

CST204 Database Management System

Module 1

Introduction

Syllabus – Overview

- Module 1 : Introduction and Entity Relationship (ER) model
- Module 2 : Relational Model
- Module 3 : SQL DML and Physical Data Organization
- Module 4 : Normalization
- Module 5: Transaction, concurrency and recovery, recent topics

Introduction

Syllabus – Module 1

- Concepts and Overview of Database Management System (DBMS)
 - Characteristics of Database System
 - Database Users
 - Structured, semi-structured and unstructured data
 - Data Models and Schema
 - Three Schema architecture
 - Database Languages
 - Database architectures and Classification
- ER model
 - Basic concepts
 - entity set & attributes, Notations, Relationships and constraints, cardinality, Participation, Notations, weak entities, relationships of degree 3

Introduction to Databases

Database

- Database is a collection of related data
- Raw facts are called Data (data is plural, datum is singular)
- Data means known facts that can be recorded and have implicit meaning
- For example, the names, telephone numbers, and address of people can be recorded in an indexed address book
- This collection of related data with an implicit meaning is a database

Introduction to Databases

Database

- A database has the following properties:
- A database represents some aspect of the real world, sometimes called the mini world or the universe of discourse (UoD)
- A database is a logically coherent collection of data with some inherent meaning
- A database is designed, built, and populated with data for a specific purpose

Introduction to Databases

Database

- A database has some source from which the data is derived , some degree of interaction with events in the real world, and an audience that is actively interested in its contents
- A database can be of any size and complexity
- A database can be generated and maintained manually or it may be computerized
- For example, a library card catalog is a database that may be created and maintained manually

Introduction to Databases

Database Management System (DBMS)

- A computerized database may be created and maintained either by a group of application programs written specifically for that task or by a Database Management System (DBMS)
- A database management system (DBMS) is a computerized system that enables users to create and maintain a database
- The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications

Introduction to Databases

Database Management System (DBMS)

- Defining a database involves specifying the data types, structures, and constraints of the data to be stored in the database
- The database definition or descriptive information is also stored by the DBMS in the form of a database catalog or dictionary; it is called meta-data
- Constructing the database is the process of storing the data on some storage medium that is controlled by the DBMS
- Manipulating a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data

Introduction to Databases

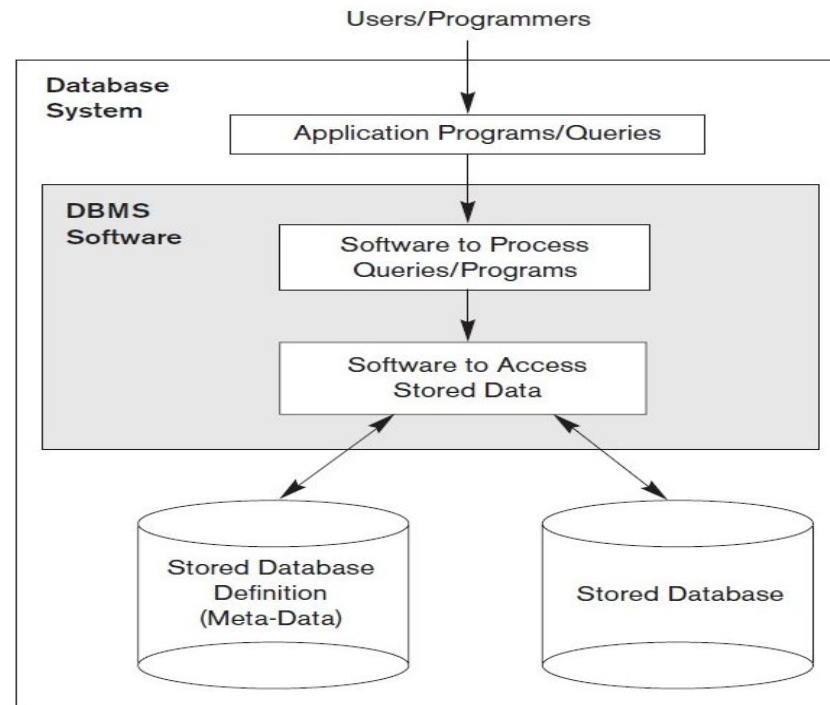
Database Management System (DBMS)

- Sharing a database allows multiple users and programs to access the database simultaneously
- An application program accesses the database by sending queries or requests for data to the DBMS
- A query typically causes some data to be retrieved; a transaction may cause some data to be read and some data to be written into the database
- Other important functions provided by the DBMS include protecting the database and maintaining it over a long period of time
- Protection includes system protection against hardware or software malfunction (or crashes) and security protection against unauthorized or malicious access

Introduction to Databases

Database Management System (DBMS) - Example

- A university database for maintaining information concerning students, courses, and grades in a university environment



Introduction to Databases

Database Management System (DBMS) - Example

- The database is organized as five files, each of which stores data records of the same type
- The STUDENT file stores data on each student
- The COURSE file stores data on each course
- The SECTION file stores data on each section of a course
- The GRADE_REPORT file stores the grades that students receive in the various sections they have completed
- The PREREQUISITE file stores the prerequisites of each course

Introduction to Databases

Database Management System (DBMS) - Example

- To define this database, we must specify the structure of the records of each file by specifying the different types of data elements to be stored in each record
- Each STUDENT record includes data to represent the student's Name, Student_number, Class, and Major (such as mathematics or 'MATH' and computer science or 'CS')
- Each COURSE record includes data to represent the Course_name, Course_number, Credit_hours, and Department (the department that offers the course), and so on

Introduction to Databases

Database Management System (DBMS) - Example

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

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	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

Introduction to Databases

Database Management System (DBMS) - Example

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

Introduction to Databases

Database Management System (DBMS) - Example

- We must also specify a data type for each data element within a record
- For example, we can specify that Name of STUDENT is a string of alphabetic characters, Student_number of STUDENT is an integer, and Grade of GRADE_REPORT is a single character from the set {'A', 'B', 'C', 'D', 'F', 'I'}
- We may also use a coding scheme to represent the values of a data item
- For example, we represent the Class of a STUDENT as 1 for freshman, 2 for sophomore, 3 for junior, 4 for senior, and 5 for graduate student

Introduction to Databases

Database Management System (DBMS) - Example

- To construct the UNIVERSITY database, we store data to represent each student, course, section, grade report, and prerequisite as a record in the appropriate file
- Notice that records in the various files may be related
- For example, the record for Smith in the STUDENT file is related to two records in the GRADE_REPORT file that specify Smith's grades in two sections
- Similarly, each record in the PREREQUISITE file relates two course records: one representing the course and the other representing the prerequisite
- Most medium-size and large databases include many types of records and have many relationships among the records

Introduction to Databases

Database Management System (DBMS) - Example

- Database manipulation involves querying and updating
- Examples of queries are as follows:
 - Retrieve the transcript—a list of all courses and grades—of ‘Smith’
 - List the names of students who took the section of the ‘Database’ course offered in fall 2008 and their grades in that section
 - List the prerequisites of the ‘Database’ course
- Examples of updates include the following:
 - Change the class of ‘Smith’ to sophomore
 - Create a new section for the ‘Database’ course for this semester
 - Enter a grade of ‘A’ for ‘Smith’ in the ‘Database’ section of last semester

Introduction to Databases

Characteristics of Database System

- In traditional file processing, each user defines and implements the files needed for a specific software application as part of programming the application
- Redundancy in defining and storing data results in wasted storage space and in redundant efforts to maintain common up to date data
- In the database approach, a single repository maintains the data that is defined once and then accessed by various users

Introduction to Databases

Characteristics of Database System

- The main characteristics of database approach versus the file processing approach is as follows:
 - The self describing nature of database system
 - Insulation between programs and data, and data abstraction
 - Support of multiple views of the data
 - Sharing of data and multiuser transaction processing

The self describing nature of database system

- A fundamental characteristic of the 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

Introduction to Databases

Characteristics of Database System

The self describing nature of database system

- This definition is stored in the DBMS catalog, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data
- The information stored in the catalog is called meta-data, and it describes the structure of the primary database
- The DBMS catalog will store the definitions of all the files in the University database

Introduction to Databases

Characteristics of Database System

The self describing nature of database system

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

Introduction to Databases

Characteristics of Database System

The self describing nature of database system

- Whenever a request is made to access, say, the Name of a STUDENT record, the DBMS software refers to the catalog to determine the structure of the STUDENT file and the position and size of the Name data item within a STUDENT record
- By contrast, in a typical file-processing application, the file structure and, in the extreme case, the exact location of Name within a STUDENT record are already coded within each program that accesses this data item

Introduction to Databases

Characteristics of Database System

Insulation between programs and data, and data abstraction

- In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access that file
- DBMS access programs do not require such changes in most cases
- The structure of data files is stored in the DBMS catalog separately from the access programs
- We call this property *program-data independence*

Introduction to Databases

Characteristics of Database System

Insulation between programs and data, and data abstraction

- For example, a file access program may be written in such a way that it can access only STUDENT records
- If we want to add another piece of data to each STUDENT record, say the Birth_date, such a program will no longer work and must be changed
- By contrast, in a DBMS environment, we only need to change the description of STUDENT records in the catalog to reflect the inclusion of the new data item Birth_date; no programs are changed
- The next time a DBMS program refers to the catalog, the new structure of STUDENT records will be accessed and used

Introduction to Databases

Characteristics of Database System

Insulation between programs and data, and data abstraction

- An operation (also called a function or method) is specified in two parts
- The **interface** (or signature) of an operation includes the operation name and the data types of its arguments (or parameters)
- The **implementation** (or method) of the 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
- This may be termed program-operation independence

Introduction to Databases

Characteristics of Database System

Insulation between programs and data, and data abstraction

- The characteristic that allows program data independence and program operation independence is called data abstraction
- A data model is a type of data abstraction that is used to provide conceptual representation
- 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
- The data model hides storage and implementation details that are not of interest to most database users

Introduction to Databases

Characteristics of Database System

Support of multiple views of data

- A database has many users, each user may require a different perspective or view of the database
- A view may be a subset of the database or it may contain virtual data that is derived from the database file but is not explicitly stored
- A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views

Introduction to Databases

Characteristics of Database System

Sharing of data and multiuser transaction processing

- A multiuser DBMS must allow multiple users to access the database at the same time
- This 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 are correct

Introduction to Databases

Characteristics of Database System

Sharing of data and multiuser transaction processing

- A fundamental role of multiuser DBMS software is to ensure that concurrent transactions operate correctly and efficiently
- A transaction is an executing program or process that includes one or more database accesses, such as reading or updating database records
- Each transaction is supposed to execute a logically correct database access if executed in its entirety without interference from other transactions

