DBMS

UNIT-1

Introduction to Database and Conceptual Design using ERD

feedback/corrections: vibha@pesu.pes.edu

VIBHA MASTI

Data

- known facts that can be recorded

Database

- collection of data
- logically coherent collection
- specific purpose

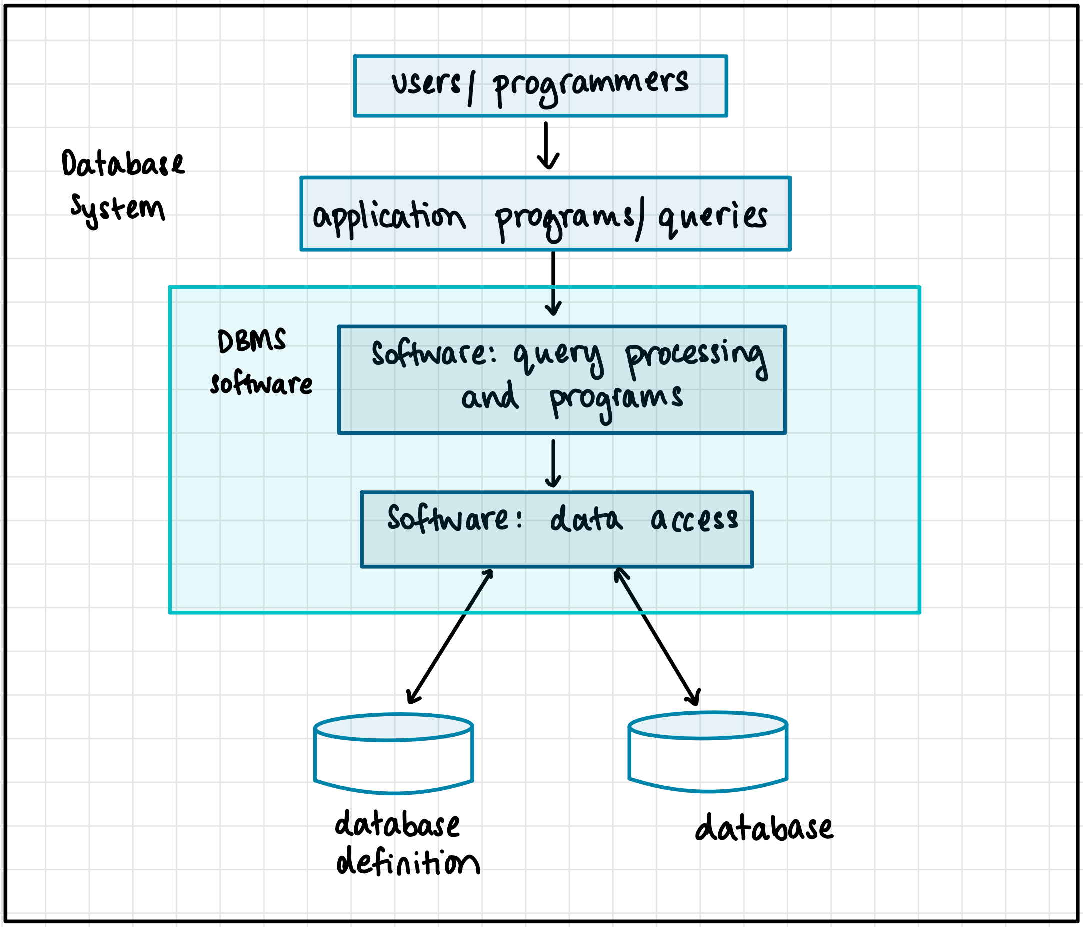
Database Management System

- software that manages data
- relational models (RDBMS), object-oriented (OODBMS), NOSQL etc.
- allows user to
  - define datatypes & constraints
  - construct (store) data
  - manipulate data through queries

impact of database technology

- businesses (banking, insurance, transport, manufacturing)
- service industries (financial, real estate, legal, e-commerce, healthcare)
  vamsha vruksha
- social networks
- science and research
- education
- personal apps

Database System Environment

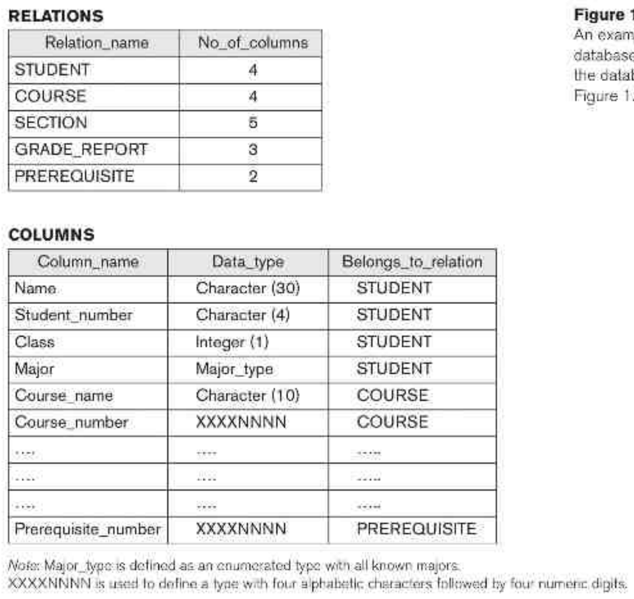


Example of a Database with Conceptual Model

- University — mini world or universe of discourse (UOD)
- Students, courses, departments, instructors, sections etc. (different entities)
- Model diagram — Entity-Relationship data model

Characteristics of DB Approach

- Self-describing nature of DB system
  - metadata stored for each DB
  - stored in a catalog on disk
  - datatypes, constraints (specified by DDL statements) data definition language ✓
  - NOSQL DBs do not have structure and may not have metadata



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 enumerated type with all known majors.
XXXXNNNN is used to define a type with four alphabetic characters followed by four numeric digits.

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

- Insulation between programs and data
  - program-data independence
  - change data structures and storing organisation without changing the DBMS access programs

- Data abstraction
  - users provided only with conceptual view of DB ; no internal structures
  - programs access data model constructs

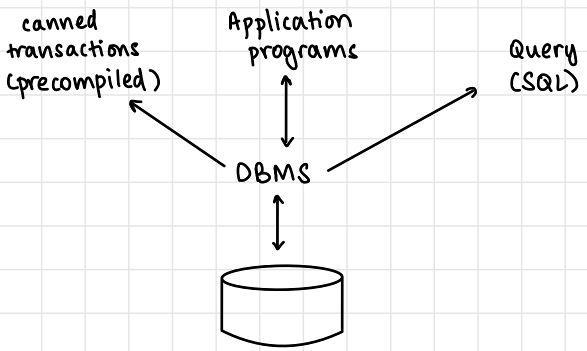
- Support of multiple views of data
  - views differ based on user's interest

- Sharing of data and multi-user transaction processing
  - ACID : atomicity, concurrency, isolation, durability
  - Atomicity: transaction either fully complete or do nothing
  - Concurrency: multiple concurrent transactions
  - Isolation: all transactions isolated from one-another (independence)
  - Durability: transactions reflected with permanent changes
  - query: fetch data , transaction: modify data
  - OLTP

