DATABASE MANAGEMENT SYSTEM
Module-1

Introduction to Databases

Database

1. What is database?

A database is a collection of logically related data with implicit meaning example:

1. Collection ofnames,home address and telephonenumbers
2. collection of words to make paragraph in a page

Database Management System

2. What is Database Management System?

- Is a collection of programs that enables users to create and maintain a database.
- The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, and manipulating databases for various applications

Functions of DBMS

3. What are the Functions of DBMS?

- **Defining:** a database involves specifying the data types, structures, and constraints for the data to be stored in the database
- **Constructing:** the database is the process of storing the data itself on some storage medium that is controlled by the DBMS.
- **Manipulating:** a database includes such functions as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data.
- **Protection or security:** measures to prevent unauthorized access
- **Active processing** to take action on data
- **Maintaining the databases** and associated programs over the lifetime of the database application

Database system

4. What is database system?

The database and DBMS software together a database system

An Example:-

- UNIVERSITY database for maintaining information concerning students, courses, and grades in a university environment
- define: file (records), data elements, data type (for each data element)
- construct: store data in the appropriate files (note that records may be related between files)
- Manipulation: querying, updating- informal queries and updates must be specified precisely in the database system language before they can be processed.

Figure 1.2 An example of a database that stores student records and their grades.

STUDENT	Name	StudentNumber	Class	Major
	Smith	17	1	CS
	Brown	8	2	CS

COURSE	CourseName	CourseNumber	CreditHours	Department
	Intro to Computer Science	CS1310	4	CS
	Data Structures	CS3320	4	CS
	Discrete Mathematics	MATH2410	3	MATH
	Database	CS3380	3	CS

SECTION	SectionIdentifier	CourseNumber	Semester	Year	Instructor
	85	MATH2410	Fall	98	King
	92	CS1310	Fall	98	Anderson
	102	CS3320	Spring	99	Knuth
	112	MATH2410	Fall	99	Chang
	119	CS1310	Fall	99	Anderson
	135	CS3380	Fall	99	Stone

GRADE_REPORT	StudentNumber	SectionIdentifier	Grade
	17	112	B
	17	119	C
	8	85	A
	8	92	A
	8	102	B
	8	135	A

PREREQUISITE	CourseNumber	PrerequisiteNumber
	CS3380	CS3320
	CS3380	MATH2410
	CS3320	CS1310

Some mini world entities:-

1. STUDENTS
2. COURSES(SUBJECTS)
3. SECTIONS(CLASSES0
4. GRADE REPORTS
5. PREREQUISITES

Some mini world relationships:-

- SECTIONS are of specific COURSES
- STUDENTS takes SECTIONS

5. Explain database system with neat diagram?

Database system environment:-

- Database system consists of Application Program or Queries + DBMS Software+Stored Database Definition and Actual Stored Data

- DBMS Software consists of software to process queries/program and software to access stored data

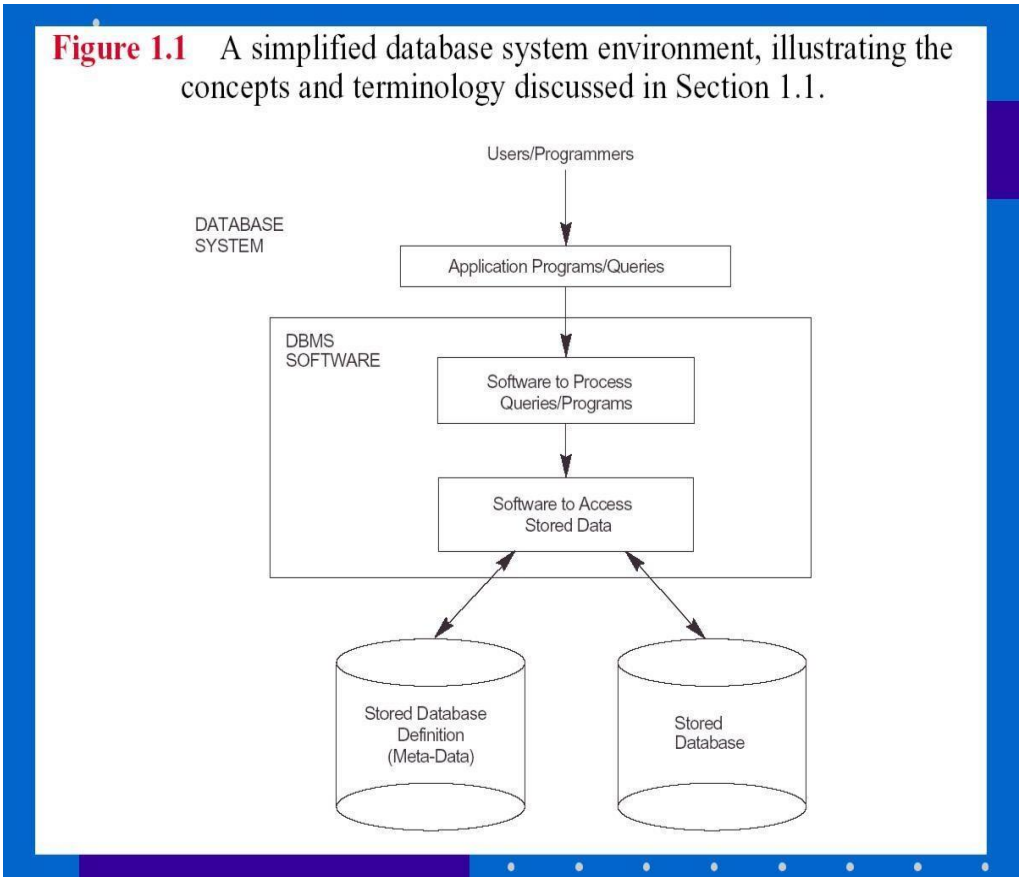
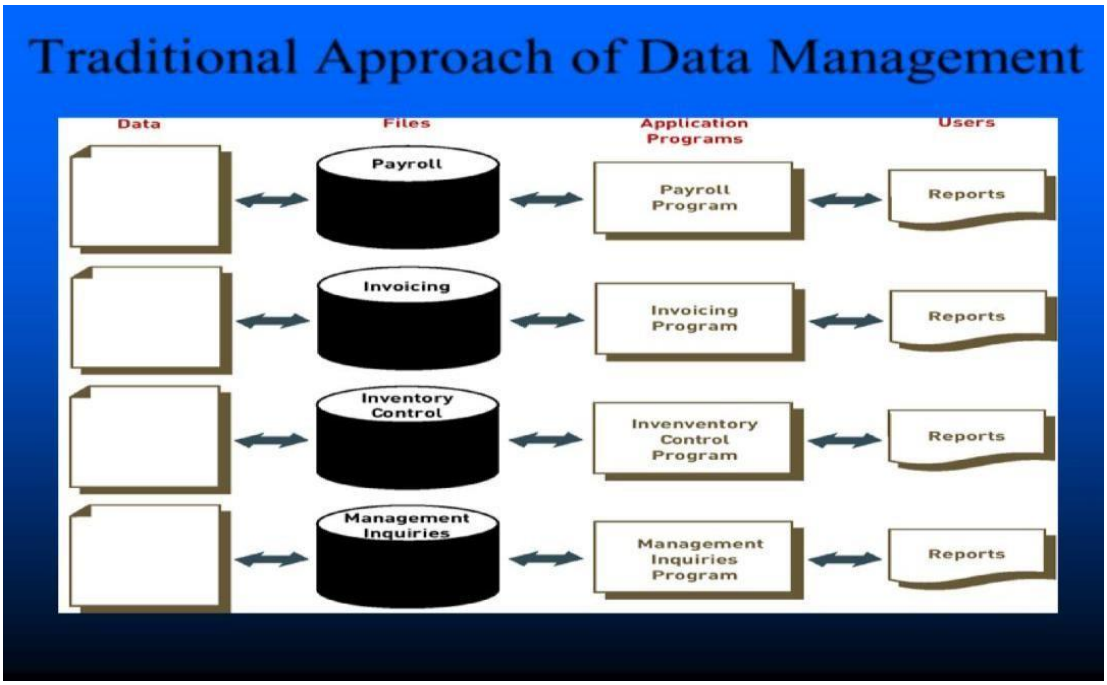
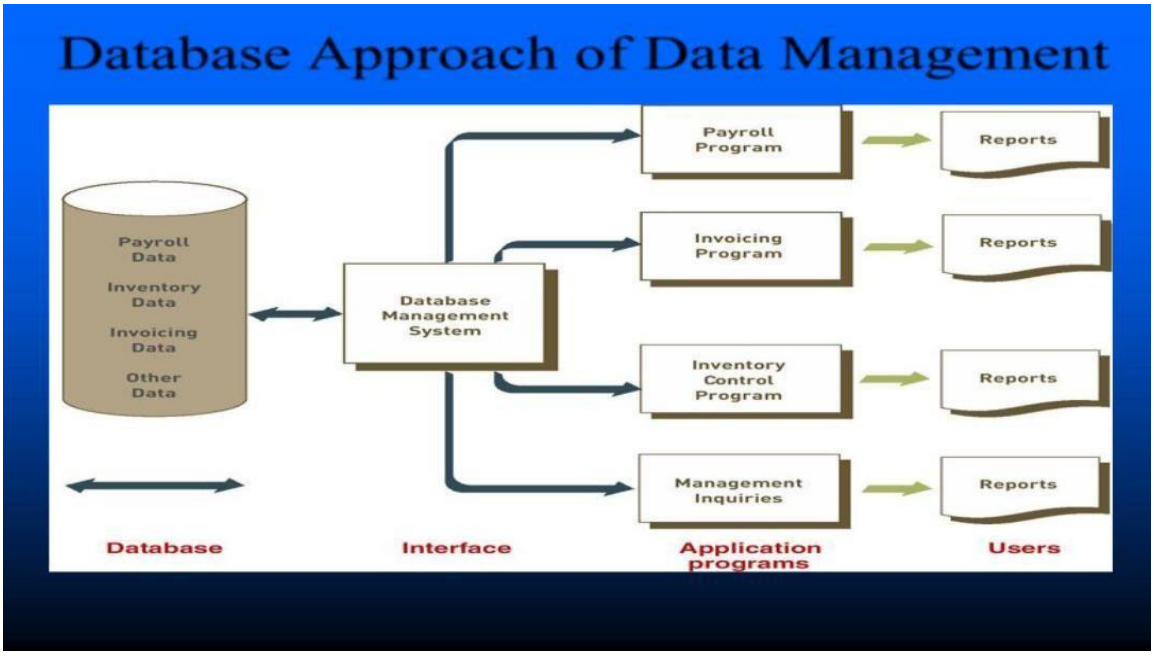


Figure 1.1 A simplified database system environment, illustrating the concepts and terminology discussed in Section 1.1.

6. Difference between Database approach Vs file processing approach(Programming with files)?





DataBase approach	Fileprocessing approach (traditional approach)
A single repository of data is maintained that is defined once and then is accessed by various users. No redundancy of data and wastage of storage	In traditional file processing, each user defines and implements the files needed for a specific application as part of programming the application Example:- one user, the grade reporting office, may keep a file on students and their grades
	second user, the accounting office, may keep track of students' fees and their payments • each user maintains separate files—and programs to manipulate these files—because each requires some data not available from the other user's files • redundancy in defining and storing data results in wasted storage space and in redundant efforts to maintain common data up-to-date
In database approach is that the database system contains not only the database itself but also a complete definition or description of the database structure and constraints and stored separately (catalog) • A general-purpose DBMS software package is not written for a specific database application • The DBMS software must work equally well with any number of database applications— example, a university database, a banking database, or a company database—as long as the database definition is stored in the catalog.	In traditional file processing, data definition is typically part of the application programs themselves. • Hence, these programs are constrained to work with only one specific database, whose structure is declared in the application programs • C++ program may have "struct" or "class" declarations; and a COBOL program has Data Division statements to define its files.

DBMS software can access diverse databases by extracting the database definitions from the catalog and then using these definitions	Not possible
In database approach the structure of data files is separated from the access programs, so any changes to the structure of a file may not require changing all programs that access this file	. In traditional file processing, the structure of data files is embedded in the access programs, so any changes to the structure of a file may require changing all programs that access this file.
Data isolation-all related data is available in one file	Data isolation-all related data is not available in one file
Program data independent	Program data dependent
Secure data because not easily accessible	Not secure data because easily accessible
Flexibility is possible	Lack of flexibility
No Concurrent access anomalies-same piece of data is not allowed to update simultaneously which provide consistencies	Concurrent access anomalies-same piece of data is allowed to update simultaneously which causes inconsistencies

Main characteristics of the database approach

7. What are the main characteristics of the database approach versus the file- processing approach?

a) Self-Describing Nature of a Database System

- single repository of data is maintained
- Contains not only the database itself but also a complete definition or description of the database structure and constraints (system catalogue).
- information stored in the catalog is called meta-data
- Catalog used by the DBMS and users.
- The DBMS software work equally well with any number of database applications.

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Note: Major_type is defined as an enumerared type with all known majors. XXXXNNNN is used to define a type with four alpha characters followed by four digits

Figure 1.3
An example of a database catalog for the database in Figure 1.2.

b) Insulation between Programs and Data, and Data Abstraction

- Program-data independence
- The characteristic that allows program-data independence and program-operation independence is called data abstraction
- A DBMS provides users with a conceptual representation of data that does not include many of the details of how the data is stored or how the operations are implemented. (data model)
- data model hides storage and implementation details that are not of interest to most database users.
- A data model is a type of data abstraction that is used to provide this conceptual representation.
- Data models use logical concepts (objects, properties, and relationships) that are easier to understand for most users
- In object-oriented and object-relational databases, abstraction is carried one level further to include not only the data structure but also the operations on the data

Example:-

In case RDBMS (program-data independence)

Student :-name,address,age

Add DOB to student record only change in structure of student no change in program accessing DB.

In case OODB/ORDBMS (program-operation independence)

Data + operation - for access data part of definition Operation divded into two part

1. Interface(signature)-without implementation operation name and the data types of its arguments
2. Implementation (or method)
   - operation is specified separately and can be changed without affecting the interface
   - User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented

Data Abstraction

8. What is data abstraction?

The characteristic that allows program-data independence and program- operation independence is called **data abstraction**

– Database users refer to the conceptual representation of the files, and the DBMS extracts the details of file storage from the catalog when these are needed by the DBMS software

c) Support of Multiple Views of the Data

Support for different perspective or view of the same database for different user group.

– A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored
– Some users may not need to be aware of whether the data they refer to is stored or derived.
– A multiuser DBMS whose users have a variety of applications must provide facilities for defining multiple views

Example:

(a)The transcript(marks card) of each student (b)The course(subject) prerequisite view

Figure 1.4 Two views derived from the example database shown in Figure 1.2. (a) The student transcript view. (b) The course prerequisite view.

(a)

TRANSCRIPT	StudentName	Student Transcript				
		CourseNumber	Grade	Semester	Year	SectionId
	Smith	CS1310	C	Fall	99	119
		MATH2410	B	Fall	99	112
	Brown	MATH2410	A	Fall	98	85
		CS1310	A	Fall	98	92
		CS3320	B	Spring	99	102
		CS3380	A	Fall	99	135

(b)

PREREQUISITES	CourseName	CourseNumber	Prerequisites
	Database	CS3380	CS3320
			MATH2410
	Data Structures	CS3320	CS1310

d) Sharing of Data and Multiuser Transaction Processing

- It allows multiple users to access the database at the same time which is essential if data for multiple applications is to be integrated and maintained in a single database.
- The DBMS must include concurrency control software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct

Example:-

- Reservation clerks try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one clerk at a time

These types of applications are generally called on-line transaction processing (OLTP) applications.

