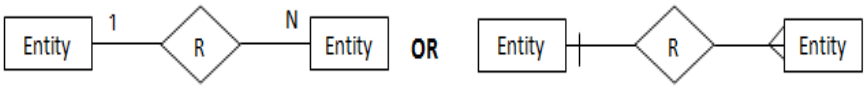
Relationship: A diamond shape is used to show the relationship between the entities. A mapping with weak entity is shown using double diamond. Relationship name will be written inside them.

Cardinality of Relationship: Different developers use different notation to represent the cardinality of the relationship. Not only for cardinality, but for other objects in ER diagram will have slightly different notations. But main difference is noticed in the cardinality. For not to get confused with many, let us see two types of notations for each.

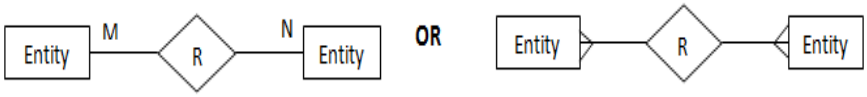
One-to-One relation: - A one-to-one relationship is represented by adding '1' near the entities on the line joining the relation. In another type of notation one dash is added to the relationship line at both ends.



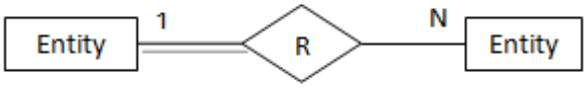
One-to-Many relation: A one-to-many relationship is represented by adding '1' near the entity at left hand side of relation and 'N' is written near the entity at right side. Other type of notation will have dash at LHS of relation and three arrow kind of lines at the RHS of relation as shown below.



Many-to-Many relation: A one-to-many relationship is represented by adding 'M' near the entity at left hand side of relation and 'N' is written near the entity at right side. Other type of notation will have three arrow kinds of lines at both sides of relation as shown below.



Participation Constraints: Total participation constraints are shown by double lines and partial participations are shown as single line.



Let us create a simple ER diagram for a STUDENT database. What is the requirement of this database?

'Student attends class. Each class is divided into one or more sections. Each class will have its own specified subjects. Students have to attend all the subjects of the class that he attends'.

Now let us identify what are the entities? STUDENT, CLASS, SECTION, SUBJECT are the entities. Attributes of these entities are not specified here. But we know what could be the entities of each of the entities. We can list them as below at this point of time.

STUDENT	CLASS	SECTION	SUBJECT
STUDENT_ID	CLASS_ID	SECTION_ID	SUBJECT_ID
STUDENT_NAME	CLASS_NAME	SECTION_NAME	SUBJECT_NAME
ADDRESS			

DOB			
AGE			
CLASS_ID			
SECTION_ID			

What are the relationships we have? ‘Attends’, ‘has section’, ‘have subjects’ and ‘studies subjects’ are the relations here. With this knowledge of requirement, we can draw the ER diagram as below.

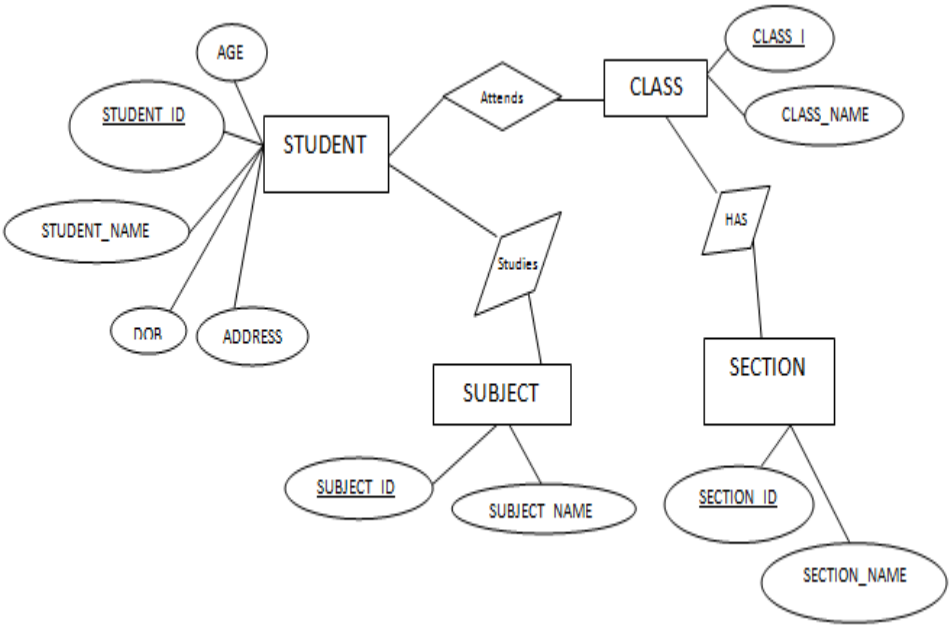


Fig: Example ER Diagram with Entities, Attribute, and Relationships

Observe the diagram carefully. Did we miss or drew it correctly? Are we missing anything on the diagram? Is it inferring correct requirement? What are our observations?

- Age attribute can be derived from DOB. Hence we have to draw dashed oval.
- Address is a composite attribute. We have to draw its sub attributes too. So that we will be very clear about his address details.
- If we see the SECTION entity, by section id, will we be able get the section that student attends? There is no relation mentioned between Student and Section. But Section is mapped only with Class. What do we understand from this? Section is a weak entity. Hence we have to represent it properly.

- If we look at ‘attends’ relationship between STUDENT and CLASS, we can have ‘Joining Date’ and ‘Total Number of Hours’ attributes. But it is an attribute of relation. We have to show them in the diagram.
- Since each class is having different subjects and Students attends those subjects, we can modify the relation ‘studies’ to ‘has’ relation on the **relation** ‘attends’.

Now the diagram will change to reflect all above points.