Database Users

1. Actors on the scene
   - use and control DB content
   - end users
     - casual users (occasional use)
     - naive or parametric users (canned transactions)
     - sophisticated end users
     - standalone end user
     - DB admins (authorise DB access)
     - DB designers (convert requirements to DDL statements)

2. Workers behind the scene
   - system designers and implementors
   - tool developers
   - operators and maintenance personnel



Advantages of DBMS Approach

- Redundancy of data is avoided
- Restricted and protected access to data
- Permanent / persistent storage for program objects
- Storage structures and indexes for efficient query processing and retrieval
- Backup and recovery services
- Efficient query processing

Development of Database Technology

- Hierarchical Data Model — HDBMS } 1960's and 70's
- Network Data Model — NDBMS
- Relational Data Model — RDBMS } 1970's and 80's
- Object Oriented Data Model — OODBMS (jagged array)
- NOSQL Data Model — NOSQL

data models

- Describes structure of DB
  - datatypes
  - relationship b/w entities
  - constraints
  - operations : retrievals and modifications

Operations

- Basic operations (eg: insert, delete, update)
- User-defined operations (like user-defined functions)

CATEGORIES of DATA MODELS

1. Conceptual or High-Level data model
   - ER model
   - Concepts close to users' perception of data

2. Physical or Low-level Data Model
   - data structures for storage

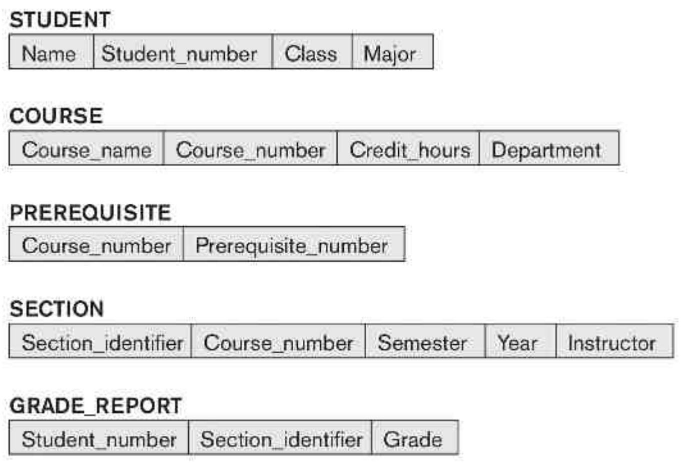
3. Implementation or Representational Data Model
   - Commercial DBMS (relational)

4. Self-describing Data Model
   - NOSQL databases

DATABASE SCHEMA

- Description of database & database structure
- Database is populated based on schema
- Also called intension

Figure 2.1
Schema diagram for the database in Figure 1.2.



STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

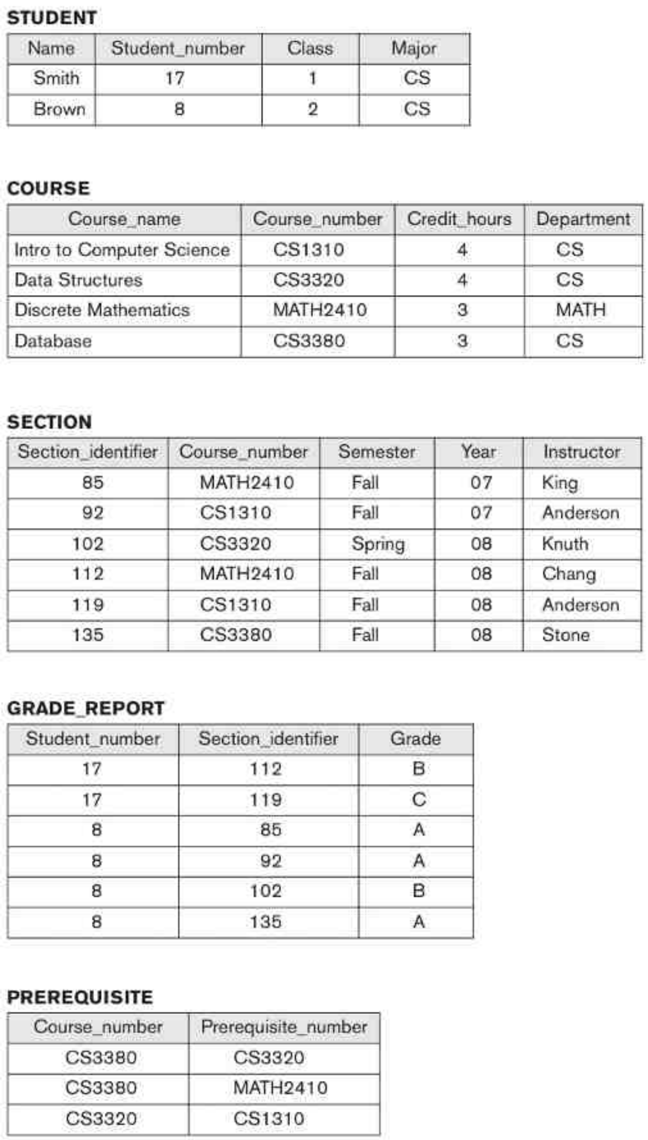
Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

Database State

- Database content at any given point of time
- Valid state: database state follows constraints
- Initial state: state when loaded into the system
- Also called extension



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

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

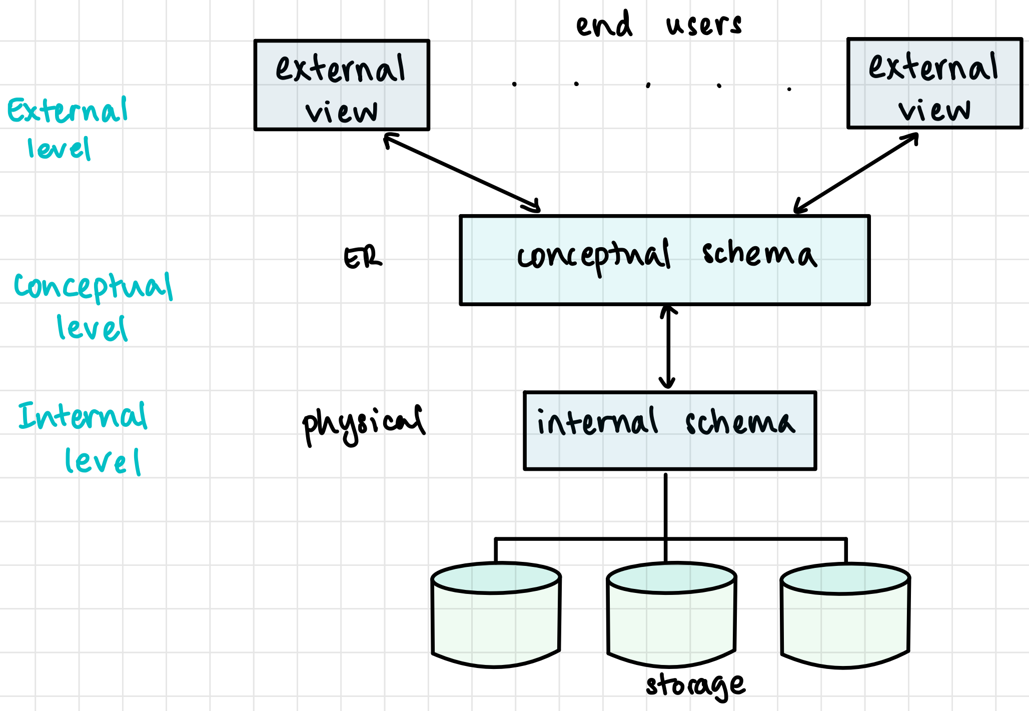
Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