Introduction to Databases

Database Users

- Many people are involved in the design, use and maintenance of database
- Major database users are
 - Database Administrators
 - Database Designers
 - End Users
 - System Analysts and Application Programmers (Software Engineers)

Introduction to Databases

Database Users

Database Administrator

- When many people use the same resources, there is a need for a chief administrator to oversee and manage these resources
- In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software
- Administering these resources is the responsibility of the database administrator (DBA)
- DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed
- DBA is accountable for problems such as security breaches and poor system response time
- In large organization, the DBA is assisted by a staff that carries out these function

Introduction to Databases

Database Users

Database Designers

- They are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store the data
- These tasks are mostly undertaken before the database is actually implemented and populated with data
- It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meets these requirements
- In many cases the designers are on the staff of DBA and may be assigned other staff responsibilities after the database design is completed
- Database designers typically interact with each potential group of users and develop views of the database that meet the data and processing requirements of these group
- Each view is 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

Introduction to Databases

Database Users

End Users

- End Users are the people whose jobs require access to the database for querying, updating, and generating reports
- There are several categories of end users
- **Casual end users:** they occasionally access the database, but they may need different information each time
- They use a sophisticated database query language to specify their requests and are typically middle or high level managers or other occasional browsers
- **Naïve or parametric end users:** they make up a sizable portion of database end users
- Their main job function revolves around constantly querying and updating database, using standard types of queries and updates – called **canned transactions** – that have been carefully programmed and tested
- Bank tellers check account balances and post withdrawals and deposits

Introduction to Databases

Database Users

End Users

- Reservation agents for airlines, hotels, and car rental companies check availability for a given request and make reservations
- **Sophisticated end users:** include engineers, scientists, business analyst, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their own applications to meet their complex requirements
- **Standalone users:** maintain personal databases using ready-made program packages that provide easy-to-use menu based or graphics based interfaces
- An example is the user of a tax package that stores a variety of personal financial data for tax purpose

Introduction to Databases

Database Users

System Analysts and Application Programmers (Software Engineers)

- System Analysts determine the requirements of end users, especially naïve and parametric end users, and develop specifications for standard canned transaction that meet these requirements
- Application programmers implement these specifications as programs then they test, debug, document, and maintain these canned transactions
- Such analyst and programmers commonly referred to as software developers or software engineers should be familiar with the full range of capabilities provided by the DBMS to accomplish their tasks

Introduction to Databases

Database Users

Other Users

- In addition to those who design, use and administer a database, others are associated with the design, development, and operation of DBMS software and system environment
- They include
 - DBMS system designers and implementers
 - Tool developers
 - Operators and maintenance personnel

Introduction to Databases

Database Users

DBMS system designers and implementers

- They design and implement the DBMS modules and interfaces as a software package
- A DBMS is a very complex software system that consists of many components, or modules, including modules for implementing the catalog, query language processing, interface processing, accessing and buffering data, controlling concurrency and handling data recovery and security
- The DBMS must interface with other system software such as the operating system and compilers for various programming languages

Introduction to Databases

Database Users

Tool developers

- They design and implement tools – the software packages that facilitate database modelling and design, database system design, and improved performance
- Tools are optional packages that are often purchased separately
- They include packages for database design, performance monitoring, natural language or graphical interfaces, prototyping, simulations, and test data generations
- In many cases, independent software vendors develop and market these tools

Introduction to Databases

Database Users

Operators and maintenance personnel

- They are responsible for the actual running and maintenance of the hardware and software environment for database system
- Although these categories of users are instrumental in making the database system available to end users, they typically do not use the database contents for their own purposes

Introduction to Databases

Structured, Semi-structured and unstructured data

- There are three forms of data that are relevant for all kinds of business applications

Structured data

- Structured data is generally tabular data that is represented by columns and rows in a database
- Databases that hold tables in this form are called relational databases
- The mathematical term “relation” specifies a formed set of data held as a table
- In structured data, all rows in a table have the same set of columns
- SQL (Structured Query Language) programming language used for structured data

Introduction to Databases

Structured, Semi-structured and unstructured data

Semi-Structured data

- Semi-structured data is information that doesn't consist of Structured data (relational database) but still has some structure to it

Unstructured data

- Unstructured data is information that either does not organize in a pre-defined manner or not have a pre-defined data model
- Unstructured information is a set of text-heavy but may contain data such as numbers, dates, and facts as well
- Videos, audio, and binary data files might not have a specific structure
- They're assigned to as unstructured data

Introduction to Databases

Data models, and schemas

Data models

- Data abstraction generally refers to the suppression of details of data organization and storage, and the highlighting of the essential features for an improved understanding of data
- A data model is a collection of concepts that can be used to describe the structure of a database and provides necessary means to achieve the abstraction
- Structure of a database means the data types, relationships and constraints that apply to the data
- Most data models include a set of operations for specifying retrievals and updates on the database

Introduction to Databases

Data models, and schemas

Categories of Data models

- Many data models have been proposed which can be categorized according to the types of concepts they use to describe the database structure
- **High Level conceptual data models** provide concepts that are close to the way many users perceive data
- **Low level or physical data models** provide concepts that describe the details of how data is stored on the computer storage media generally meant for computer specialists not for end users
- **Representational or implementation data models** provide concepts that may be easily understood by end users but that are not far from the physical data model

Introduction to Databases

Data models, and schemas

Categories of Data models

- Conceptual data models use concepts such as entities, attributes and relationships
- An **entity** represents a real world object or concept such as employee
- An **attribute** represents some property of interest that further describes the entity such as name of the employee
- A **relationship** among two or more entities represents the association among the entities such as a works on relationship between an employee and a project
- An **Entity Relationship model** is an example of high level conceptual data model

Introduction to Databases

Data models, and schemas

Categories of Data models

- **Relational data model, network and hierarchical model** are examples of Representational or implementational data model
- Representational data models represent data by using record structures and hence are sometimes called **Record based data models**
- **Object data model** is an example of a new family of higher level implementational data models that are closer to high level conceptual data model

Introduction to Databases

Data models, and schemas

Schemas

- The description of database is called **database schema**
- Schema is specified during database design and is not expected to change frequently
- Most data models have certain conventions for displaying schemas as diagrams
- A displayed schema is called **schema diagram**
- A typical schema diagram for university database is shown below:

Introduction to Databases

Data models, and schemas

Schemas

STUDENT

Name	Student_number	Class	Major
------	----------------	-------	-------

COURSE

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

PREREQUISITE

Course_number	Prerequisite_number
---------------	---------------------

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

GRADE_REPORT

Student_number	Section_identifier	Grade
----------------	--------------------	-------

Introduction to Databases

Data models, and schemas

Schemas

- The schema diagram displays the structure of each record type but not the actual instances of the record
- A schema diagram displays only some aspects of a schema such as name of record types and data items, and some types of constraints
- Other aspects are not specified in the schema diagram
- The actual data in a database may change quite frequently
- The data in the database at a particular moment in time is called a **database state** or **snapshot**
- It is also called the current set of **occurrences** or **instances** in database

Introduction to Databases

Data models, and schemas

Schemas

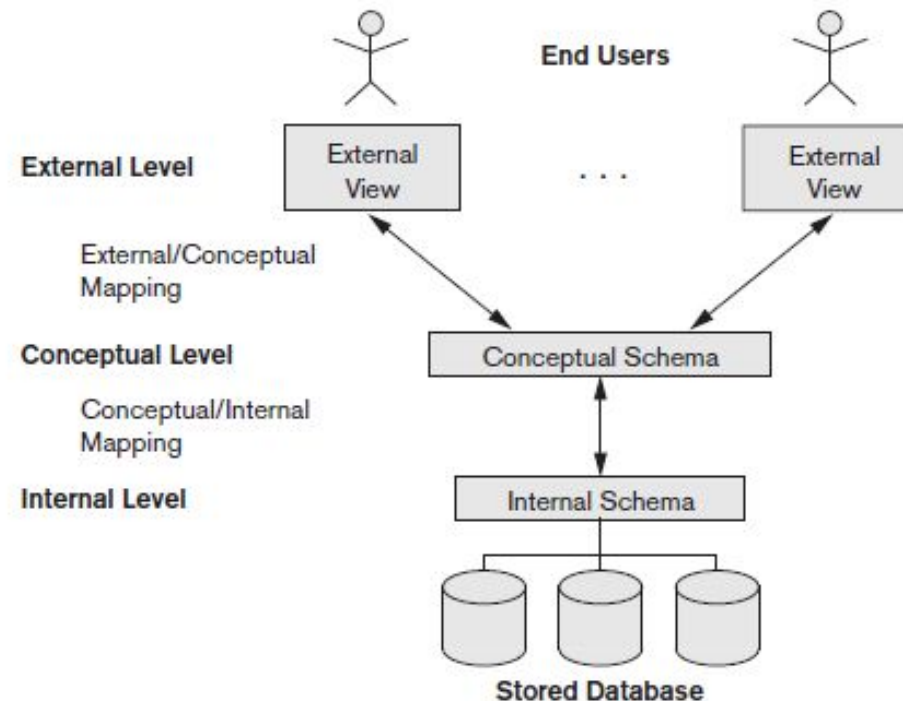
- When we define a new database we specify its schema only to the DBMS
- At this point, the corresponding database state is the empty state with no data
- We get the initial state of the database when the database is first populated with initial data
- DBMS stores the schema description in the meta data of DBMS catalog so that it can be referred whenever needed
- The schema is sometimes called the intension and a database state is called an extension of the schema

Introduction to Databases

Data models, and schemas

Three Schema Architecture

- The goal of three schema architecture is to separate the user application from the physical database



Introduction to Databases

Data models, and schemas

Three Schema Architecture

- In this architecture, schemas can be defined at the following three levels:
- **Internal Level** has an internal schema, which describes the physical storage structure of the database
- Internal schema uses a physical data model and describes the complete details of data storage and access path for the database
- **Conceptual level** has a conceptual schema which describes the structure of the whole database for a community of users
- It hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints

Introduction to Databases

Data models, and schemas

Three Schema Architecture

- Usually a representational data model is used to describe the conceptual schema when a database system is implemented
- Programmers and database administrators work at this level
- The **external level** or view level includes a number of external schemas or user views
- Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from the user group
- As in the previous level, each external schema is typically implemented using a representational data model

Introduction to Databases

Data models, and schemas

Three Schema Architecture

- The external schema describes the end user interaction with database systems
- Three schema architecture is a convenient tool with which the user can visualize the schema level in a database system
- Mapping is used to transform the request and response between various database levels of architecture
- In External / Conceptual mapping, it is necessary to transform the request from external level to conceptual schema
- In Conceptual / Internal mapping, DBMS transform the request from the conceptual to internal level

