CS4.301 Data & Applications

Ponnurangam Kumaraguru ("PK")
#ProfGiri @ IIIT Hyderabad

pk.profgiri /in/ponguru @ponguru Ponnurangam.kumaraguru

Data Modeling Using the Entity-Relationship (ER) Model

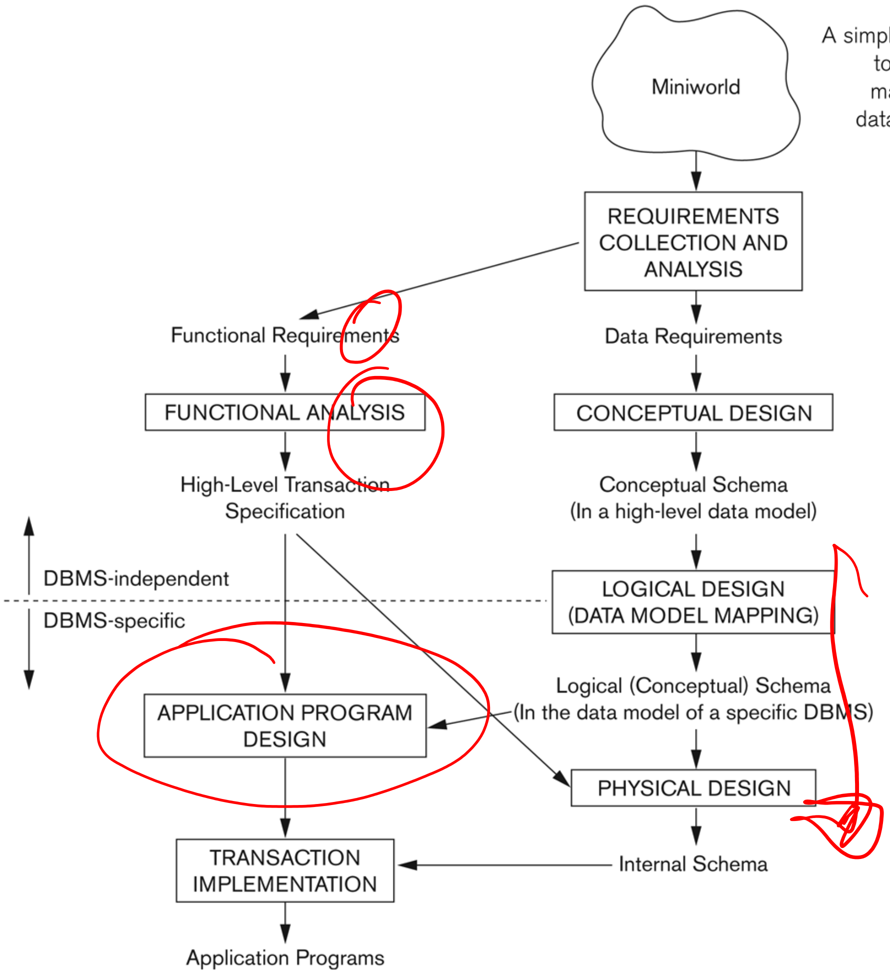


Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Overview of Database Design Process

Example COMPANY Database

We need to create a database schema design based on the following (simplified) **requirements** of the COMPANY Database:

- The company is organized into DEPARTMENTs. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager. A department may have several locations.
- Each department *controls* a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.

Example COMPANY Database (Continued)

- The database will store each EMPLOYEE's social security number, address, salary, gender, and birthdate.
  - Each employee *works for* one department but may *work on* several projects.
  - The DB will keep track of the number of hours per week that an employee currently works on each project.
  - It is required to keep track of the *direct supervisor* of each employee.
- Each employee may *have* a number of DEPENDENTs.
  - For each dependent, the DB keeps a record of name, gender, birthdate, and relationship to the employee.

ER Model Concepts

Entities and Attributes

- Entity is a basic concept for the ER model. Entities are specific things or objects in the mini-world that are represented in the database.
  - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT
- Attributes are properties used to describe an entity.
  - For example an EMPLOYEE entity may have the attributes Name, SSN, Address, gender, BirthDate
- A specific entity will have a value for each of its attributes.
  - For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', gender='M', BirthDate='09-JAN-55‘
- Each attribute has a *value set* (or data type) associated with it – e.g. integer, string, date, enumerated type, …

Types of Attributes (1)

Simple

Each entity has a single atomic value for the attribute. For example, SSN or gender.

Composite

The attribute may be composed of several components. For example:

- Address (Apt#, House#, Street, City, State, ZipCode, Country), or
- Name (FirstName, MiddleName, LastName).

Multi-valued

Multiple values for the attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT.

- Denoted as {Color} or {PreviousDegrees}.

Entity Types and Key Attributes (1)

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

For example, the 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 of the entity type.

For example, SSN of EMPLOYEE.

Entity Types and Key Attributes (2)

A key attribute may be composite.

