Unit 1

classmate
Date
Page

Introduction to Database and Conceptual Design using ERD

Introduction to DBMS

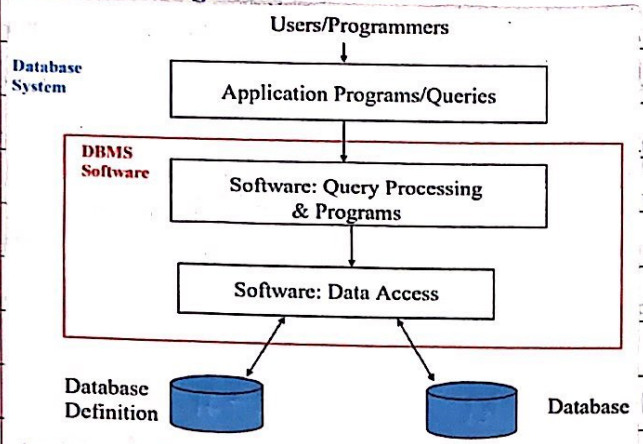
Data is known facts that can be recorded. A key resource.
Database is a collection of data that represents some aspect of the real world, logically coherent collection [not random], designed, built and populated for a specific purpose.
Database Management System is software that manages data.
Databases help us store data [file structures, disk management], understand data [data models], secure data [security and recovery], find and utilise data [Query language, Data Analytic Tools]
DBMSes are a key part of many systems in the industry. They are used in banking, hospitality, online catalogues, employee informatic and many others.

What do DBMS provide ? Where are they used ?

DBMS provide facilities to:
(i) Define data - specify data types, structures and constraints for the data to be stored in the database
(ii) Store data
(iii) Manipulate data - pose queries to retrieve specific data, update data or generate reports based on the data.
(iv) Additional features such as
1) Protection or Security measures to prevent unauthorised acces
2) Active processing to take internal actions on data
3) Presentation and Visualisation of data
4) Maintenance of database and associated programs over the lifetime of database application. [software, system maintenance]

The rising need for a proper system to keep track of the large amount of data that was being created on a daily basis by social media, web-pages, search engines etc. The emergence of Big Data storage systems involving large clusters of distributed computers, causes rise in demand for DBMS. A large amount of data now resides on the cloud which means it is in huge data centers using thousand of machines. DBMS are used in buisnesses, social networking apps, education, medicine, etc.

Database System Environment



Characteristics of Database Approach

1) Self-describing nature of database system

A DBMS catalog stores the description of a particular database. This description is called meta data. This allows the DBMS software to work with different database applications. Some newer systems such as NOSQL systems need no meta-data; they store data definition within it's structure, making it self describing.

2) Independence between programs and data

This is called program-data independence. It allows for changing data structures and data organization without having to change the DBMS access programs.

3) Data Abstraction

A data model is used to hide storage details and present the users with a conceptual view of database. Programs refer to the data model constructs rather then data storage details. This allows a person who doesn't know how DBMS work exactly, still be able to access database.

4) Support multiple views of data

Each user may see a different view of the database, which describes only the data of interest of that user.

5) Sharing of data and multi-user transaction processing

Allowing a set of concurrent users to retrieve from and to update the database. Concurrency control within DBMS guarantees each transaction is correctly executed or aborted. Recovery subsystem ensures that each completed transaction has it's effect permanently recorded in database

OLTP [Online Transaction Processing] is major part of database applications

Database Users

Users may be divided into two distinct groups!

(i) Those who actually use and control the database content [Actors on the Scene]

(ii) Those who design and develop the DBMS software and related tools and the computer system operators [Workers behind the Scene]

1) Actors on the Scene

a) Database Adminstrators

Responsible for authorizing access to database, coordinating and monitoring use of database, acquring software and hardware resources, controlling use of database and monitoring efficiency of operations. These are usually people who maintain the database or own the database.

b) Database Designers

Responsible to define the content, structure, constraints, functions and transactions against the database. They communicate with the end-users and on the basis of their needs, define the above features of the database.