Three-Schema Architecture

- To ensure
  - program-data independence
  - multiple views
  - catalogs

- end user ⟶ mini statement ⟶ conceptual schema



- Data independence: change one schema without changing other schemas

  (i) Logical data independence
  - change conceptual schema without having to change external schema

  (ii) Physical data independence
  - change internal schema without having to change conceptual schema

DBMS Languages

(i) Data Definition Language (DDL)
- DBA and DB designers
- Define schemas
- DDL compiler

(ii) Data Manipulation Language (DML)
- High-level/ non-procedural language (can be embedded, SQL)
- Low-level /procedural language (must be embedded in prog lang)

DBMS Interface

1. Stand - alone query language interfaces
   - eg: SQL queries in interactive interface

2. Programmer interfaces to embed DML in programming languages

3. User-friendly interfaces
   - eg: menu-based, form-based, GUI-based

4. Mobile interfaces
   - eg: mobile transactions

DATABASE SYSTEM UTILITIES

1. Loading utility
- Load data from files into a DB
- Data conversion tools from old DB format to new DB format

2. Backup Utility
- Backing up the DB periodically
- Incremental or total

3. Storage Reorganisation
- Reorganising database file structures
- Create new access paths

4. Performance monitoring
- monitors DB use
- Sends stats to DBA

Other Tools

- Data dictionary / repository
  - schema descriptions, design decisions etc
  - active dictionary : DBMS software, users / DBA
  - passive dictionary : users / DBA only
- Application Development Environment and Computer Aided Software Engineering (CASE) tools
  - PowerBuilder (Sybase)
  - JBuilder (Borland)
  - JDeveloper 10G (Oracle)

DBMS Component Modules

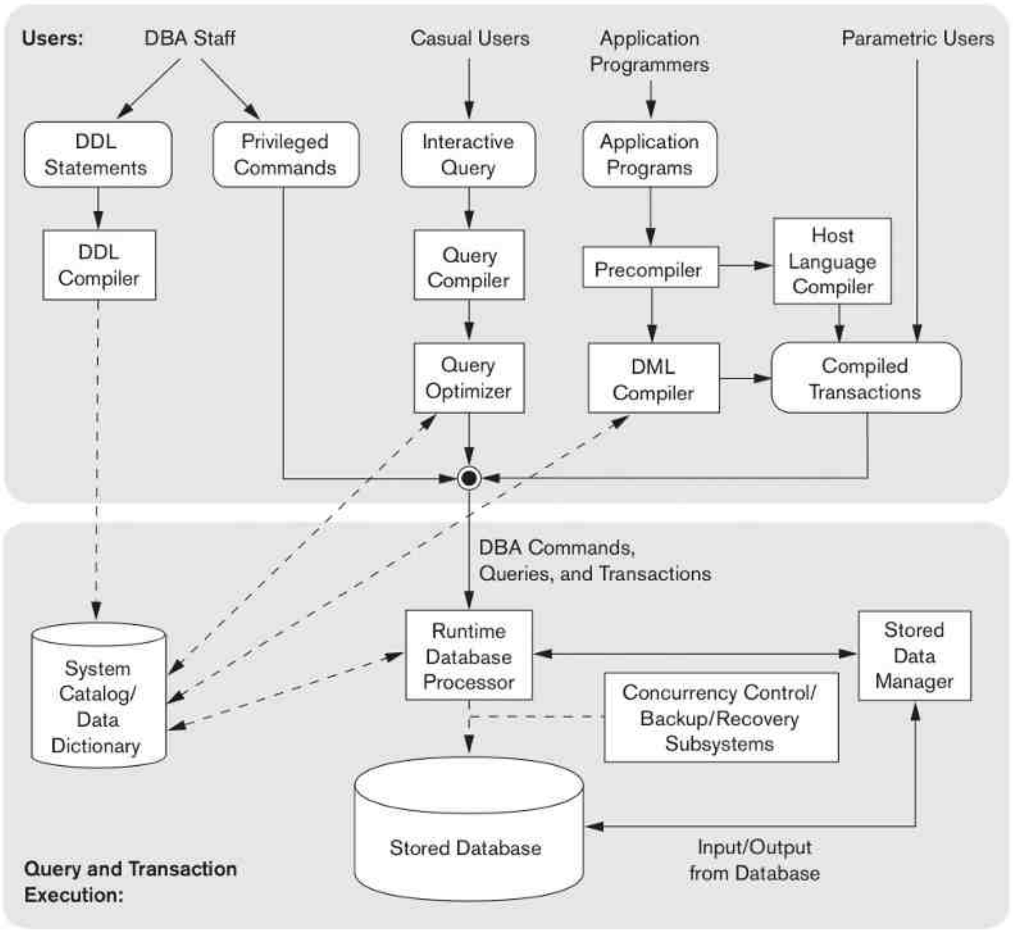


Figure 2.3
Component modules of a DBMS and their interactions.

Centralised & Client-Server DBMS Architectures

1. Centralised Architecture

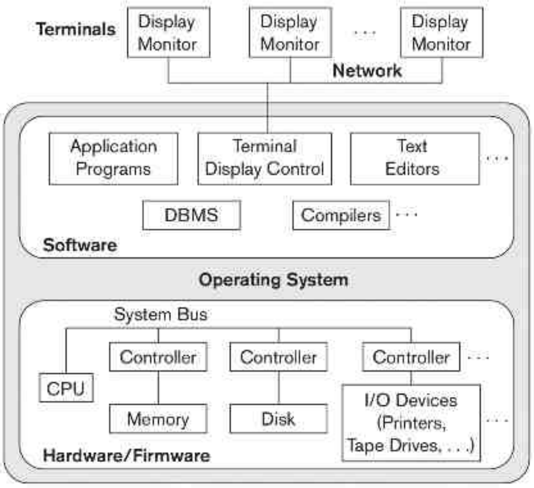


Figure 2.4
A physical centralized architecture.

2. Two-Tier Client-Server Architecture

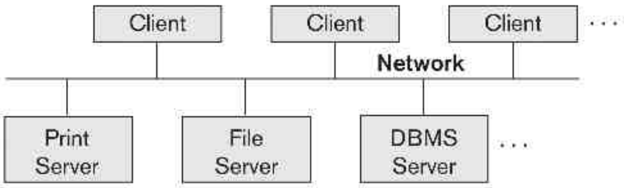


Figure 2.5
Logical two-tier client/server architecture.