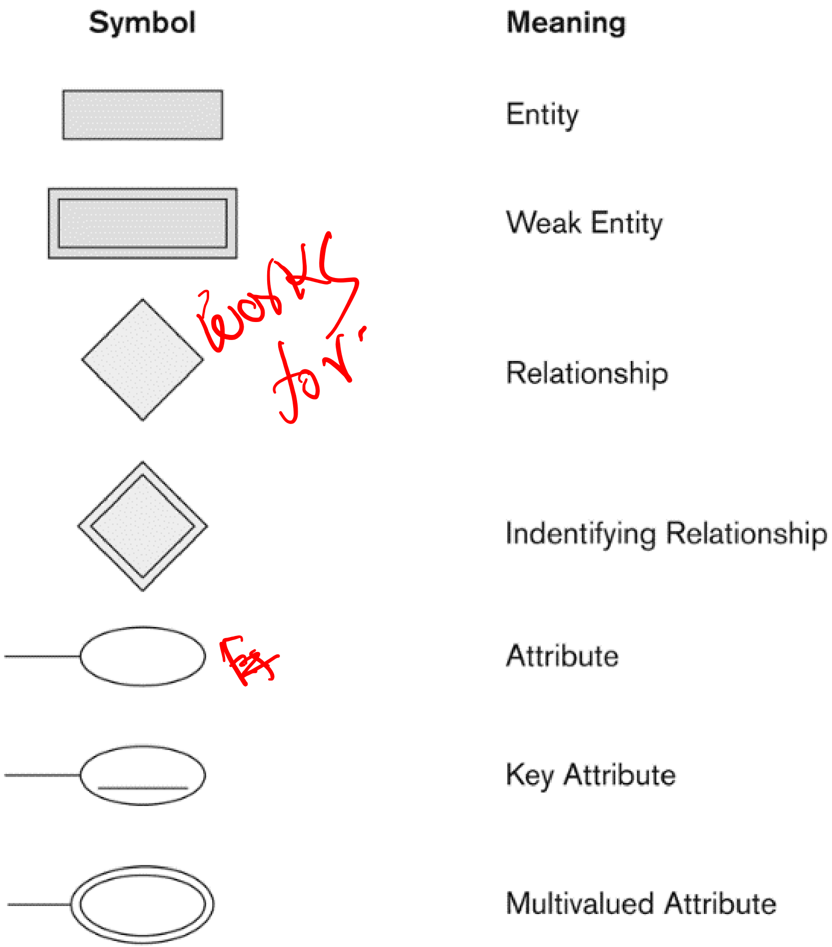
- VehicleTagNumber is a key of the CAR entity type with components (Number, State).

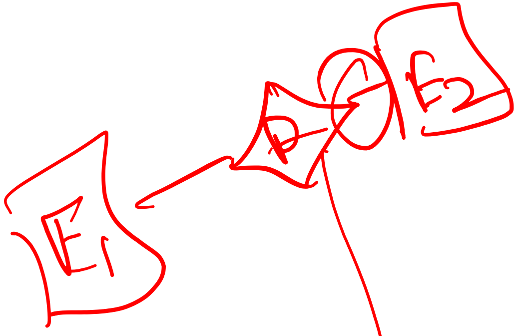
An entity type may have more than one key.

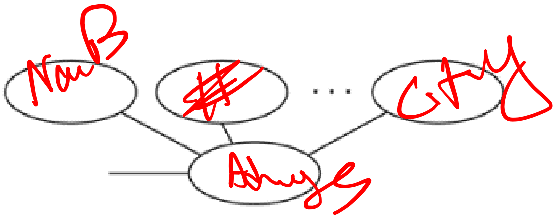
- The CAR entity type may have two keys:
  - VehicleIdentificationNumber (popularly called VIN)
  - VehicleTagNumber (Number, State), aka license plate number.

<u>Each key </u>is <u>underlined</u>

Figure 3.14
Summary of the notation for ER diagrams.

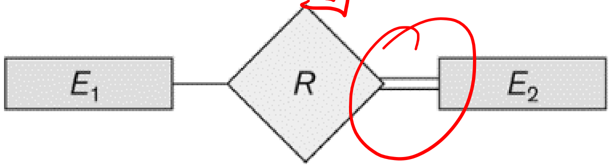




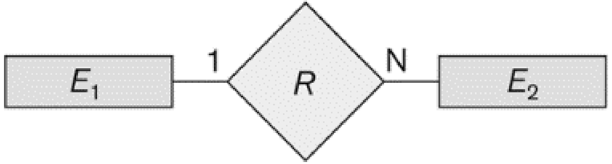


Composite Attribute

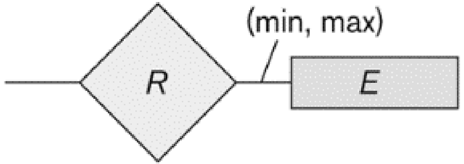
Derived Attribute



Total Participation of E_2 in R



Cardinality Ratio 1: N for E_1:E_2 in R



Structural Constraint (min, max) on Participation of E in R

Initial Conceptual Design of Entity Types for the COMPANY Database Schema

Based on the requirements, we can identify four initial entity types in the COMPANY database:

- DEPARTMENT
- PROJECT
- EMPLOYEE
- DEPENDENT

Their initial conceptual design is shown on the following slide

The initial attributes shown are derived from the requirements description

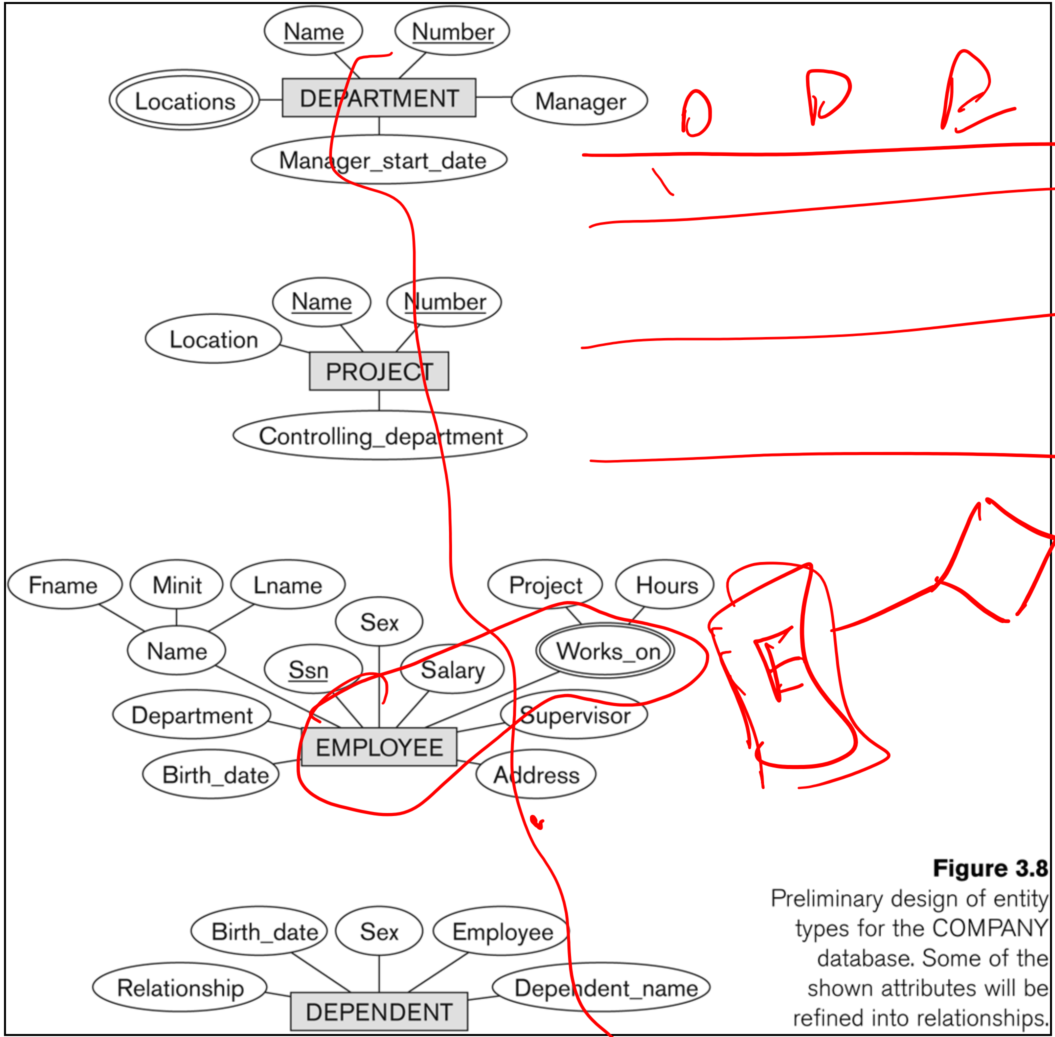


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Initial Design of Entity Types:

EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

Relationships and Relationship Types (1)

A **relationship** relates two or more distinct entities with a specific meaning.

- For example, EMPLOYEE John Smith *works on* the ProductX PROJECT, or EMPLOYEE Franklin Wong *manages* the Research DEPARTMENT.

Relationships of the same type are grouped or typed into a **relationship type**.

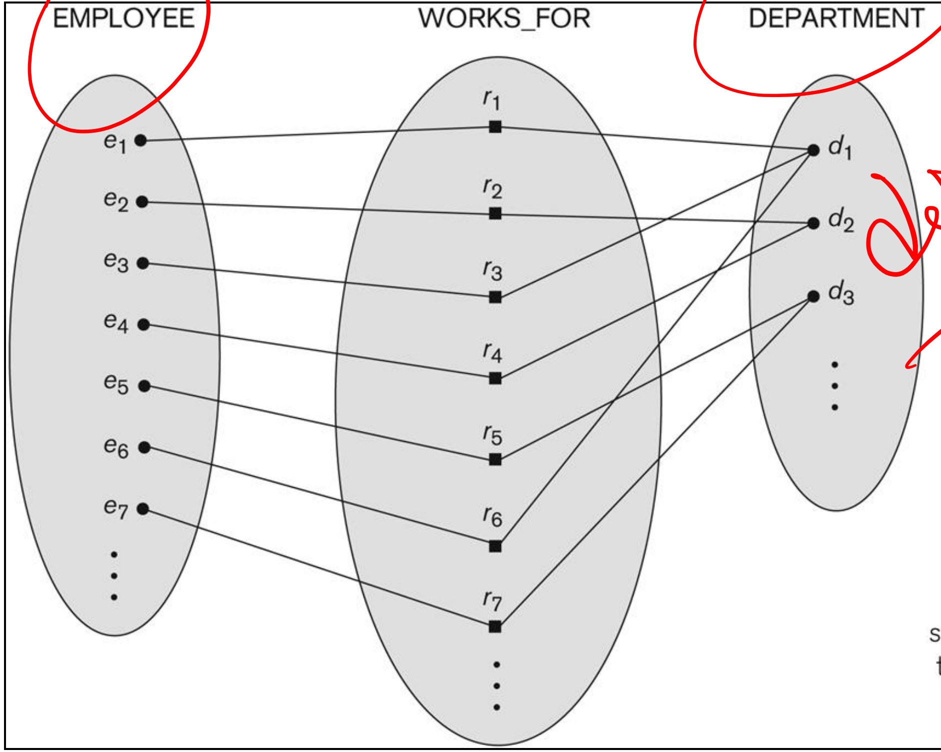
- For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.