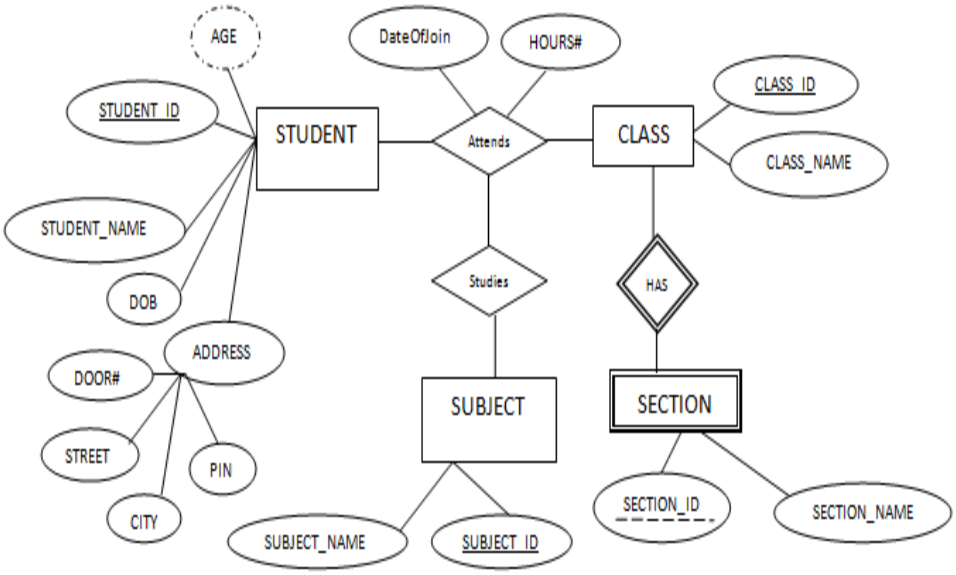


Fig: Example ER Diagram with Strong Entities, Weak Entities, and Simple attribute, Composite Attribute, Keys and Relationships etc.

Is the above diagram is a complete diagram? No, we have not added the cardinality and participation in the diagram.

4.7.1 Participation Constraints

- All the Students attend any one of the class, but class can have only certain group of students. Hence total participation of Students and partial participation of class in ‘Attends’ relation.
- All the class has section and all the section has class. Hence both are total participation.
- All the Students study some of the subject’s specific for their class and each class has only some group of subjects. Hence partial participation of both STUDENT and CLASS. Each subject will be studied by some students and it will be part of some class. Hence this also partial participation.

4.7.2 Cardinalities of All the Relationship

- Each Student attends only one class at a time. Hence it is a **1: 1** relation.
- Each class has one or more sections. Hence it can be considered as **1: N** relation.
- Each student attends many subjects and each class has many subjects. Hence it is a **1: N** relation.

Note: If you look at STUDENT and CLASS relationship as many Students attend one class, then it would be an **M: 1** relation. It is all up to the developer, how he looks at the requirement.

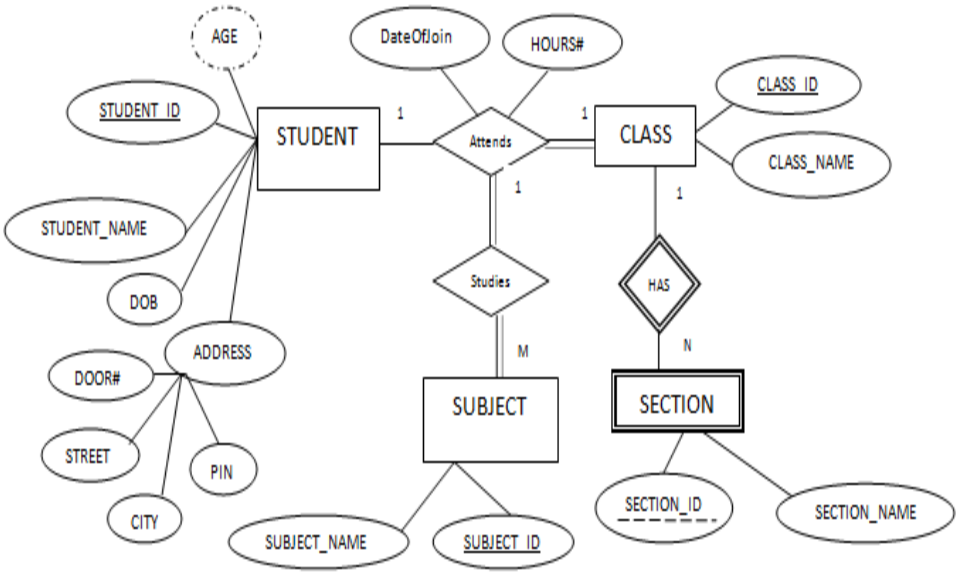


Fig: A Complete ER diagram for simple Student database

Now it is a complete ER diagram for simple Student database including the essential components of ER Model.

4.8 Conversion of ER diagram to Relations

As the database grows, the ER diagram representation becomes more complex and crowded. It is creates a difficult situation to understand the requirement and their structure as a whole. Similarly, if ER diagram is represented at very high level, it again creates a difficulty in understanding the system. But representation at high level and till the minute levels is very necessary to understand the system well. These concepts are well defined by some extended concepts called **generalization** and **specialization**. Sometimes, we would have divided the entities into two or more

entities to be more accurate in design. But when compared to the whole database or user, it can be combined to one entity. Such a process is called as **aggregation**.

Once designing ER diagram is complete, we need to put it into logical structure. But how it can be done? Let us discuss this in the last section.

4.8.1 Generalization

- The refinement from an initial entity set into successive levels of entity sub groupings 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. The database designer may have first identified a customer entity set with the attributes name, street, city, and customer-id, and an employee entity set with the attributes name, street, city, employee-id, and salary. There are similarities between the customer entity set and the employee entity set in the sense that they have several attributes in common. This commonality can be expressed by generalization, which is a containment relationship that exists between a higher-level entity set and one or more lower-level entity sets. In our example, person is the higher-level entity set and customer and employee are lower-level entity sets.
- Higher- and lower-level entity sets also may be designated by the terms super class and subclass, respectively. The person entity set is the super class of the customer and employee subclasses.
- For all practical purposes, generalization is a simple inversion of specialization. We will apply both processes, in combination, in the course of designing the E-R schema for an enterprise. In terms of the E-R diagram itself, we do not distinguish between specialization and generalization. New levels of entity representation will be distinguished (specialization) or synthesized (generalization) as the design schema comes to express fully the database application and the user requirements of the database.