Client

- diskless machines / PCs / workstations
- connected to servers through a network (LAN, wireless etc.)
- User machine that provides UI capabilities and local processing

DBMS Server

- System that provides services to clients when requested (file access, printing, archiving)

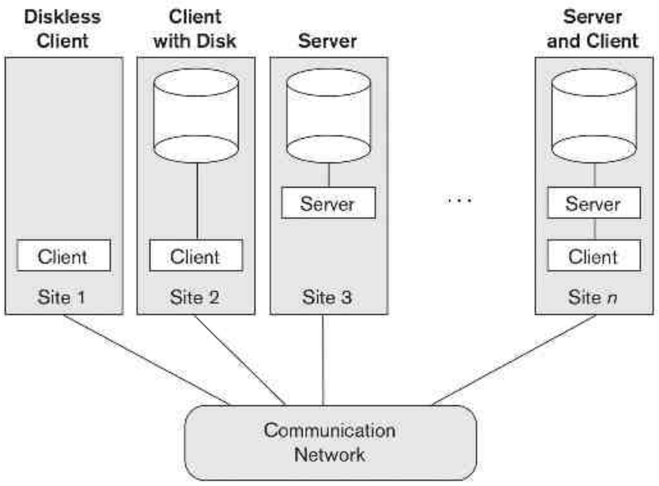


Figure 2.6 Physical two-tier client/server architecture.

3. Three-Tier Client-Server Architecture

- Intermediate layer between client and server

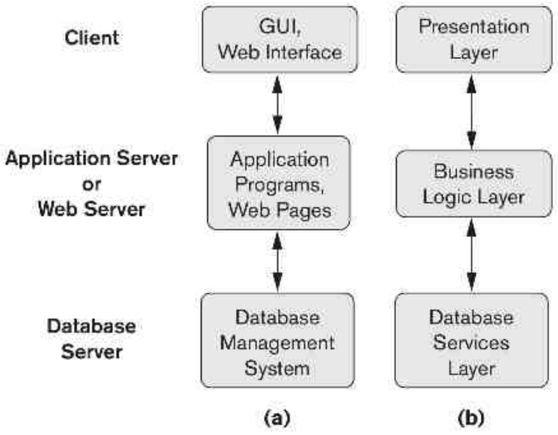


Figure 2.7 Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

Classification of DBMS

1. Based on data model

- Legacy: network, hierarchical
- Currently used: relational, object-oriented, object-relational
- Recent: document-based, column-based, graph-based, key-value based

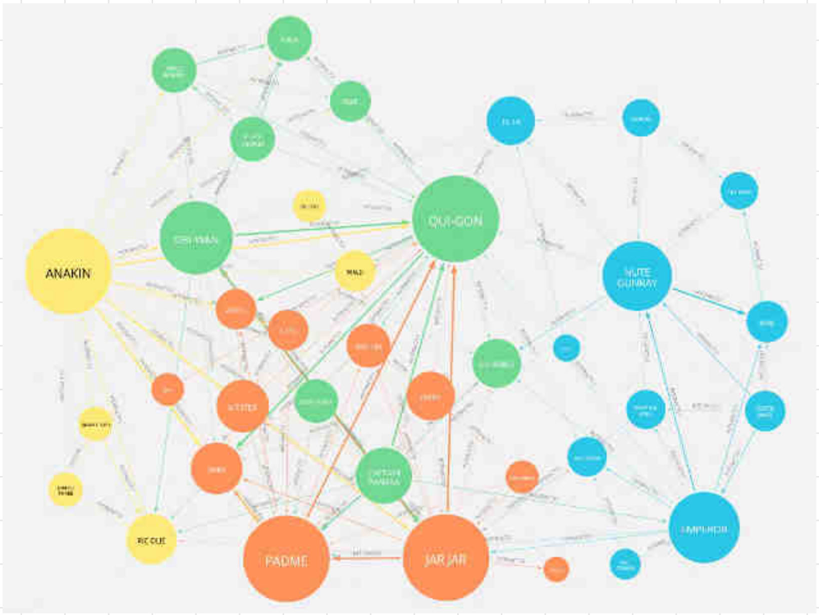
2. Other classifications

- Single-user vs multi-user
- Centralised vs distributed
- Homogeneous vs heterogeneous distributed DBMS

  ↑ same DBMS at all sites　　↑ different DBMS at all sites

- Eg: graph database (Neo4j) — Star wars

https://towardsdatascience.com/presenting-multiple-node-label-support-and-graph-mutability-features-of-the-neo4j-graph-data-a0b0ea744884



Database Design Process

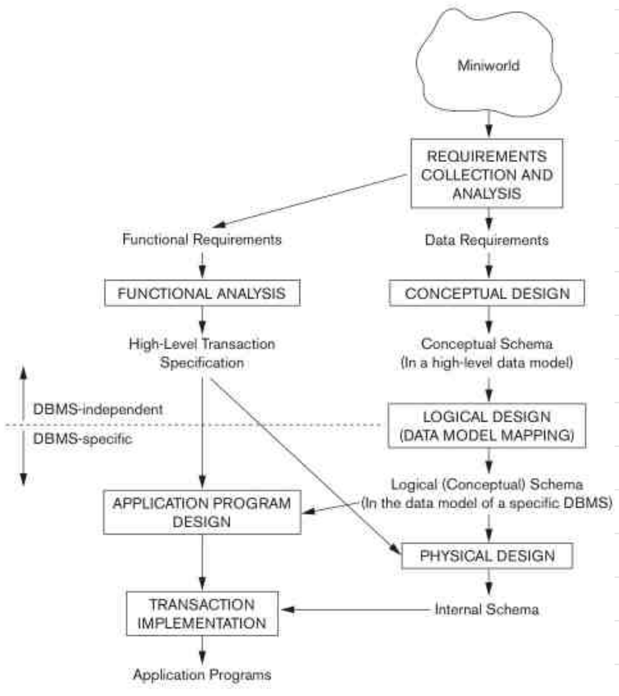

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Requirements of the COMPANY Database

- Company organised into DEPARTMENTs
- Each department has a name, number and an employee who manages the department
- Keep track of start date of department manager
- Department may have several locations
- Each department controls a number of PROJECTs
- Each project has a unique name, number and location
- DB stores each EMPLOYEE's SSN, address, salary, sex, birthdate
- Each employee works for one department but may work on several projects
- DB keeps track of no. of hours per week that an employee works on a project

- Keep track of direct supervisor of each employee
- Each employee has a number of DEPENDENTs
- DB records each dependent's name, sex, birthplace, relationship with employee

Entity

- Basic concept for ER model (relation)
- Specific things/object in the mini world
- Physical or conceptual (degree: no. of attributes)
- Entities have attributes
- Entity Set: collection of similar entities
- Strong (single line border) and weak (double line)
- Rectangle
- Type: class/abstract template

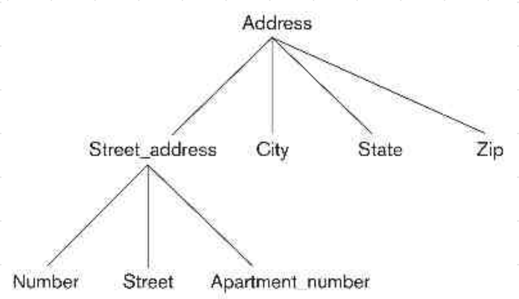
Attribute

- Properties used to describe an entity
- Associated with a domain/value set
- Oval
- Each attribute has a type