The degree of a relationship type is the number of participating entity types.

- Both MANAGES and WORKS_ON are *binary* relationships.

Relationship instances of the WORKS_FOR N:1 relationship between EMPLOYEE and DEPARTMENT

Emp



dept

Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Relationship instances of the M:N WORKS_ON relationship between EMPLOYEE and PROJECT

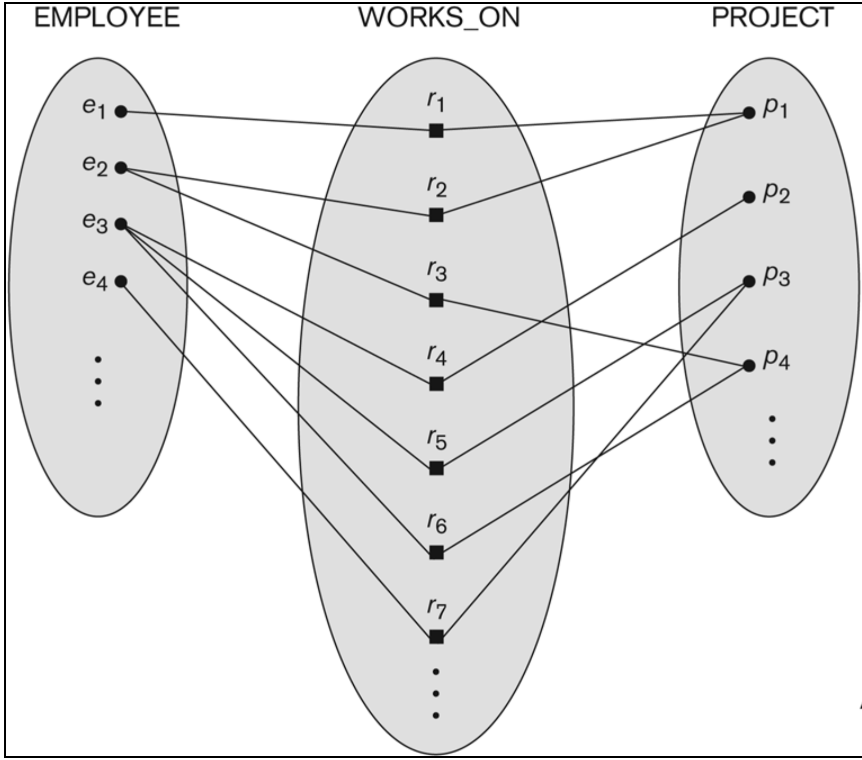


Figure 3.13
An M:N relationship, WORKS_ON.

Refining the COMPANY database schema by introducing relationships

By examining the requirements, six relationship types are identified

All are *binary* relationships (degree 2)

Listed below with their participating entity types:

- WORKS_FOR (between EMPLOYEE, DEPARTMENT)
- MANAGES (also between EMPLOYEE, DEPARTMENT)
- CONTROLS (between DEPARTMENT, PROJECT)
- WORKS_ON (between EMPLOYEE, PROJECT)
- SUPERVISION (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))
- DEPENDENTS_OF (between EMPLOYEE, DEPENDENT)