DATABASE USERS

They are two types of users

- **Actors on the Scene-**Those who actually use and control the database content and those who design, develop and maintain database applications
- **Actors behind the Scene**- Those who design and develop the DBMS software and related tools and the computers systems operators

Actors on the Scene

a) Database Administrators
b) Database Designers
c) End Users
d) System Analysts and Application Programmers (Software Engineers)

Database Administrators:-

- In a database environment, the primary resource is the database itself and the secondary resource is the DBMS and related software
- authorizing access to the database
- coordinating and monitoring its use
- acquiring software and hardware resources as needed

9. What are main functions of DBA?

The database administrator (DBA) is responsible for overall control of the database system. Responsibilities include:

- Deciding the information content of the database, i.e. identifying the entities of interest to the enterprise and the information to be recorded about those entities. This is defined by writing the conceptual schema using the DDL
- Deciding the storage structure and access strategy, i.e. how the data is to be represented by writing the storage structure definition. The associated internal/conceptual schema must also be specified using the DDL
- Liaising with users, i.e. to ensure that the data they require is available and to write the necessary external schemas and conceptual/external mapping (again using DDL)
- Defining authorization checks and validation procedures. Authorization checks and validation procedures are extensions to the conceptual schema and can be specified using the DDL
- Defining a strategy for backup and recovery. For example periodic dumping of the database to a backup tape and procedures for reloading the database for backup. Use of a log file where each log record contains the values for database items before and after a change and can be used for recovery purposes

- Monitoring performance and responding to changes in requirements, i.e. changing details of storage and access thereby organizing the system so as to get the performance that is best for the enterprise

Database Designers

- identifying the data to be stored in the database
- choosing appropriate structures to represent and store this data undertaken before the database is actually implemented and populated with data
- communicate with all prospective database users, in order to understand their requirements
- develop a view of the database that meets the data and processing requirements for each group of users
- These views are then analyzed and integrated with the views of other user groups. The final database design must be capable of supporting the requirements of all user groups

End Users

– access to the database for querying, updating, and generating reports

Casual end users:

– occasionally access the database
– need different information each time
– Learn only a few facilities that they may use repeatedly.
– use a sophisticated database query language to specify their requests
– typically middle-or high-level managers or other occasional browsers

Naive or parametric end users

– constantly querying and updating the database, using standard types of queries and updates called canned transactions that have been carefully programmed and tested
– need to learn very little about the facilities provided by the DBMS example:-
– Bank tellers check account balances and post withdrawals and deposits
– Reservation clerks for airlines, hotels, and car rental companies check availability for a given request and make reservations
– Clerks at receiving stations for courier mail enter package identifications via bar codes and descriptive information through buttons to update a central database of received and in-transit packages

Sophisticated end users

– Engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS so as to implement their applications to meet their complex requirements
– Try to learn most of the DBMS facilities in order to achieve their complex requirements

Stand-alone users

– Maintain personal databases by using ready-made program packages that provide easy-to-use menu- or graphics-based interfaces. An example is the user of a tax package that stores a variety of personal financial data for tax purposes
– Typically become very proficient in using a specific software package Example:-tally (income tax return)

System Analysts and Application Programmers

– Determine the requirements of end users, especially naive and parametric end users, and develop specifications for canned transactions that meet these requirements
– Application programmers implement these specifications as programs; then they test, debug, document, and maintain these canned transactions

Worker Behind the Scene

DBMS system designers and implementers

– design and implement the DBMS modules (for implementing the catalog, query language, interface processors, data access, concurrency control, recovery, and security.) and interfaces as a software package
– The DBMS must interface with other system software, such as the operating system and compilers for various programming languages.

Tool developers

– include persons who design and implement **tools**—the software packages that facilitate database system design and use, and help improve performance
– Tools are optional packages that are often purchased separately
– include packages for database design, performance monitoring, natural language or graphical interfaces, prototyping, simulation, and test data generation

Operators and maintenance personnel

– system administration personnel who are responsible for the actual running and maintenance of the hardware and software environment for the database system

Advantages of using the DBMS approach

10. What are the Advantages of using the DBMS approach?

1. Controlling Redundancy
2. Restricting Unauthorized Access
3. Providing Persistent Storage for Program Objects and Data Structures
4. Permitting Inferencing and Actions Using Rules
5. Providing Multiple User Interfaces 1.6.6 Representing Complex Relationships Among Data
6. Enforcing Integrity Constraints
7. Providing Backup and Recovery

OR

Controlling Redundancy

For consistency, in Database design that stores each logical data item—such as a student's name or birth date—in *only one place* in the database. This does not permit inconsistency, and it saves storage space.

The DBMS should have the capability to control this redundancy so as to prohibit inconsistencies among the files

Restricting Unauthorized Access

– When multiple users share a database, it is likely that some users will not be authorized to access all information in the database.

Example:-

- financial data is often considered confidential, and hence only authorized persons are allowed to access such data
- some users may be permitted only to retrieve data, whereas others are allowed both to retrieve and to update.

The DBMS should then enforce these restrictions automatically

Example:-

Parametric users may be allowed to access the database only through the **canned transactions** developed for their use.

Providing Persistent Storage for Program Objects and Data Structures

– Databases can be used to provide **persistent storage** for program objects and data structures.

Permitting Inferencing and Actions Using Rules

– **Deductive database systems:** when database systems provide capabilities for defining deduction rules for Inferencing new information from the stored database facts

Example:-

There may be complex rules in the mini world application for determining when a student is on probation. (rules)

Providing Multiple User Interfaces

– Many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces A DBMS provides a variety of user interfaces:

1. query languages-casual users
2. programming language interfaces-application programmers
3. forms and command codes - parametric users
4. menu-driven interfaces and natural language interfaces - stand-alone users.

– Capabilities for providing World Wide Web access to a database

Representing Complex Relationships Among Data

– A database may include numerous varieties of data that are interrelated in many ways
– A DBMS must have the capability to represent a variety of complex relationships among the data as well as to retrieve and update related data easily and efficiently.

Enforcing Integrity Constraints

– The simplest type involves specifying a data type for each data item
– A more complex type involves specifying that a record in one file must be related to records in other files.
– Another type specifies uniqueness on data item values
– It is the database designers responsibility identify integrity constraints during database design.

Example:-

Student – name varchar (5)-it should be alphabets and maximum length 5

Providing Backup and Recovery

– provide facilities for recovering from hardware or software failures.(The backup and recovery subsystem)

Implications of the Database Approach

– Potential for Enforcing Standards
– Reduced Application Development Time
– Flexibility Availability of Up-to-Date Information

– Economies of Scale

History of database applications.

1. The Hierarchical and Network Models
2. Relational Model based Systems
3. Object-Oriented Database Management Systems (OODBMSs)

Extended relational systems add further capabilities (e.g. for multimedia data, XML, and other data types)

Overview of Database Languages and Architectures:

Data Model

1. What is data model?

A set of concepts to describe the ***structure*** of a database, the ***operations*** for manipulating these structures, and certain ***constraints*** that the database should obey.

- Provide data abstraction

a) Data Model Structure and Constraints:

- Constructs are used to define the database structure
- Constructs typically include ***elements*** (and their ***data types***) as well as groups of elements (e.g. ***entity, record, table***), and ***relationships*** among such groups
- Constraints specify some restrictions on valid data; these constraints must be enforced at all times

b) Data Model Operations:

- generic operation: insert, delete, modify, retrieve
- ・user-defined operations
  - These operations are used for specifying database *retrievals* and *updates* by referring to the constructs of the data model.
  - Operations on the data model may include ***basic model operations*** (e.g. generic insert, delete, update) and ***user-defined operations*** (e.g. compute_student_gpa, update_inventory)

Categories of Data Models

2. What are the Categories of Data Models?

a) Conceptual (high-level, semantic) data models:

Provide concepts that are close to the way many users perceive data.

- (Also called *entity-based* or *object-based* data models.)
  - entity ・attribute ・relationship

b) Physical (low-level, internal) data models:

Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals

- ・record formats ・record ordering ・access paths

c)Implementation (representational) data models:

– Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial system such as ・relational ・network ・hierarchical

3. Define Schemas, Instances, Schema Diagram, Schema Construct and Database State?

Schemas

1) **Database Schema(meta-data)**:
   a) The *description* of a database.
   b) Includes descriptions of the database structure, data types, and the constraints on the database.

Schema Diagram

2) **Schema Diagram:**
   a) An *illustrative* display of (most aspects of) a database schema.

Schema Construct

3) **Schema Construct:**
   a) A *component* of the schema or an object within the schema, e.g., STUDENT, COURSE

Database State

4) **Database State(instance)**:
   a) The actual data stored in a database at a ***particular moment in time***. This includes the collection of all the data in the database. Also called database instance (or occurrence or snapshot).
   b) The term *instance* is also applied to individual database components, e.g. *record instance, table instance, entity instance*

Database Schema vs. Database State

Database State:

- Refers to the *content* of a database at a moment in time.

Initial Database State:

- Refers to the database state when it is initially loaded into the system.

Valid State:

- A state that satisfies the structure and constraints of the database.

Distinction

- The ***database schema*** changes very infrequently. **Schema** is also called **intension**.
- The ***database state*** changes every time the database is updated. **State** is also called **extension**

Example of a Database Schema

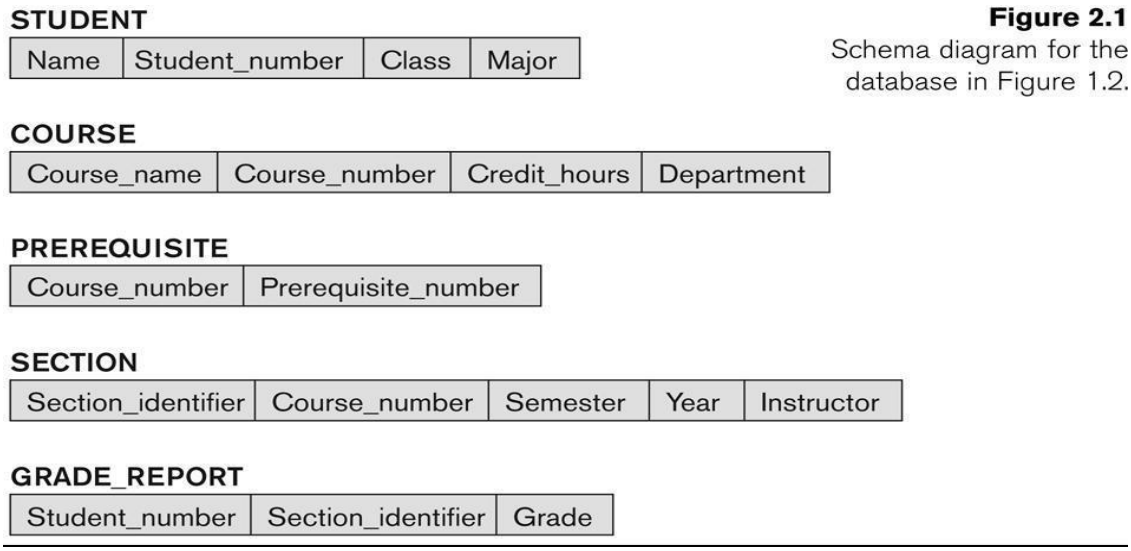
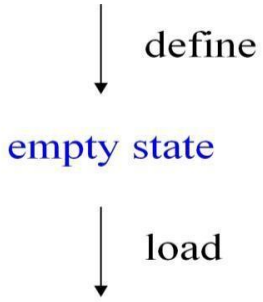


Figure 2.1
Schema diagram for the database in Figure 1.2.

Example of a database state

Concepts and Architecture :-



COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

valid state satisfy database

Three schema architecture

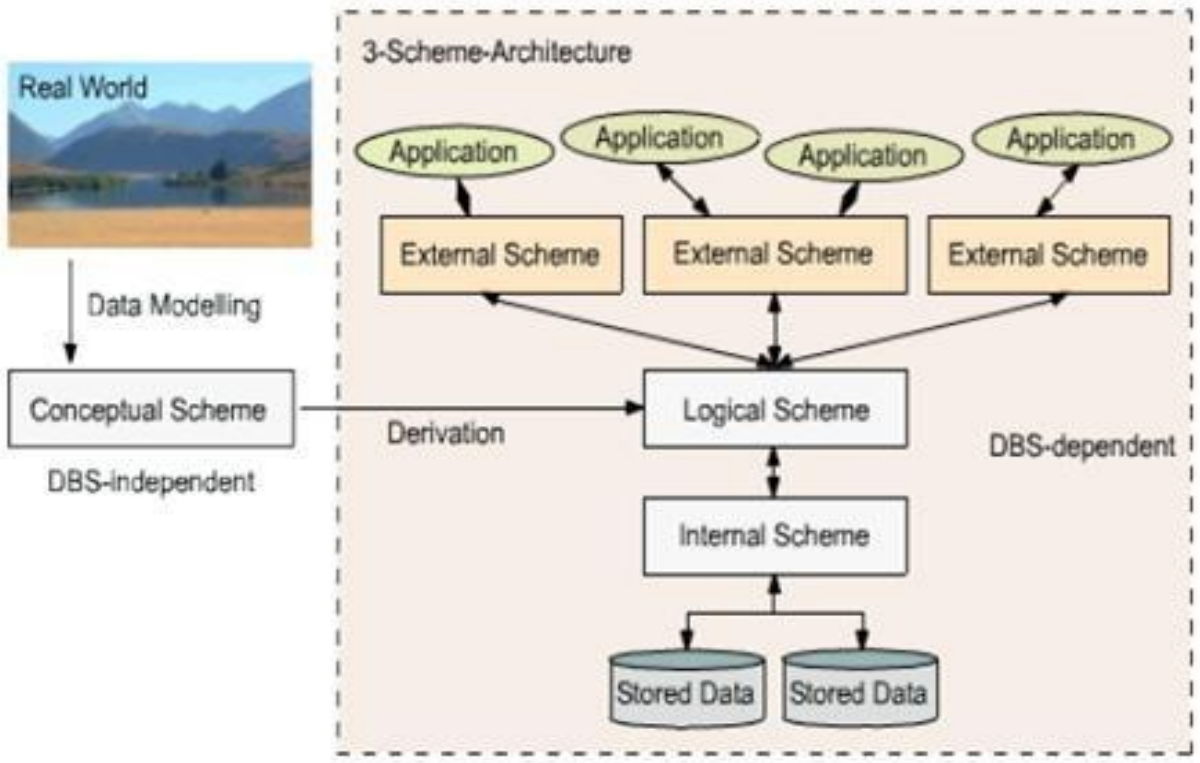
4. Explain three schema architecture?

An architecture for database systems, called the three-schema architecture, which was proposed to help achieve characteristics

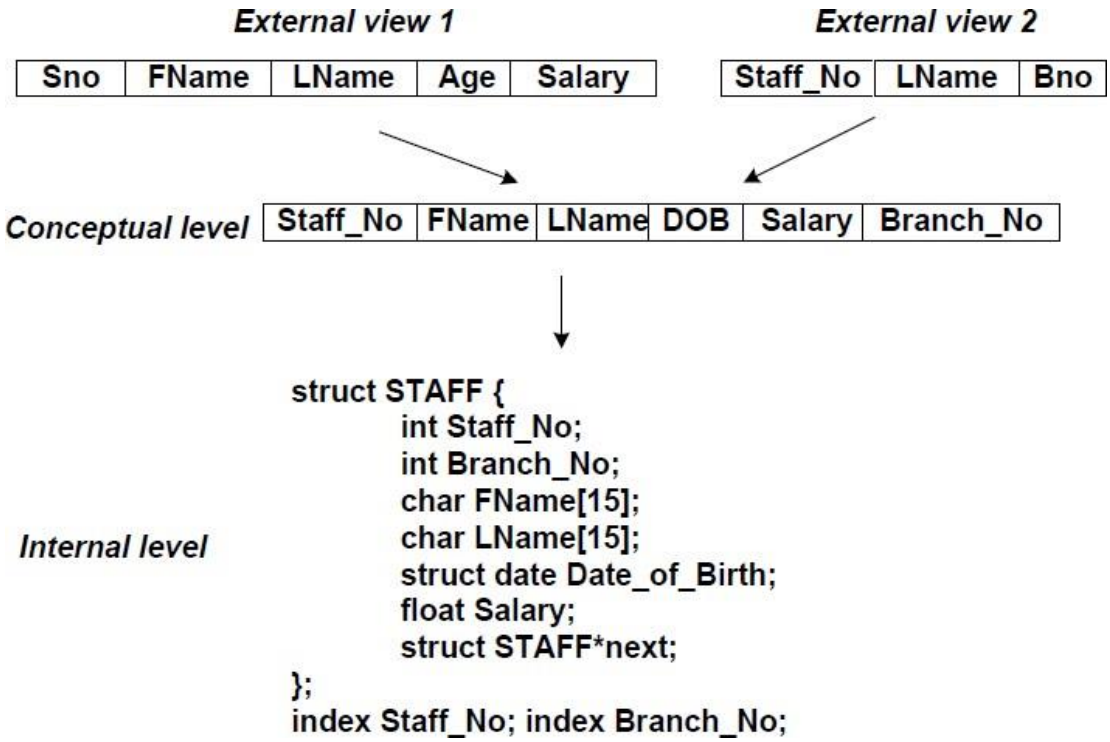
- The goal of the three-schema architecture, is to separate the user applications and the physical database

Proposed to support DBMS characteristics of:

- Program-data independence. Supportofmulti-
- pleviewsofthedata.
- Use of a catalog to store the database description (schema).



Example:-



- **Internal schema** at the internal level to describe and access paths (e.g indexes)- Typically uses a **physical** data model.

- **Conceptual schema or logical schema** at the conceptual level to describe the structure and constraints for the whole database for a community of users.-Uses a **conceptual** or an **implementation** data model.

- **External schemas** at the external level to describe the various user views. Usually uses the same data model as the conceptual level or high-level data model.

Mappings among schema levels are also needed. Programs refer to an external schema and are mapped by the DBMS to the internal schema for execution

- In a DBMS based on the three-schema architecture, each user group refers only to its own external schema using SQL statement.
- Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database.
- If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user's external view.
- The processes of transforming requests and results between levels are called mappings.
- Mappings among schema levels are needed to transform requests and data.
- Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.
- Data extracted from the internal DBMS level is reformatted to match the user's external view (e.g. formatting the results of an SQL query for display in a Web page)

Data independence

5. Explain concept program data independence or logical and physical data independence?

The three-schema architecture can be used to explain the concept of **data independence,** which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

Logical Data Independence

The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.