Types of Attributes

(i) Simple/composite Attributes

- Simple: single atomic valued attribute
  - eg: Aadhaar number
- Composite: composed of several components; tree
  - eg: Telephone number (STD code, number)



(ii) Single-valued | multi-valued Attributes

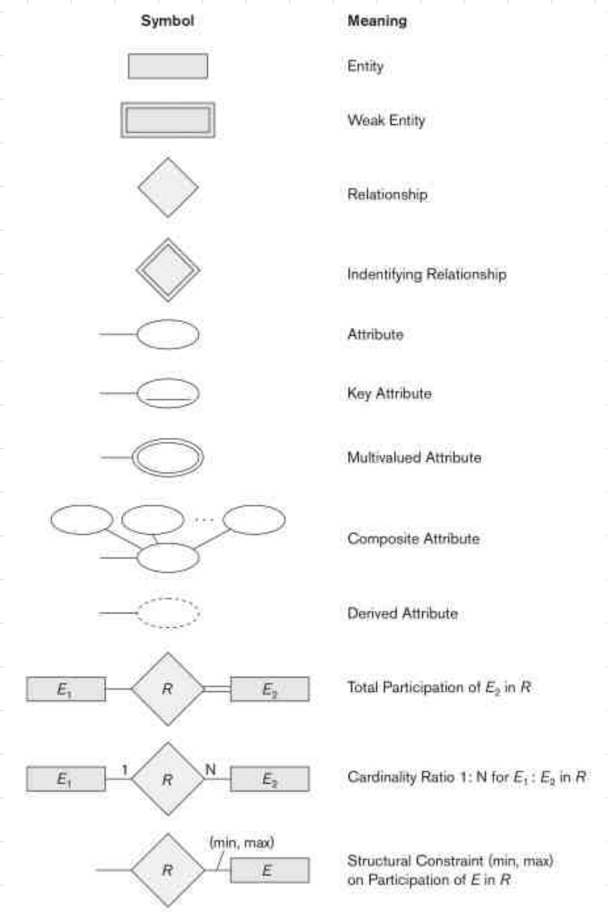
- Single-valued: single value for an attributed; single oval
  - eg: DOB
- Multi-valued: may have multiple values ; two oval lines
  - eg: Course Set = { DDCO, AFLL, DBMS }

(iii) Stored / derived Attributes

- Age can be derived from DOB
- Derived: dotted line

(iv) Complex Attributes

- Combination of multi-valued and composite



Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
E_1 R E_2	Total Participation of E_2 in R
E_1 1 R N E_2	Cardinality Ratio 1: N for $E_1 : E_2$ in R
R (min, max) E	Structural Constraint (min, max) on Participation of E in R

E-R Model

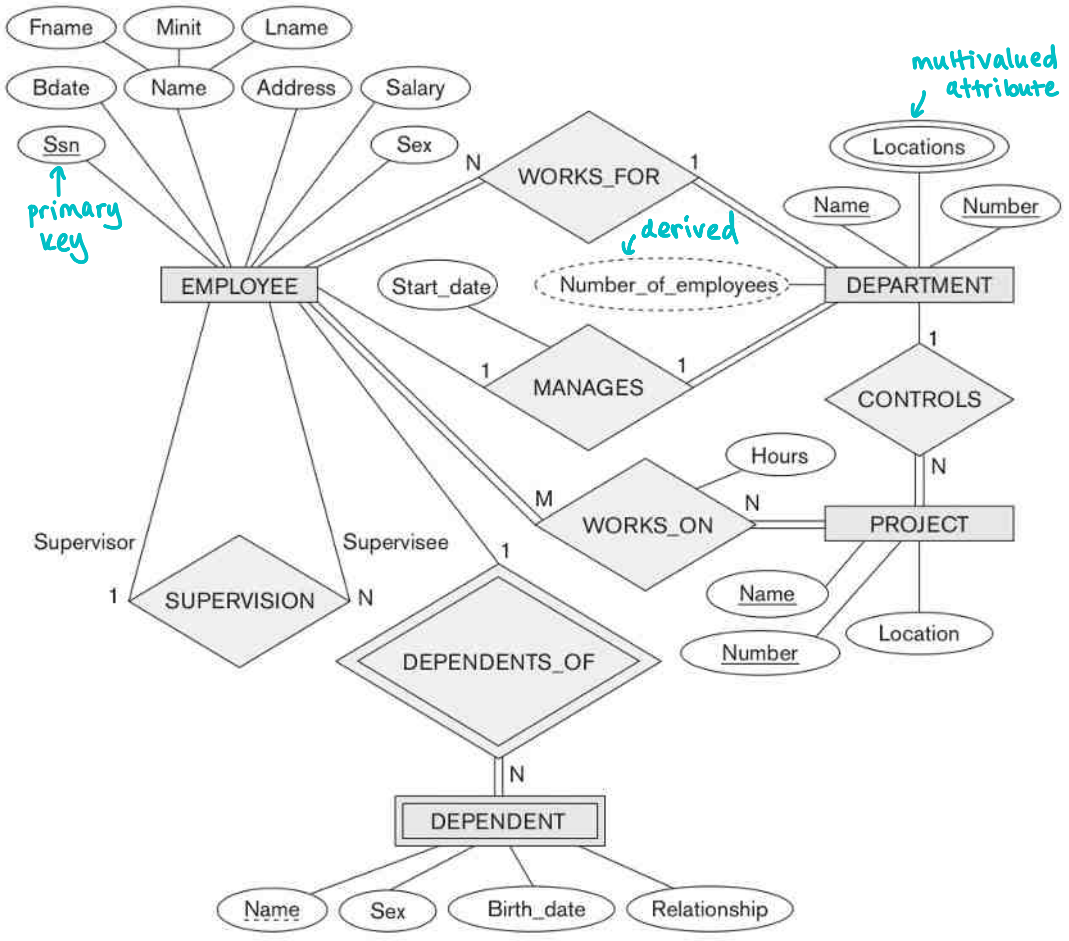


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 3.14.