c) End users

They use data for queries, reports and some of them update database conten

- Casual: access database occasionally/when needed [Managers, Occasional browsers]
- Naive or Parametric: use previously well defined functions in the form of canned transactions against the database. Users of mobile apps fall in this category. [Bank-tellers, Reservation clerks, Social Media Users]
- Sophisticated: These include people who are thoroughly familiar with the capabilities of the system. Many use tools in the form of software packages that work closely with stored database. [Buisness Analysts, Scientists, Engineers]
- Stand-alone: Mostly maintain personal databases using ready-to-use packaged applications [Personal photos and videos]

d) System Analysts and Application Developers

2) Workers Behind the Scene

a) System Designers and Implementors

They design and implement DBMS packages in the form of modules and interfaces and test and debug them. The DBMS must interface with applications, language compilers, OS components etc.

b) Tool Developers

Design and implement software systems called tools for modelling and designing databases, performance monitoring, prototyping, test data generation, UI creation, simulation etc. These facilitate building of applications and allow using database effectively.

c) Operators and Maintenance Personnel

Manage the actual running and maintenance of the database system hardware and software environment.

Advantages of using Database Approach

- Controlling redundancy in data storage and in development and maintenance efforts. This is implemented by sharing of data among multiple users
- Restricting unauthorized access to data. This is implemented by database administrators, who are the only people who can access priveleged commands and facilities
- Providing persistent storage for program objects [Object-Oriented DBMS used]
- Providing storage structures for efficient query processing.
- Providing optimization of queries for efficient processing.
- Providing backup and recovery services, in case of loss of data.
- Providing multiple interfaces to different class of users. This allows each user to view data in the way they understand.
- Representing complex relationships among data, as well as defining new relationships as they are formed.
- Drawing inferences and actions from the stored data using deductive and active rules and triggers
- Enforcing integrity constraints on database.

(Disadvantages) Additional Implications of Using Database Approach

- Standards need to be fixed. This is crucial for the success of database applications in large organizations. Standards refer to data item names, display formats, meta data, report structures.
- Reduced application development time. Incremental time to add each new application is reduced.

- Maintain flexibility to change data structures. The database structure may evolve as new requirements are defined.
- Availability of current information. This info must be available immediately to ensure that there is concurrency in the data stored
- Economies of Scale. We need to ensure wasteful overlap of resources and personnel should be avoided by consolidating data and application across departments.

Historical Development of Database Technology

Extending Database Capabilities

When not to use a DBMS?

When there is high initial investment and a lot of overheads. If the database and applications are simple and won't change. When multiple users won't access. In real-time applications, where deadlines need to be met and overhead not needed. When data is complex, not supported.

Data Models

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. Constructs are used to define, the database structure. Constructs include elements as well as groups of elements and relationships among such groups. Constructs are used to make or form database by combining or arranging elements of the database. Constraints specify restrictions on valid data that must always be enforced. Operations on databases are used for specifying database retrievals and updates by referring to constructs of data model. These could be basic model operations [CRUD] or user-defined operations

Categories of Data Model

a) Conceptual [High Level] data model

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

Also called entity-based or object-based model

b) Physical [low level] data model

Provide concepts that describe details of how data is stored in the computer. Specified in an ad-hoc manner through DBMS design and adminstration manuals. Concepts provided are meant for computer specialists.

Ad-hoc =>

c) Implementation / Representational data model

Provide concepts that fall between the above two, commonly used by many commercial DBMS implementations. Requires knowledge of how database works abstractly.

Relational data models used in many commercial systems

d) Self-describing data model

Combine the description of data with data values.

Eg. XML, Key-value stores, NOSQL systems

Database Schema vs Database State

Database schema is the description of the database. This includes descriptions of the database structure, data types and the constraints on the database. An illustrative display of database schema is schema diagram.

A component of the schema or an object within schema => Schema Construct

Database state is the actual data stored in a database at a particular moment in time, including collection of all data in the database. This is also called, data instance/occurrence/snapshot. A valid state is a state that satisfies the structure and constraints of database.

The schema changes very infrequently and state changes every time updated.