Example:-

- By adding or removing a record type or data item to expand the database or reduce the database
- A change to the conceptual definition means that the conceptual/external mapping must be changed accordingly, so that the external schema may remain invariant, achieving logical data independence.

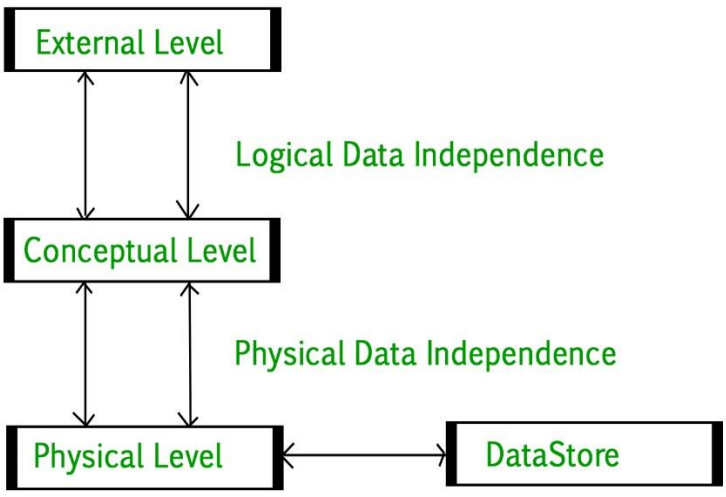
Physical Data Independence

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

- A change to the storage structure definition means that the conceptual/internal mapping must be changed accordingly, so that the conceptual schema may remain invariant, achieving physical data independence

Example:-

The internal schema may be changed when certain file structures are reorganized or new indexes are created to improve database performance



DBMS Languages and Interfaces

6. Explain DBMS Languages?

1. Once the design of a database is completed and a DBMS is chosen to implement the database, to specify conceptual and internal schemas for the database and any mappings between the two.

- In many DBMSs - data definition language (DDL), is used by the DBA and by database designers to define both schemas.
- The DBMS will have a DDL compiler whose function is to process DDL statements in order to identify descriptions of the schema constructs and to store the schema description in the DBMS catalog.

2. Once the database schemas are compiled and the database is populated with data, users must have some means to manipulate the database. Typical manipulations include retrieval, insertion, deletion, and modification of the data.

- The DBMS provides a data manipulation language (DML) for these purposes.
- SQL relational database language which represents a combination of DDL, VDL, and DML, as well as statements for constraint specification and schema evolution

There are two main types of DMLs. A high-level or nonprocedural DML can be used on its own to specify complex database operations in a concise manner.

High-Level or Non-procedural Languages: These include the relational language-SQL

- May be used in a standalone way using terminal or may be embedded in a programming language

Low Level or Procedural Language

These must be embedded in a programming language

1. **Data Definition Language (DDL)**
2. **Data Manipulation Language (DML)**

7. Explain DDL?

Data Definition Language (DDL):

- Used by the DBA and database designers to specify the conceptual schema of a database.
- In many DBMSs, the DDL is also used to define internal and external schemas (views).
- In some DBMSs, separate **storage definition language (SDL)** and **view definition language (VDL)** are used to define internal and external schemas.
- SDL is typically realized via DBMS commands provided to the DBA and database designers

8. Explain DML?

Data Manipulation Language (DML)

- User specify database retrievals and updates

- DML commands (data sublanguage) can be *embedded* in a general-purpose programming language (host language), such as COBOL, C, C++, or Java.
- A library of functions can also be provided to access the DBMS from a programming language
- Alternatively, stand-alone DML commands can be applied directly (called a *query language*).

9. What are the Types of DML?

High Level or Non-procedural Language: For example, the SQL relational language Are set-oriented and specify what data to retrieve rather than how to retrieve it. Also called declarative languages.

Low Level or Procedural Language: Retrieve data one record-at-a-time; Constructs such as looping are needed to retrieve multiple records, along with positioning pointers.

10. Types of DBMS Interfaces supported to access Database?

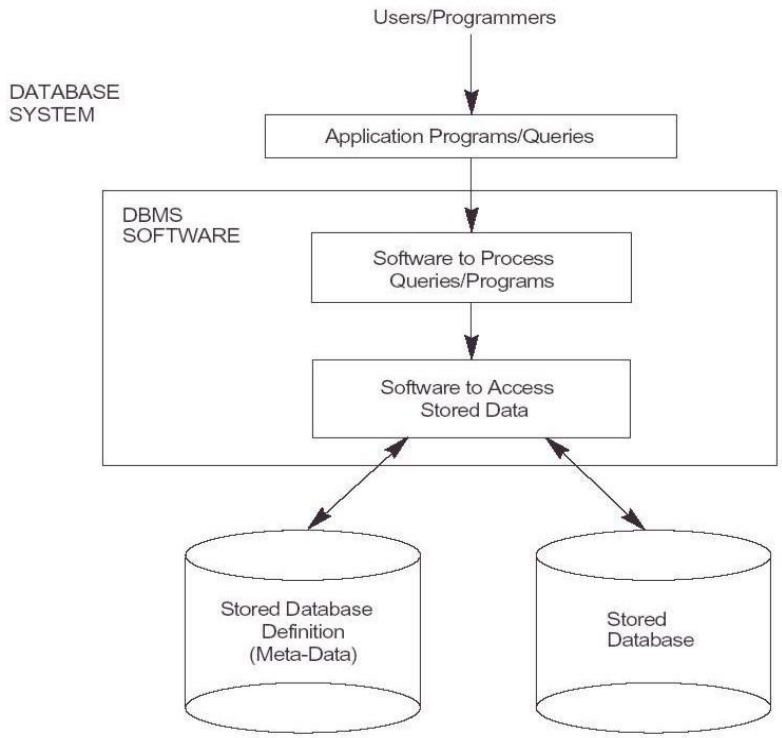
1. Menu-Based Interfaces for Browsing
2. Forms-Based Interfaces
3. Graphical User Interfaces
4. Natural Language Interfaces Interfaces for Parametric Users
5. Interfaces for the DBA
6. Stand-alone query language interfaces
   Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. SQL*Plus in ORACLE, MYSQL)
7. Programmer interfaces for embedding DML in programming languages
8. User-friendly interfaces
   Menu-based, forms-based, graphics-based, etc.

Database System environment

Database system consists of Application Program or Queries + DBMS Software + Stored Database Definition and Actual Stored Data

- DBMS Software consists of software to process queries/program and software to access stored data

Figure 1.1 A simplified database system environment, illustrating the concepts and terminology discussed in Section 1.1.



11. Explain component modules of DBMS with neat diagram?

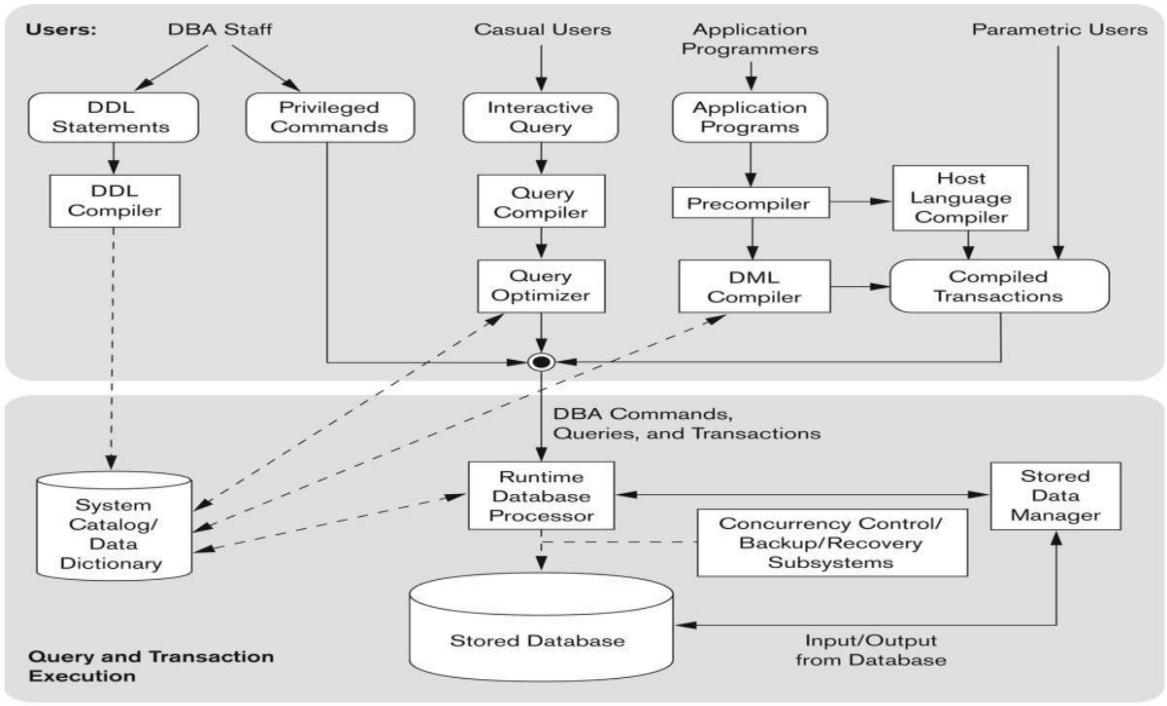


Figure 2.3
Component modules of a DBMS and their interactions.

The above Figure illustrates, in a simplified form, the typical DBMS components. The database and the DBMS catalog are usually stored on disk. Access to the disk is controlled primarily by the **operating system (OS),** which schedules disk input/output.

Stored data manager

A module to control access to DBMS information that is stored on disk, whether it is part of the database or the catalog.

DDL compiler

A module to process schema definition, specified in the DDL, and store description of the schema (meta-data) in the DBMS catalog.

DML compiler

It translates the DML commands into object code for database access.

- run-time database processor

Query compiler

It handles high-level queries that are entered interactively. It parses, analyzes, and compiles or interprets a query by creating database access code, and then generates calls to the rum-time processor for executing the code.

Pre-compiler

It extracts DML commands from an application program which is written in host programming language like C, Pascal, etc

Run time data processor:-

- Object database access code from query / program are input Handles data base access at run-time, **retrieves' or update operations and carries them out on the data base**.
- Execute privileged commands ,executable query plans and canned transaction with running parameters
  - It works with system catalog/dictionary and update its statistics
  - It works with stored data manager which in turn uses OS services to carrying out low level input and output operations between disk and main memory.

Stored Data base: The database and DBMS catalog are stored ,access to the disk is controlled by the operating system which schedules disk input/output

Host language compiler: Other than DML commands, remaining commands are sent to the host language compiler, generates the object code.

Canned Transaction: It contains the object code of the DML commands and other program codes, also includes calls to the run-time data base processor

Database System Utilities

To perform certain functions such as:

- Loading data stored in files into a database. Includes data conversion tools.
- Backing up the database periodically on tape.
- Reorganizing database file structures.
- Report generation utilities.
- Performance monitoring utilities.
- Other functions, such as sorting, user monitoring, data compression, etc.

Tools, Application Environments, and Communications Facilities Data dictionary / repository:

Used to store schema descriptions and other information such as design decisions, application program descriptions, user information, usage standards, etc.

- **Active data dictionary** is accessed by DBMS software and users/DBA.
- **Passive data dictionary** is accessed by users/DBA only

When Not to Use a DBMS

Main inhibitors (costs) of using a DBMS:

- High initial investment and possible need for new hardware.
- Overhead for providing generality, security, concurrency control, recovery, and integrity functions.

When a DBMS may be unnecessary:

- Ifthedatabaseandapplicationsaresimple,welldefined,andnotexpected to change
- If there are stringent real-time requirements that may not be met because of DBMS overhead.
- If access to data by multiple users is not required.

When no DBMS may suffice:

– If the database system is not able to handle the complexity of data because of modelling limitations.

– If the database users need special not supported operations.

Centralized and Client-Server DBMS Architectures 12.Explain

centralized architecture with neat diagram?

Centralized DBMSs architectures

- Combines everything into single system including- DBMS software, hardware, application programs, and user interface processing software.
- User can still connect through a remote terminal – however, all processing is done at centralized site

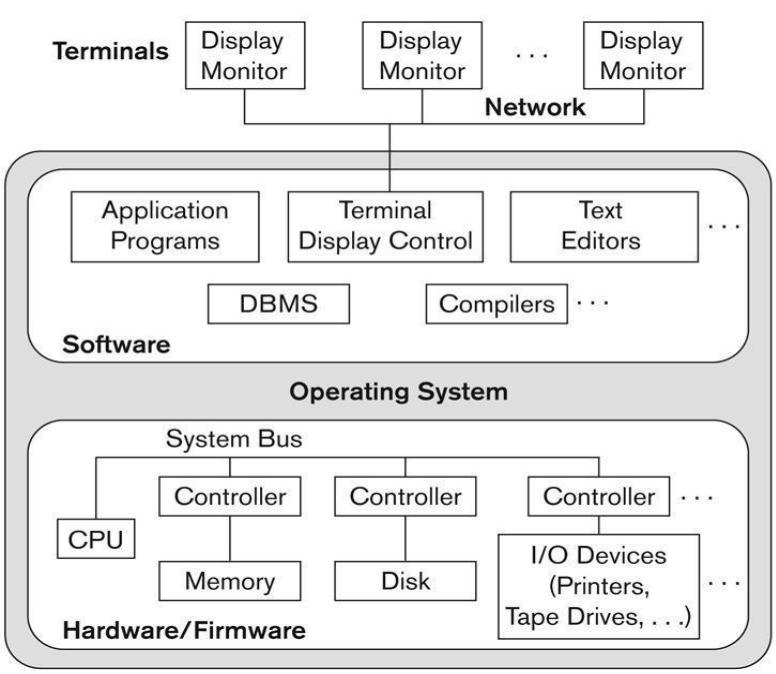
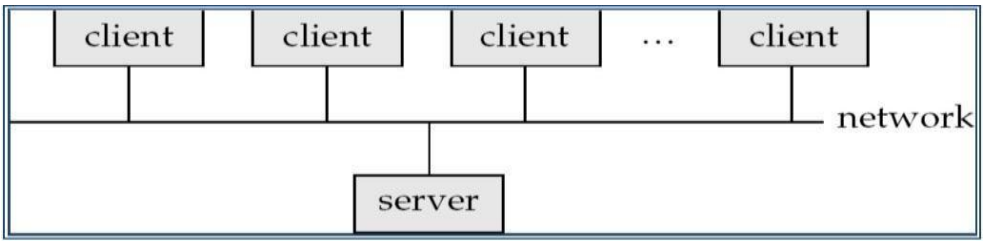


Figure 2.4
A physical centralized architecture.

12. Explain functions of client server architecture?Functions of client and server module

In a basic **client-server architecture**, the system functionality is distributed between two types of modules.
Server systems satisfy requests generated at *m* client systems, whose general structure is shown below:

- A client **module** is typically designed so that it will run on a user workstation or personal computer.
- Application programs and user interfaces that access the database run in the client module. Hence, the client module handles user interaction and provides the user-friendly interfaces such as forms or menu-based GUIs (*g*raphical *u*ser *i*nterfaces).
- A **server module,** typically handles data storage, access, search, and other functions



Basic 2-tier Client-Server Architectures

- The Client-Server Architecture was developed to deal with computing environment in which a large numbers of PCs, workstations, file servers, Print server, DBMS server, Email server and other equipment are connected via a network
- Resources provided by these server can be access by many clients
- Specialized Servers with Specialized functions which connect number of PCs/workstations
- Print server-connected to various printer, all print request by client forwarded to these machine.
- File server-maintains files of client
- DBMS server- maintains database of client Web server-maintains information of the client
- Email server-maintains messages of the client Clients can access the specialized servers as needed
- Client machine provides appropriate interfaces to utilize these servers as well as with local processing power to run local applications

Example:-

- DBMS software in server
  Client interface in client

Logical two-tier client server architecture

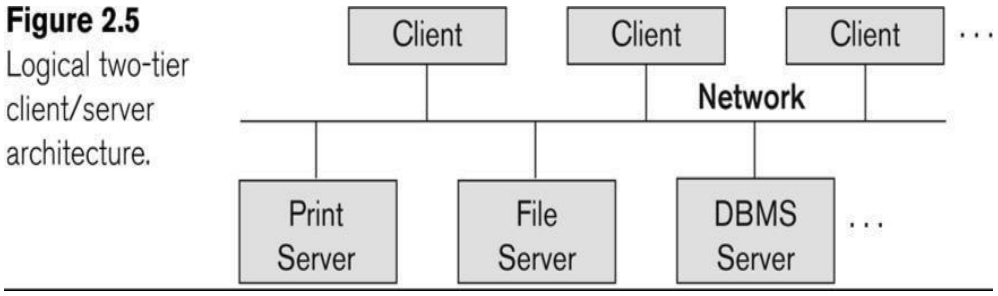


Figure 2.5
Logical two-tier client/server architecture.