Cardinality Ratio

- 1:1, 1:N, N:1, N:1, M:N
- Can also use (min,max) representation

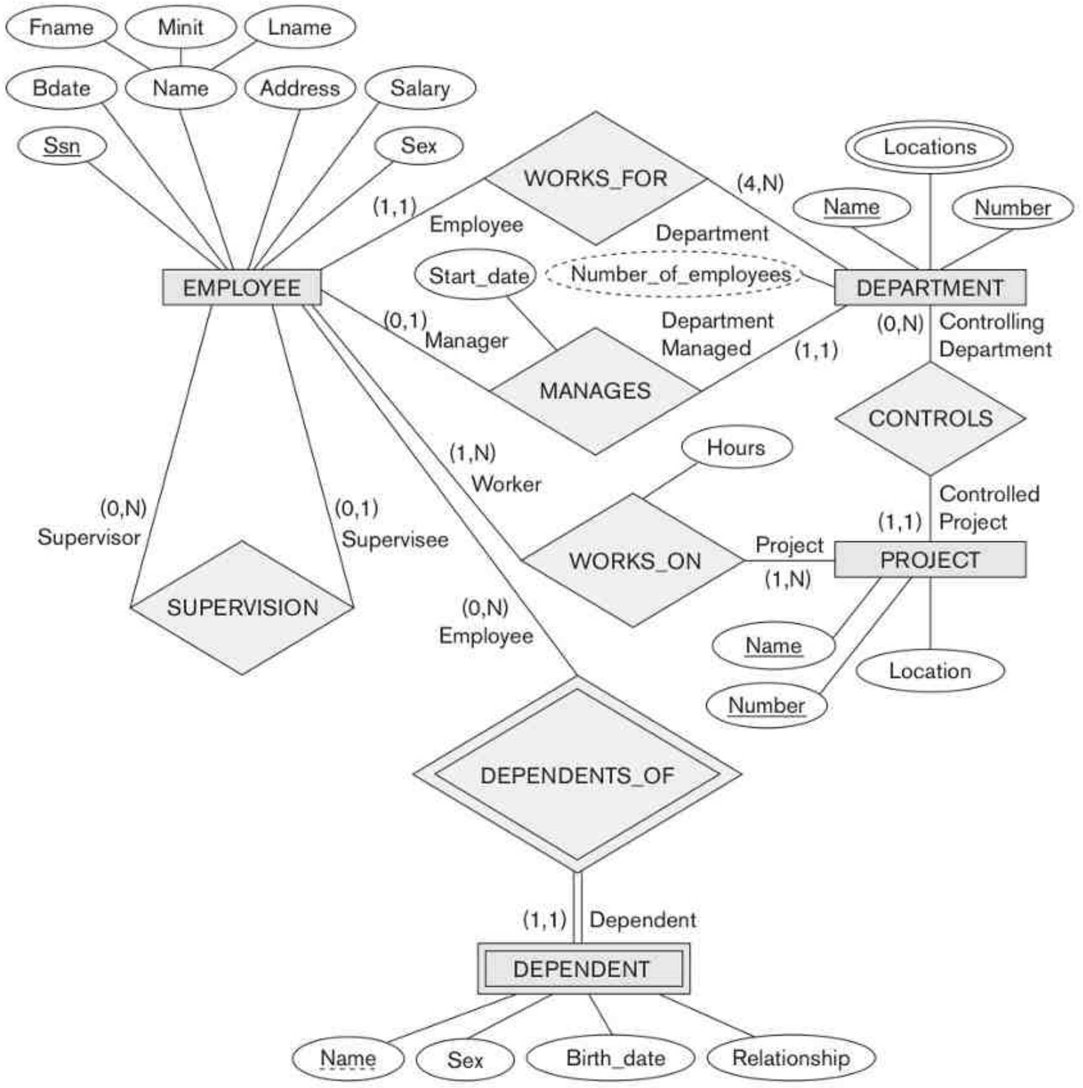


Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

(i) 1:1

- 1 employee manages 1 dept
- 1 dept managed by 1 employee
- can pick a side

T1

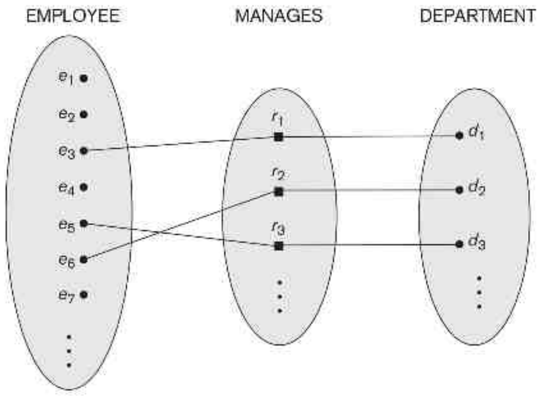
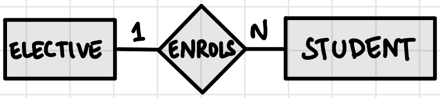


Figure 3.12
A 1:1 relationship, MANAGES.

[8]N stands for *any number* of related entities (zero or more). In some notations, the asterisk symbol (*) is used instead of N.

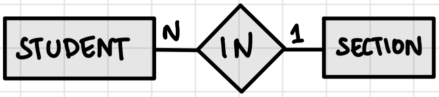
(ii) 1:N

- N students in 1 elective
- 1 elective can have N students
- stored at N side
- 1 student has only one elective



(iii) N:1

- N students in 1 section
- 1 section has N students
- stored at N side
- 1 student is in only 1 class



(iv) M:N

- M students register for N courses
- N courses have M registered students in total
- create separate table

T1

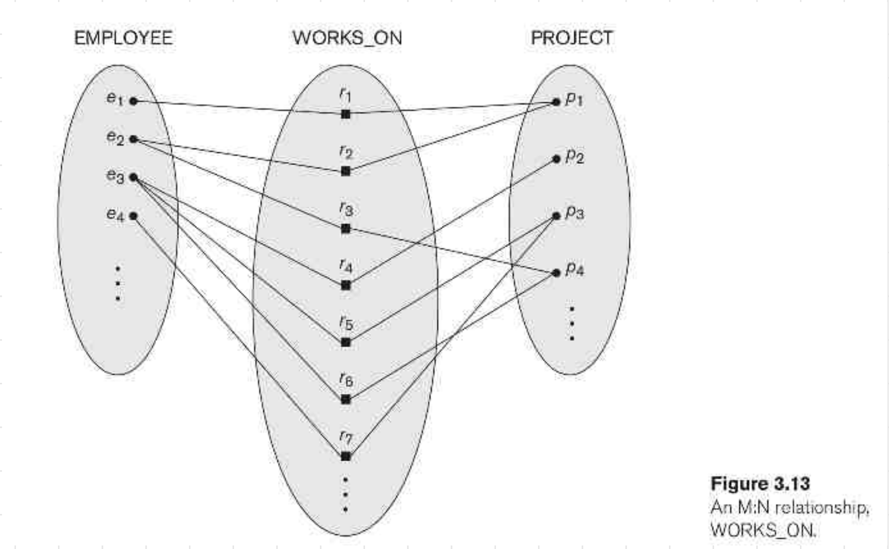
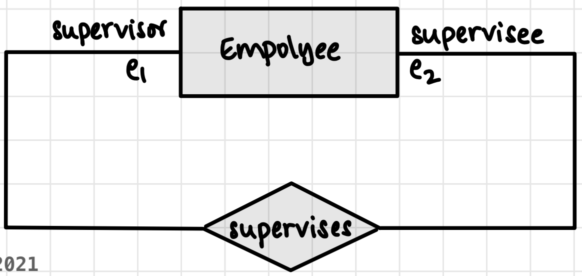


Figure 3.13
An M:N relationship, WORKS_ON.