Schema = [Intension] State = [Extension]

Three Schema Architecture

This was proposed to support DBMS characteristic of program-data independence and supports multiple views of database. Not used or seen explicitly, but useful in explaining database system organization

1) Internal Schema

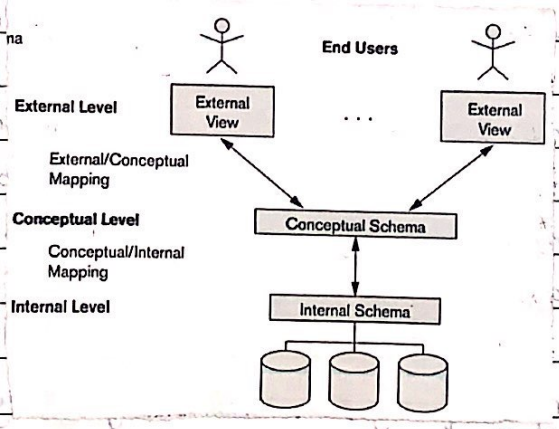
This is at internal level to describe physical storage structures and access paths. Typically uses a physical data model.

2) Conceptual Schema

This is at conceptual level to describe the structure and constraints for the whole database for a community of users.

3) External Schema

This is at the external level to describe various user views.



Data Independence

1) Logical Data Independence

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

2) Physical Data Independence

Capacity to change internal schema without having to change conceptual schema. Example is when the internal schema may be changed when certain file structures are reorganized or new indexes created to improve performance.

When a schema at a lower level is changed, only the mappings between this schema and higher level schemas need to be changed in a DBMS that supports data independence. Higher level schemas are unchanged.

DBMS Languages

1) Data Definition Language

This is used by database adminstrators and designers to specify the conceptual schema of a database. [In many, used for internal, external as well]
In a few DBMS, 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 database adminstrators and database designers

2) Data Manipulation Language

This is used to specify database retrievals and updates. DBL commands can be embedded in a general purpose programming language. 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. [Query Language]

(a) High level / Non-procedural language

"Set" oriented and specify what data to retrieve rather then how to retrieve it. Also called declarative languages. Eg. SQL

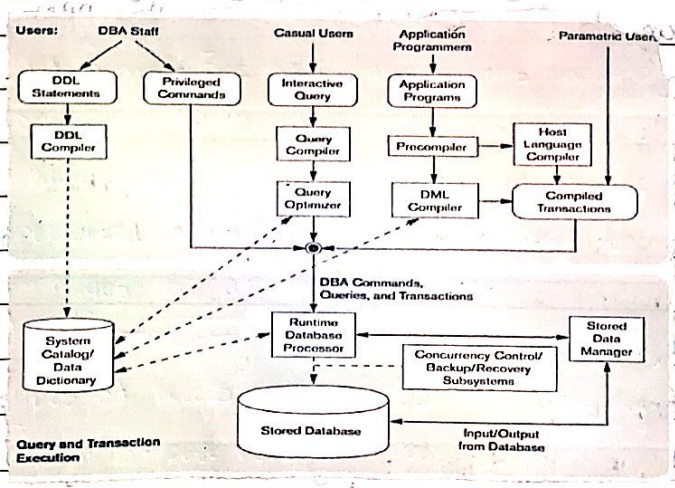
(b) Low level / Procedural language

Retrieve data one record at a time. Constructs such as looping are needed to retrieve multiple records, along with positioning pointers. This must be embedded in a programming language

DBMS Interfaces

Database System Utilities

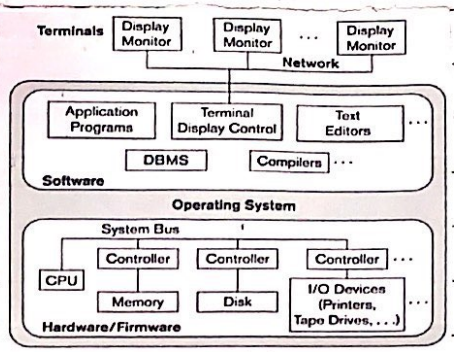
Typical DBMS Component Modules



Database System Environments

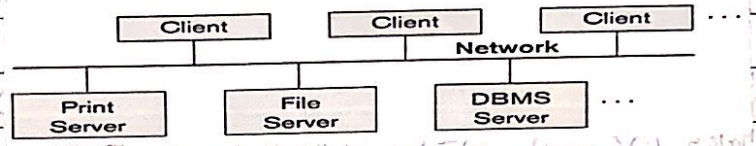
1) Centralized DBMS

This combines everything into a 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