Clients

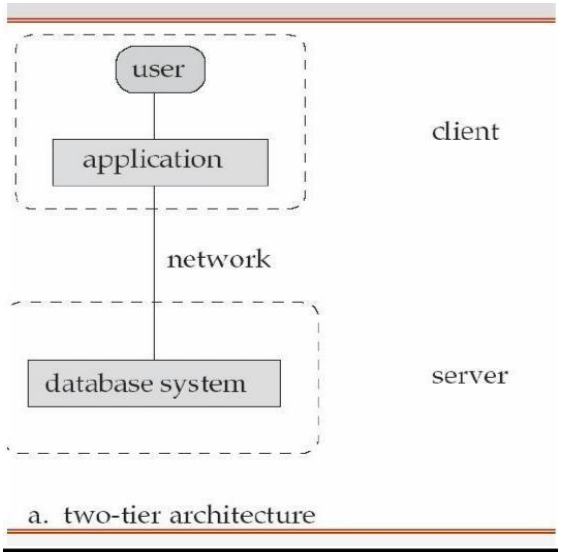
- Provide appropriate interfaces through a client software module to access and utilize the various server resources.
- ClientsmaybedisklessmachinesorPCsorWorkstationswithdiskswithonlythe client software installed.
- Connected to the servers via some form of a network.(LAN: local area network, wireless network, etc.)

DBMS Server

- Provides database query and transaction services to the clients
- Relational DBMS servers are often called SQL servers, query servers, or transaction servers
- Applications running on clients utilize an Application Program Interface (API) to access server databases via standard interface such as:
    - ODBC: Open Database Connectivity standard
    - JDBC: for Java programming acce
- Client and server must install appropriate client module and server module software for ODBC or JDBC

13. Explain client server architecture either 2 tier or 3 tier? Two

Tier Client-Server Architecture for DBMSs



- A client program may connect to several DBMSs, sometimes called the data sources.
- In general, data sources can be files or other non-DBMS software that manages data.
- Other variations of clients are possible: e.g., in some object DBMSs, more functionality is transferred to clients including data dictionary functions, optimization and recovery across multiple servers, etc

Three Tier Client-Server and n tier Architecture for web application

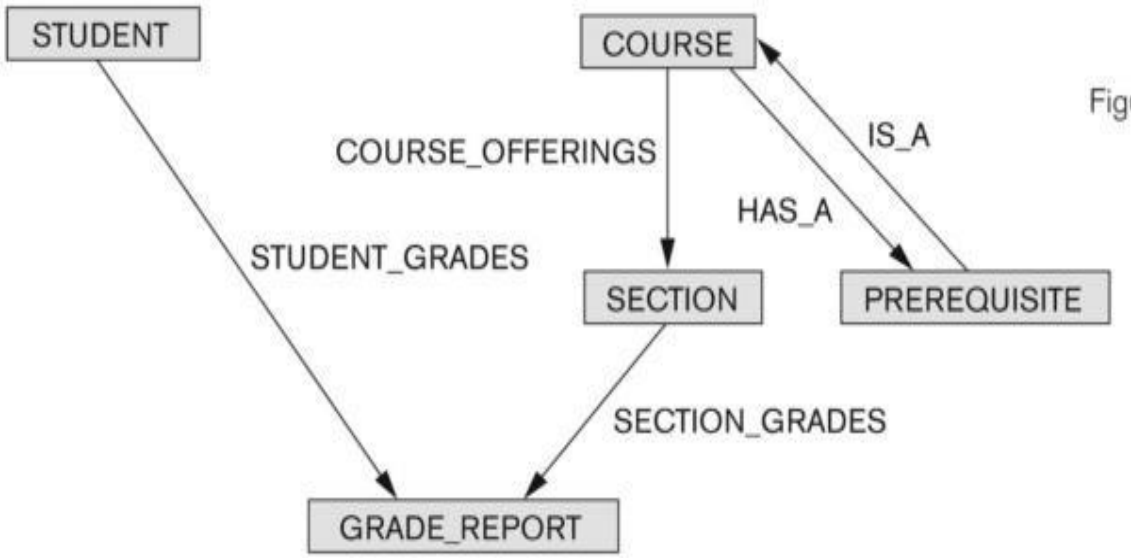
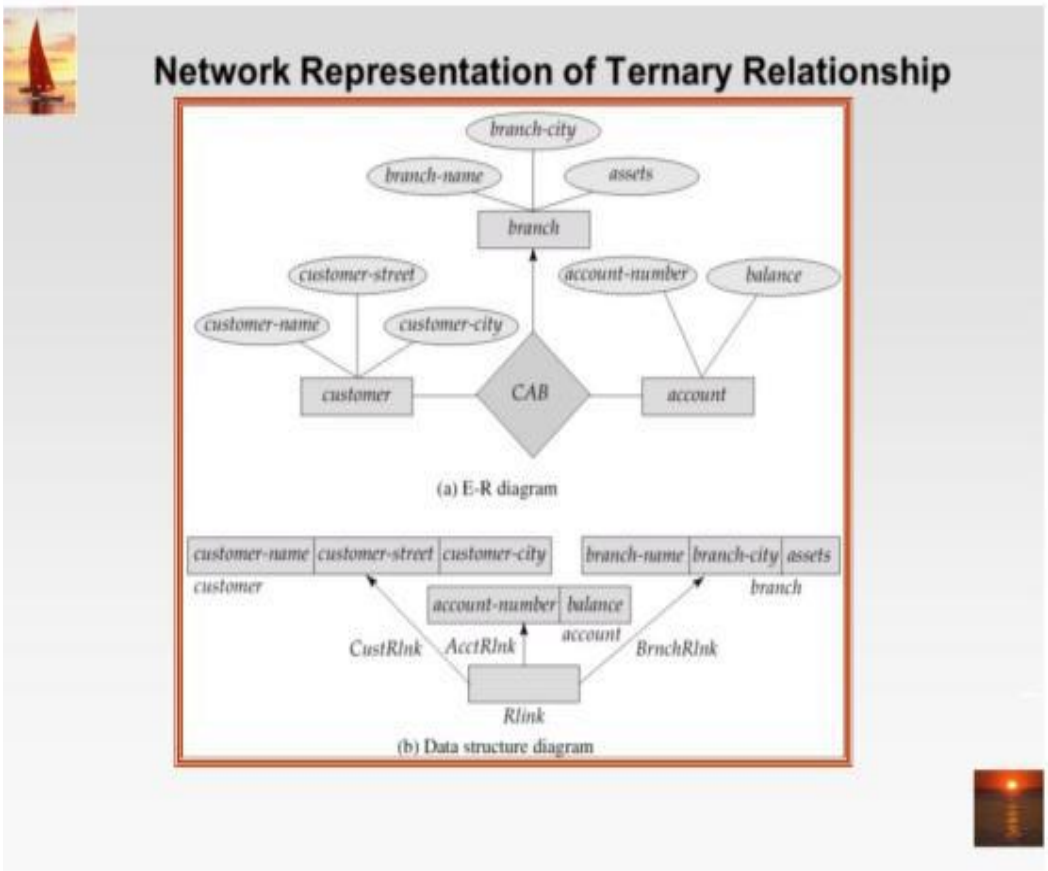


Figure 2.8
The schema of Figure 2.1 in network model notation.



- Common for Web applications
- Intermediate Layer called Application Server or Web Server:
- Stores the web connectivity software and the business logic part of the application used to access the corresponding data from the database server Acts like a conduit for sending partially processed

data between the database server and the client.

- Three-tier Architecture Can Enhance Security: Database server only accessible via middle tier Clients cannot directly access database server

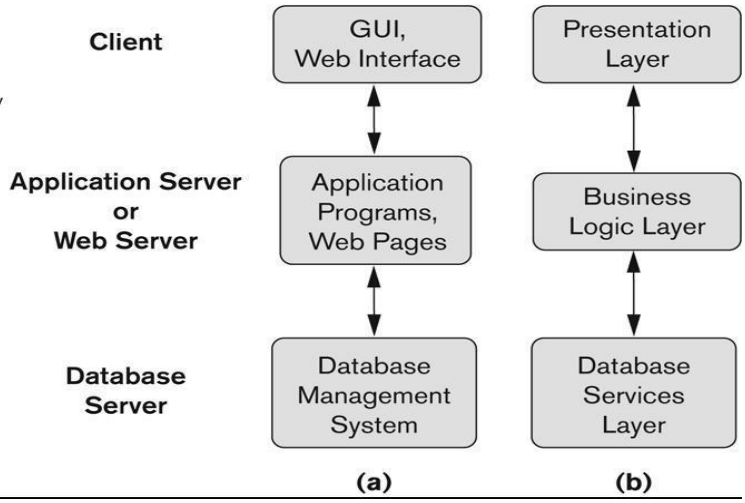


Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

Classification of DBMSs

14. Explain what are the parameter consider for classification of DBMS?

Several criteria are normally used to classify DBMSs.

- ✓ **Based on the data model used:**
- Data models

-Traditional: Relational, Network (see 2-19), Hierarchical

-Emerging:Object-oriented,Semantic,Entity-Relationship,other.

Other classifications:

- ✓ **Number of users** : Single-user (typically used with personal computers) vs. multi-user (most DBMSs)
- ✓ **Number of sites**:
- ✓ Centralized (uses a single computer) vs. distributed (uses multiple computers). Homogeneous vs. Heterogeneous
- ✓ **Cost of DBMS software**.$10,000~100,000 and$100~3,000
- ✓ **Types of access paths used.** (inverted file structures, …)
- ✓ **Purpose**
  - general purpose
  - special purpose

e.g. airline reservations, telephone directory, on-line transaction processing system

Data Modeling Using the Entity-Relationship Model

History of Data model

- **Network Model**
- **Hierarchical Model**
- **Relational Model**
- **Object-oriented Data Models Object-**
- **Relational Models**

Relational Model:

- Proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82.
- Now in several commercial products (e.g. DB2, ORACLE, MS SQL Server, SYBASE, INFORMIX).
- Several free open source implementations, e.g. MySQL, PostgreSQL
- Currently most dominant for developing database applications.
- SQL relational standards: SQL-89 (SQL1), SQL-92 (SQL2), SQL-99, SQL3, …

Object-oriented Data Models:

- Several models have been proposed for implementing in a database system.
- One set comprises models of persistent O-O Programming Languages such as C++ (e.g., in OBJECTSTORE or VERSANT), and Smalltalk (e.g., in GEMSTONE).
- Additionally, systems like O2, ORION (at MCC - then ITASCA), IRIS (at H.P. - used in Open OODB).
- Object Database Standard: ODMG-93, ODMG-version 2.0, ODMG-version 3.0.

Object-Relational Models:

- Most Recent Trend. Started with Informix Universal Server.
- Relational systems incorporate concepts from object databases leading to object- relational
- Exemplified in the latest versions of Oracle-10i, DB2, and SQL Server and other DBMSs

Standards included in SQL-99 and expected to be enhanced in future SQL standards

Modeling using Entities and Relationships

Using High-Level Conceptual Data Models for Database Design

Conceptual modelling is an important phase in designing a successful database application. the modelling concepts of the **Entity-Relationship (ER) model,** which is a popular high-level conceptual data model. This model and its variations are frequently used for the conceptual design of database applications, and many database design tools employ its concepts

What are different phases in database design?

A simplified diagram to illustrate the main phases of database design

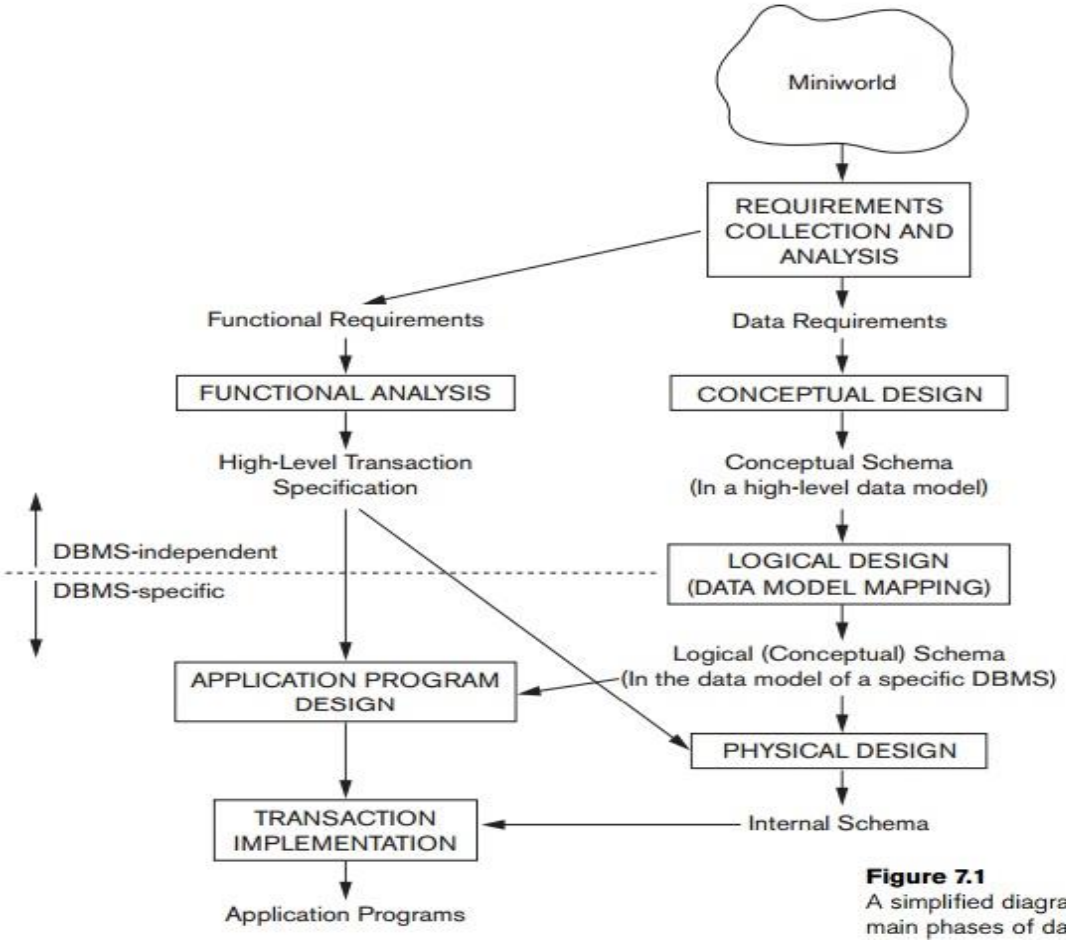


Figure 7.1
A simplified diagram to illustrate the main phases of database design.

- **Requirement Analysis**: The database designers interview prospective database users to understand and document their **data requirements** Collecting information from the users what they want to store in the database. F**unctional requirements** of the application is user-defined **operations** (or **transactions**) that will be applied to the database, and they include both retrievals and updates

- **Conceptual Database Design:** Create a **conceptual schema** for the database includes detailed descriptions of the entity types, relationships, and constraints and do not include implementation details, using a high-level conceptual data model. This step is called **conceptual design.** High level description of data to be stored in the database. ER model is the ideal one. The basic data model operations can be used to specify the high-level user operations identified during functional analysis.
- **Logical Database Design (data model mapping):** The conceptual schema is transformed from the high-level data model into the implementation data model Selecting an appropriate DBMS for implementation.
- **Schema Refinement:** This is done through normalization theories. **Physical Database Design:** During which the internal storage structures, access paths, and file organizations for the database files are specified Low level implementation like indexing and clustering of tables.
- **Application and Security Design:** Developing a complete application using front- end tools like VB, Java, etc. Unified Modelling Language (UML) may be used.