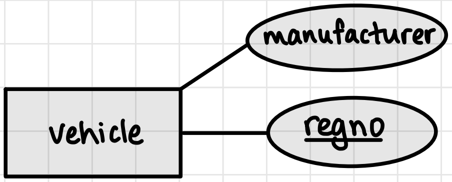
Relationships

- Can be binary, ternary etc but advisable to use only binary
- Recursive: one employee manages N empolyees — self-referential relationship (mandatory to specify role names)



Key Attribute

- unique identifier for an attribute
- can have more than one key attribute
- specified with underline

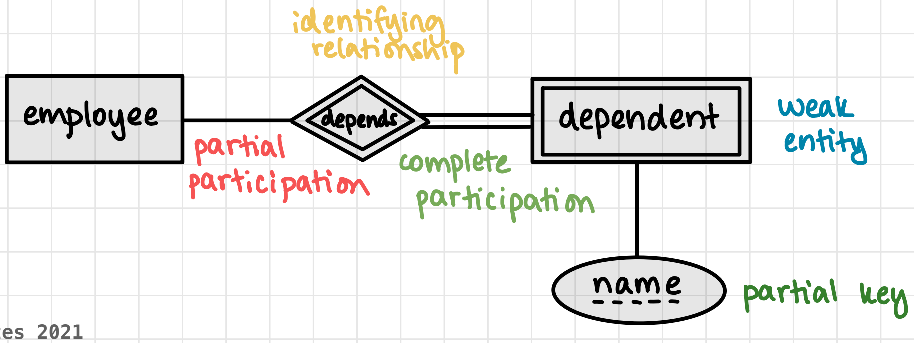


Disadvantages of E-R Diagram

- Normalisation: Reduce the NULL entries; cannot be achieved with E-R diagram
- Cannot represent datatypes in ER diagram

Weak Entity

- Partial key: Dotted line; not always unique
- Weak entities do not have a unique key; must be associated with at least one strong entity
- Called identifying relationship



Case Study

- The university is organized into colleges (COLLEGE), and each college has a unique name (CName), a main office (COffice) and phone (CPhone), and a particular faculty member who is dean of the college. Each college adminis- ters a number of academic departments (DEPT). Each department has a unique name (DName), a unique code number (DCode), a main office (DOffice) and phone (DPhone), and a particular faculty member who chairs the department. We keep track of the start date (CStartDate) when that fac- ulty member began chairing the department.

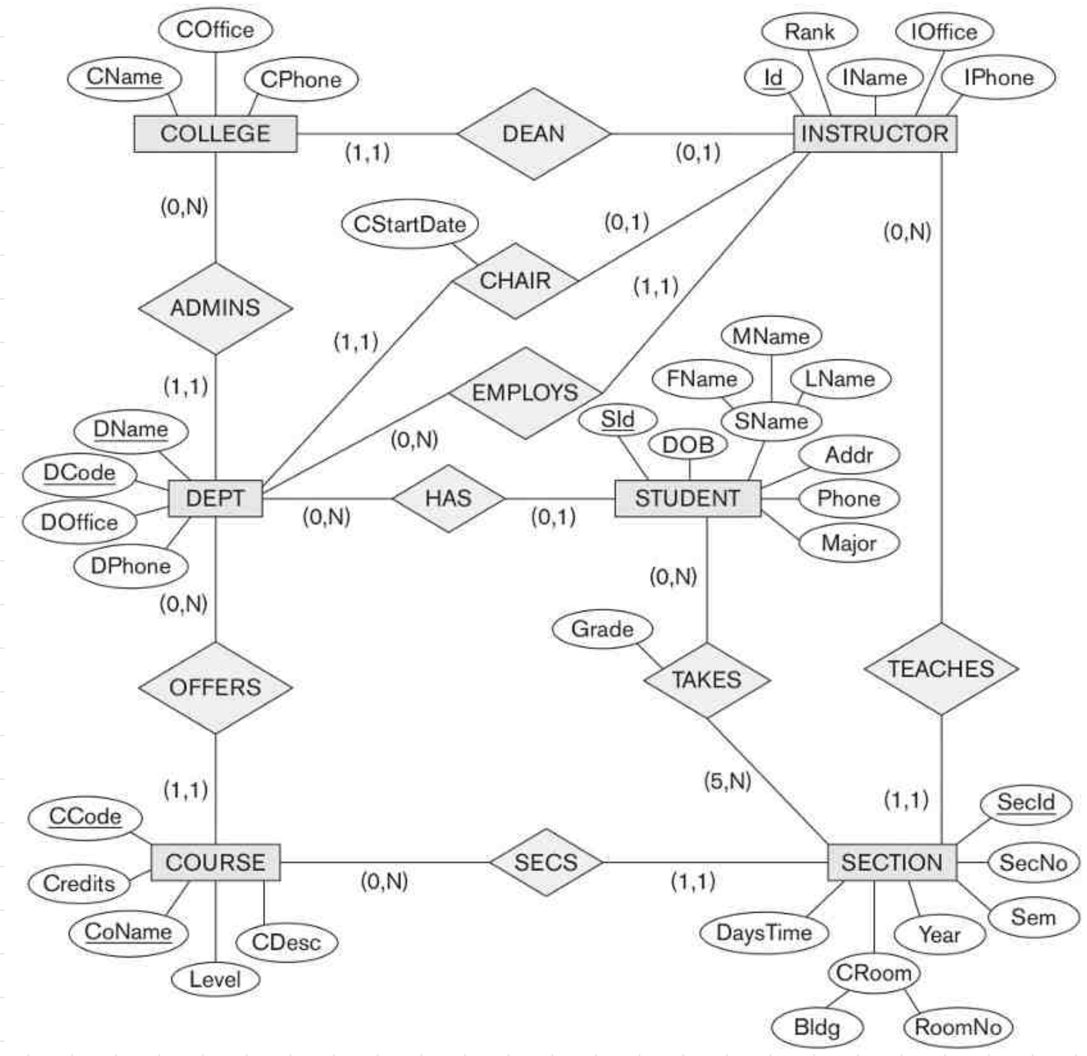
- A department offers a number of courses (COURSE), each of which has a unique course name (CoName), a unique code number (CCode), a course level (Level: this can be coded as 1 for freshman level, 2 for sophomore, 3 for junior, 4 for senior, 5 for MS level, and 6 for PhD level), a course credit hours (Credits), and a course description (CDesc). The database also keeps track of instructors (INSTRUCTOR); and each instructor has a unique iden- tifier (Id), name (IName), office (IOffice), phone (IPhone), and rank (Rank); in addition, each instructor works for one primary academic department.

- The database will keep student data (STUDENT) and stores each student's name (SName, composed of first name (FName), middle name (MName), last name (LName)), student id (Sid, unique for every student), address (Addr), phone (Phone), major code (Major), and date of birth (DoB). A stu- dent is assigned to one primary academic department. It is required to keep track of the student's grades in each section the student has completed.

- Courses are offered as sections (SECTION). Each section is related to a single course and a single instructor and has a unique section identifier (SecId). A section also has a section number (SecNo: this is coded as 1, 2, 3, . . . for mul- tiple sections offered during the same semester/year), semester (Sem), year (Year), classroom (CRoom: this is coded as a combination of building code (Bldg) and room number (RoomNo) within the building), and days/times (DaysTime: for example, 'MWF 9am-9.50am' or 'TR 3.30pm-5.20pm'— restricted to only allowed days/time values). (Note: The database will keep track of all the sections offered for the past several years, in addition to the current offerings. The SecId is unique for all sections, not just the sections for a particular semester.) The database keeps track of the students in each section, and the grade is recorded when available (this is a many-to-many relationship between students and sections). A section must have at least five students.