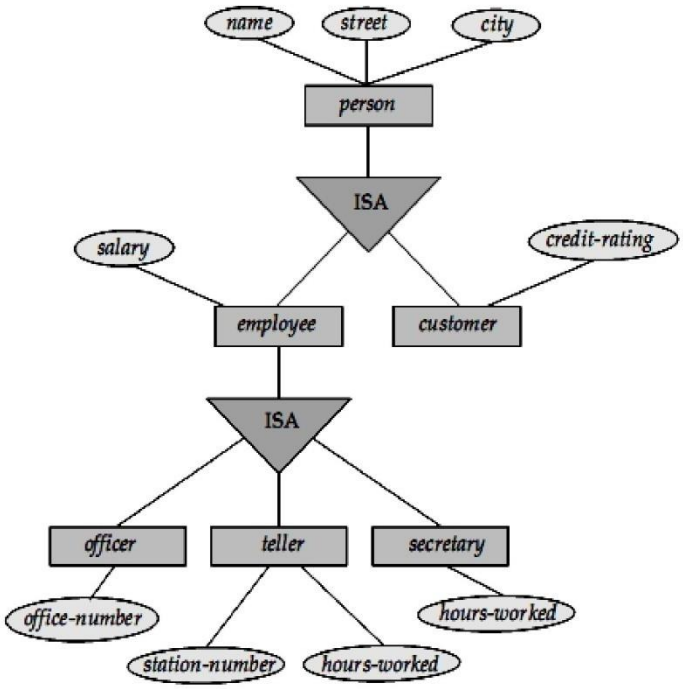


Fig: Specialization and generalization

4.8.2 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 E-R model provides a means for representing these distinctive entity groupings.
- Consider an entity set person, with attributes name, street, and city. A person may be further classified as one of the following:
- Customer
- Employee
- Each of these person types is described by a set of attributes that includes all the attributes of entity set person plus possibly additional attributes.
- For example, customer entities may be described further by the attribute customer-id, whereas employee entities may be described further by the attributes employee-id and salary.
- The process of designating sub groupings within an entity set is called specialization.

- The specialization of person allows us to distinguish among persons according to whether they are employees or customers.
- As another example, suppose the bank wishes to divide accounts into two categories, checking account and savings account. Savings accounts need a minimum balance, but the bank may set interest rates differently for different customers, offering better rates to favored customers.
- Checking accounts have a fixed interest rate, but offer an overdraft facility; the overdraft amount on a checking account must be recorded.
- The bank could then create two specializations of account, namely savings-account and checking-account.
- As we saw earlier, account entities are described by the attributes account-number and balance.
- The entity set savings-account would have all the attributes of account and an additional attribute interest-rate.
- The entity set checking-account would have all the attributes of account, and an additional attribute overdraft-amount.
- We can apply specialization repeatedly to refine a design scheme. For instance, bank employees may be further classified as one of the category such as Officer, Teller, Secretary, etc.
- Each of these employee types is described by a set of attributes that includes all the attributes of entity set employee plus additional attributes. For example, officer entities may be described further by the attribute office-number, teller entities by the attributes station-number and hours-per-week, and secretary entities by the attribute hours-per-week. Further, secretary entities may participate in a relationship secretary-for, which identifies which employees are assisted by a secretary.
- An entity set may be specialized by more than one distinguishing feature. In our example, the distinguishing feature among employee entities is the job the employee performs. Another, coexistent, specialization could be based on whether the person is a temporary (limited-term) employee or a permanent employee, resulting in the entity sets temporary- employee and permanent-employee. When more than one specialization is formed on an entity set, a particular entity may belong to multiple specializations. For instance, a given employee may be a temporary employee who is a secretary.
- In terms of an E-R diagram, specialization is depicted by a triangle component labeled ISA. The label ISA stands for "is a" and

represents, for example, that a customer "is a" person. The ISA relationship may also be referred to as a super class- subclass relationship. Higher- and lower-level entity sets are depicted as regular entity sets—that are, as rectangles containing the name of the entity set.

4.8.3 Aggregation

- One limitation of the E-R model is that it cannot express relationships among relationships.
- To illustrate the need for such a construct, consider the ternary relationship works-on, which we saw earlier, between a employee, branch and job.
- Now, suppose we want to record managers for tasks performed by an employee at a branch; that is, we want to record managers for (employee, branch, job) combinations. Let us assume that there is an entity set manager.
- One alternative for representing this relationship is to create a quaternary relationship manages between employee, branch, job, and manager. (A quaternary relationship is required—a binary relationship between manager and employee would not permit us to represent which (branch, job) combinations of an employee are managed by which manager.)
- Using the basic E-R modeling constructs, we obtain the E-R diagram as follows:

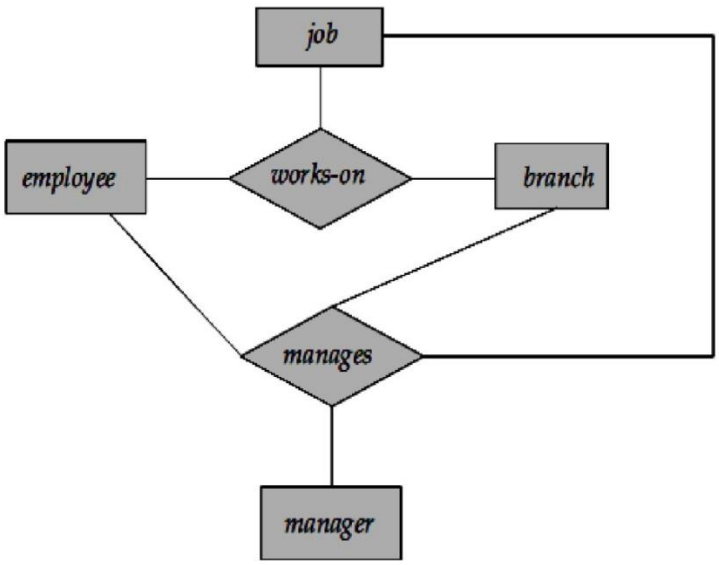


Fig: E-R diagram with redundant relationships

- It appears that the relationship sets works-on and manages can be combined into one single relationship set.
- Nevertheless, we should not combine them into a single relationship, since some employee, branch, job combinations many not have a manager.
- There is redundant information in the resultant figure, however, since every employee, branch, job combination in manages is also in works-on.
- If the manager were a value rather than a manager entity, we could instead make manager a multi valued attribute of the relationship works-on.
- But doing so makes it more difficult (logically as well as in execution cost) to find, for example, employee-branch-job triples for which a manager is responsible. Since the manager is a manager entity, this alternative is ruled out in any case.
- The best way to model a situation such as the one just described is to use aggregation.
- Aggregation is an abstraction through which relationships are treated as higher-level entities.
- Following figure shows a notation for aggregation commonly used to represent the above situation.

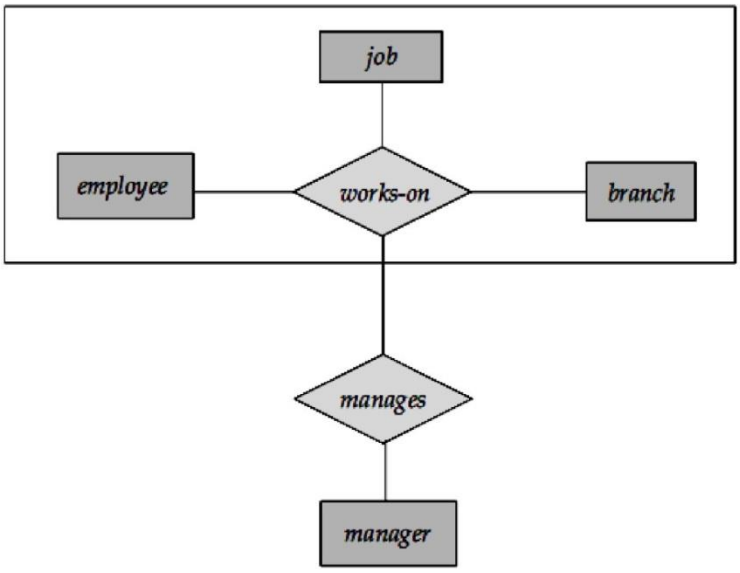


Fig: E-R diagram with aggregation

4.8.4 Transform ER Diagram into Tables

Since ER diagram gives us the good knowledge about the requirement and the mapping of the entities in it, we can easily convert them as tables and columns. i.e.; using ER diagrams one

can easily create relational data model, which is nothing but the logical view of the database.

There are various steps involved in converting it into tables and columns. Each type of entity, attribute and relationship in the diagram takes their own depiction here. Consider the ER diagram below and will see how it is converted into tables, columns and mappings.

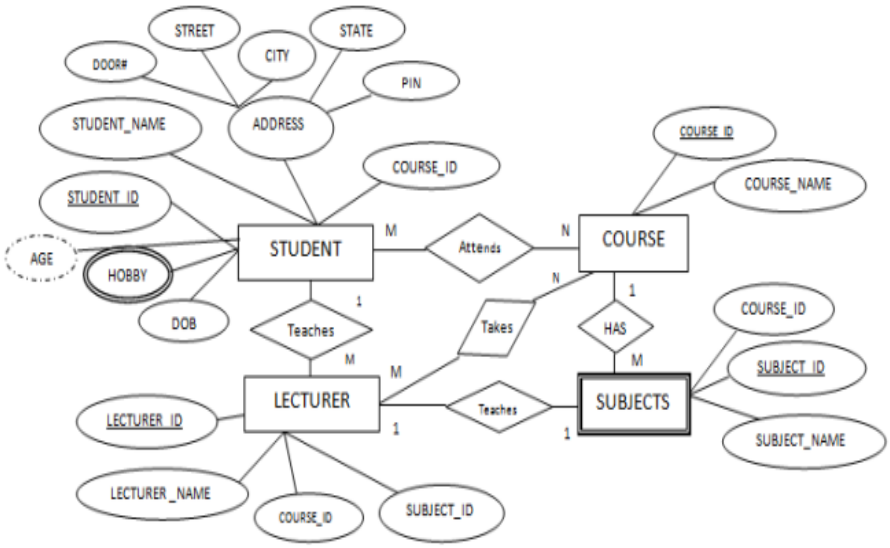


Fig: Transform ER Diagram into Tables

Rules of Converting ER-Diagrams into tables

The basic rules for converting the ER diagrams into tables is are as follows.

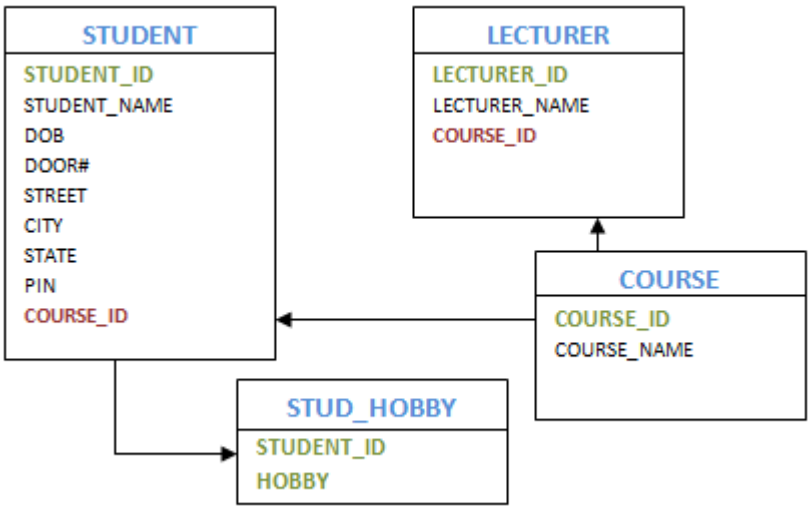
1. Convert all the Entities in the diagram to tables.