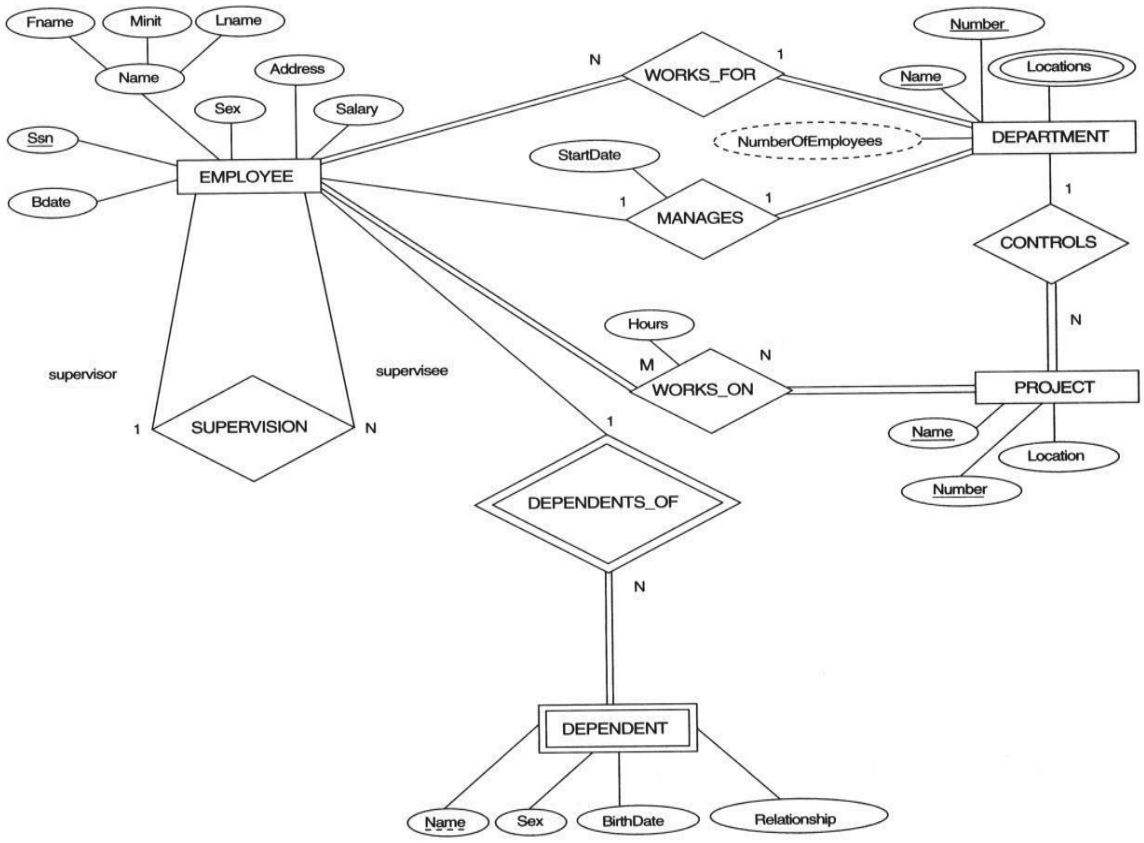
An Example Database Application:-

An example database application, called COMPANY, which serves to illustrate the ER model concepts and their use in schema design. The COMPANY database keeps track of a company's **employees, departments, projects and dependents**

List the data requirements for the database

- The company is organized into departments. Each department has a **unique name, a unique number, and a particular employee** who **manages** the department. Keep track of the **start date** when that employee began managing the department. A department may have **several locations.**
- A **department controls** a number of **projects,** each of which has a **unique name, a** Unique number, and a single location.
  We store each employee's **name, social security number, address, salary, sex, and birth date.** An employee is assigned to one department but may **work on several projects**, which are not necessarily controlled by the same department. Keep track of the **number of hours** per week that an **employee** works on **each project**. Keep track of the direct supervisor of each employee.
- Keep track of the **dependents** of each employee for insurance purposes. Keep each
  Dependent's first name, sex, birth date, and relationship to the employee.

  Figure shows how the schema for this database application can be displayed by means of the graphical notation known as **ER diagrams.**



Introduction to Entity-Relationship (E-R) Modeling:-

Notation uses three mainconstructs

- Data entities
- Attributes
- Relationships

2. Define entity type and entity set with example?

Entity Types and Entity Sets

Entities: An entity is anything that exists in a real world with an independent existence.

Eg. Maruti Car, Accounts Department , DBMS Book, etc.

- An entity is described using a set of attributes
- The same entity may have different prominence in different UoDs

Example:-

- In the Company database, an employee's car is of lesser importance
- In the Department of Transportation's registration database, cars may be the most important concept
- In both cases, cars will be represented as entities but with different levels of detail

Entity Set: Collection of similar entities(e.g. all employees).

- All entities in an entity set have the same set of attributes
- Each entity set has a*key*
- Each attribute has a *domain*
- Can map entity set to a relation easily

Eg.Employees–CollectionofAccountsdepartmentemployeesandHRdepartment employees.

EMPLOYEES

SSN	NAME	SAL
321-23-3241	Kim	23,000
645-56-7895	Jones	45,000

Entity Type: A collection of entities that share common properties or the same attributes (e.g. EMPLOYEE) or characteristics

- Each Entity Type is described by its **NAME** and attributes
- The Entity Type describes the ―Schema‖ or ―Intension‖ for a set of entities oCollectionofallentities ofaparticularentitytypeatagivenpointintime
- is called the ―Entity Set‖ or ―Extension‖ of an Entity Type
- EntityType and EntitySet are customarily referred tobythesame name

3. Define Attributes of an Entity Type?

Attributes

The properties of an entity can be described by what is known as attributes.

Eg. SSN, Name, Age may be the attributes of the entity set Employees. These attributes are common for all the entities in this entity set.

4. Define Key Attributes of an Entity Type?

Key attributes (Candidate keys and identifiers):

Each entity type must have an attribute or set of attributes that distinguishes one instance from other instances of the same type

Example:-

CAR entity type with two key attributes, Registration and VehicleID

(a)

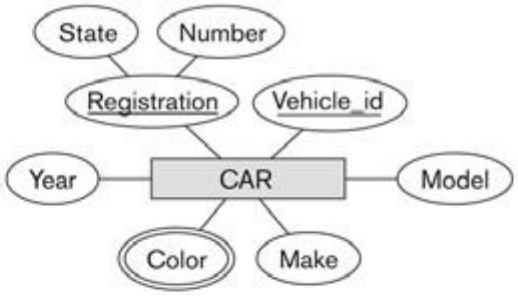


Figure 3.7
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

(b)

CAR

Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR_1
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR_2
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

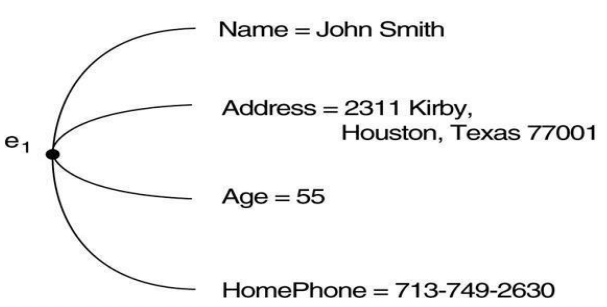
CAR_3
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

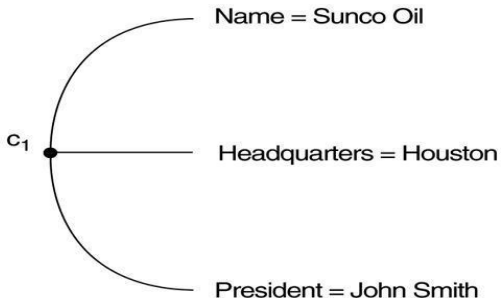
⋮

Key attribute

Attribute (or combination of attributes) that uniquely identifies each instance of an entity type

Two entities, employee e_1 and company c_1, and their attributes





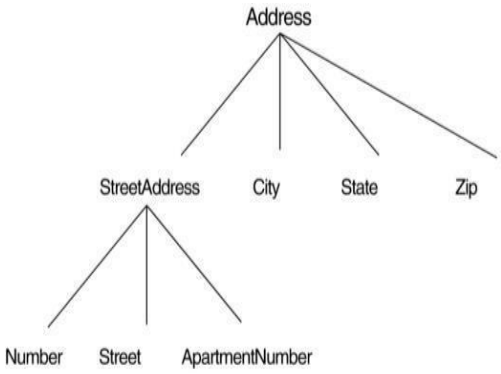
Value Sets (Domains) of Attributes

5. What are the types of attributes and Explain with example?

Each attribute is associated with a set of values called as *dom*(Ai). If the attribute Age takes, say, values between 18 and 58 years, the domain of this attribute is 18-58

Attribute Types

- **Simple or Atomic** - hold single value and not composed of any other attributes. Example:RegNo,Age,Date_of_Birth,Dept_No,Sex,etc.
- **Composite:** attributes that can be sub-dived into some more attributes are called as composite attributes.
  - Eg. Address - A hierarchy of composite attributes.



- **Single valued:** hold only one value for a particular entity. Eg. Age, Sex
- **Multivalued:** attributes having more than one value. Eg. DLocation, Qualification
- **Stored:** non-derivable Eg. Sex,DOB
- **Derived:** can be derived from other attribute(s). Eg. Age, Total

Null Valued Attributes:

A particular entity may not have an applicable value for an attribute

- **Tertiary-Degree:** Not applicable for a person with no university education
- **Home-Phone:** Not known if it exists
- **Height:** Not known at present time

Type of Null Values

- Not Applicable
- Unknown
- Missing

Complex Attributes: Composite and multi valued attributes can be nested in an arbitrary way. Parenthesis ( ) for composite attributes and Brackets { } for multi-valued attributes

Example:-

Assume a person can have more than one residence and each residence can have multiple telephones

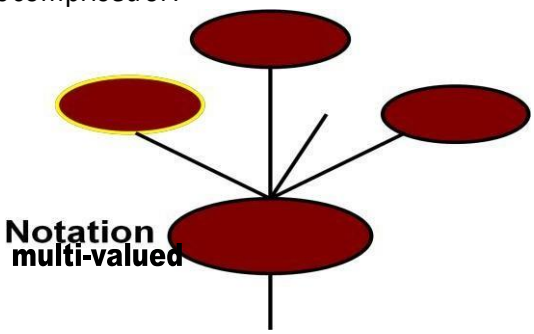
{AddressPhone ({ Phone (AreaCode,PhoneNum) },Address (StreetAddresss (Number, Street, AptNo), City,State,PostalCode)) }

Difference between:-

a) **Composite Vs. Simple Attributes**

Composite attributes can be divided into smaller parts which represent simple attributes with independent meaning

- **Simple Attribute:** Aircraft-Type
- **Complex Attribute:** Aircraft-Location which is comprised of:
  - Aircraft-Latitude
  - Aircraft-Longitude
  - Aircraft-Altitude



b) **Single Vs Multivalued Attributes**

Simple attributes can either be single-valued or Single-valued:

- Gender =F

Notation 

- Multivalued: Degree = {BSc, MInfTech}

Notation 

… An attribute in the relational model is always single valued – Values are atomic!

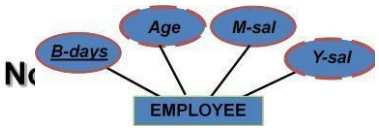
Derived Vs. Stored Attributes

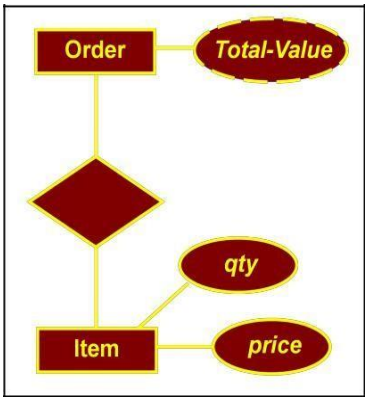
Some attribute values can be derived from

related attribute values:

Age®Date-B-day Y-Sal®

12*M-Sal

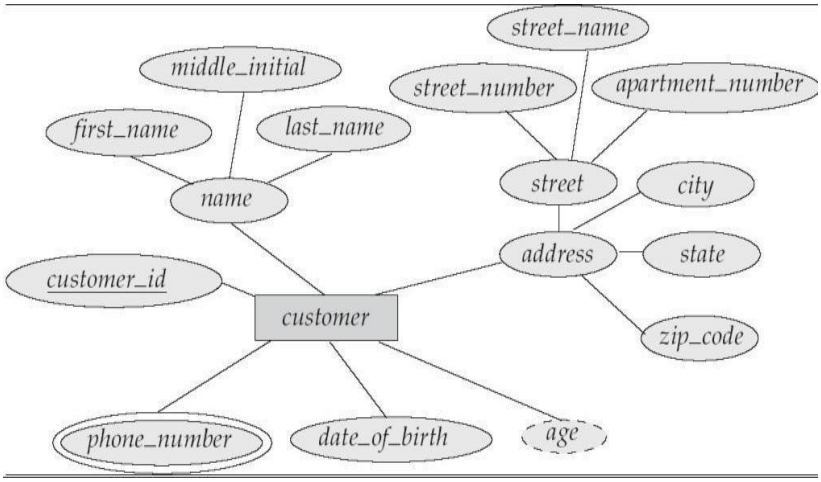




Some attribute values can be **derived from attributed** values of re***lated entities***
total-value ® sum (qty * price)

E-R Diagram With Composite, Multivalued, and Derived Attributes

- **Ellipses** represent attributes
- **Double ellipses** represent multi valued attributes.
- **Dashed ellipses** denote derivedattributes



6. Write schema diagram for company database ?

1. An entity type DEPARTMENT with attributes **Name, Number, Locations, Manager, and Man-agerStartDate. Locations** is the only multivalued attribute.Specify that both Name and Number are (separate) key attributes, because each was specified to be unique.

2. An entity type PROJECT with attributes **Name, Number, Location, and Controlling Department.** BothNameandNumberare(separate)keyattributes.

3. An entity type EMPLOYEE with attributes **Name, SSN (for social security number), Sex, Address, Salary, BirthDate, Department, and Supervisor**. Both Name and Address may be composite attributes.

4. An entity type DEPENDENT with attributes **Employee, DependentName, Sex, BirthDate, and Relationship** (to the employee).

DEPARTMENT
Name, Number, {Locations}, Manager, ManagerStartDate

PROJECT
Name, Number, Location, ControllingDepartment

EMPLOYEE
Name (FName, MInit, LName), SSN, Sex, Address, Salary,
BirthDate, Department, Supervisor, {WorksOn (Project, Hours)}

DEPENDENT
Employee, DependentName, Sex, BirthDate, Relationship

7. Define Relationships, Relationship Types, Roles, and Structural Constraints?

Relationship Types, Sets and Instances:

Relationship

There is an association among two or more entities (e.g EMPLOYEE(John) works in DEPT(Pharmacy department))

Relationship Type

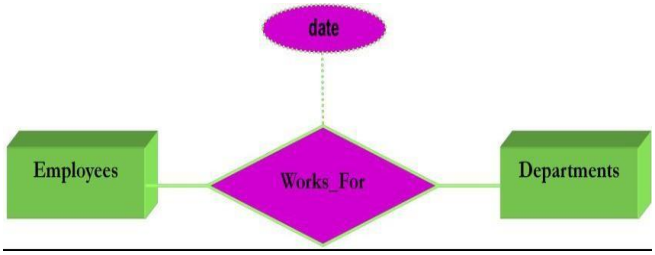
Defines the relationship and defines the structure of the Relationship Set

Relationship Set

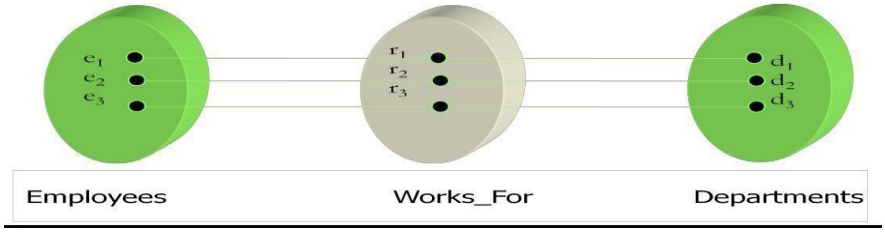
Represents a set of relationship instances

Relationship Type and corresponding Set are customarily referred to by the same name

In ER diagrams, relationship types are displayed as diamond-shaped boxes, which are connected by straight lines to the rectangular boxes representing the participating entity types. The relationship name is displayed in the diamond-shaped box



Instance of Works_For



8. Define Relationship Degree, Role Names, and Recursive Relationships?

Degree of a Relationship Type

The degree of a relationship type is defined as —the number of participating entities.

2. entities: Binary Relationship

3 entities: Ternary Relationship

n entities: N-ary Relationship

Same entity type could participate in multiple relationship **types.** More than one relationship type can exist with the same participating entity types.

Example:-

MANAGES and WORKS_FOR are distinct relationships between EMPLOYEE and DEPARTMENT, but with different meanings and different relationship instances.