Introduction to Databases

Data models, and schemas

Three Schema Architecture – Data Independence

- Data independence is the capacity to change the schema at one level of a database system without having to change the schema at the next higher level
- Two types of data independence – logical data independence, physical data independence
- **Logical data independence** 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 items, to change constraints, or to reduce the database by removing a record type or data item

Introduction to Databases

Data models, and schemas

Three Schema Architecture – Data Independence

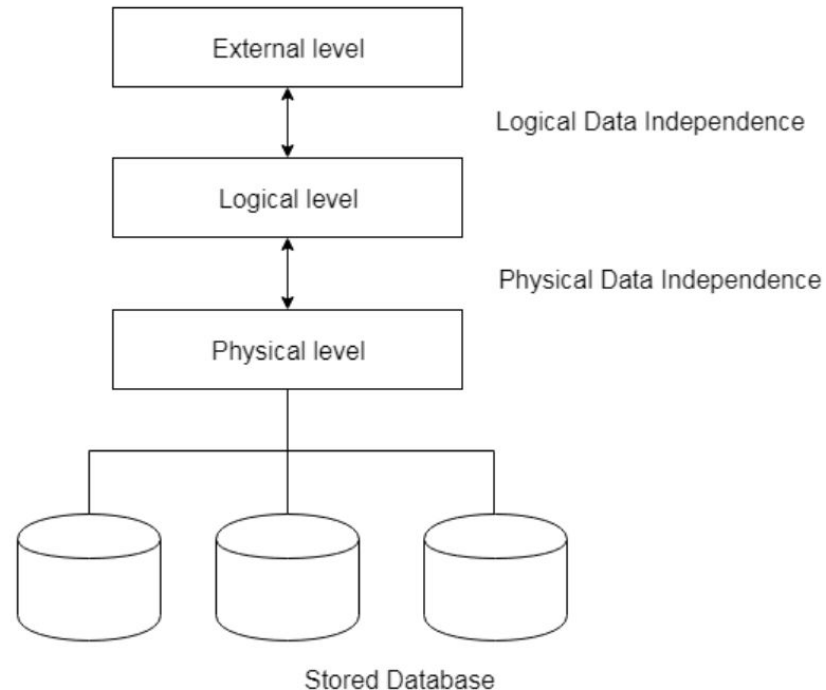
- **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema
- Hence the external schemas need not be changed as well
- Changes to internal schema may needed because some physical files were reorganized to improve the performance of retrieval or update
- 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 mapping information in the catalog

Introduction to Databases

Data models, and schemas

Three Schema Architecture – Data Independence

- Data independence occurs because when the schema is changed at some level, the schema at next higher level remains unchanged; only mapping between the two levels is changed



Introduction to Databases

Database languages

- Once the design of a database is completed and a DBMS is chosen to implement database, the first step is to specify conceptual and internal schema for the database and any mapping between the two
- In many DBMSs where no strict separation of levels is maintained, one language, called **Data Definition Language (DDL)** is used by the DBA and by database designers to define both schemas
- In many DBMSs where a clear separation of levels is maintained, DDL is used to specify the conceptual schema only; another language called **Storage Definition Language (SDL)** is used to specify internal schema
- There is no specific language that performs the role of SDL

Introduction to Databases

Database languages

- **View Definition Language (VDL)** is used to specify user views and their mappings to the conceptual schema but in most DBMSs the DDL is used to define both conceptual and external schema
- Once the database schema is completed and the database is populated with data we use **Data Manipulation Language (DML)** to manipulate the database – retrieval, insertion, deletion and modification of data
- **Data Control Language (DCL)** is used to retrieve the stored or saved data from the database
- **Transaction Control Language (TCL)** is used to run the changes made by the DML statement
- In current DBMSs the above mentioned languages are usually not considered distinct languages; rather a comprehensive integrated language is used – **Structured Query Language (SQL)**

Introduction to Databases

Database languages

- SQL represents a combination of DDL, VDL, DML, TCL and DCL
- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

Introduction to Databases

Database Architectures

- Architectures for database can be classified into two – Centralized DBMS Architecture (single tier) and Client – Server Architecture (Multi tier)

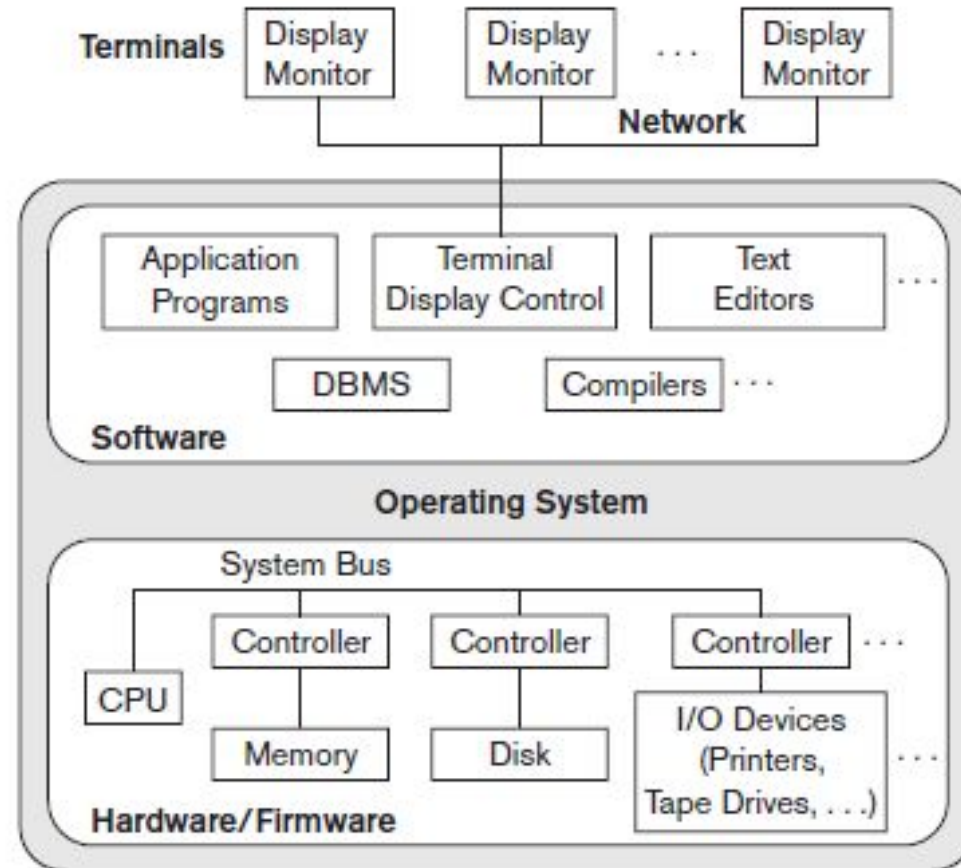
Centralized DBMS architecture

- In this architecture, the database is directly available to the user
- It means the user can directly sit on the DBMS and uses it
- Any changes done here will directly be done on the database itself
- It doesn't provide a handy tool for end users
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response

Introduction to Databases

Database Architectures

Centralized DBMS architecture



Introduction to Databases

Database Architectures

Client – Server architecture

- The idea is to define specialized servers with specific functionalities
- For example a machine designated as printer server by being connected to various printers; all print requests by the client are forwarded to this machine
- The client machines provide the user with appropriate interfaces to utilize these servers as well as with local processing powers to run local applications
- A client is typically a user machine that provides user interface capabilities and local processing
- A server is a system containing both hardware and software that can provide services to the client machines such as file access, printing, archiving or database access

Introduction to Databases

Database Architectures

Client – Server architecture

- Two main types of this category of architecture is **two tier** and **three tier**

Two Tier Client – Server Architecture

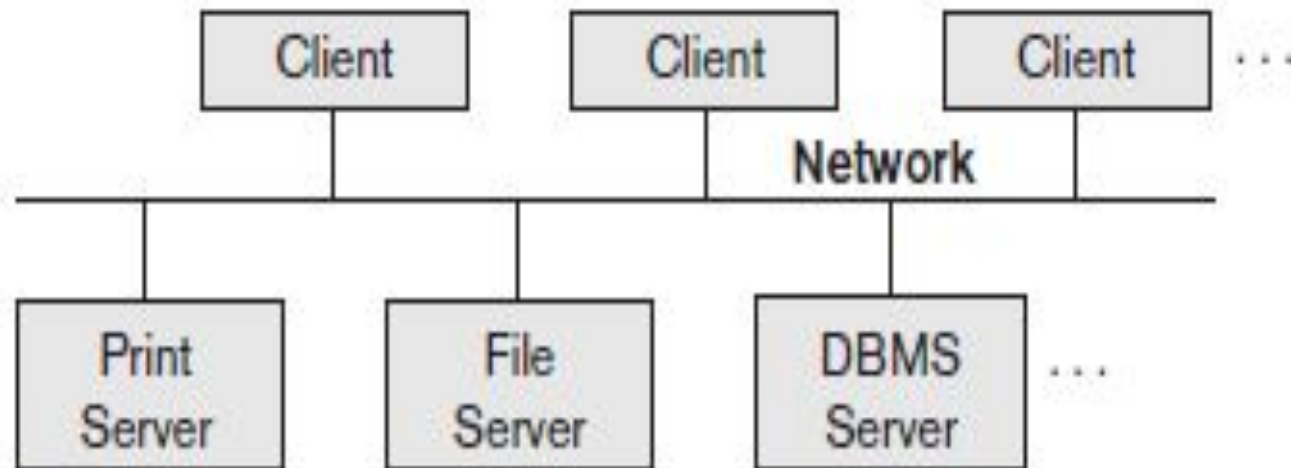
- The user interface and application program runs on the client side
- When DBMS access is required the program establishes a connection to the DBMS (which is on the server side)
- Once the connection is created, the client program can communicate with the DBMS
- A standard called Open Database Connectivity (ODBC) provides an application programming interface (API), which allows client-side programs to call the DBMS

Introduction to Databases

Database Architectures

Client – Server architecture

Two Tier Client – Server Architecture



Introduction to Databases

Database Architectures

Client – Server architecture

Two Tier Client – Server Architecture

- The architecture is called two-tier architectures because the software components are distributed over two systems: client and server
- The advantages of this architecture are its simplicity and seamless compatibility with existing systems