   All the entities represented in the rectangular box in the ER diagram become independent tables in the database. In the below diagram, STUDENT, COURSE, LECTURER and SUBJECTS forms individual tables.
2. Concert all single valued attributes of an entity is converted to a column of the table

   All the attributes, whose value at any instance of time is unique, are considered as columns of that table. In the STUDENT Entity, STUDENT_ID, STUDENT_NAME form the columns of STUDENT table. Similarly, LECTURER_ID, LECTURER_NAME form the columns of LECTURER table. And so on.
3. Convert the Key attributes in the ER diagram to the Primary keys of the table.

In diagram above, STUDENT_ID, LECTURER_ID, COURSE_ID and SUB_ID are the key attributes of the entities. Hence we consider them as the primary keys of respective table.

4. Declare the foreign key column, if applicable.
   In the diagram, attribute COURSE_ID in the STUDENT entity is from COURSE entity. Hence add COURSE_ID in the STUDENT table and assign it foreign key constraint. COURSE_ID and SUBJECT_ID in LECTURER table forms the foreign key column. Hence by declaring the foreign key constraints, mapping between the tables are established.
5. Any multi-valued attributes are converted into new table.
   A hobby in the Student table is a multi valued attribute. Any student can have any number of hobbies. So we cannot represent multiple values in a single column of STUDENT table. We need to store it separately, so that we can store any number of hobbies, adding/ removing / deleting hobbies should not create any redundancy or anomalies in the system. Hence we create a separate table STUD_HOBBY with STUDENT_ID and HOBBY as its columns. We create a composite key using both the columns.
6. Any composite attributes are merged into same table as different columns.
   In the diagram above, Student Address is a composite attribute. It has Door#, Street, City, State and Pin. These attributes are merged into STUDENT table as individual columns.
7. One can ignore derived attribute, since it can be calculated at any time.
   In the STUDENT table, Age can be derived at any point of time by calculating the difference between Date of Birth and current date. Hence we need not create a column for this attribute. It reduces the duplicity in the database.
   These are the very basic rules of converting ER diagram into tables and columns, and assigning the mapping between the tables. Table structure at this stage would be as below:



Let us see some of the special cases.

4.8.4.1 Converting Weak Entity

Weak entity is also represented as table. All the attributes of the weak entity forms the column of the table. But the key attribute represented in the diagram cannot form the primary key of this table. We have to add a foreign key column, which would be the primary key column of its strong entity. This foreign key column along with its key attribute column forms the primary key of the table.

In our example above, SUBJECTS is the weak entity. Hence, we create a table for it. Its attributes SUBJECT_ID and SUBJECT_NAME forms the column of this table. Although SUBJECT_ID is represented as key attribute in the diagram, it cannot be considered as primary key. In order to add primary key to the column, we have to find the foreign key first. COURSE is the strong entity related to SUBJECT. Hence the primary key COURSE_ID of COURSE is added to SUBJECT table as foreign key. Now we can create a composite primary key out of COURSE_ID and SUBJECT_ID.

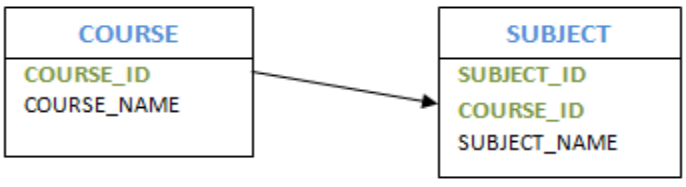


Fig: Creating a composite primary key out of COURSE_ID and SUBJECT_ID.

Representing 1:1 relationship

Imagine SUBJECT is not a weak entity, and we have LECTURER teaches SUBJECT relation. It is a 1:1 relation. i.e.; one lecturer teaches only one subject. We can represent this case in two ways

1. Create table for both LECTURER and SUBJECT. Add the primary key of LECTURER in SUBJECT table as foreign key. It implies the lecturer name for that particular subject.
2. Create table for both LECTURER and SUBJECT. Add the primary key of SUBJECT in LECTURER table as foreign key. It implies the subject taught by the lecturer.

   In both the case, meaning is same. Foreign key column can be added in either of the table, depending on the developer's choice.

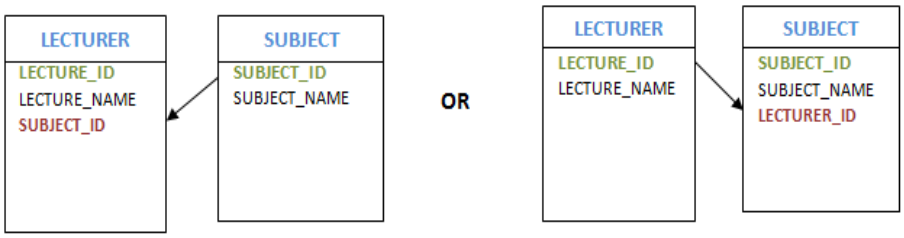


Fig: Representing 1:1 relationship

Representing 1: N relationship

Consider SUBJECT and LECTURER relation, where each Lecturer teaches multiple subjects. This is a 1: N relation. In this case, primary key of LECTURER table is added to the SUBJECT table. i.e.; the primary key at 1 cardinality entity is added as foreign key to N cardinality entity

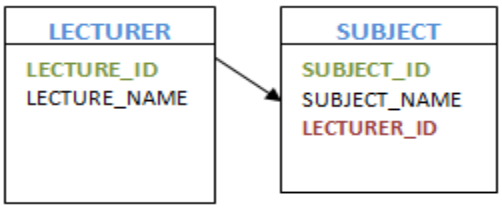


Fig: Representing 1: N relationship

Representing M: N relationship

Consider the example, multiple students enrolled for multiple courses, which is M: N relation. In this case, we create STUDENT and COURSE tables for the entities. Create one more table for the relation 'Enrolment' and name it as STUD_COURSE. Add the

primary keys of COURSE and STUDENT into it, which forms the composite primary key of the new table.