9. Explain types of relationship with example?

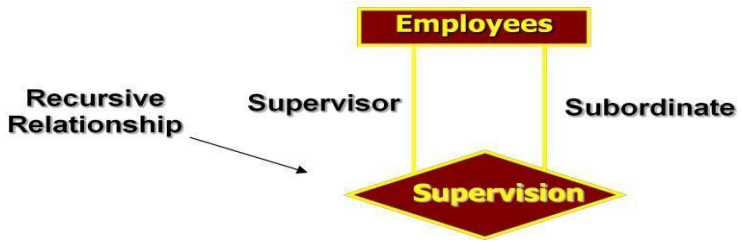
Unary relationship/Recursive Relationship:

1. If only one entity is participating in forming the relationship.
2. Same entity type can participate more than once in the same relationship type under different **"roles"** Such relationships are called —**Recursive Relationships**
3. Each entity type that participates in a relationship type plays a particular **role** in the relationship.
4. The **role name** signifies the role that a participating entity from the entity type plays in **each relationship instance**, and helps to **explain what the relationship means.**
5. ***Same* entity type participates more than once in a** relationship type in *different roles.* In such cases the role **name becomes essential for distinguishing** the meaning of each participation

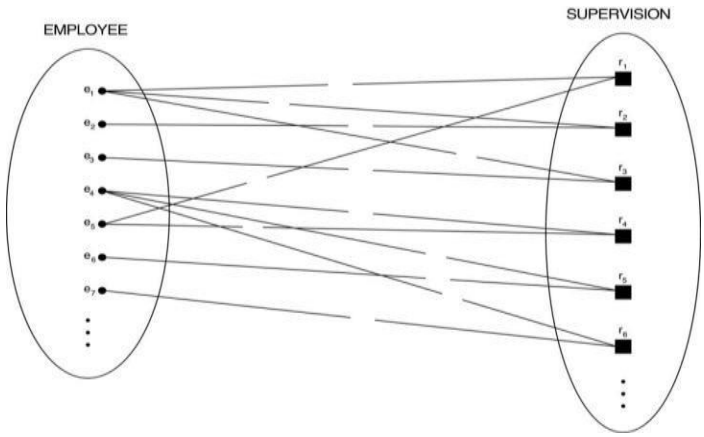
Such relationship types are called **recursive relationships**

Example:-

Each relationship instance in SUPERVISION associates two employee entities ej and ek, one of which plays the role of supervisor and the other the role of supervisee.

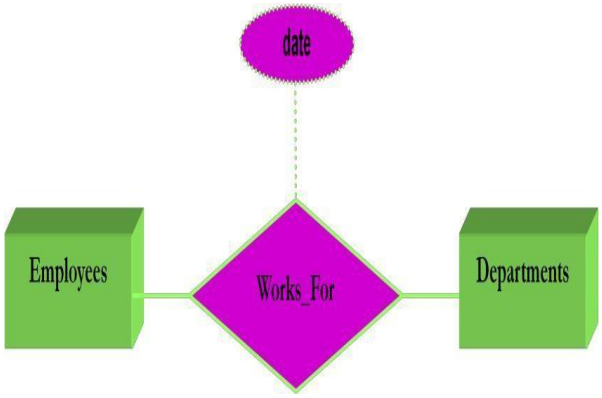


Instance for supervision:-

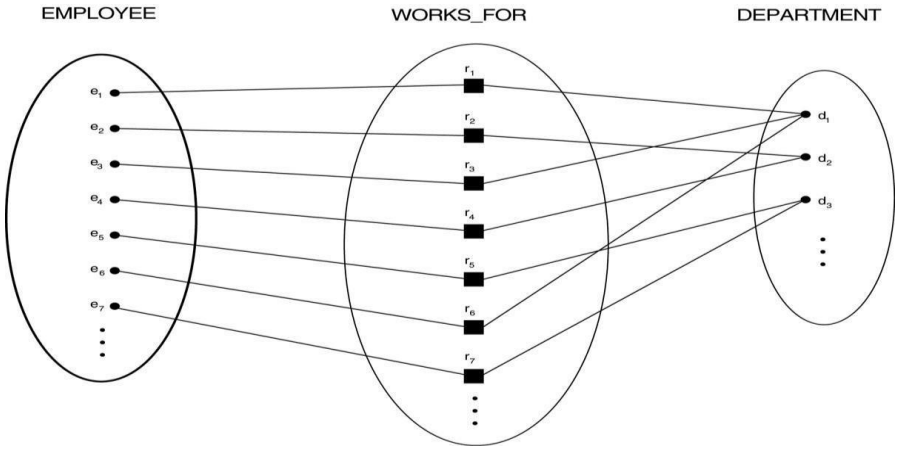


Binary relationship

:If the number of participating entities is two, then it is called as binary relationship.

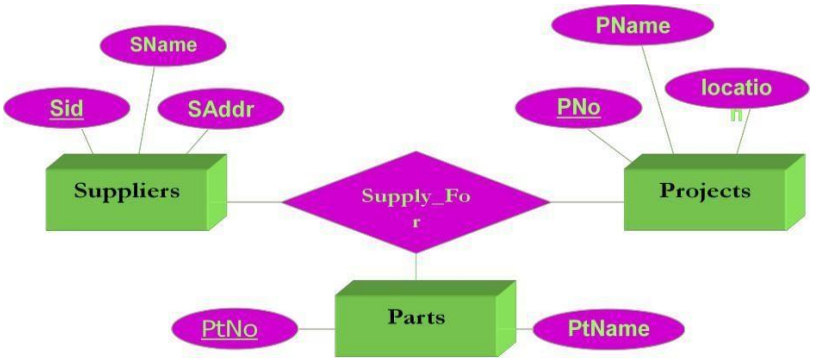


Instance of -Works_For

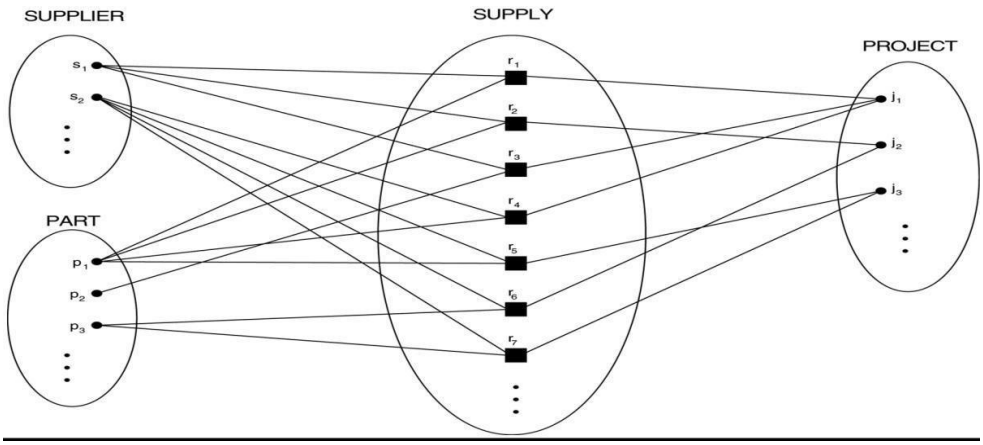


Ternary Relationship

Supposeifthenumberofparticipatingentitiesarethree,thenitiscalledasternary relationship/N-ary relationship.



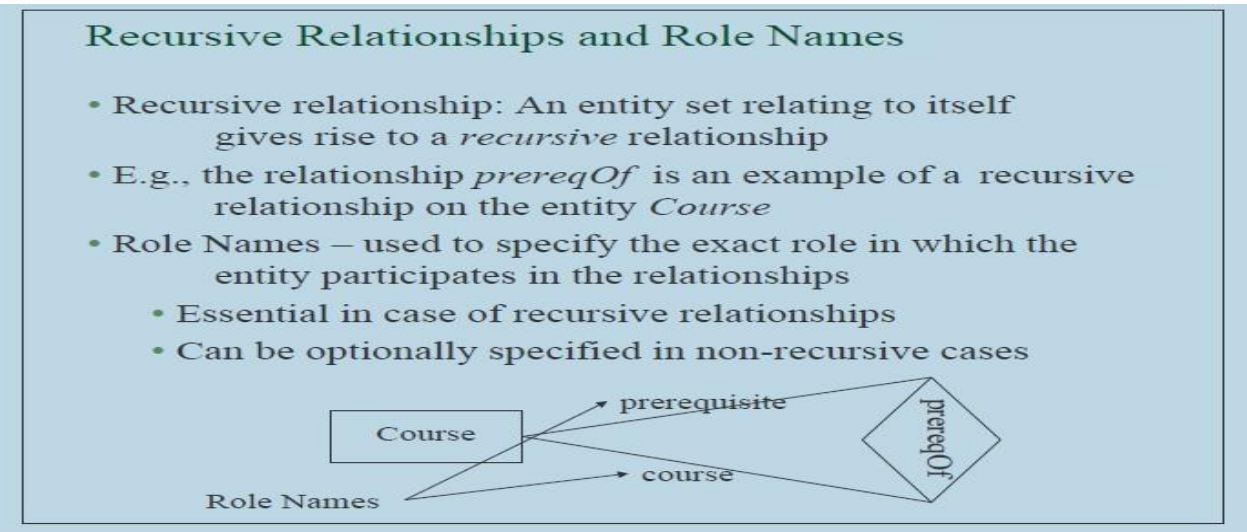
Instance of ternary Relationship Set-Supply_For



10. Explain how role names are assigned in case of recursive relationships? Illustrate

this concept with an example.

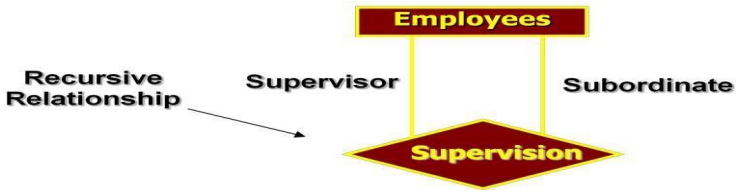
An entity set relating to itself give rise to a recursive relationship



Role Names and Recursive Relationships Chapter 3

- Each entity type that participates in a relationship type plays a particular **role** in the relationship.
- The **role name** signifies the role that a participating entity from the entity type plays in **each relationship instance**, and helps to **explain what the relationship means.**
- ***same* entity type participates more than once in a** relationship type in *different roles.* In such cases the role **name becomes essential for distinguishing** the meaning of each participation Such relationship types are called **recursive relationships**
- Each relationship instance in SUPERVISION associates two employee entities ej and ek, one of which plays the role of supervisor and the other the role of supervisee.

Example1:- need to mention role names

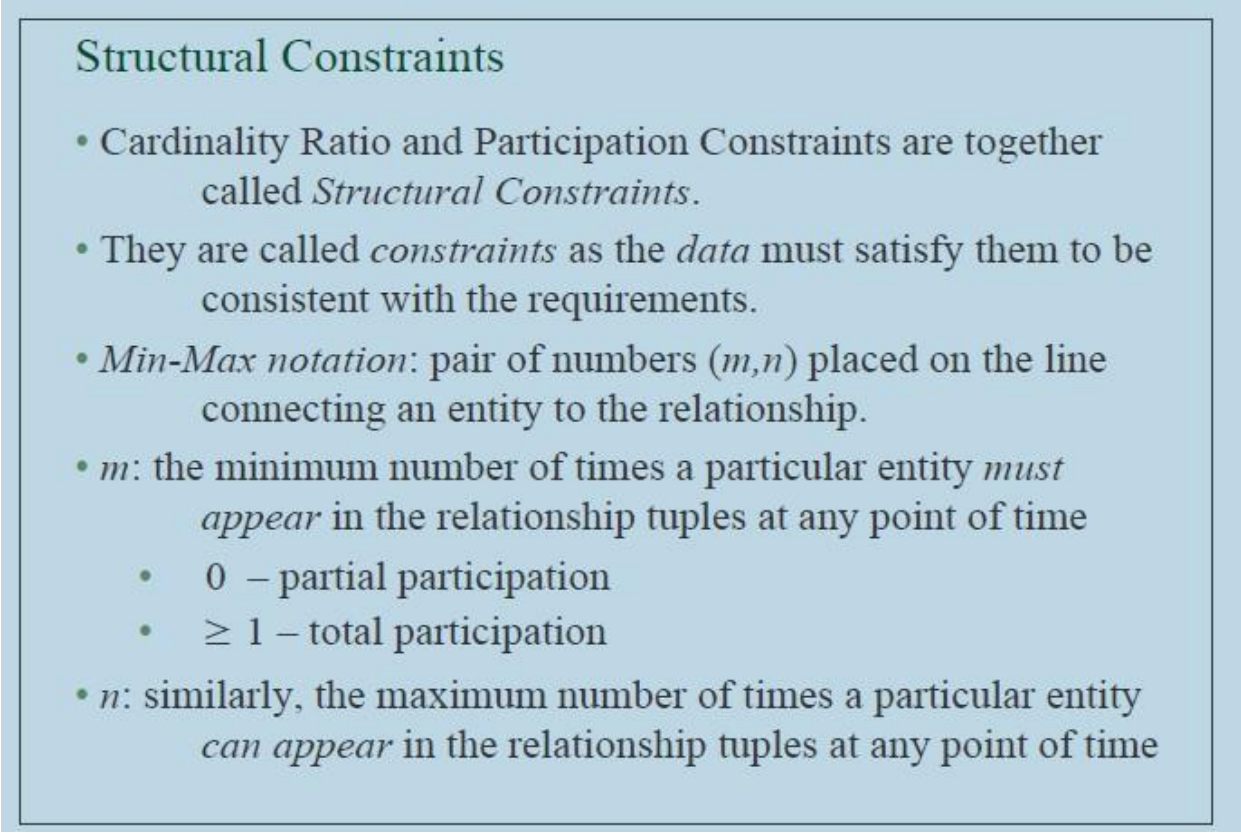


Structural constraint in relationship

11. Explain structural constraint with example?

It is represented by combination of cardinality and participationconstraint

- Together called —Structural Constraints.
- StructuralConstraints–onewaytoexpresssemanticsofrelationships
- Constraintsarerepresentedbyspecificnotationin theERdiagram



Cardinality constraint

- is a constraint on a relationship
- it characterizes relationshipsfurther
- given as (mapping) cardinality : how many entities of an entity set participate in a relationship
- especially useful for binary relationships

Cardinality ratio: The *cardinality ratio* for a binary relationship specifies the number of relationship instances that an entity can participate in

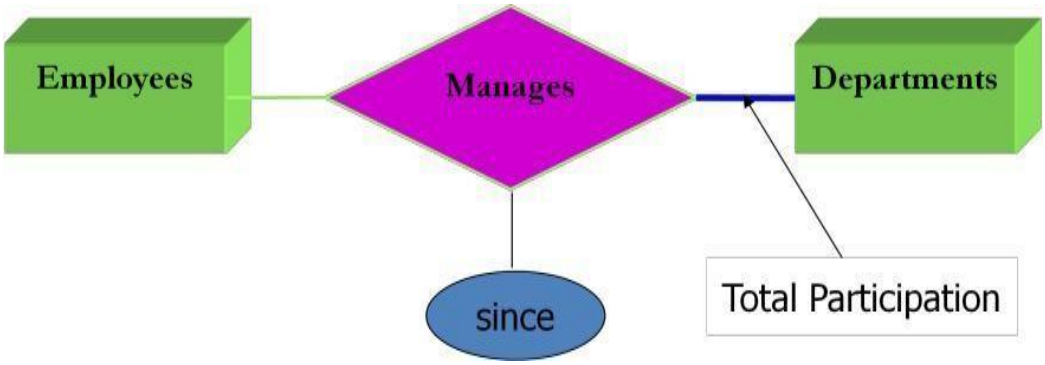
Participation constraint

Whether all entities or few entities participated in a relation represent the constraint

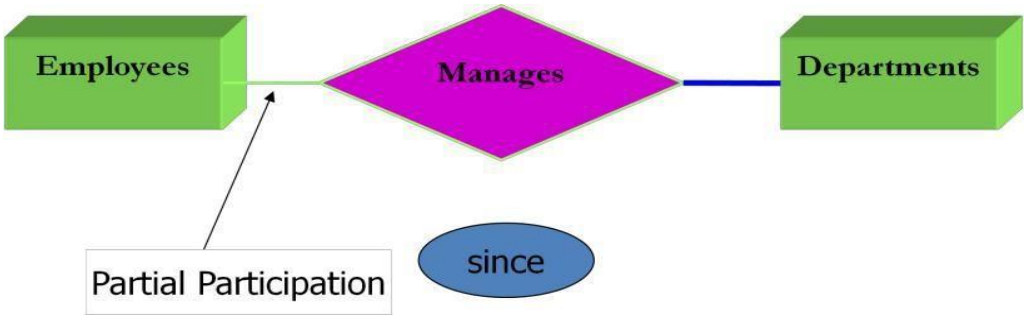
Types of participation

12. What are different types of Participation Constraints?

Total Participation: assume that the company policy is that every department should have a manager. This implies that there will be a full or total participation between Departments and Manages. Therefore, this type of participation is called as total participation (represented as thick or double line).