Three Tier Client – Server Architecture

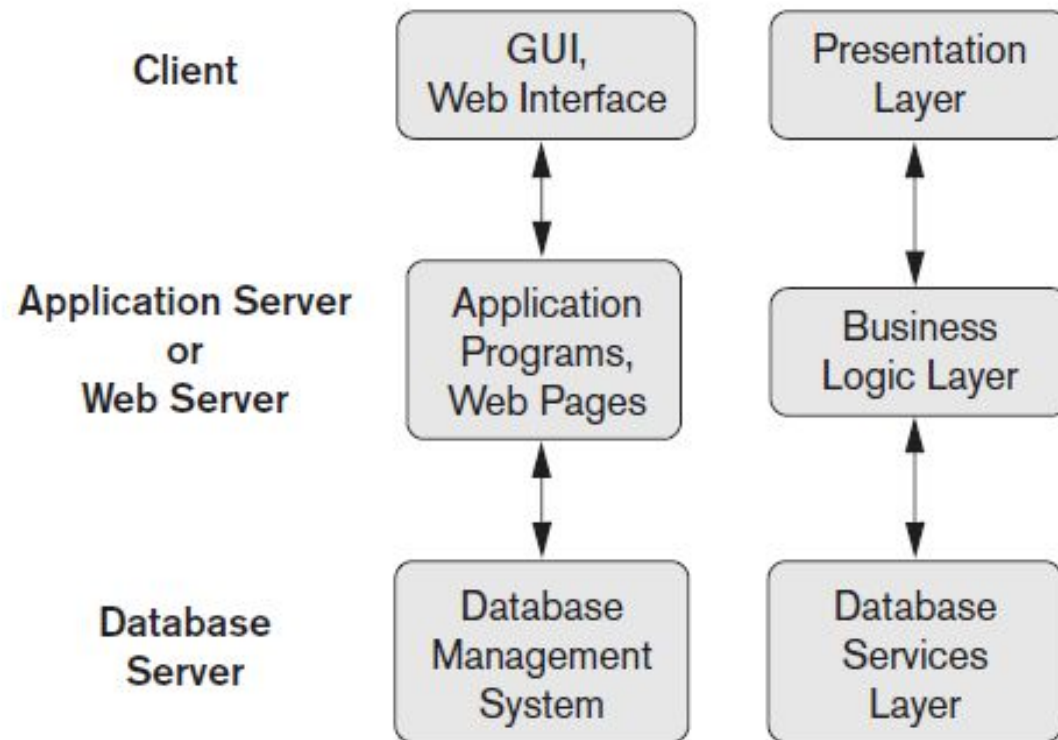
- Many Web applications use an architecture called the three-tier architecture, which adds an intermediate layer between the client and the database server

Introduction to Databases

Database Architectures

Client – Server architecture

Three Tier Client – Server Architecture



Introduction to Databases

Database Architectures

Client – Server architecture

Three Tier Client – Server Architecture

- The intermediate layer or middle tier is called the application server or the Web server, depending on the application
- This server plays an intermediary role by running application programs and storing business rules (procedures or constraints) that are used to access data from the database server
- It can also improve database security by checking a client's credentials before forwarding a request to the database server
- Clients contain user interfaces and Web browsers

Introduction to Databases

Database Architectures

Client – Server architecture

Three Tier Client – Server Architecture

- The intermediate server accepts requests from the client, processes the request and sends database queries and commands to the database server, and then acts as a conduit for passing (partially) processed data from the database server to the clients, where it may be processed further and filtered to be presented to the users
- Thus, the user interface, application rules, and data access act as the three tiers
- In another way, the presentation layer displays information to the user and allows data entry
- The business logic layer handles intermediate rules and constraints before data is passed up to the user or down to the DBMS
- The bottom layer includes all data management services

Introduction to Databases

DBMS Classification

- The major criteria used to classify DBMS are
 - Data model on which the DBMS is based
 - Number of users supported by the system
 - Number of sites over which the database is distributed
 - Cost
 - Purpose

Data model on which the DBMS is based

Based on the data model we have:

- **Relational DBMS (RDBMS):**
- Relational data model represents a database as a collection of tables, where each table can be stored as a separate file
- Most relational databases use the high-level query language called SQL

Introduction to Databases

DBMS Classification

Data model on which the DBMS is based

- **Object DBMS (ODBMS):**
 - Object data model defines database in terms of objects, their properties, and their operations
 - Objects with the same structure and behavior belong to a class, and classes are organized into hierarchies
 - The operations of each class are specified in terms of predefined procedures called methods
- **Big Data DBMS:**
 - Big data systems are based on various data models, with the following four data models most common

Introduction to Databases

DBMS Classification

Data model on which the DBMS is based

- **Big Data DBMS:**
- The *key-value data model* associates a unique key with each value (which can be a record or object) and provides very fast access to a value given its key
- The *document data model* is based on JSON (Java Script Object Notation) and stores the data as documents, which somewhat resemble complex objects
- The *graph data model* stores objects as graph nodes and relationships among objects as directed graph edges
- The *column-based data models* store the columns of rows clustered on disk pages for fast access and allow multiple versions of the data

Introduction to Databases

DBMS Classification

Data model on which the DBMS is based

- **XML model based DBMS**
- This model has emerged as a standard for exchanging data over the Web and has been used as a basis for implementing several prototype native XML systems
- XML uses hierarchical tree structures
- It combines database concepts with concepts from document representation models
- Data is represented as elements; with the use of tags, data can be nested to create complex tree structures
- This model conceptually resembles the object model but uses different terminology

Introduction to Databases

DBMS Classification

Data model on which the DBMS is based

- **Network model based DBMS**

- The network model represents data as record types and also represents a limited type of 1:N relationship, called a set type
- A 1:N, or one-to-many, relationship relates one instance of a record to many record instances using some pointer linking mechanism in these models

- **Hierarchical model based DBMS**

- The hierarchical model represents data as hierarchical tree structures. Each hierarchy represents a number of related records
- There is no standard language for the hierarchical model

Introduction to Databases

DBMS Classification

Number of users supported by the system

- **Single-user systems**
 - Support only one user at a time and are mostly used with PCs
- **Multiuser systems**
 - Include the majority of DBMSs, support concurrent multiple users

Number of sites over which the database is distributed

- **Centralized DBMS**
 - If the data is stored at a single computer site
 - A centralized DBMS can support multiple users, but the DBMS and the database reside totally at a single computer site
- **Distributed DBMS (DDBMS)**
 - Can have the actual database and DBMS software distributed over many sites connected by a computer network

Introduction to Databases

DBMS Classification

Cost

- **Paid and free DBMS**

Purpose

- **General Purpose DBMS**
 - When performance is not a prime concern general purpose DBMS is used
- **Special Purpose DBMS**
 - When performance is a primary consideration, a special-purpose DBMS can be designed and built for a specific application
 - Such a system cannot be used for other applications without major changes

Entity Relationship (ER) Model

Database Design Process

- Requirement collection and analysis – Database designers collect all requirements from database users
- Conceptual Design – Designing conceptual schema using high level conceptual model like E R Model
- Logical Design – Using SQL conceptual schema is converted in to actual implementation data model
- Physical Design – Internal storage structures, file organizations, indexes for database files are specified
- The above steps are phases in database design much like phases in software engineering
- E R model comes in the second step – Conceptual Design

Entity Relationship (ER) Model

Entity

- Thing or object in the real world with an independent existence
- An entity may be an object with a physical existence eg: person, car, house, or employee
- An entity may be an object with a conceptual existence eg: a company, a job, or a university course

Attribute

- Each entity has attributes—the particular properties that describe it
- For example, an EMPLOYEE entity may be described by the employee's name, age, address, salary, and job
- An entity will have a value for each of its attributes
- The attribute values that describe each entity become a major part of the data stored in the database

Entity Relationship (ER) Model

Example

- The EMPLOYEE entity e1 has four attributes: Name, Address, Age, and Home_phone
- Their values are 'John Smith,' '2311 Kirby, Houston, Texas 77001', '55', and '713-749-2630', respectively.
- The COMPANY entity c1 has three attributes: Name, Headquarters, and President
- Their values are 'Sunco Oil', 'Houston', and 'John Smith', respectively
- Several types of attributes occur in the ER model:
 - Simple versus composite
 - Single valued versus multivalued
 - Stored versus derived

Entity Relationship (ER) Model

Simple and Composite Attribute

- Simple Attribute - Attributes that are not divisible are called simple or atomic attributes
- Composite attributes - Attributes that can be divided into smaller subparts, which represent more basic attributes with independent meanings.
- For example, the Address attribute of the EMPLOYEE entity shown can be subdivided into Street_address, City, State, and Zip, with the values '2311 Kirby', 'Houston', 'Texas', and '77001'
- Composite attributes can form a hierarchy; for example, Street_address can be further subdivided into three simple component attributes: Number, Street, and Apartment_number

Entity Relationship (ER) Model

Single and Multivalued Attribute

- Single-valued Attribute: Attributes that have a single value for a particular entity
- For example, Age is a single-valued attribute of a person
- Multi-valued Attribute: An attribute can have a set of values for the same entity—for instance, a Colors attribute for a car, or a College_degrees attribute for a person
- Cars with one color have a single value, whereas two-tone cars have two color values
- Similarly, one person may not have any college degrees, another person may have one, and a third person may have two or more degrees

Entity Relationship (ER) Model

Stored and Derived Attribute

- Derived Attributes: In some cases, two (or more) attribute values are related—for example, the Age and Birth_date attributes of a person
- For a particular person entity, the value of Age can be determined from the current (today's) date and the value of that person's Birth_date
- The Age attribute is hence called a derived attribute and is said to be derivable from the Birth_date attribute, and is represented in a dashed oval in ER Diagram
- Stored Attribute: the Birth_date attribute is called a stored attribute
- Some attribute values can be derived from related entities
- For example, an attribute Number_of_employees of a DEPARTMENT entity can be derived by counting the number of employees related to (working for) that department

Entity Relationship (ER) Model

Null Attribute

- In some cases, a particular entity may not have an applicable value for an attribute
- For example, the Apartment_number attribute of an address applies only to addresses that are in apartment buildings and not to other types of residences, such as single-family homes
- Similarly, a College_degrees attribute applies only to people with college degrees
- For such situations, a special value called NULL is created
- An address of a single-family home would have NULL for its Apartment_number attribute, and a person with no college degree would have NULL for College_degrees
- NULL can also be used if we do not know the value of an attribute for a particular entity