This lecture

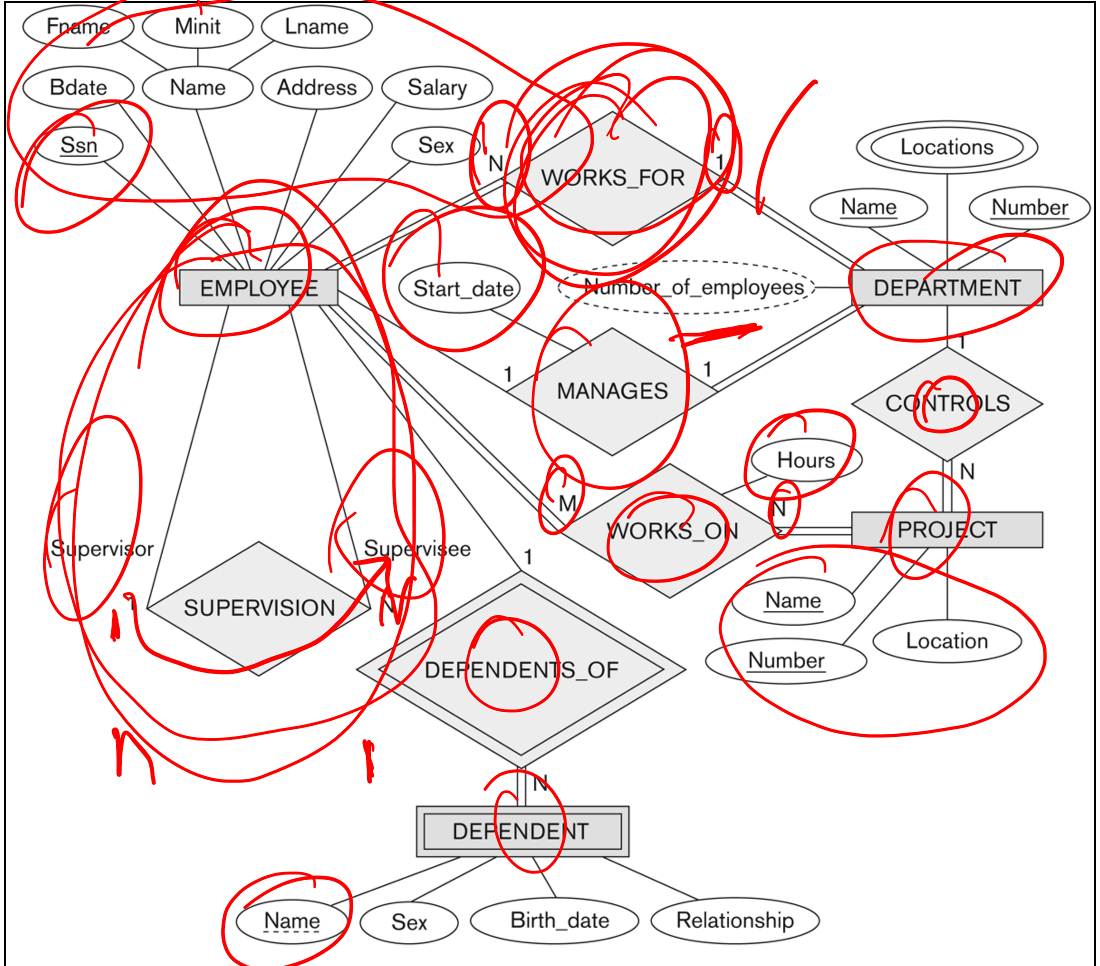


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

ER DIAGRAM – Relationship Types are:

WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF

Discussion on Relationship Types

In the refined design, some attributes from the initial entity types are refined into relationships:

- Manager of DEPARTMENT -> MANAGES
- Works_on of EMPLOYEE -> WORKS_ON
- Department of EMPLOYEE -> WORKS_FOR
- etc

In general, more than one relationship type can exist between the same participating entity types

- MANAGES and WORKS_FOR are distinct relationship types between EMPLOYEE and DEPARTMENT
- Different meanings and different relationship instances.

Constraints on Relationships

Constraints on Relationship Types

- (Also known as ratio constraints)
- Cardinality Ratio (specifies *maximum* participation)
  - One-to-one (1:1)
  - One-to-many (1:N) or Many-to-one (N:1)
  - Many-to-many (M:N)
- Existence Dependency Constraint (specifies *minimum* participation) (also called participation constraint)
  - zero (optional participation, not existence-dependent)
  - one or more (mandatory participation, existence-dependent)

Many-to-one (N:1) Relationship

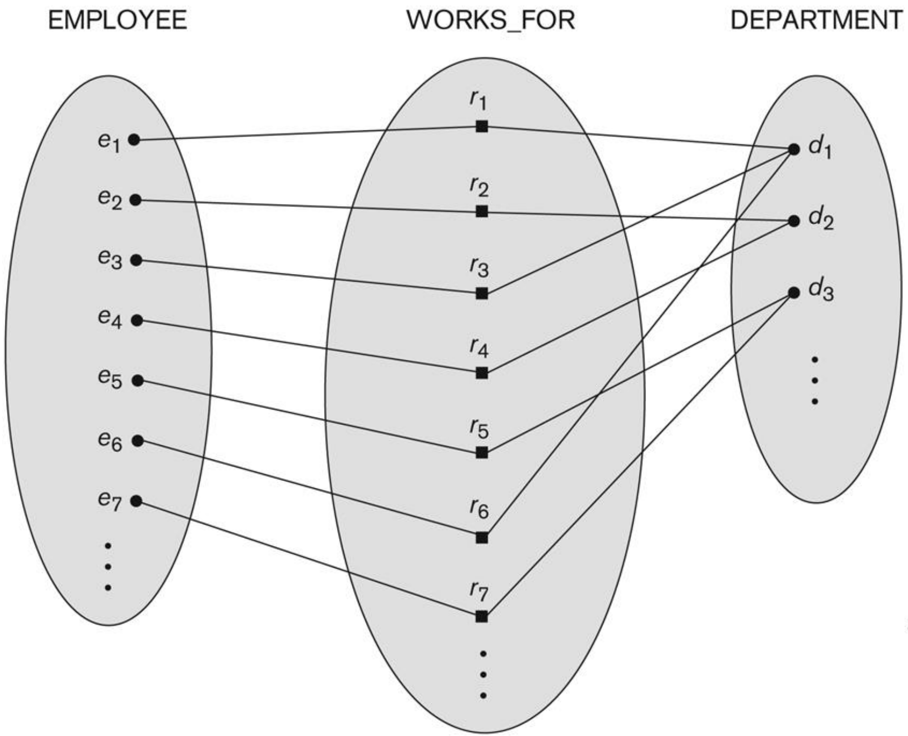


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Many-to-many (M:N) Relationship