Using (min, max) notation

- min & max no. of relationship instances



N-Ary Relationships

- Ternary relationships that can be represented by multiple binary relationships are said to be redundant
- In 3.17, binary relationship cannot replace ternary relationship
- Can represent as a weak entity type and associate it with identifying relationship

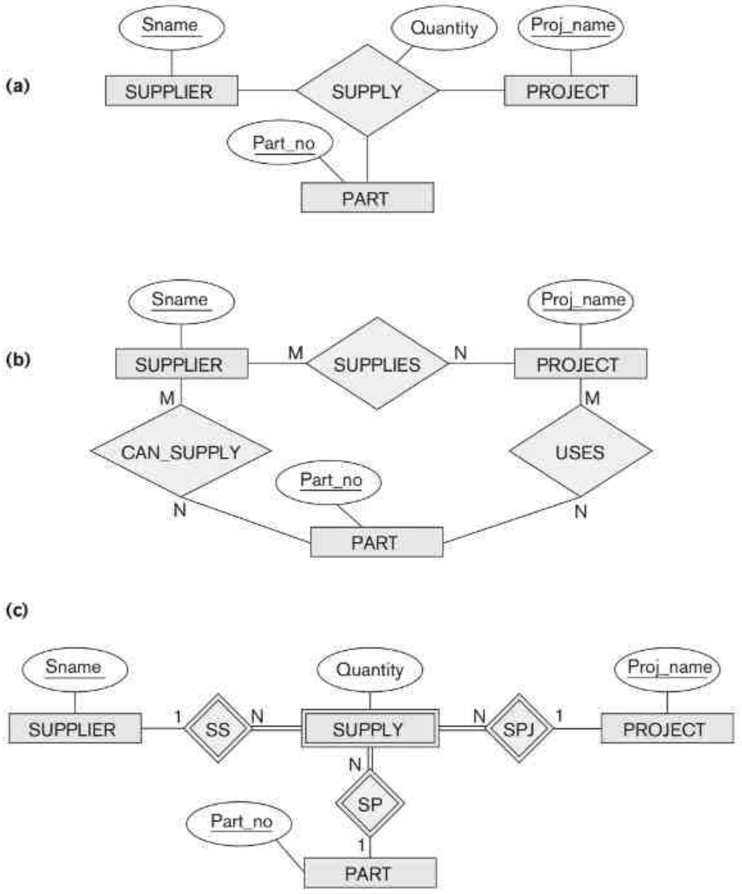
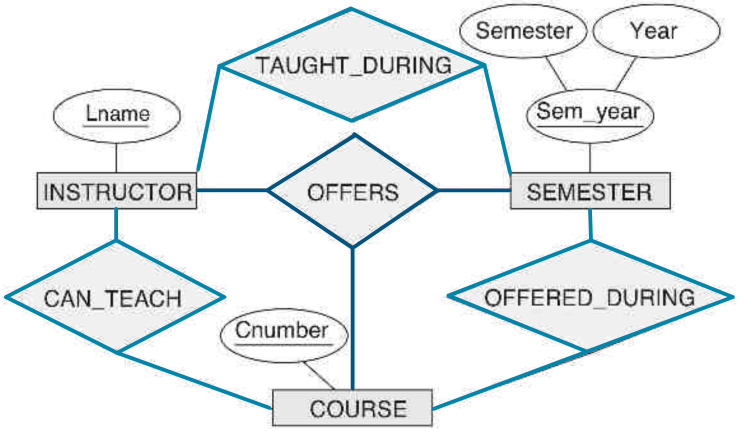


Figure 3.17
Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

- Another example

binary relationships
ternary relationship

Figure 3.18 Another example of ternary versus binary relationship types.



R1

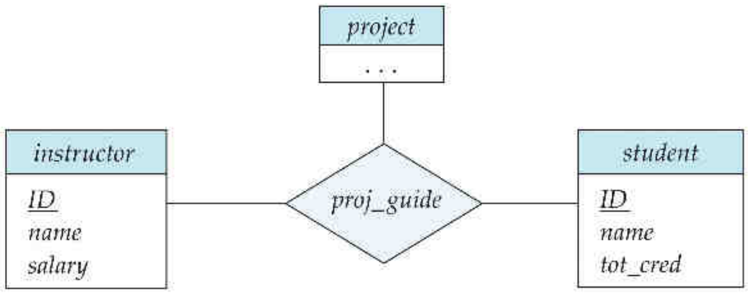
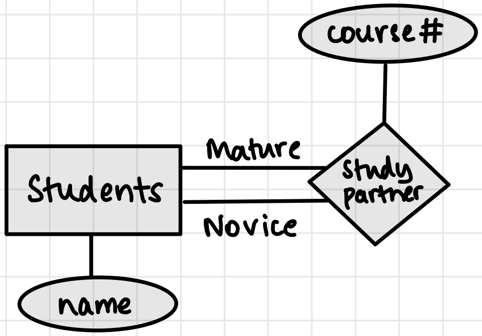


Figure 7.13 E-R diagram with a ternary relationship.

Recursive Relationship

- Entities can be related to themselves



R1

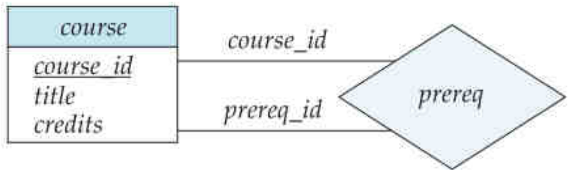


Figure 7.12 E-R diagram with role indicators.

T1

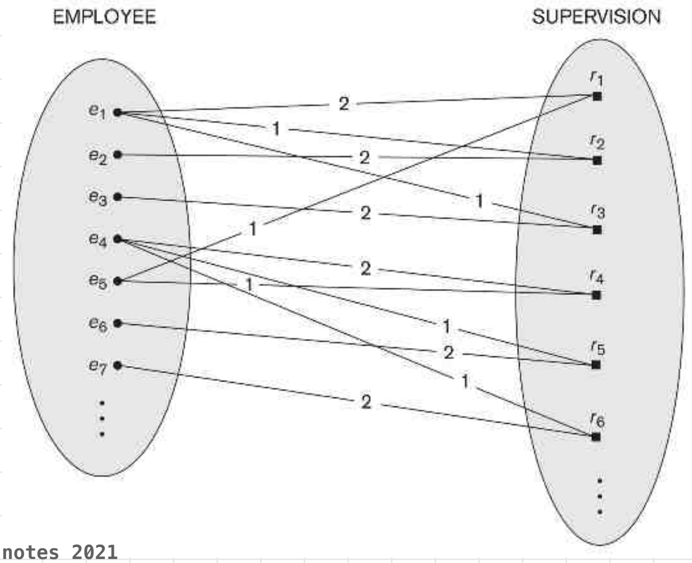


Figure 3.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).