Entity Relationship (ER) Model

Composite Attribute

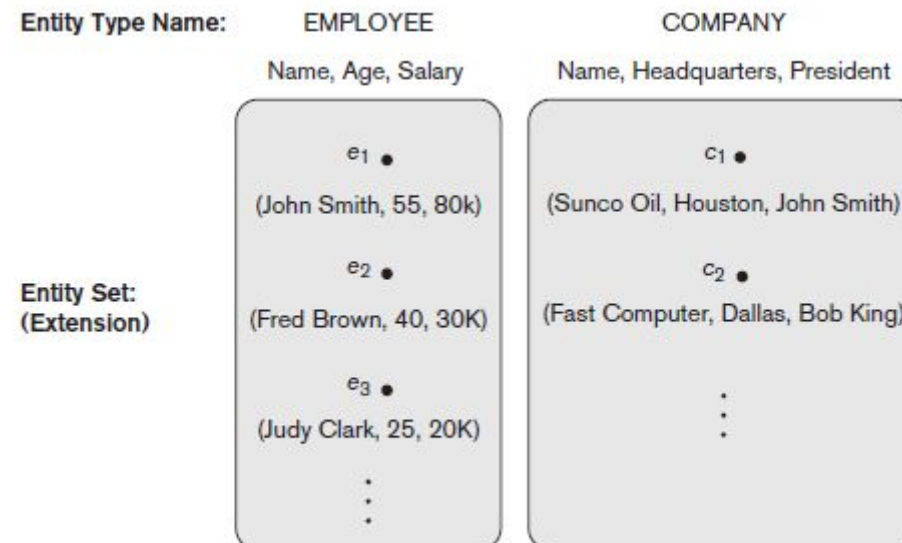
- In general, composite and multivalued attributes can be nested arbitrarily
- We can represent arbitrary nesting by grouping components of a composite attribute between parentheses () and separating the components with commas, and by displaying multivalued attributes between braces { }
- Such attributes are called complex attributes
- For example, if a person can have more than one residence and each residence can have a single address and multiple phones, an attribute Address_phone for a person can be specified as

```
{Address_phone({Phone(Area_code,Phone_number)},Address(Street_address(Number,Street,Apartment_number),City,State,Zip) )}
```

Entity Relationship (ER) Model

Entity types and Entity sets

- A database usually contains groups of entities that are similar and collection of entities that have same attributes even though they have different values are called entity types
- collection of all entities of a particular entity type in the database at any point in time is called entity set



Entity Relationship (ER) Model

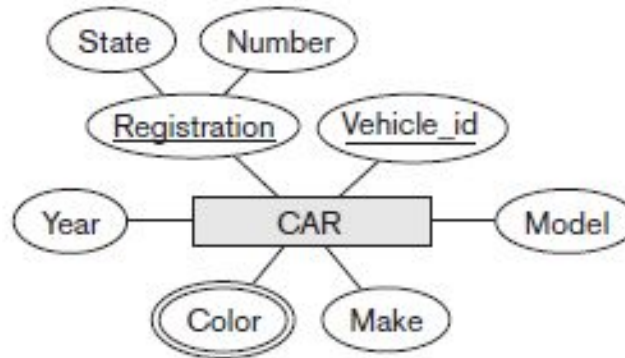
Entity types and Entity sets

- An entity type is represented in ER diagram as a rectangular box enclosing the entity type name
- Attribute names are enclosed in ovals and are attached to their entity type by straight lines
- Composite attributes are attached to their component attributes by straight lines
- Multivalued attributes are displayed in double ovals
- An entity type usually has one or more attributes whose values are distinct for each individual entity in the entity set
- Such an attribute is called a **key attribute**, and its values can be used to identify each entity uniquely

Entity Relationship (ER) Model

Entity types and Entity sets

- For example, the Name attribute is a key of the COMPANY entity type in the figure above because no two companies are allowed to have the same name
- For the PERSON entity type, a typical key attribute is Ssn (Social Security number)
- In ER diagrammatic notation, each key attribute has its name underlined inside the oval



- An entity type may also have no key, in which case it is called a **weak entity type** otherwise called **strong entity type**

Entity Relationship (ER) Model

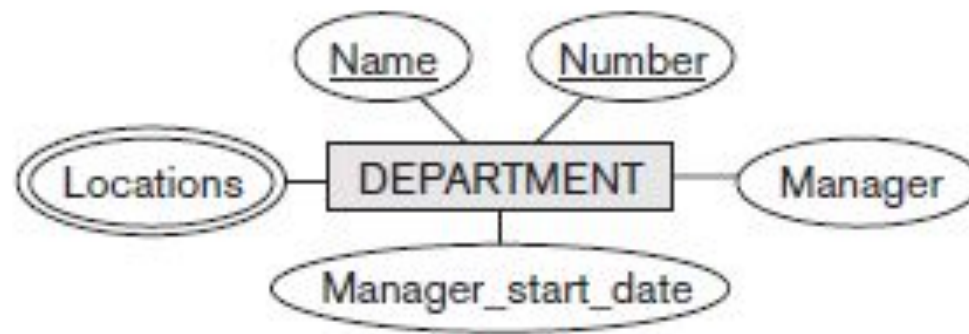
Entity types and Entity sets

- Each simple attribute of an entity type is associated with a **value set** (or domain of values), which specifies the set of values that may be assigned to that attribute for each individual entity
- For example if the range of ages allowed for employees is between 16 and 70, we can specify the value set of the Age attribute of EMPLOYEE to be the set of integer numbers between 16 and 70
- Value set for the Name attribute to be the set of strings of alphabetic characters separated by blank characters, and so on
- Value sets are not typically displayed in basic ER diagrams and are similar to the basic data types available in most programming languages, such as integer, string, Boolean, float, enumerated type, subrange, and so on

Entity Relationship (ER) Model

E R Diagram Examples

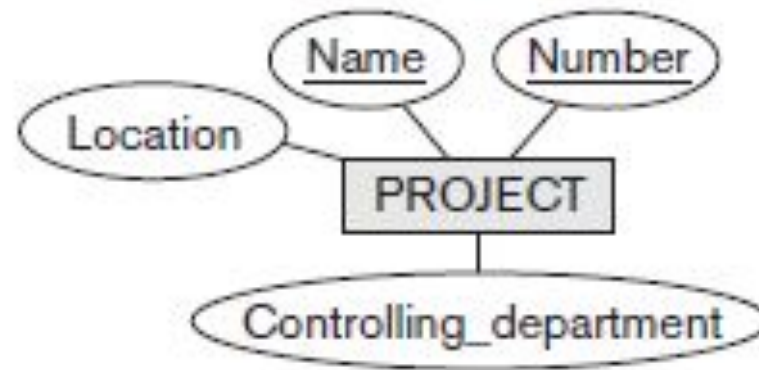
- An entity type DEPARTMENT with attributes Name, Number, Locations, Manager, and Manager_start_date
- Locations is the only multivalued attribute
- We can specify that both Name and Number are (separate) key attributes because each was specified to be unique.



Entity Relationship (ER) Model

E R Diagram Examples

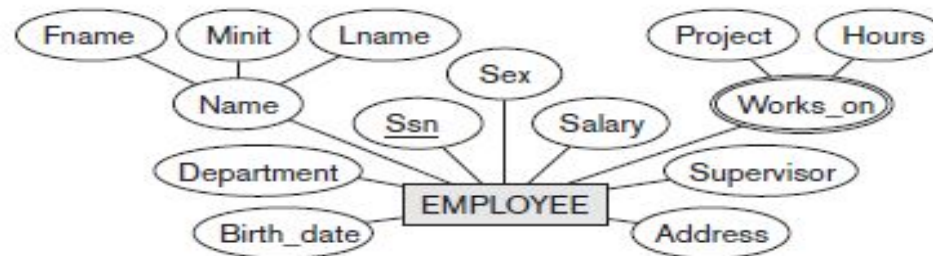
- An entity type PROJECT with attributes Name, Number, Location, and Controlling_department
- Both Name and Number are (separate) key attributes



Entity Relationship (ER) Model

E R Diagram Examples

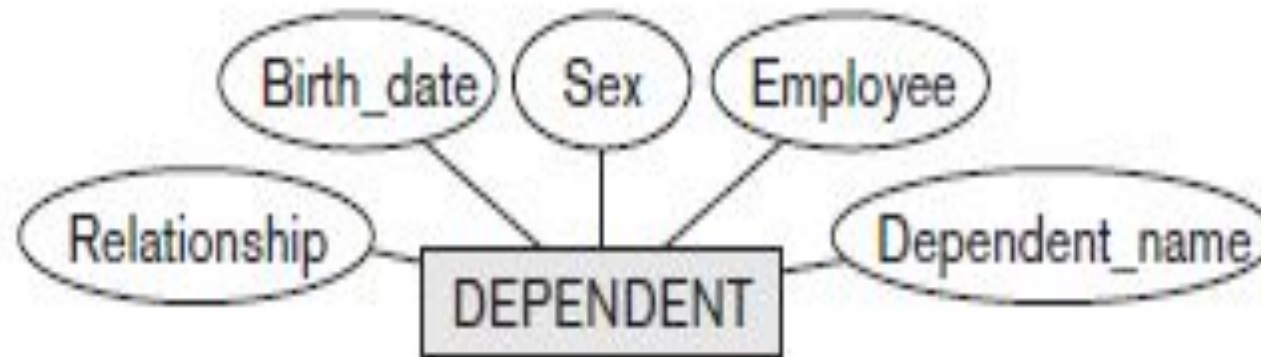
- An entity type EMPLOYEE with attributes Name, Ssn, Sex, Address, Salary, Birth_date, Department, and Supervisor
- Both Name and Address may be composite attributes; however, this was not specified in the requirements
- We must go back to the users to see if any of them will refer to the individual components of Name—First_name, Middle_initial, Last_name—or of Address
- In our example, Name is modeled as a composite attribute, whereas Address is not, presumably after consultation with the users



Entity Relationship (ER) Model

E R Diagram Examples

- An entity type DEPENDENT with attributes Employee, Dependent_name, Sex, Birth_date, and Relationship (to the employee)