Partial Participation: When an entity set E does not participate fully with a relationship and in turn to the participating entity, such a constraint is called as partial participation



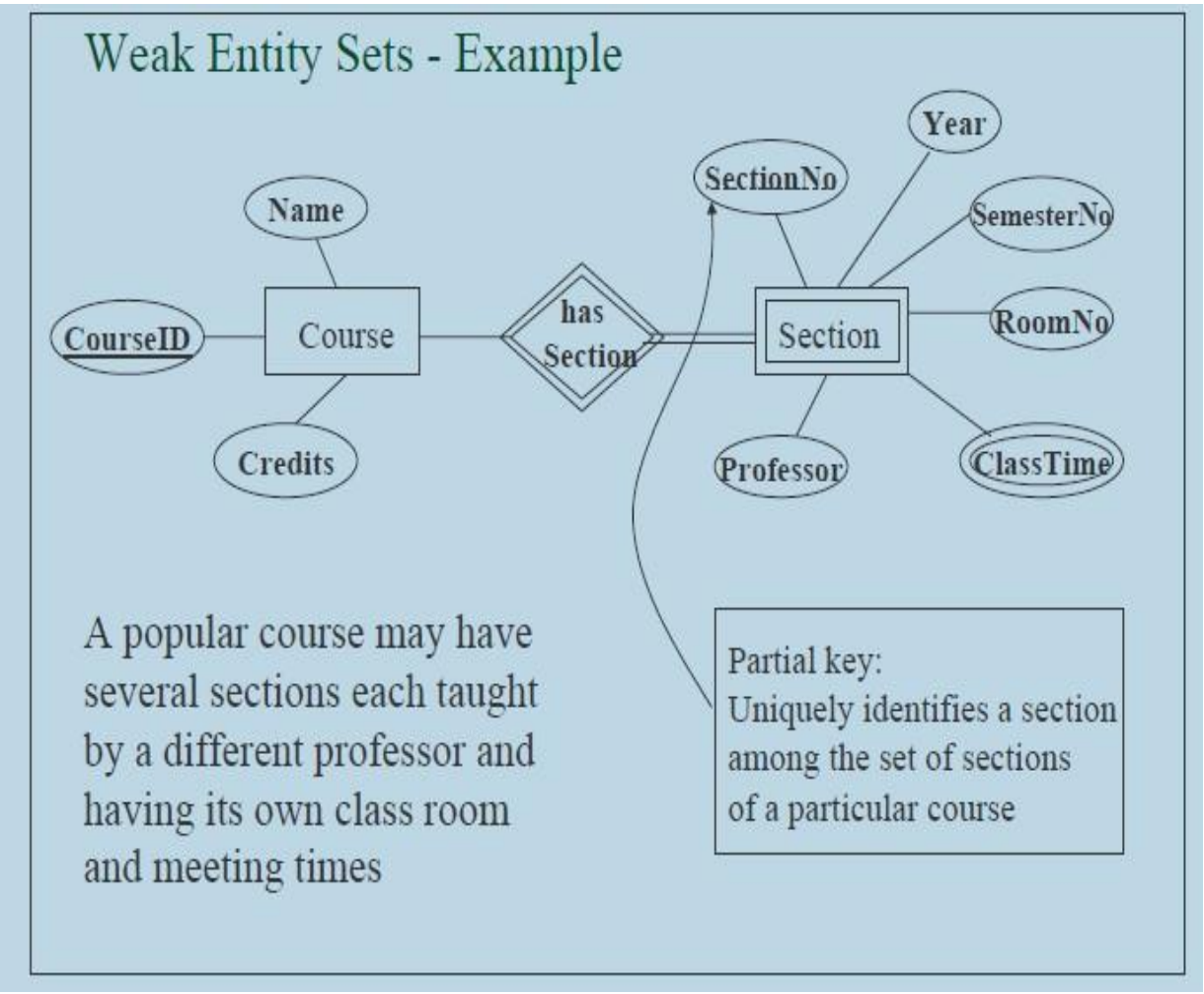
Representing Participation

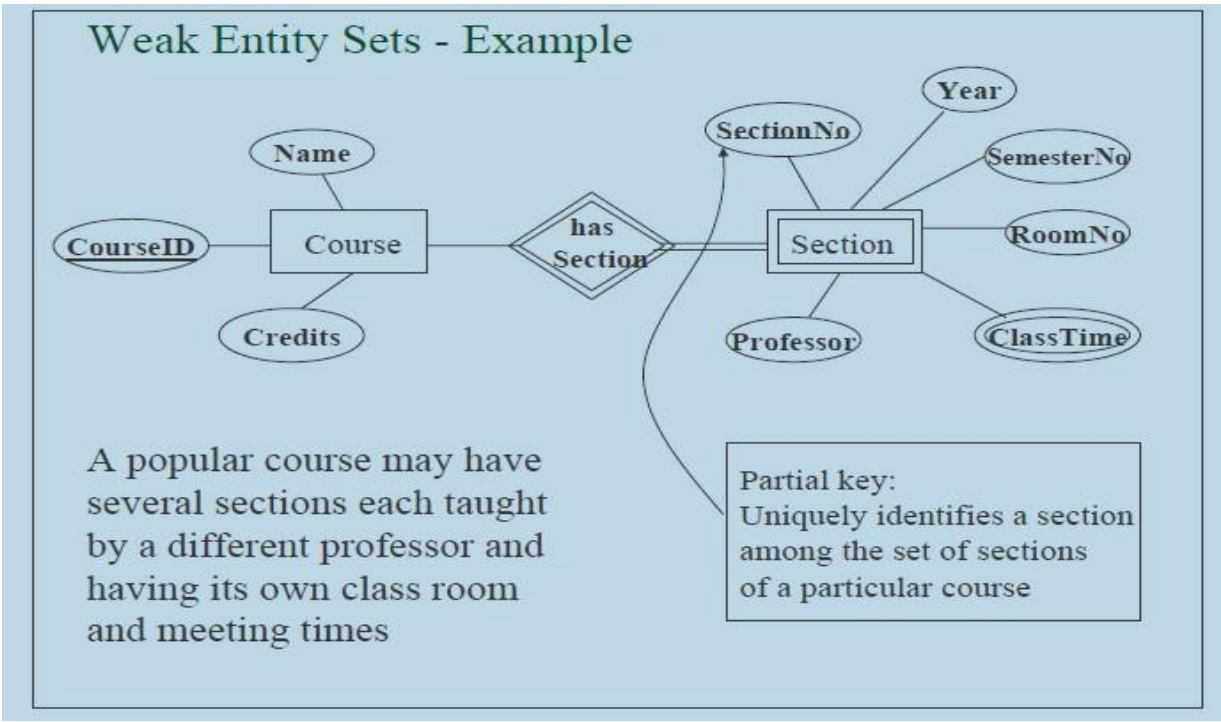
- Every employee must work for a department

Weak entity

13. Explain weak entity with example?

- A weak entity set is an entity set such that its key is composed of attributes some or all are of which belong to another entity set.
- Weak entity sets do not have their own key attributes (only partial key allowed). A weak entity will always
- have total participation with the identifying owner entity. because a weak entity cannot be identified without an owner entity





- The owner entity set and the weak entity set must participate in 1:M relationship
- Primary key of weak entity = primary key of its strong entity + discriminating Attribute of weak entity within the context of strong entity
- Relationship type that relates a weak entity type to its owner the identifying relationship of the weak entity type

Example:

Suppose that a DEPENDENT entity is identified by the dependent's first name and birhtdate, *and* the specific EMPLOYEE that the dependent is related to. DEPENDENT is a weak entity type with EMPLOYEE as its identifying entity type via the identifying relationship type

DEPENDENT_OF

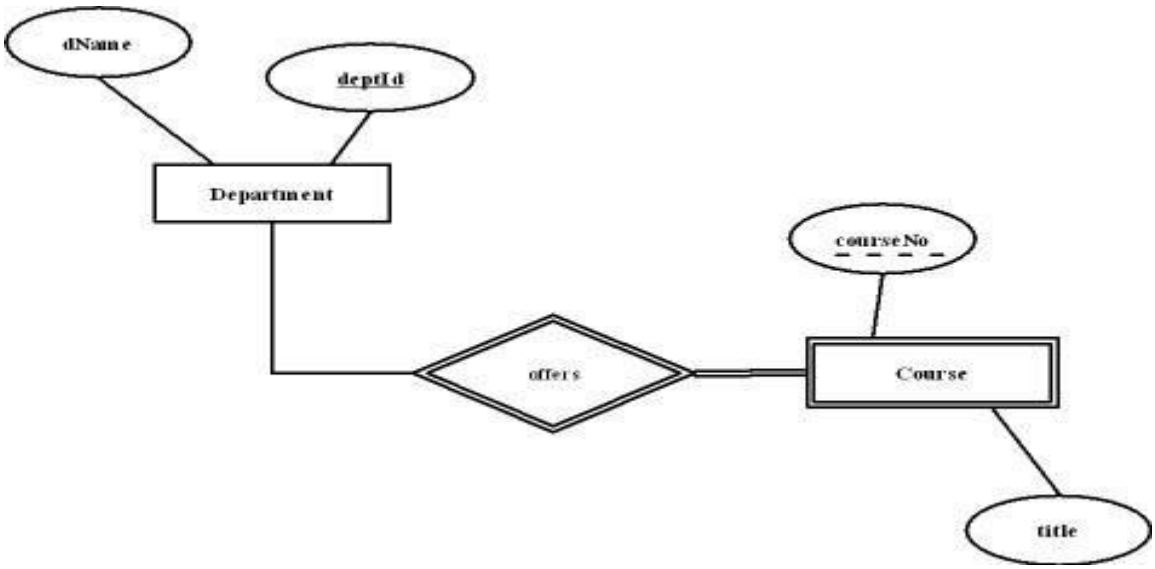
DepName – partial key cannot uniquely identify dependent. Rather to identify a dependent we must include the employee number (SSN) attribute as well from owner entity

14. Define and explain Partial Key, with example?

An attribute is a Partial Key if a Key from a related entity type must be used in conjunction with the attribute in question to uniquely identify instances of a corresponding entity set.

For example, suppose "Course Number" is an attribute of the Course entity type in our design for a University database. Suppose Course Number alone cannot uniquely identify courses. Rather, to identify a course we must include the Department Number attribute as well.

At the U of W, keys for a Course entity type include the Key of the Department entity set: ACS-2914, ACS-3902, BIO-1914



ER Diagram

Refining the ER Design for the COMPANY Database

Refine the database design of above by changing the attributes that represent relationships into relationship types.

1. MANAGES, a 1:1 relationship type between EMPLOYEE and DEPARTMENT. EMPLOYEE participation is partial. DEPARTMENT participation is full (total participation. The attribute Start Date is assigned to this relationship type.

2. WORKS_FOR, a 1:N relationship type between DEPARTMENT and EMPLOYEE. Both participations are total.

3. CONTROLS, a 1:N relationship type between DEPARTMENT and PROJECT. The participation of PROJECT is total, whereas that of DEPARTMENT is determined to be partial..

4. SUPERVISION, a 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role). Both participations are determined to be Partial, after the users indicate that not every employee is a supervisor and not every employee has a supervisor.

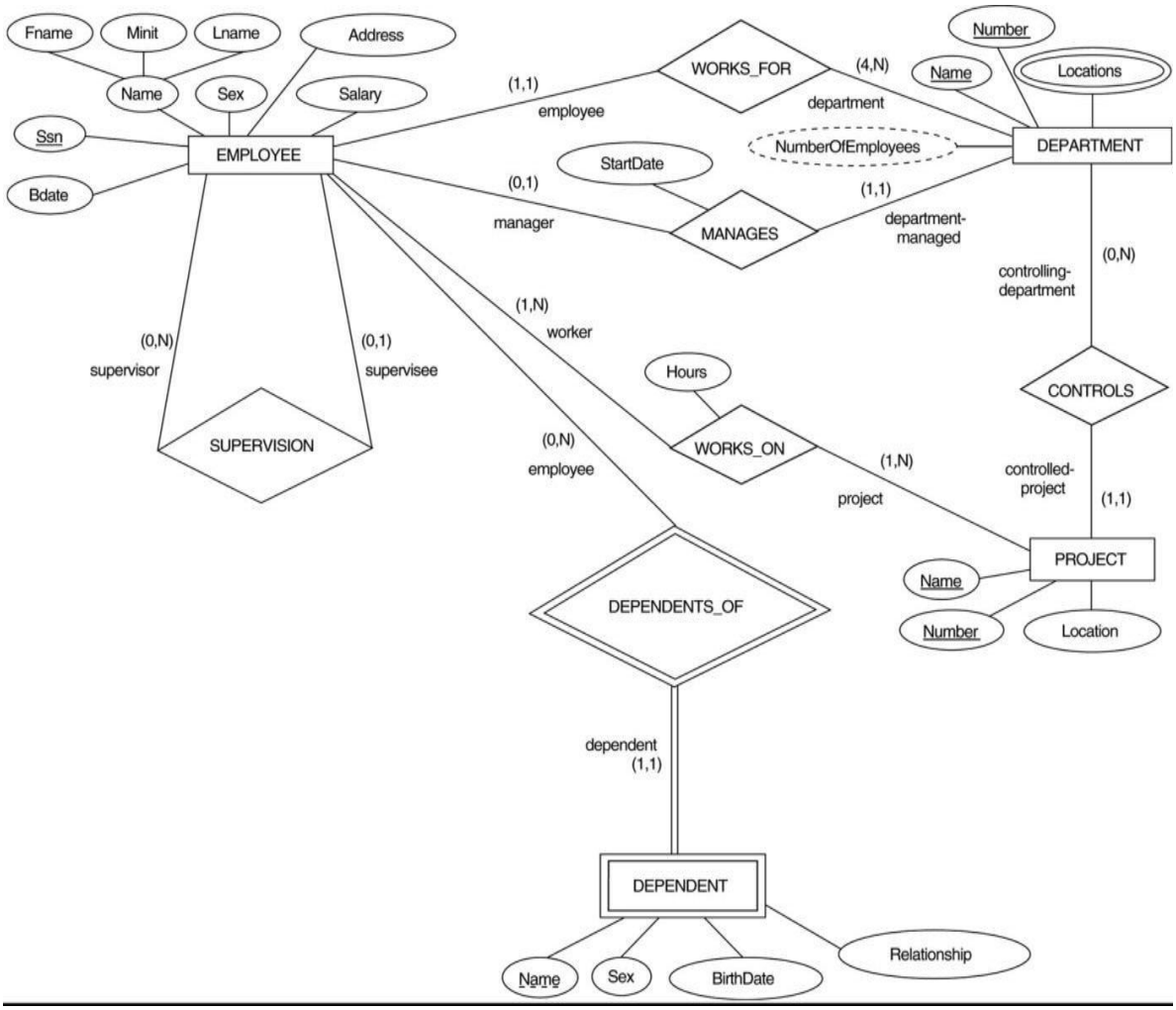
5. WORKS_ON, determined to be an M:N relationship type with attribute Hours, after the users indicate that a project can have several employees working on it. Both participations are determined to be total.

6. DEPENDENTS_OF, a 1:N relationship type between EMPLOYEE and DEPENDENT, which is also the identifying

relationship for the weak entity type DEPENDENT. The participation of EMPLOYEE is partial, whereas that of DEPENDENT is total.

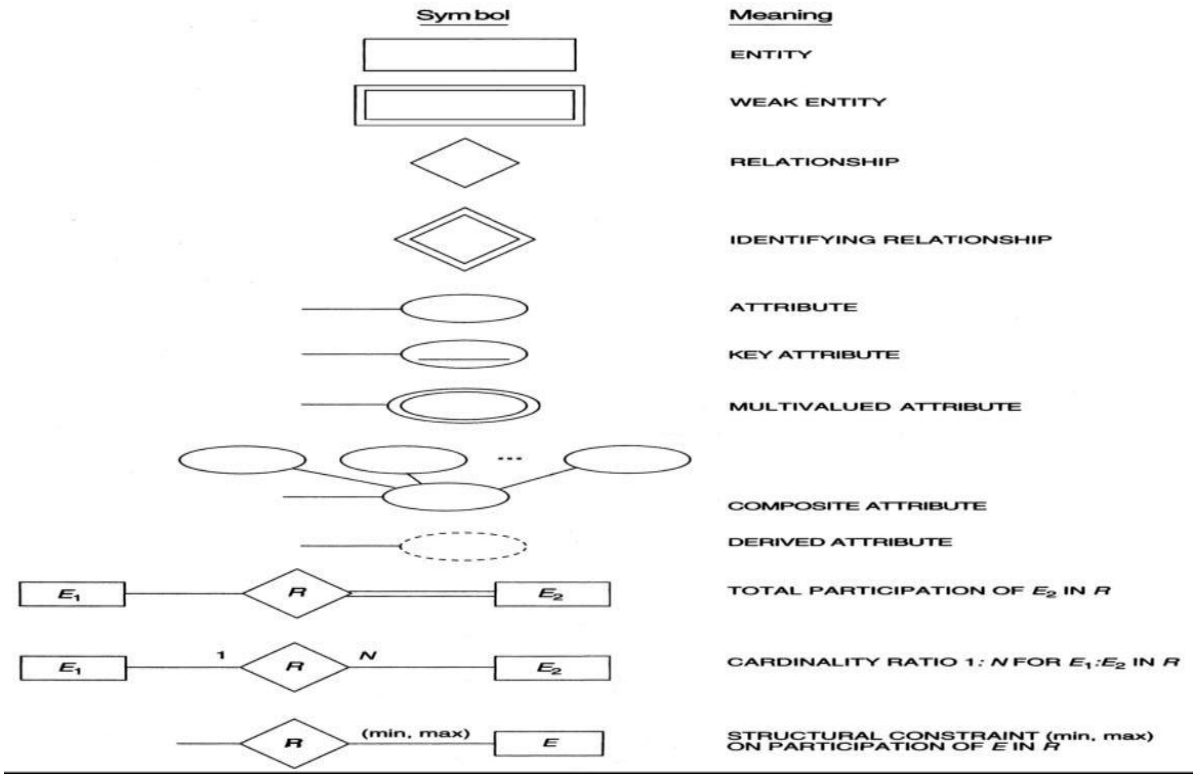
Remove from the entity types in above initial Figure all attributes that have been refined into relationships. These include Manager and ManagerStartDate from DEPARTMENT; Controlling Department from PROJECT; Department, Supervisor, and WorksOn from EMPLOYEE; and Employee from DEPENDENT

ER diagrams for the COMPANY schema, with structural constraints specific using (min, max) notation.