That is, in this case both the participating entities are converted into tables, and a new table is created for the relation between them. Primary keys of entity tables are added into new table to form the composite primary key. We can add any additional columns, if present as attribute of the relation in ER diagram.

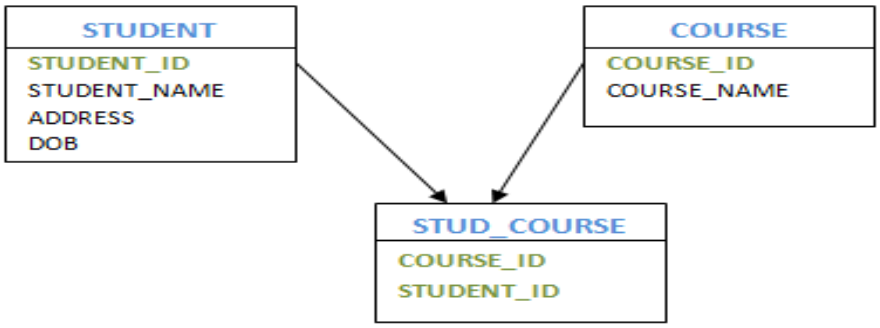


Fig: Representing M: N relationship

Self Referencing 1: N relation

Consider the example of HOD and Lecturers. Here one of the Lecturers is a HOD of the department. i.e.; one HOD has multiple lecturers working with him. In this case, we create LECTURER table for the Lecturer entity. Create the columns and primary keys as usual. In order to represent HOD, we add one more column to LECTURER table which is same column as primary key, but acts as a foreign key. i.e.; LECTURER_ID is the primary key of LECTURER table. We add one more column HOD, which will have LECTURER_ID of the HOD. Hence LECTURER table will show HOD's Lecturer ID for each Lecturer. In this case, primary key column acts as a foreign key in the same table.

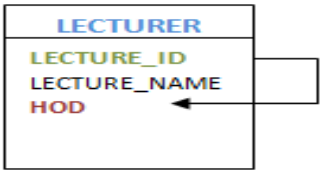


Fig: Self Referencing 1: N relation

Self Referencing M: N relation

Consider Student and Teacher example as 'Manny students have Many Teachers teaching the subjects'. Here relation between

Student and Teacher is M: N. In this case, creates independent tables for student and teacher, and set their primary keys.

Then we create a new table for the relationship 'have' as STUDENT_TEACHER, which will have student and teacher combination, and any other columns if applicable. Basically, student-teacher combination is the two primary key columns from respective tables, hence establishing the relationship between them. Both the primary keys from both tables act as a composite primary key in the new table. This reduces the storing of redundant data and consistency in the database.

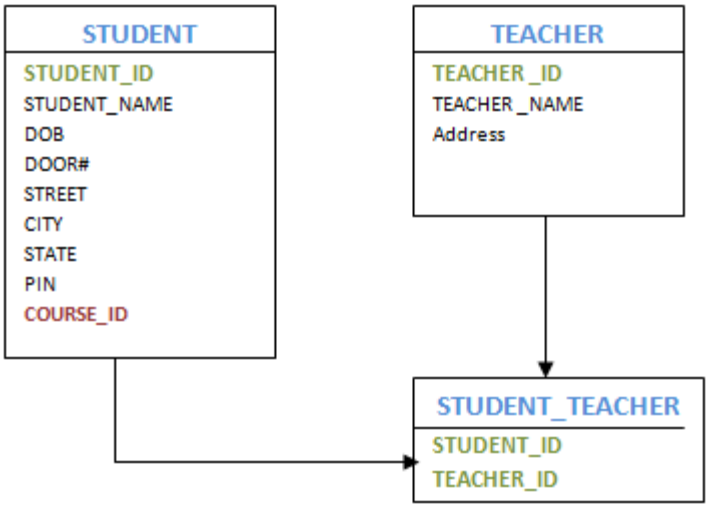


Fig: Self Referencing M: N relation

4.9 Let Us Sum Up

The conceptual design needs a thorough requirements analysis that yields a high level description of data to be stored at the conceptual level. ER model is a popular model for conceptual design, the ER diagrams are more expressive, and better represents way in which the entities in the real world are interlinked.

Some basic constructs of ER Diagram are entities, relationships, and attributes. Some additional constructs: ISA hierarchies, cardinality ratios.

Several kinds of integrity constraints can be expressed in the ER model: key constraints, structural constraints, constraints on specializations

Some of them can be expressed in SQL when translating entity and relationship types into tables. Not all constraints can be expressed in

the ER model. Constraints play an important role in determining a good database design for an application domain.

Ensuring a good database design includes analyzing and further refining relational schema obtained through translating ER schema.

4.10 Self Assessment Questions

1. Define each of the following concepts in the context of the relational data model:

a) Relation
b) Attribute
c) Domain
d) Tuple
e) Relational database.

.....
.....
.....
.....
.....
.....
.....
.....

2. Discuss the properties of a relational table.

……
……
……
…….

3. Define an attribute. Discuss different types of attributes in relational databases.

……
……
……
……

4. What is a Key? What are different types of Keys in relational databases?

……
……
…….
…….

5. Differentiate between the strong and weak entity.

……
……
……
……

4.11 Model Questions

1. What are different types of constraints in relational databases? Discuss with examples.
2. What is an ER Diagram? Explain different symbols used in an ER diagram.
3. Define a relationship. Discuss different types of relationship with example.
4. Write the Rules of Converting ER-Diagrams into tables.