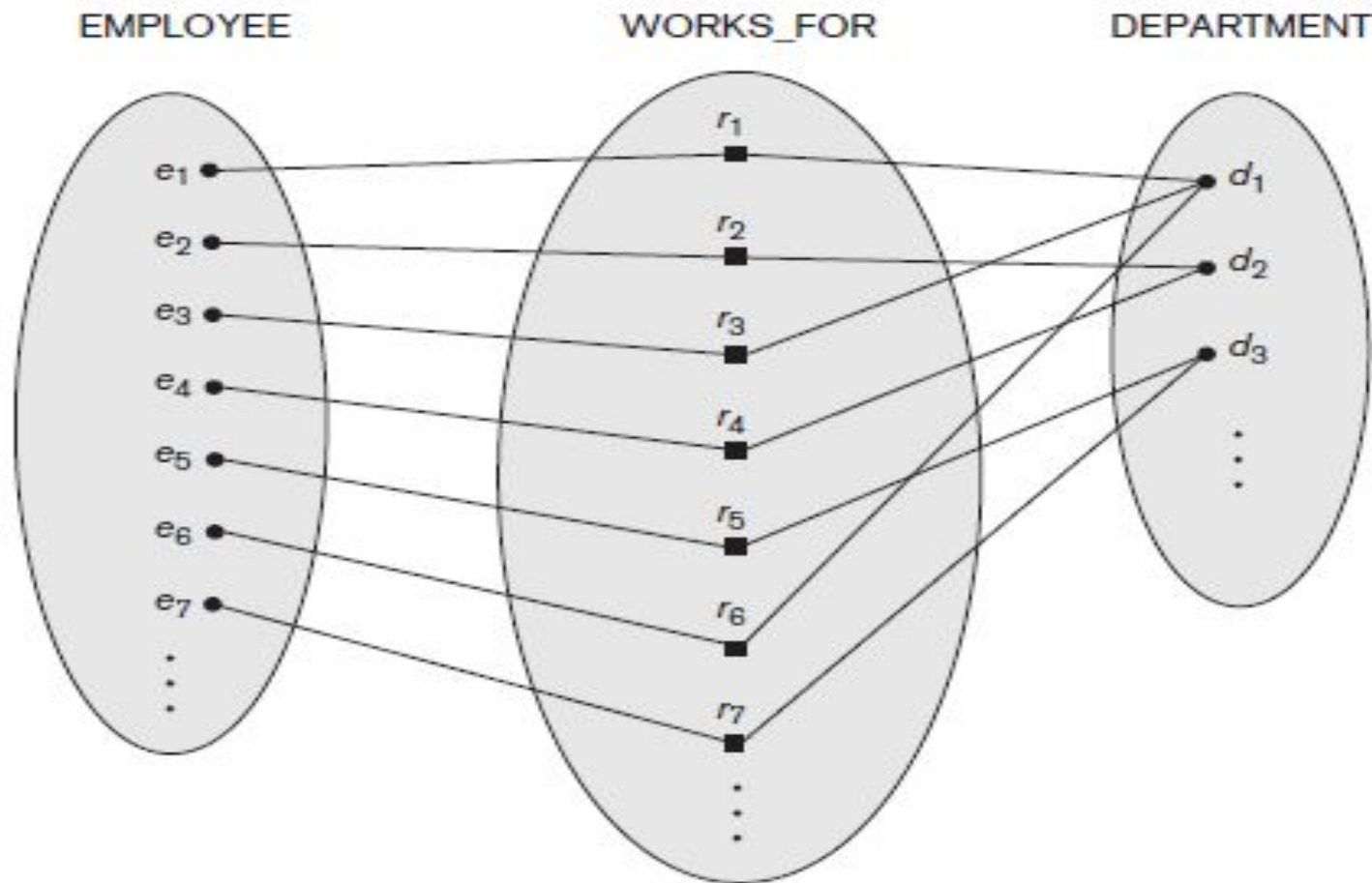
Entity Relationship (ER) Model

Relationship

- A **relationship** relates two or more distinct entities with a specific meaning
- For example, EMPLOYEE John Smith works on the ProductX PROJECT
- EMPLOYEE Franklin Wong manages the Research DEPARTMENT
- Relationship of the same type are grouped into a **relationship type**
- A **relationship type** R among n entity types E_1, E_2, \dots, E_n defines a set of associations or a relationship set among entities from these entity types
- More than one relationship type can exist with the same participating entity types
- For example, MANAGES and WORKS_FOR are distinct relationships between EMPLOYEE and DEPARTMENT

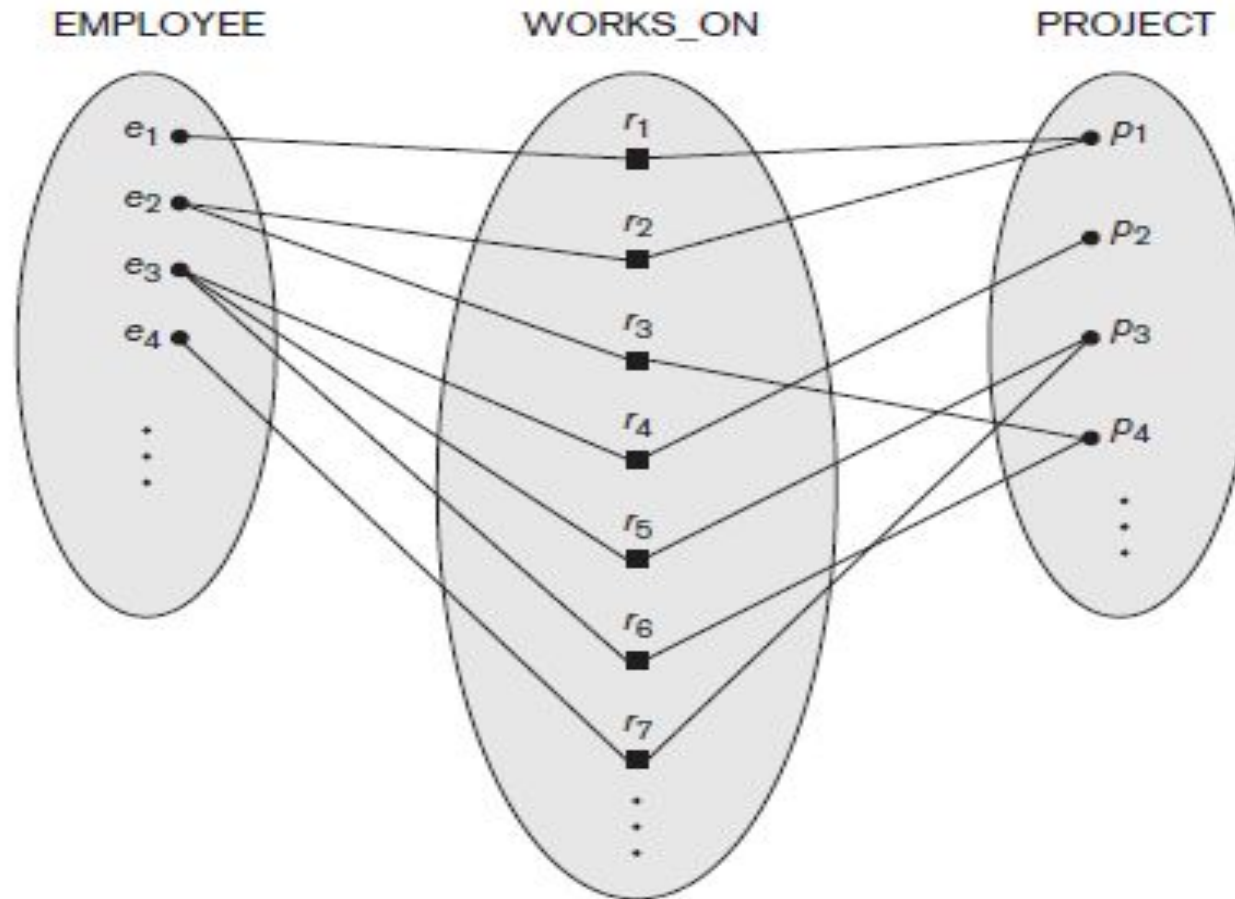
Entity Relationship (ER) Model

Relationship



Entity Relationship (ER) Model

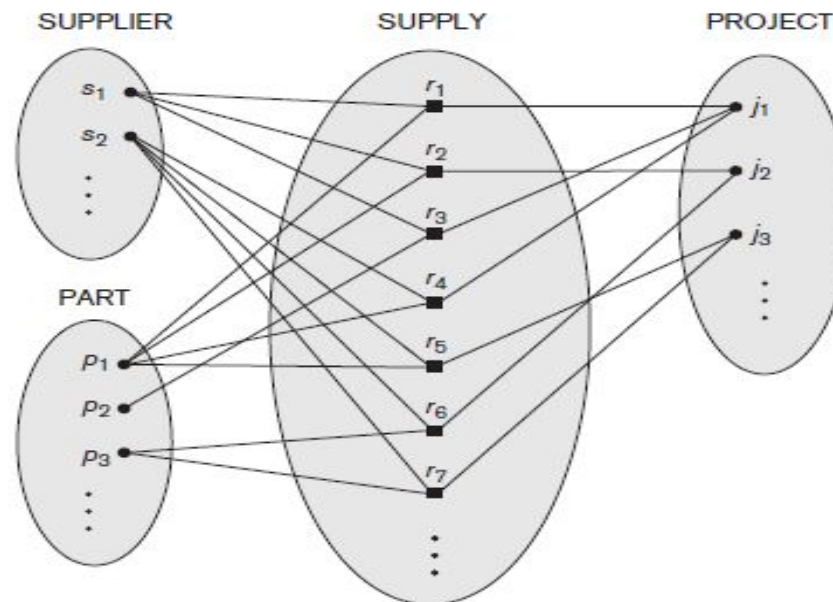
Relationship



Entity Relationship (ER) Model

Relationship

- **Degree** of a relationship type is the number of participating entity types
- For example, WORKS_FOR relationship is of degree two
- Relationship type of degree two is called **binary relationship**
- Relationship type of degree three is called **ternary relationship**