15. Explain notation used in ER diagrams?

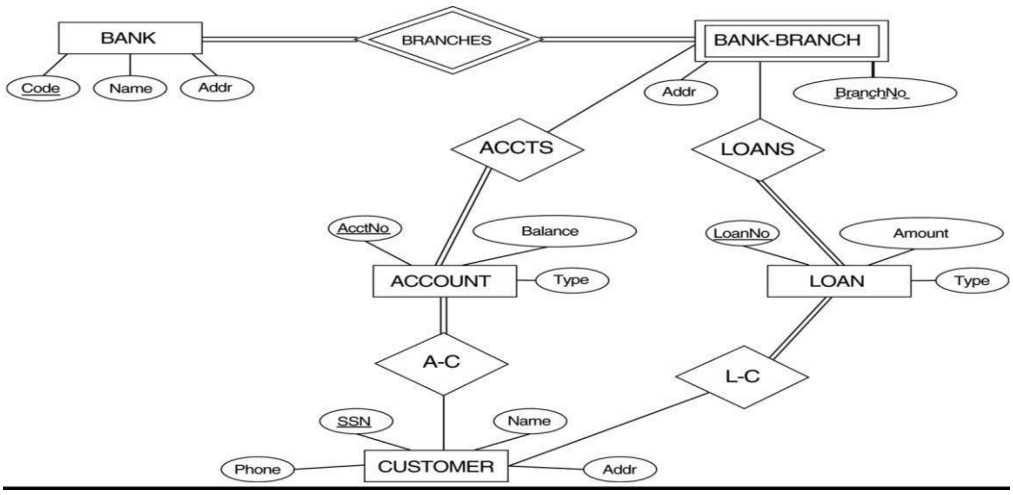
Summary of Notation for ER Diagrams

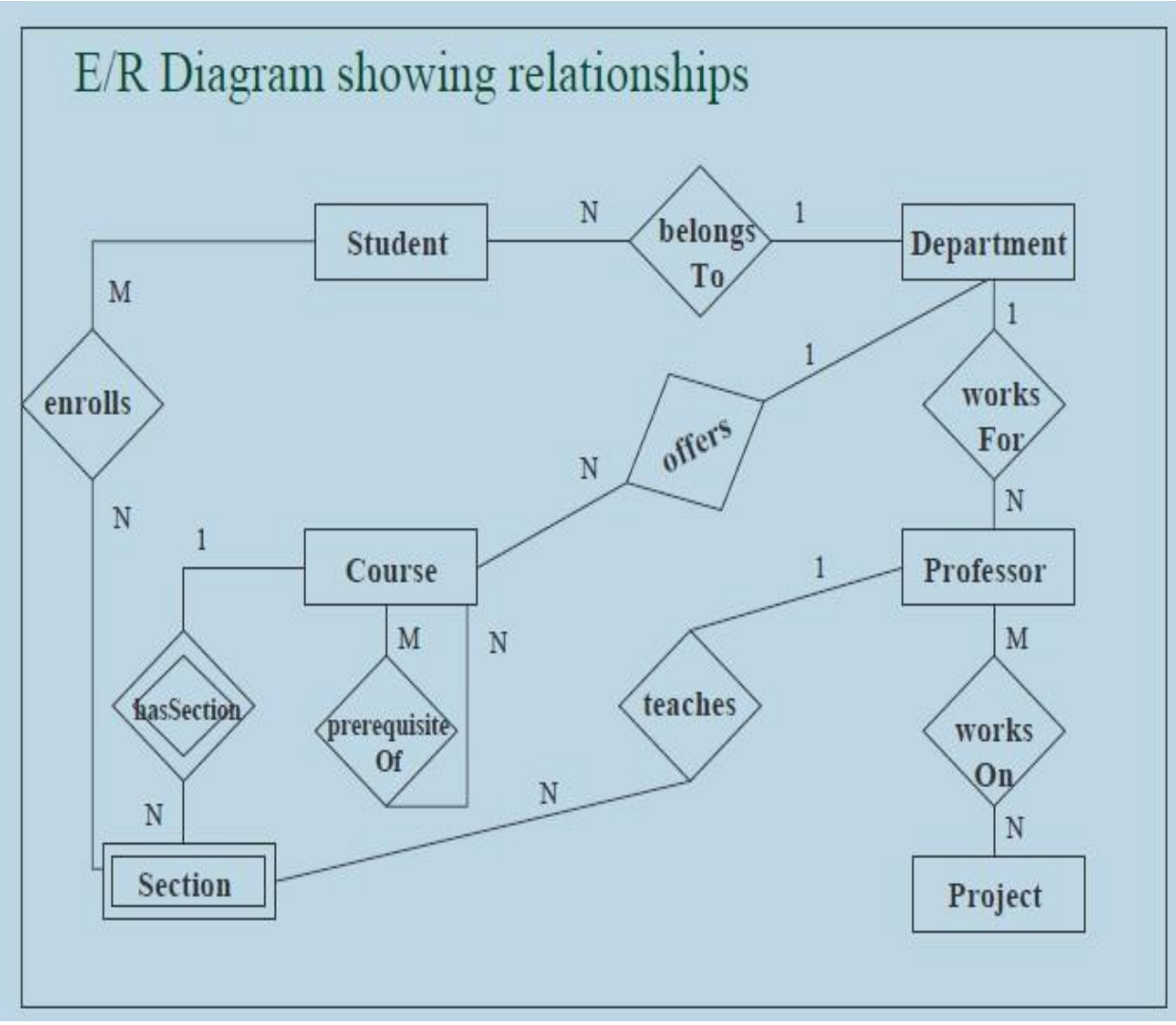


Alternative Notations for ER Diagrams

- one alternative ER notation for specifying structural constraints on relationships.
- This notation involves associating a pair of integer numbers (min, max) with each participation of an entity type E in a relationship type R, where 0 1 min 1 max and max 1. The numbers mean that, for each entity e in E, e must participate in at least min and at most max relationship instances in R at any point in time.
- In this method, min = 0 implies partial participation, whereas min > 0 implies total participation
- one uses either the cardinality ratio/single line/double line notation or the min/max notation

An ER diagram for a BANK database schema





Strong vs. Weak Entity Sets

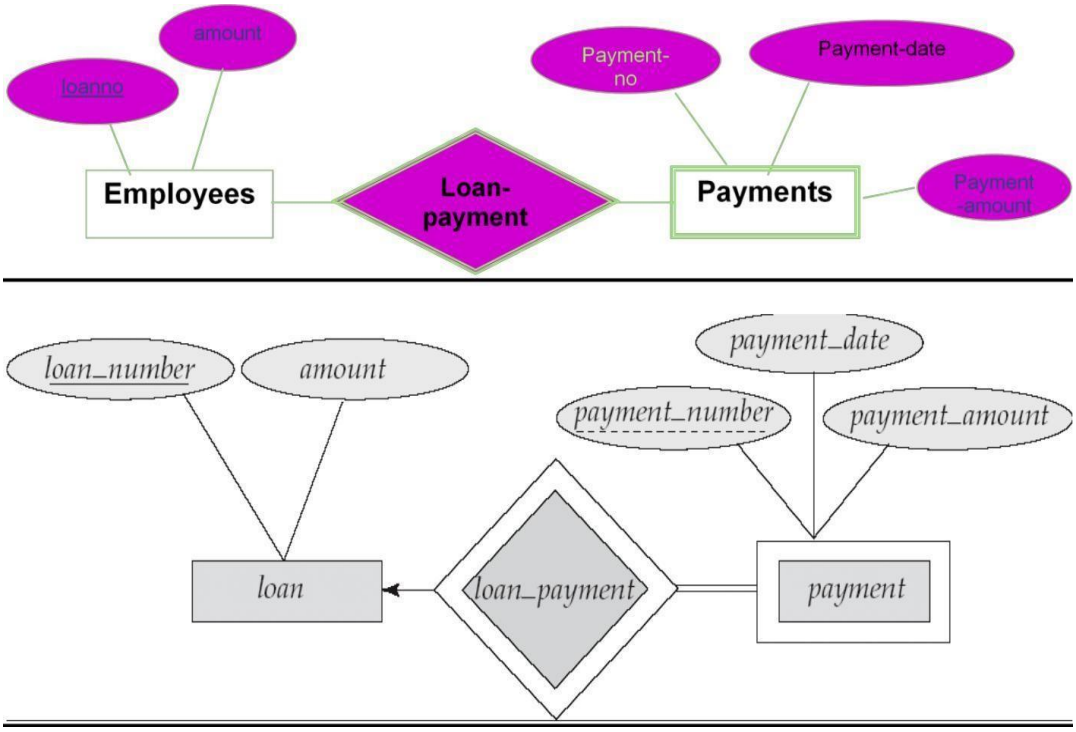
Strong entity set:– Has sufficient attributes to form a primary key

Weak entity set:– Lacks sufficient attributes to form *any* key

But every entity set needs a key; What to do? Must *import* attributes from strong entity set(s)

– A weak entity set member is subordinate to the dominant entity from strong entity set providing attributes to complete its key

Example



Example of Other Notation: UML Class Diagrams

TheUMLmethodologyisbeingusedextensivelyinsoftwaredesignandhasmanytypes of diagrams for various software design purposes.

InUMLclassdiagrams,a**class**(similartoanentitytypeinER)isdisplayedasabox that includes threesections:

1. The Top section - **class name** (similar to entity type name)
2. The middle section-the **attributes**
3. The last section includes **operations** that can be applied to individual objects (similartoindividualentities inanentityset)of the class. butOperationsare*not* specified in ER diagrams.

Example:-

Consider the EMPLOYEE class in Figure below Its attributes are Name, Ssn, Bdate, Sex, Address, and Salary.

1. Thedesignercanoptionally specifythe**domain**(ordatatype)ofanattributeif desired, by placing a colon (:) followed by the domain name or description, as illustratedbytheName,Sex,and BdateattributesofEMPLOYEEinFigure below.

2. A **composite attribute** ismodeled as a **structureddomain**,asillustratedbythe Name attribute ofEMPLOYEE.
3. A **multivalued attribute** will generally be modeled as a separate class, as illustrated by the LOCATION class in Figure below
4. Relationship types are called **associations** in UML terminology, and relationship instances are called **links**.

- A **binary association** (binary relationship type) is represented as a line connecting the participating classes (entity types), and may optionally have a name.
- A relationship attribute, called a **link attribute**, is placed in a box that is connected to the association's line by a dashed line.
- The (min, max) notation is used to specify relationship constraints, which are called **multiplicities** in UML terminology.
- Multiplicities are specified in the form *min..max*,and an asterisk (*) indicates no maximum limit on participation.
- However, the multiplicities are placed *on the opposite ends of the relationship* when compared with the (min, max) notation.
- InUML,a single asterisk indicates a multiplicity of0..*,andasingle1indicatesa multiplicity of 1..1.
- A recursive relationship type is called a **reflexive association** in UML, and the role names—like the multiplicities— are placed at the opposite ends of an association when compared with the placing of role names.
- In UML, there are two types of relationships: association and aggregation. **Aggregation** is meant to represent a relationship between a whole object and its component parts, and it has a distinct diagrammatic notation.
- UML also distinguishes between **unidirectional** and **bidirectional** associations (or aggregations). In the unidirectional case, the line connecting the classes is displayed with an arrow to indicate that only one direction for accessing related objects is needed.
- If no arrow is displayed, the bidirectional case is assumed, which is the default.
- Weak entities can be modeled using the UML construct called **qualified association** (or **qualified aggregation**); this can represent both the identifying relationship and the partial key, which is placed in a box attached to the owner class.
- This is illustrated by the DEPENDENT class and its qualified aggregation to EMPLOYEE in Figure below
- InUMLterminology,the partial key attribute Dependent_name is calledthe discriminator.

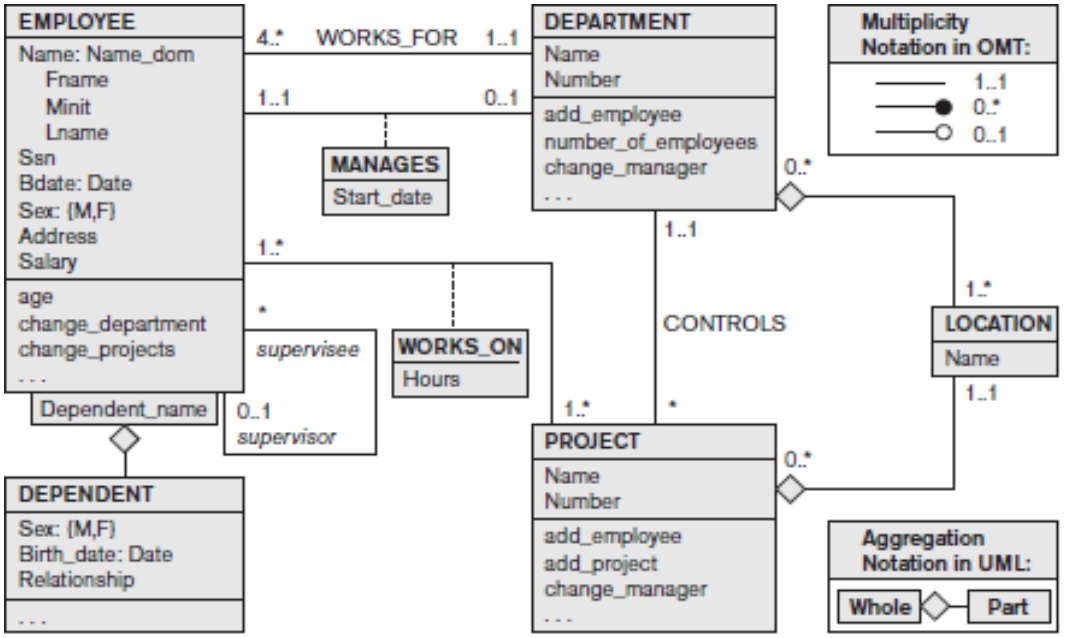


Figure 3.16
The COMPANY conceptual schema in UML class diagram notation.

Generalization and specialization

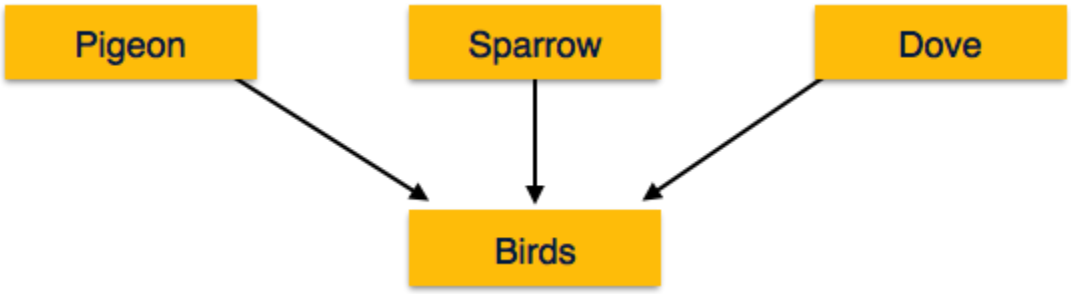
The ER Model has the power of expressing database entities in a conceptual hierarchical manner. As the hierarchy goes up, it generalizes the view of entities, and as we go deep in the hierarchy, it gives us the detail of every entity included.

Going up in this structure is called **generalization**, where entities are clubbed together to represent a more generalized view. For example, a particular student named Mira can be generalized along with all the students. The entity shall be a student, and further, the student is a person. The reverse is called **specialization** where a person is a student, and that student is Mira.

16. Brief about Generalization and specialization?

Generalization

As mentioned above, the process of generalizing entities, where the generalized entities contain the properties of all the generalized entities, is called generalization. In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics. For example, pigeon, house sparrow, crow and dove can all be generalized as Birds.



Specialization

Specialization is the opposite of generalization. In specialization, a group of entities is divided into sub-groups based on their characteristics. Take a group 'Person' for example. A person has name, date of birth, gender, etc. These properties are common in all persons, human beings. But in a company, persons can be identified as employee, employer, customer, or vendor, based on what role they play in the company.

Similarly, in a school database, persons can be specialized as teacher, student, or a staff, based on what role they play in school as entities.