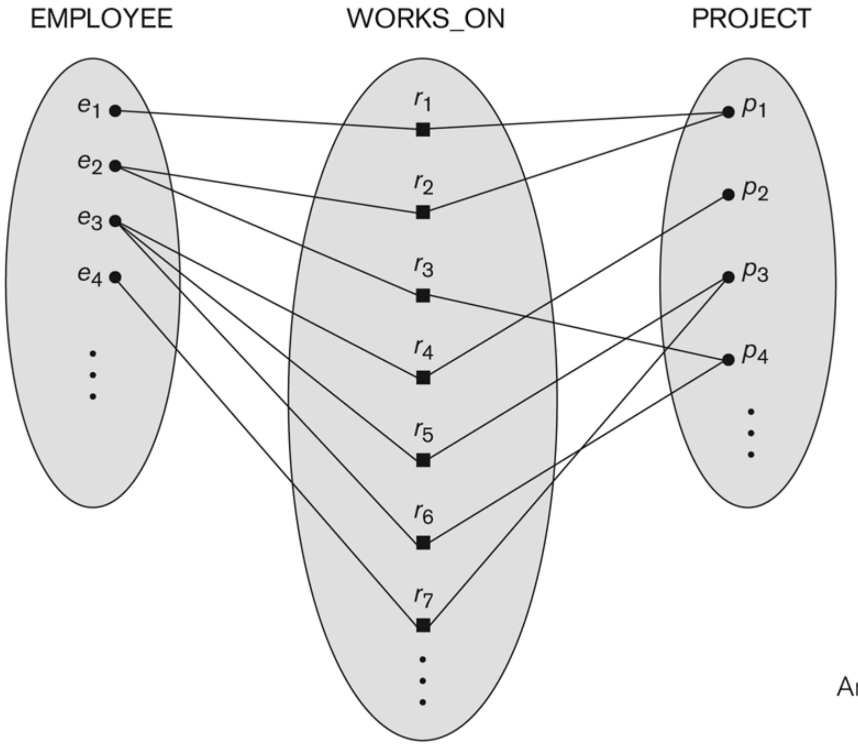


Figure 3.13
An M:N relationship, WORKS_ON.

Recursive Relationship Type

A relationship type between the same participating entity type in **distinct roles**

Also called a **self-referencing** relationship type.

Example: the SUPERVISION relationship

EMPLOYEE participates twice in two distinct roles:

- supervisor (or boss) role
- supervisee (or subordinate) role

Each relationship instance relates two distinct EMPLOYEE entities:

- One employee in *supervisor* role
- One employee in *supervisee* role

Displaying a recursive relationship

In a recursive relationship type.

- Both participations are same entity type in different roles.
- For example, SUPERVISION relationships between EMPLOYEE (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker).

In following figure, first role participation labeled with 1 and second role participation labeled with 2.

In ER diagram, need to display role names to distinguish participations.