4.12 References and Further Readings

1. Abraham Silberschatz, Henry Korth, and S. Sudarshan, "Database System Concepts", Mc Grow Hill
2. R. Elmasri and S. Navathe, "Fundamentals of Database Systems" Pearson Education.
3. S.K Singh, Database Systems, Concepts, Design and Applications, Pearson Education.
4. http://ecomputernotes.com/fundamental/what-is-a-database/type-of-data-models
5. http://ecomputernotes.com/fundamental/what-is-a-database/database-model
6. https://www.tutorialcup.com/dbms/object-based-data-models.htm

ANSWER TO SELF ASSESSMENT QUESTIONS (UNIT-1)

1. **Differentiate between Data and Information.**
   - Data is the input for a computer and information is the output that is meaningful to the user.
   - Data is unprocessed facts or figures but information is processed data which has some meaning.
   - Data does not depend on information but information depends on data.
   - Data is not specific but information is specific enough to generate meaning.
2. **List the limitations of file oriented Systems.**
   - Data Redundancy: It is possible that the same information may be duplicated in different files. This leads to data redundancy results in memory wastage.
   - Data Inconsistency: Because of data redundancy, it is possible that data may not be in consistent state.
   - Difficulty in Accessing Data: Accessing data is not convenient and efficient in file processing system.
   - Limited Data Sharing: Due to this data isolation, it is difficult to share data among different applications.
   - Integrity Problems: Data integrity means that the data contained in the database in both correct and consistent. For this purpose the data stored in database must satisfy correct and constraints.
   - Atomicity Problems: Any operation on database must be atomic. This means, it must happen in it do entirely or not at all.
   - Concurrent Access Anomalies: Multiple users are allowed to access data simultaneously. This is for the sake of better performance and faster response.
   - Security Problems: Database should be accessible to users in limited way. Each user should be allowed to access data concerning his requirements only.
3. **Define a database system. Write the characteristics of a Database System.**

   A database system is acollection of interrelated data and a set of programs to access data efficiently and provides an environment that is convenient to users and effective to use.

   A Database system must have the following characteristics.
   - Minimal data redundancy

- Data independence
- Efficient data Access
- Concurrent data access
- Consistency of data
- Integrity of data
- Data Security

4. **Write the benefits of a database system.**

Database Systems Benefits include the following.

- Improved strategic use of corporate data
- Reduced complexity of the organization's information systems
- Reduced data redundancy and inconsistency
- Enhanced data integrity
- Application-data independence
- Improved security
- Reduced application development and maintenance costs
- Improved flexibility of information systems
- Increased access and availability of data and information
- Logical & Physical data independence
- Provides central control on the system through DBA.

5. **Write the role and Responsibilities of a DBA.**

- A Person in the organization who controls the design and the use of the database is known as a database administrator.
- DBA provides necessary technical support for implementing a database.
- DBA works on such as design, development, testing, and operational phases.
- DBA is a technical person having knowledge of database technology.
- DBA does not need to be a business person. In short, DBA is a technically focused person but should understand about the business to administrator the database effectively.

Responsibilities

- Maintaining all databases required for development, testing, training and production usage
- Managing share resources used amongst applications

- Administrates all database objects, including tables, views, indexes, stored procedures, functions, packages, sequences and clusters
- Enforces and maintains database constraints to ensure integrity of the database
- Installation and configuration of DBMS server software and related products.
- Upgrading and patching/hot-fixing of DBMS server software and related products.
- Ensure that the site is running the products that are most appropriate
- Establish and maintain sound backup and recovery policies and procedures.
- Take care of the database design and implementation.

6. **What are different types of users in the database systems?**
   Besides the Database Administrator, other database users are of 4 different types:

   - Naive users: These are the unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.
   - Application programmers: These are computer professionals who write application programs, used to develop user interfaces.
   - Sophisticated users: These users interact with the database using database query language.
   - Specialized users: These users write specialized database applications to retrieve data.

7. **What are different types of database languages?**
   There are two main types of languages.
   - Data definition language (DDL)
   - Data manipulation language (DML)

   But in modern database architecture, the other languages that are used by the database are:
   - Storage Definition language (SDL)
   - Transaction control language (TCL)
   - View Definition language (VDL)
   - Data control languages (DCL)
   - Fourth Generation language (4GL)

ANSWER TO SELF ASSESSMENT QUESTIONS (UNIT-2)

1. Define a Schema. What are different types of Schema?

A schema is defined as an outline or a plan that describes the records and relationships existing at the particular level.

There are three different types of schema in the database corresponding to each data view of database. In other words, the data views at each of three levels are described by schema. These are explained as follows.

External Schema

- The level closest to the user and highest level of abstraction.
- Concerned with the way in which data are required by the user / viewed by the user.
- Each external schema is used by represented by a data model.
- Described the part of database, which is relevant to the user.

Conceptual Schema

- It described the logical structure of the data base.
- It described the entities, data types, relationships, user operations and constraints.

Internal Schema

- The lowest level of abstraction. It describes the physical storage structure of the database.
- It describes complete details of the storage and access paths.
- It describes the detailed data structure of the data items.

2. What do you mean by intension and extension of the database?

The data in the database at any particular point in time is called a database instance. Therefore, many database instances can correspond to the same database schema.

The schema is sometimes called the **intension** of the database, while an instance is called an **extension** (or state) of the database

3. What are the disadvantages of using Three-Tier Architecture?

Following are the disadvantages of using Three-Tier Architecture?

- To implement even small part of application it will consume lots of time.
- Need good expertise in object oriented concept (classes and objects).
- It is more complex to build.

4. What do you mean by data abstraction? What are the three levels of data abstraction in databases?

Data abstraction simplifies database design. Major purpose of database system is to provide users with abstract view of data i.e. the system hides certain details of how the data are stored and maintained. This process is called data abstraction.

Since database system users are not computer trained, and need not bother about the internal details of the database, developers hide the complexity from users through *3 levels of abstraction*, to simplify user's interaction with the system. The three levels of abstraction are as follows.

1) Physical level of data abstraction

This is the lowest level of abstraction which describes how data are actually stored.

2) Logical level of data abstraction

This level hides what data are actually stored in the database and what relationship exists among them.

3) View Level of data abstraction

View provides security mechanism to prevent user from accessing certain parts of database.