2) Two-tier Client-Server Architecture

There are specialized server with specialized functions that client can access as needed



(i) Clients

Provide appropriate interfaces through a client software module to access and utilize the various server resources. Clients maybe PCs or Workstations with the client software installed. They are connected to servers with the help of a network.

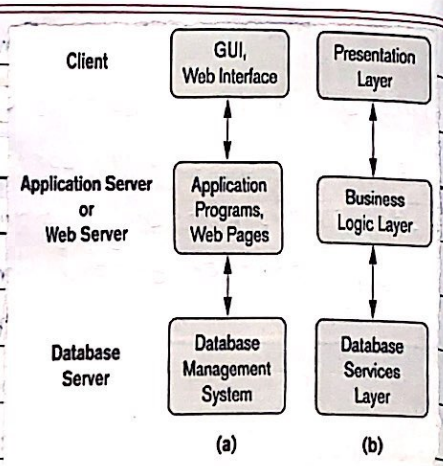
(ii) Server

These provide 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 APIs to access server databases via standard interface such as ODBC [Open Database Connectivity] or JDBC for java programming access.

Client and Server must install appropriate client module and server module. software for ODBC and JDBC. 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

History of data models - [Additional Info]

3) Three-tier client-server architecture



An intermediate layer called application or web server. Stores web connectivity software and business logic part of application used to access corresponding data from database server. Conduit for sending partially processed data btw server and client. This enhances security as the database server is only accessible via middle tier, ensuring no direct access to server. Client contains UI or Web browsers, typically a PC.

Classification of DBMS

They can be classified based on data model that is used

Done in history of Data Models
- (i) Legacy: Network, Hierarchical
- (ii) Currently used: Relational, Object-oriented, Object-relational
- (iii) Recent Technologies: Key-value storage systems, NOSQL systems [document based, column based, graph based], Native XML DBMSs

They can also be classified as single user [PC] vs multi-user

They can also be classified as centralized [Single PC, 1 database] vs distributed [Multiple Computers, Multiple DBs]

Variations of Distributed DBMSs

Homogenous DDBMS

Heterogenous DDBMS

Federated/Multidatabase: DBs are loosely coupled with high degree of autonomy

DDBS have now come to be known as client server based DBS because, they do not support totally distributed, but set of database servers for set of clients

Cost Considerations for DBMSs

- Cost Range: Free open-source systems to Configurations costing millions.
- Commercial DBMS offer additional specialized modules like time-series module, spatial data module, document module etc. These offer additional specialized functionality when purchased separately. Need to consider which cartridges are purchased/required.
- Different licensing options: max no. of users, concurrent use, single use
- Types of access path in DBMS: For efficiency of search: {inverted, full} indexing
- General purpose vs Special purpose

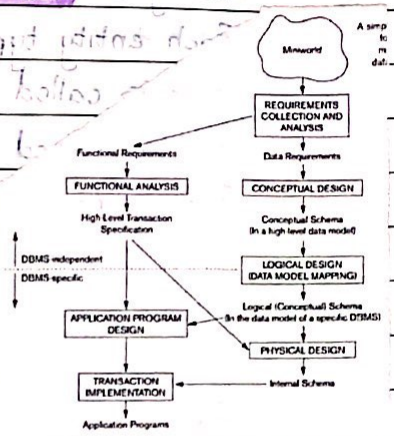
Data Modelling using the Entity-Relationship Model

In the database design process, there are two main activities, (i) database design and (ii) applications design

Generally considered part of software engineering

(i) Focus to design conceptual schema for a database application

(ii) Applications design focuses on the programs and interfaces that access the database.