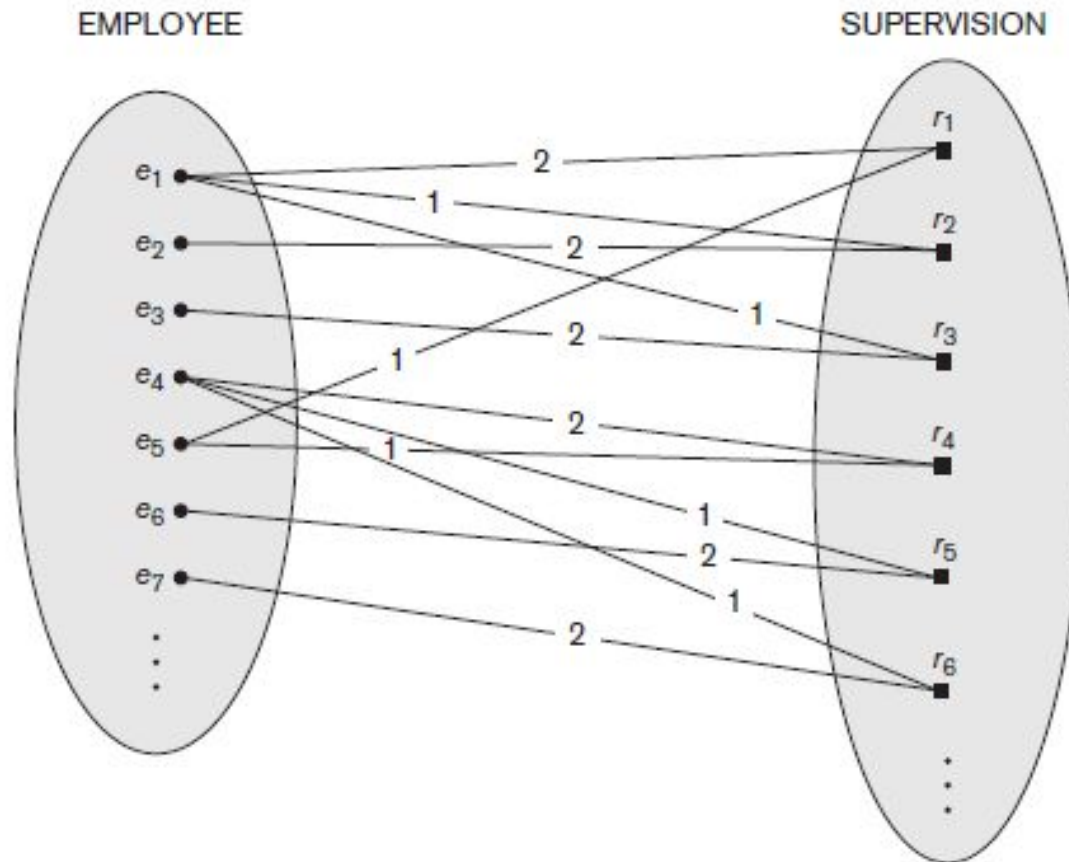
Entity Relationship (ER) Model

Relationship

- **Role names** signifies that role that a participating entity from the entity type plays in each relationship instance
- It helps to explain what the relationship means
- For example, WORKS_FOR relationship type, EMPLOYEE plays the role of employee or worker and DEPARTMENT plays the role of department or employer
- In **recursive relationship**, both participations are same entity type in different roles
- For examples, SUPERVISION relationships between EMPLOYEE (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker)

Entity Relationship (ER) Model

Relationship



1 – Supervisor
2 – Subordinate

Entity Relationship (ER) Model

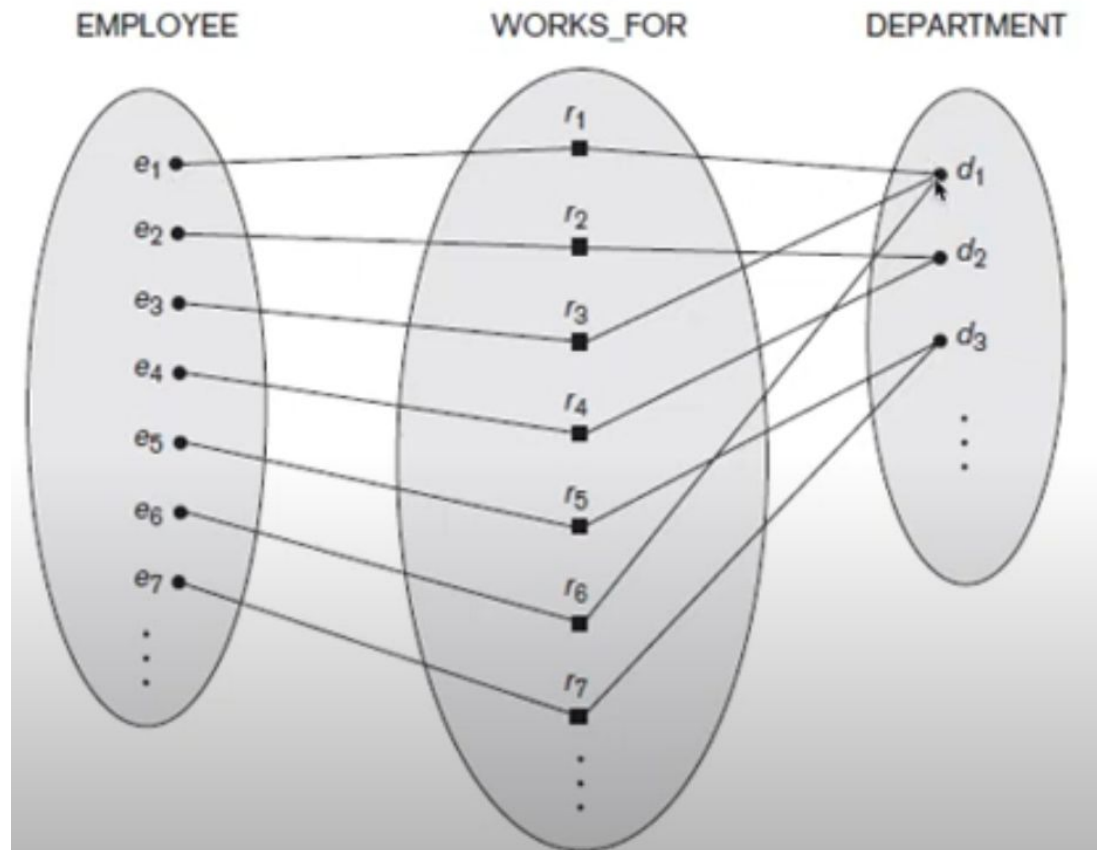
Constraints on binary relationships

- Relationship types usually have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set
- There are two major constraints:
 - Cardinality ratio
 - Participation
- They are together called structural constraints
- **Cardinality ratio**
- It specifies the maximum number of relationship instances that an entity can participate in
- For example, WORKS_FOR, DEPARTMENT: EMPLOYEE is of cardinality ratio 1:
N

Entity Relationship (ER) Model

Constraints on binary relationships

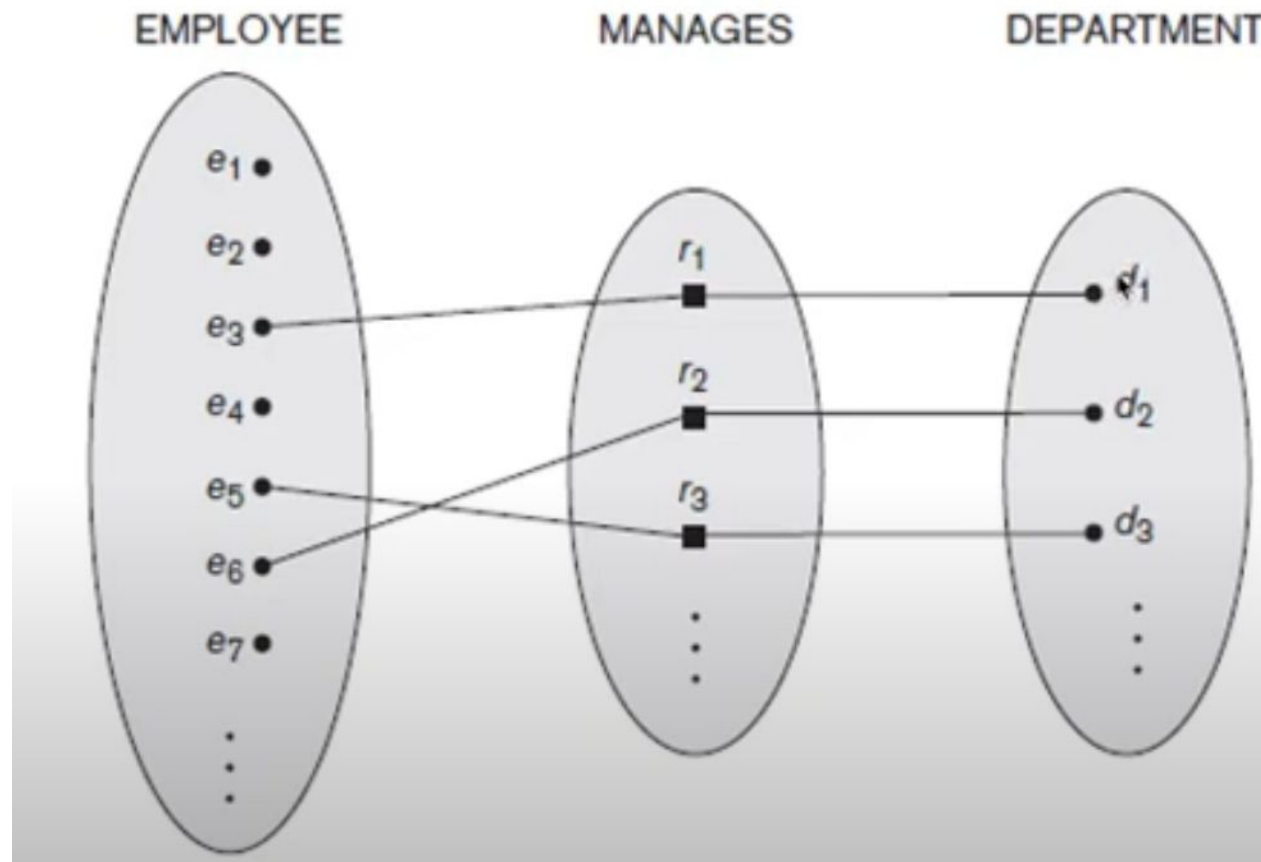
- Cardinality ratio (1 : N)



Entity Relationship (ER) Model

Constraints on binary relationships

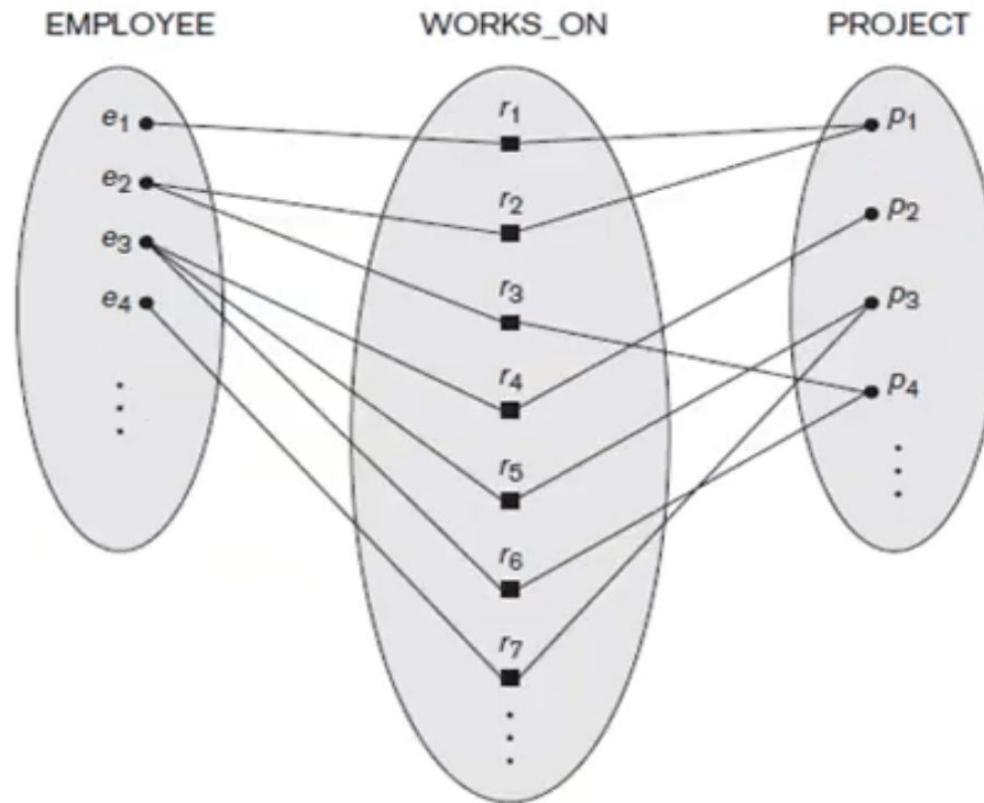
- Cardinality ratio (1 : 1)