A Recursive Relationship Supervision`

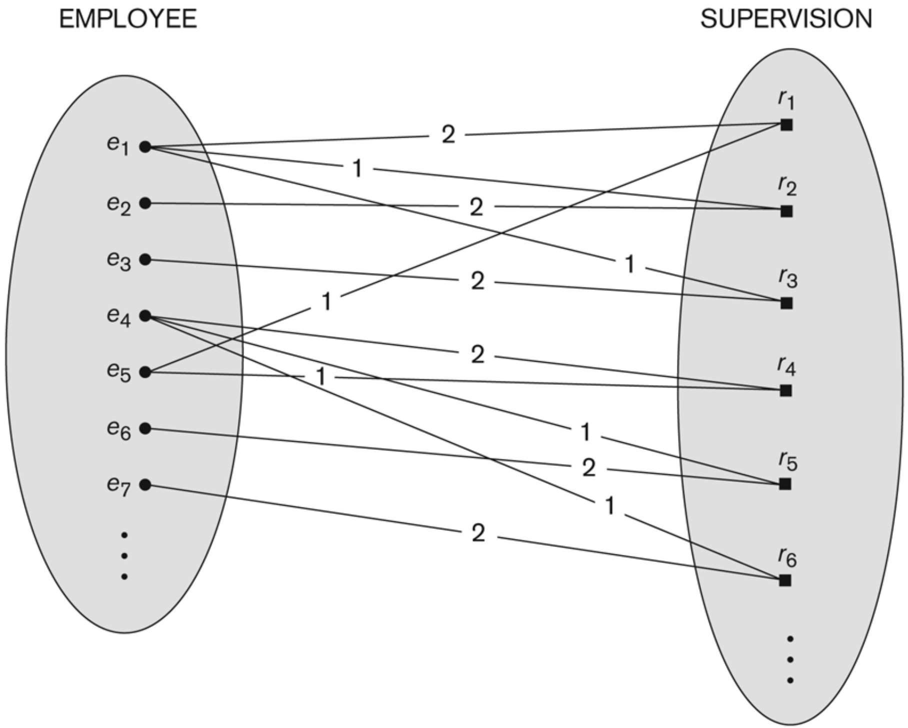


Figure 3.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

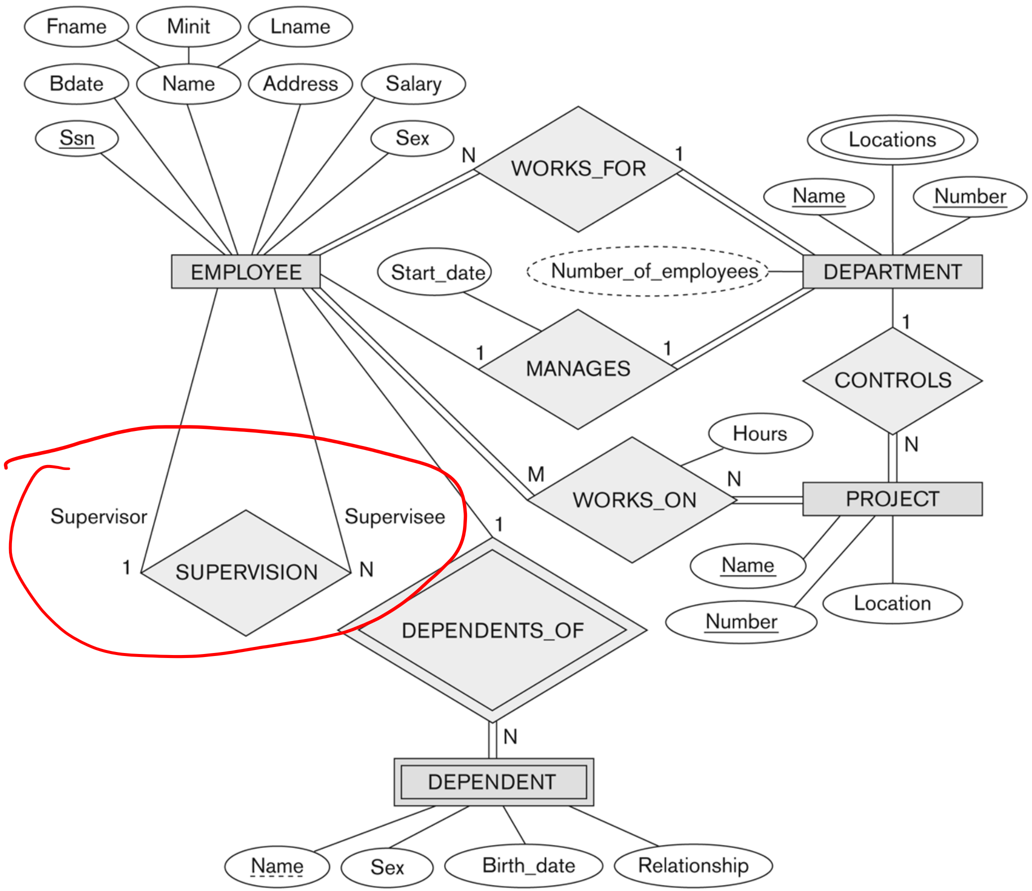


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Recursive Relationship Type is: SUPERVISION (participation role names are shown)

Weak Entity Types

An entity that does not have a key attribute and that is identification-dependent on another entity type.

A 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
- The particular entity they are related to in the identifying relationship type

Example:

- A DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE with whom the dependent is related
- Name of DEPENDENT is the *partial key*
- DEPENDENT is a *weak entity type*
- EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_OF

Attributes of Relationship types

A relationship type can have attributes:

- For example, HoursPerWeek of WORKS_ON
- Its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.
  - A value of HoursPerWeek depends on a particular (employee, project) combination
- Most relationship attributes are used with M:N relationships

Example Attribute of a Relationship Type: Hours of WORKS_ON

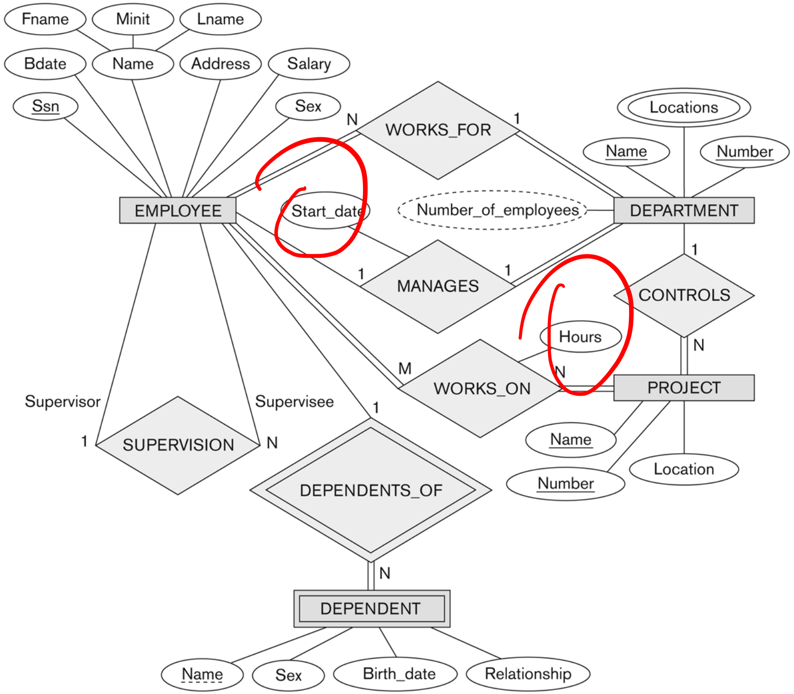


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Notation for Constraints on Relationships

Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N

Shown by placing appropriate numbers on the relationship edges.