Entities and Attributes

Entity is a basic concept for ER model. Entities are specific things or objects in the mini-world represented in the database.

Eg. EMPLOYEE, DEPARTMENT, PROJECT

Attributes are properties used to describe an entity. A specific entity will have a value for each of it's attributes. Each attribute has value set.

Eg. EMPLOYEE has Name, Address, DOB, Gender etc..

Types of Attributes

(a) Simple :- Single atomic value Eg: Id, Gender

(b) Composite :- Composed of several components. Composition may form a hierarchy where some components are themselves composite

Eg. Address (Apt #, House #, Street, City, State, Zipcode, Country)

(c) Multi-valued :- An entity may have multiple values for that attribut

Eg. Color of car {Color}, PreviousDegrees of student {PreviousDegrees}

*In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels. Eg. PreviousDegrees is denoted by composite, multi-valued attribute {PreviousDegrees (College, Year, Field, Degree)}

(d) Stored/Derived :- When an attribute is derived from another attribute

Eg. Age is derived from Date of birth.

Entity Types and Key Attributes

Entities with the same basic attributes are grounded or typed into entity type.

Eg. Entity type EMPLOYEE and PROJECT

An attribute of an entity type for which each entity must have a unique value is called a key attribute. A key attribute may be composite. An entity type may have more than one key. Key is underlined

Eg. SRN, Vehicle Tag Number (Number, State)

Entity Set

Each entity type will have a collection of entities stored in the database is called the entity set or sometimes entity collection. Same name is used to refer to both the entity type and entity set. Entity set is the current state of the entities of that type that are stored in the database.

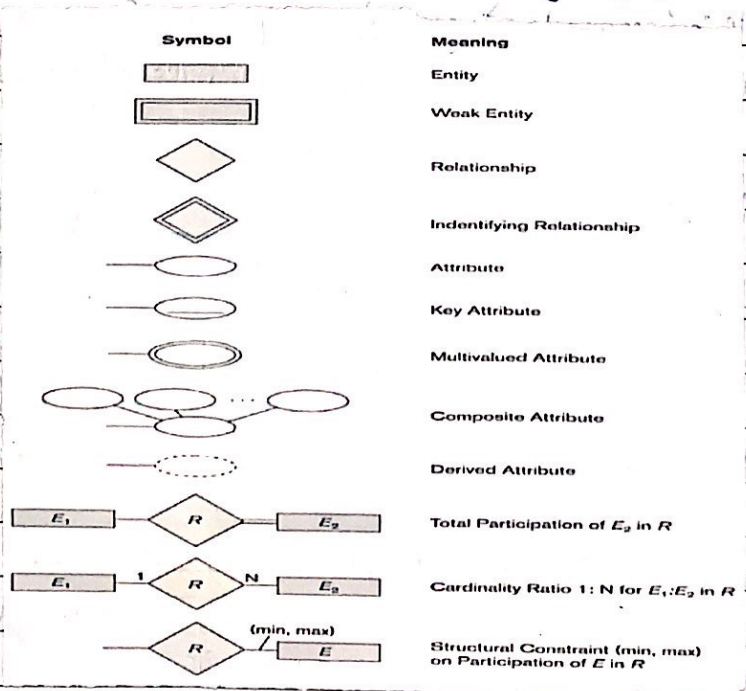
Value Sets of Attributes (Domains)

Each simple attribute is associated with a value set. A value set specifies the set of values associated with an attribute. They are similar to data types in programming language. Mathematically, an attribute A for an entity type E whose value set is V is defined as a function

$$A : E \rightarrow P(V)$$

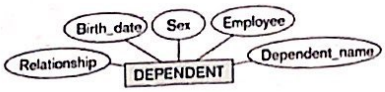
P(V) indicates a power set [all possible subsets] of V.
Attribute A for entity e as A(e)

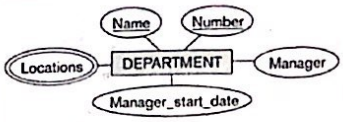
Notation for Entity Relationship Diagram