5. What are the advantages of network databases?

The following are the advantages of network databases:

Flexibility: Multiple parent/child relationships allowed a network database to represent data that did not have a simple hierarchical structure.

Standardization: The CODASYL standard boosted the popularity of the network model.

Performance: Network databases boasted performance approaching that of hierarchical databases.

Answer to Self Assessment Questions (Unit-3)

1. **Define data model. Name different types of record based data models.**

Data Model can be defined as an integrated collection of concepts for describing and manipulating data, relationships between data, and constraints on the data in an organization.

The record based data models are mainly three types.

(i) Hierarchical model: represents data as a hierarchical tree structure

(ii) Network model: represents data as record types

(iii)Relational model: represents data as relations or tables

2. **What is a Physical Data Model? What is the purpose of a Physical Data Model?**

Physical Data Model is the physical representation of the database. It describes how the data is stored.

The data model deals with

- Run-time performance of the database
- Storage utilization and compression
- File organization and access methods
- Are the physical level – managed by the *operating system (OS)*
- Provide concepts that describe the details of how data are stored in the computer's memory

3. **Discuss the advantages and disadvantages of hierarchical data model.**

Advantages:

- Simple and easy to use
- Data with hierarchical relationship can be mapped on this model.
- Suitable for application such as: Employee Dept.

Disadvantages:

- Search for (an element or) a record is difficult.
- Insertion, deletion and updating is difficult.
- Many – Many relationships cannot be established in this model.
- Data with non hierarchical relationship cannot be mapped.
- The hierarchical relationship is maintained using points which require extra storage.
- Changes in relationship require changes in entire structure of the database.
- Processing is sequential among branches of the tree so access time is high.

4. **What is an Entity Relationship Data Model? Write its advantages and disadvantages.**

ER model is a graphical representation of real world objects with their attributes and relationship. It makes the system easily understandable. This model is considered as a top down approach of designing a requirement.

Advantages

- It makes the requirement simple and easily understandable by representing simple diagrams.
- One can covert ER diagrams into record based data model easily.
- Easy to understand ER diagrams

Disadvantages

- No standard notations are available for ER diagram. There is great flexibility in the notation. It's all depends upon the designer, how he draws it.
- It is meant for high level designs. We cannot simplify for low level design like coding.

- **Explain the merits and demerits of relational data model.**

Merits:

- Tabular structure is easy to understand simple.
- Data manipulation is easy.
- We can apply mathematical operation on tables.
- Built in query language support such as SQL.
- Very flexible data organization.

Demerits:

- Size of the data base becomes large

ANSWER TO SELF ASSESSMENT QUESTIONS (UNIT-4)

1. Discuss each of the following concepts in the context of the relational data model:

(a) Relation: A table with columns and rows is called as a Relation.
(b) Attribute: A named column of a relation is called an attribute.
(c) Domain: The set of allowable values for one or more attributes.
(d) Tuple: A record or a row in a relation is called a tuple.
(e) Relational database: A collection of normalized tables is said to be a relational database.

2. Discuss the properties of a relational table.

A relational table has the following properties:

- The table has a name that is distinct from all other tables in the database.
- The cell values in a table should be atomic. . In other words each cell of the table contains exactly one value. A relational table that satisfies this property is said to be in *first normal form*.
- Each column has a distinct name.
- The values of a column are all from the same domain.
- The order of columns has no significance. In other words, provided a column name is moved along with the column values. We can interchange columns.
- Each record is distinct; there are no duplicate records.
- The order of records has no significance, theoretically.

3. Define an attribute. Discuss different types of attributes in relational databases.

An attribute is a descriptive property of an entity. In other words, the characteristics of an entity are called attributes. Ex. Roll no, name, marks, etc are the attributes of an entity student.

Depending on the values that an attribute can take, it is divided into different types.

1. Simple Attribute: These kinds of attributes have values which cannot be divided further. For example, Passport Number is unique value and it cannot be divided.
2. Composite Attribute:This kind of attribute can be divided further to more than one simple attribute. For example, address of a person.

3. Derived Attribute: Derived attributes are the one whose value can be obtained from other attributes of entities in the database. For example, Age of a person can be obtained from date of birth and current date.
4. Stored Attribute: The attribute which gives the value to get the derived attribute are called Stored Attribute. In example above, age is derived using Date of Birth. Hence Date of Birth is a stored attribute.
5. Single Valued Attribute:These attributes will have only one value. For example, passport#, driving license#, SSN etc have only single value for a person.
6. Multi-Valued Attribute:These attribute can have more than one value at any point of time. e.g. A person can have more than one email address.
7. Simple Single Valued Attribute: This is the combination of above four types of attributes. An attribute can have single value at any point of time, which cannot be divided further. For example, EMPLOYEE_ID – it is single value as well as it cannot be divided further.
8. Simple Multi-Valued Attribute:Phone number of a person, which is simple as well as he can have multiple phone numbers is an example of this attribute.
9. Composite Single Valued Attribute: Date of Birth can be a composite single valued attribute. Any person can have only one DOB and it can be further divided into date, month and year attributes.
10. Composite Multi-Valued Attribute:Shop address which is located two different locations can be considered as example of this attribute.
11. Descriptive Attribute: Attributes of the relationship is called descriptive attribute. For example, employee works for department. Here 'works for' is the relation between employee and department entities. The relation 'works for' can have attribute DATE_OF_JOIN which is a descriptive attribute.

4. What is a Key? What are different types of Keys in relational databases?

A Key or a Key attribute is the attribute that contains unique values and used to identify each entity uniquely.

Different types of keys used in relational databases are as follows.

- Super Key
- Candidate Key
- Primary Key
- Foreign
- Composite Key
- Alternate Key
- Secondary Key

5. Differentiate between the strong and weak entity.

Strong Entity: Entities having its own attribute(s) as primary key(s) are called strong entity. For example, STUDENT has Enrolment No as primary key. Hence it is a strong entity.

Weak Entity: Entities which cannot form their own attribute as primary key are known as weak entities. These entities will derive their primary keys from the combination of its attribute and primary key from its mapping entity.