Entity Relationship (ER) Model

Constraints on binary relationships

- Cardinality ratio (M : N)



Entity Relationship (ER) Model

Constraints on binary relationships

Participation Constraints

- Specifies whether the existence of an entity depends on its being related to another entity
- Specifies the minimum number of relationship instances that each entity can participate in
- Also called the minimum cardinality constraints
- There are two types of participation constraints – **total** and **partial**

Total participation

- It is also called existence dependency
- The WORKS_FOR relationship is of total participation that means every employee should work for any department which is mandatory

Entity Relationship (ER) Model

Constraints on binary relationships

Participation Constraints

Partial participation

- The MANAGES relationship is of partial participation that means some employees can manage a department
- **Attributes of Relationship types**
- A relationship type can have attributes
- For example, HoursPerWeek of WORKS_ON, means number of hours per week that an EMPLOYEE works on a PROJECT

Entity Relationship (ER) Model

Weak Entity Types

- Entity that does not have a key attribute is called a weak entity type
- Weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
 - A partial key of the weak entity type
 - Particular entity they are related to in the identifying entity type
- For example, a DEPENDENT entity is identified by the dependent's *firstname* and *birthdate* and the specific EMPLOYEE that the dependent is related to

Entity Relationship (ER) Model

Relationship

- A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities
- There are four types of relationships:
 - One to One
 - One to Many
 - Many to One
 - Many to Many
- **One to One Relationship**
 - When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship
 - For example, a person has only one passport and a passport is given to one person

Entity Relationship (ER) Model

Relationship

- One to One Relationship



- One to Many Relationship

- When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship
- For example – a customer can place many orders but a order cannot be placed by many customers

Entity Relationship (ER) Model

Relationship

- One to Many Relationship



- Many to One Relationship
- When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship
- For example – many students can study in a single college but a student cannot study in many colleges at the same time

Entity Relationship (ER) Model

Relationship

- Many to One Relationship



- Many to Many Relationship
- When more than one instances of an entity is associated with more than one instances of another entity then it is called many to many relationship
- For example, a can be assigned to many projects and a project can be assigned to many students

Entity Relationship (ER) Model


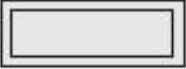




Relationship

- Many to Many Relationship



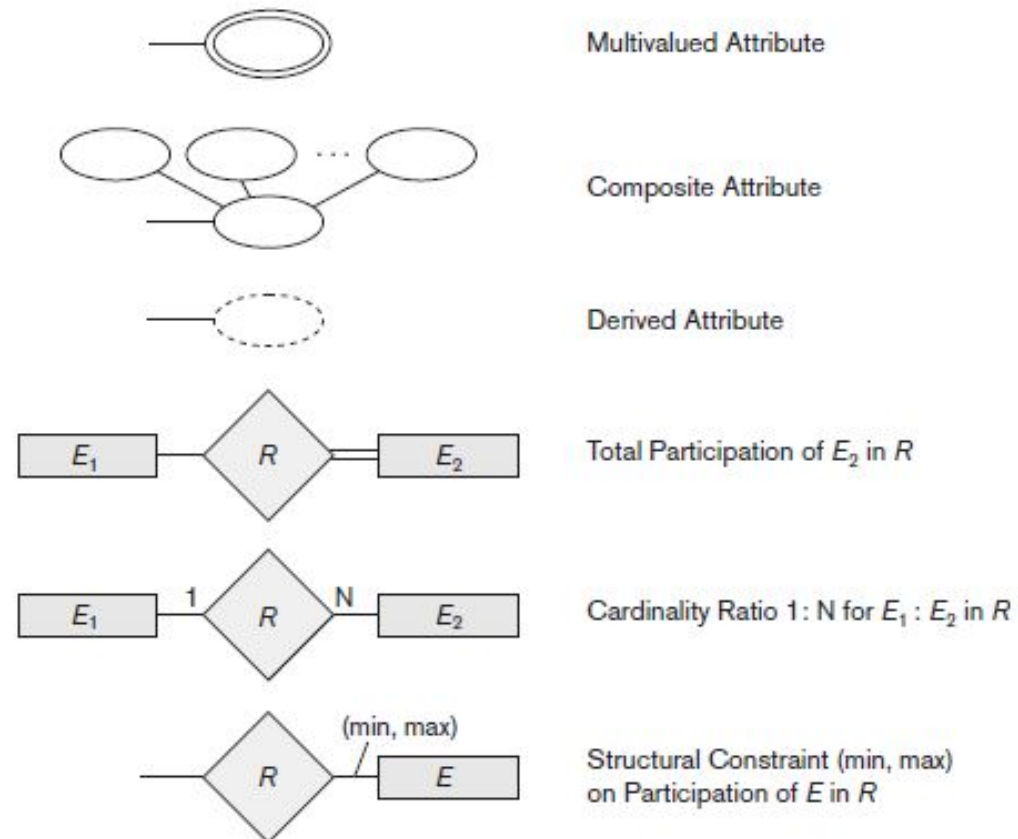
Entity Relationship (ER) Model

Notations

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute

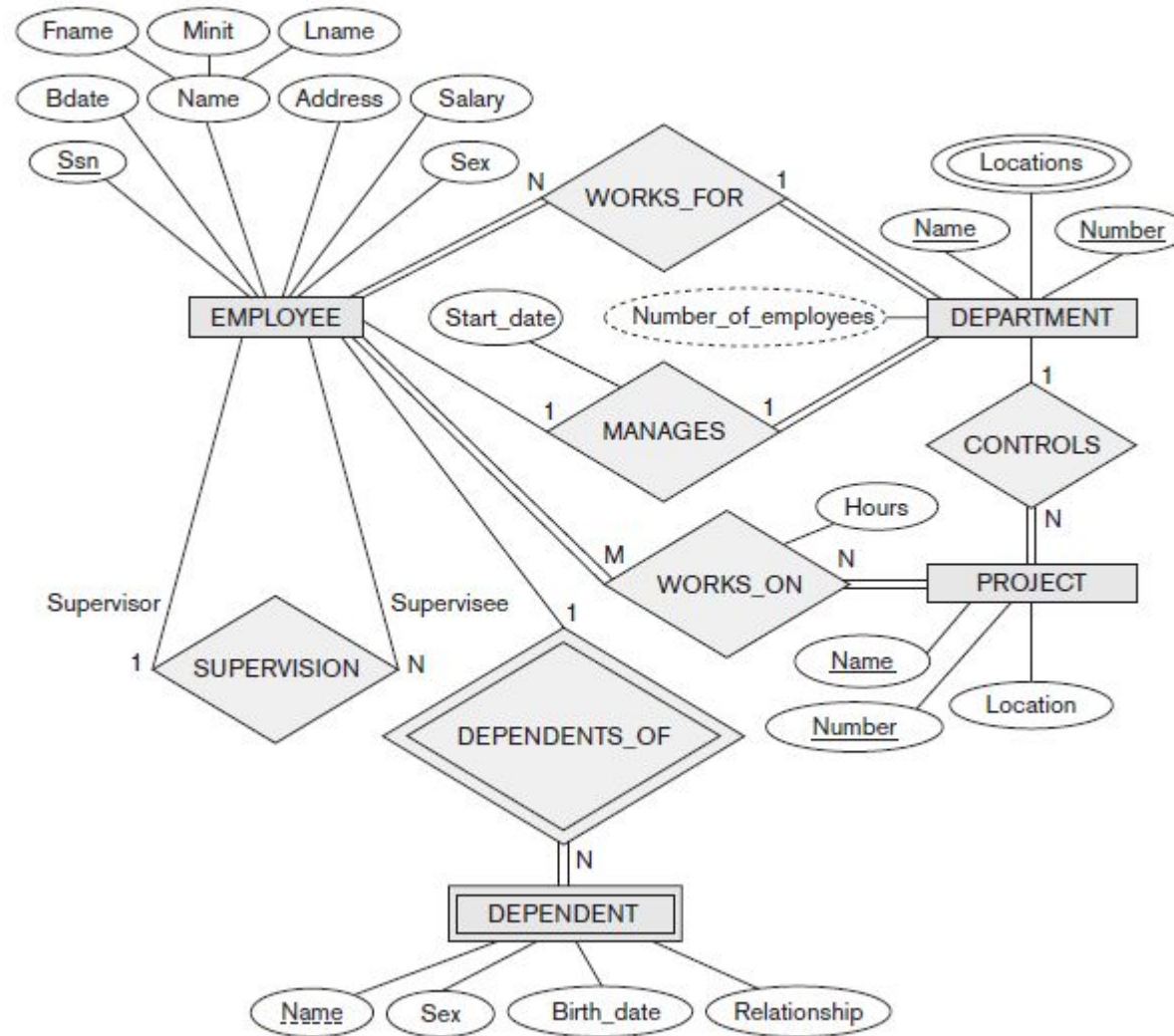
Entity Relationship (ER) Model

Notations



Entity Relationship (ER) Model

E R Diagram



Entity Relationship (ER) Model

E R Diagram

Min max notation

- Specified on each participation of an entity type E in a relationship type R
- Specifies that each entity e in E participates in at least min and at most max relationship instances of R
- For example, a department has exactly one manager and an employee can manage at most one department
- Specify (0,1) for participation of EMPLOYEE in MANAGES
- Specify (1,1) for participation of DEPARTMENT in MANAGES

Entity Relationship (ER) Model

E R Diagram

Min max notation