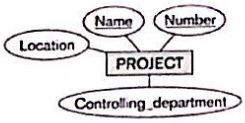


Partial key in weak entity

Example of ERD with Company Database [Only entities and attributes]



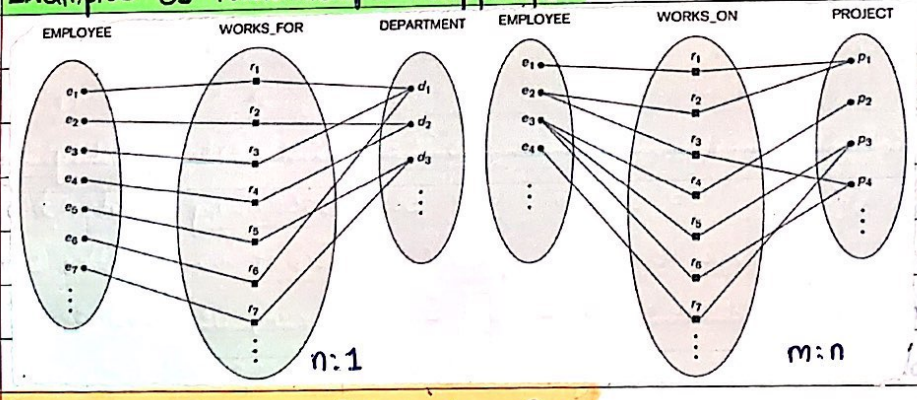




Relationships

A relationship relates two or more distinct entities with a specific meaning. Relationships of the same type are grouped or typed into a relationship type. The degree of a relationships type is number of participating entity types Relationships may have attributes as well. More then one relationship can exist between some participating entities.

Examples of relationships mappings

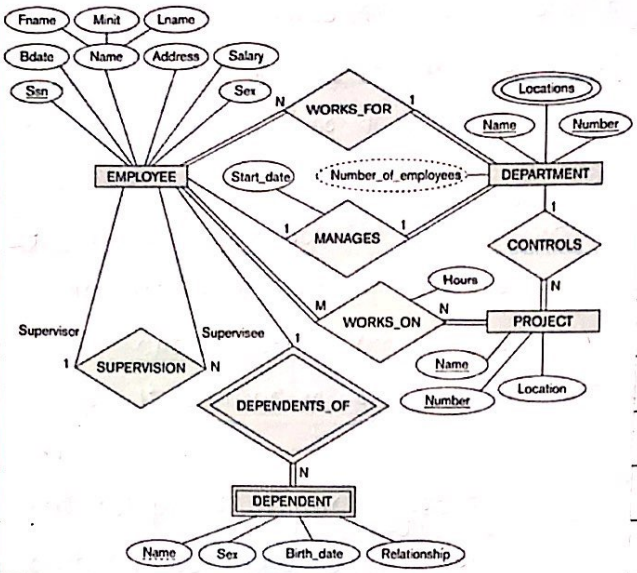


Relationship Type vs Relationship Set

Relationship type is the schema description of a relationship. It identifies the name of the relationship and the participating entity types. It also identifies certain relationship constraints.

Relationship set is the current set of relationship instances represented in the database. The current state of relationship type.

Example of ER Diagram with Company Database [with relations]



(i) WORKS_FOR EMP → DEPT
(ii) MANAGES EMP → DEPT
(iii) WORKS_ON EMP → PROJECT
(iv) DEPENDENTS_OF EMP → DEP
(v) SUPERVISION EMP → EMP
(vi) CONTROLS DEPT → PROJECT

Constraints on Relationships [Ratio Constraints]

Cardinality Ratio [Specifies maximum participation]

This could be 1:1, 1:n, n:1 or m:n. Taking above ER diagram examples are (ii), (iv),(v),(vi), (i), (iii).

Existence Dependency Constraint : participation of entity in a relationship.

Could be zero, Optional Participation

Must be one or more, Total Participation

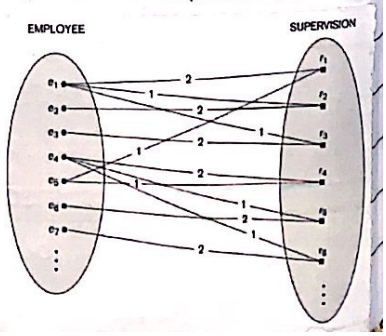
Recursive Relationship Type

A relationship type between the same participating entity type in distinct roles. This is also called self-referencing relationship type. In a recursive relationship, both participations are the same entity in different roles. In an ER diagram, we need to write the role names to distinguish between participations.

We can take the example of Supervision relation. The supervisor is also an employee and the people supervised are also employees.

1- supervisor 2- subordinate role

For each R_i, follow $E_i \xrightarrow{1} R \xrightarrow{2} E_j$



Weak Entity Types

An entity that does not have a key attribute and that is identification dependent on another entity type. This means that if an entry didn't exist in the owner entity, it wouldn't exist in the dependent identity.

A weak entity must participate in an identifying relationship type with an owner or identifying entity. A weak entity is identified by the combination of a partial key of weak entity type and a candidate key of the particular entity they are related to in identifying relationship.

Example) Dependents of Employees. If employee doesn't exist, dependent's won't. Partial Key = Name of dependent, Candidate Key =

Note)

*Participation Constraint and Cardinality Ratio Representation

In cardinality ratio :- Shown by placing appropriate numbers on relationship edges.

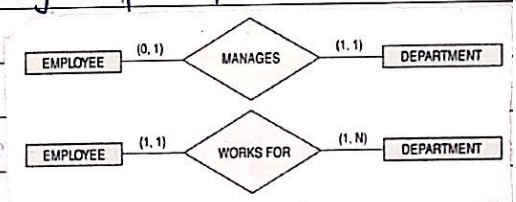
In participation constraint :- total ═, partial ─

Alternative (min, max) notation on relationship structural constraints

This is specified on each participation of an entity type E in a relationship. This specifies that each entity e participates at least min and at most max relationship instances in R.

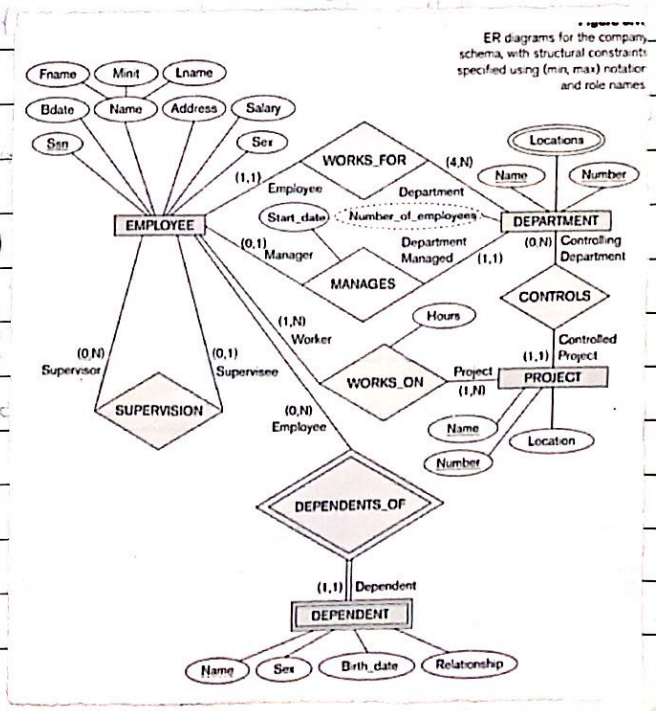
min ≥ 0 max ≥ 1 max ≥ min.

If max = n, signifies no limit.



Example) An employee may (1) or may not (0) manage a department, but a department must have only 1 manager (1,1)

An employee must work (1,1) for at least one department (1, n).



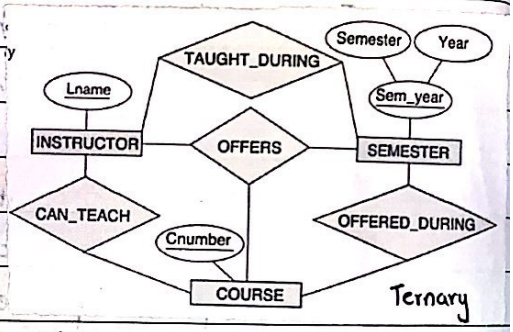
Refining ER design

We can refine our ER diagrams by using meaningful names for our entities, attributes and relationship types. Usually entities and attributes are nouns and relationship types are verbs.
It should be readable in fixed way. Left to Right and Top to Bottom

If an attribute refer to another entity type = Relationship set.
an attribute refer to multiple entity sets = Another entity.
an entity set is having one or two attributes and one relationship set = it can be modeled as an attribute

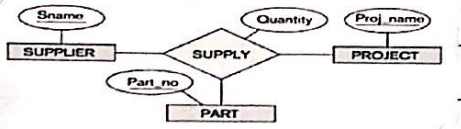
Relationships of Higher Degree

Relationships types of degreed
2 = binary, 3 = ternary, n = n-ary.
In general, an n-ary relationship is not equivalent to n binary relationships. Constraints are much harder to specify for higher degree relationships than for binary relationships.

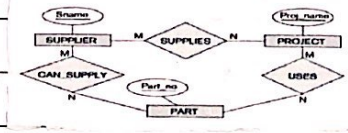


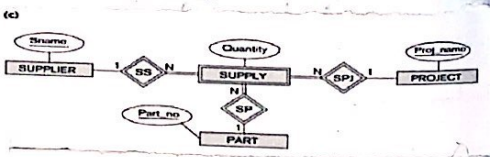
Ternary

In general, 3 binary relationships can represent a ternary relationship.
A ternary relationship can be represented as weak entity if data model allows for a weak entity to have multiple identifying relationships.



3 binary relations ↓

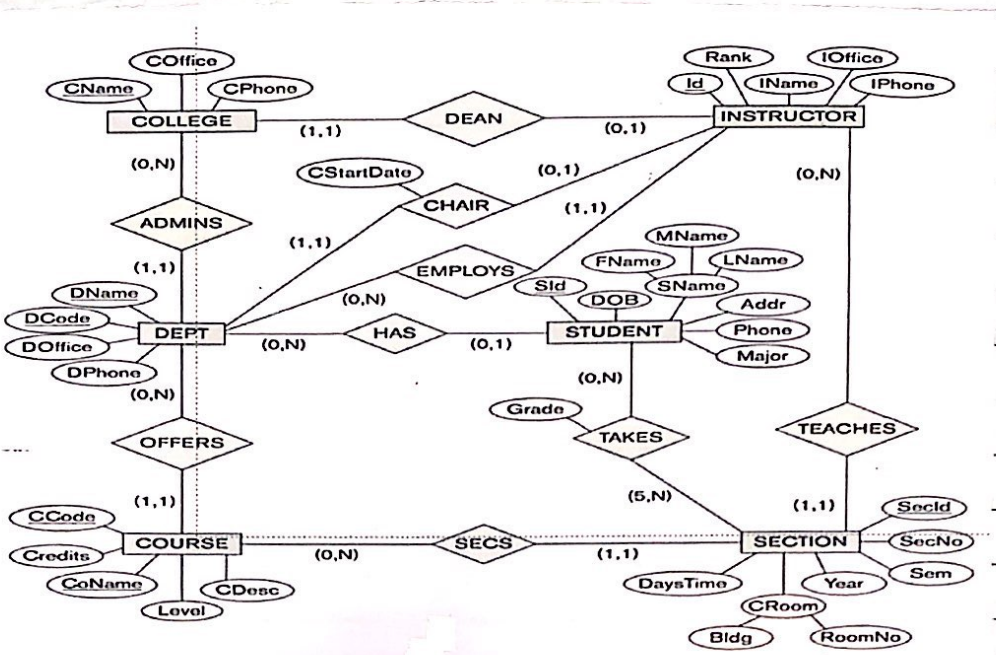




Weak Entity ↑

The (min, max) constraints can be displayed on edges but may not be fully describing the constraints.

University Database - Case Study



Generalisation

Specialization