Participation constraint (on each participating entity type): total (called existence dependency) or partial.

Total shown by double line, partial by single line.

Alternative (min, max) notation for relationship structural constraints:

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 in R

Default(no constraint): min=0, max=n (signifying no limit)

Must have min≤max, min≥0, max ≥1

Derived from the knowledge of mini-world constraints

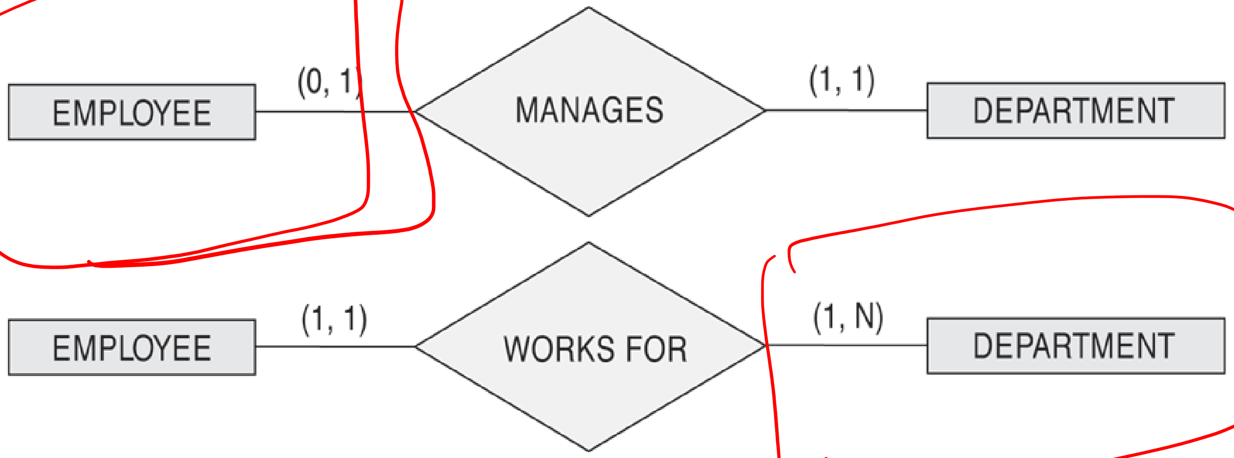
Cardinality & Participation taken together called structural constraints; (m,n); m = 0 is partial, m = 1 total

Examples:

- 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
- An employee can work for exactly one department but a department can have any number of employees.
  - Specify (1,1) for participation of EMPLOYEE in WORKS_FOR
  - Specify (0,n) for participation of DEPARTMENT in WORKS_FOR

The (min,max) notation for relationship constraints

x Nidhi
x Chaitanya



Read the min,max numbers next to the entity type and looking **away from** the entity type

COMPANY ER Schema Diagram using (min, max) notation

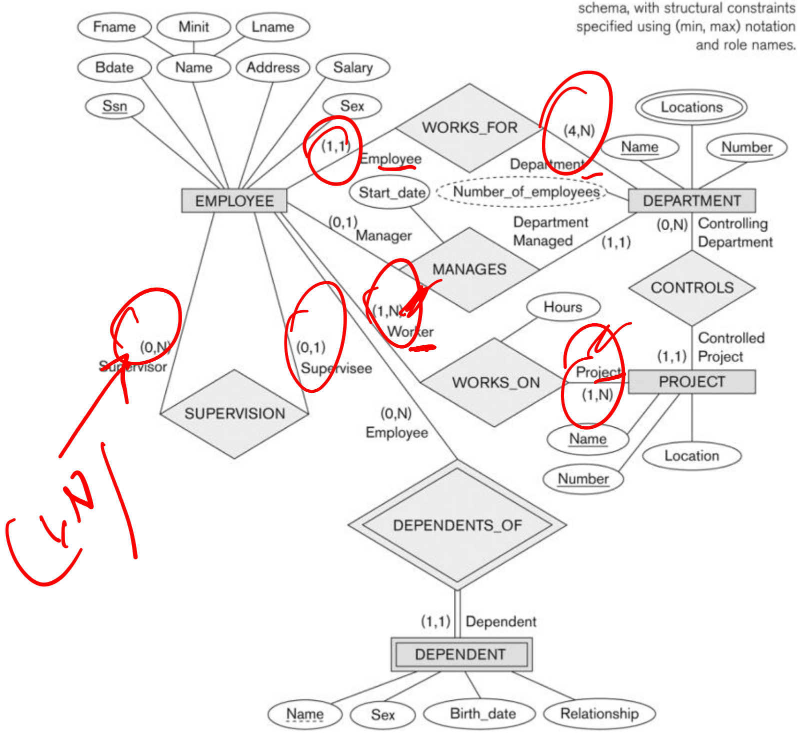


Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

Alternative diagrammatic notation

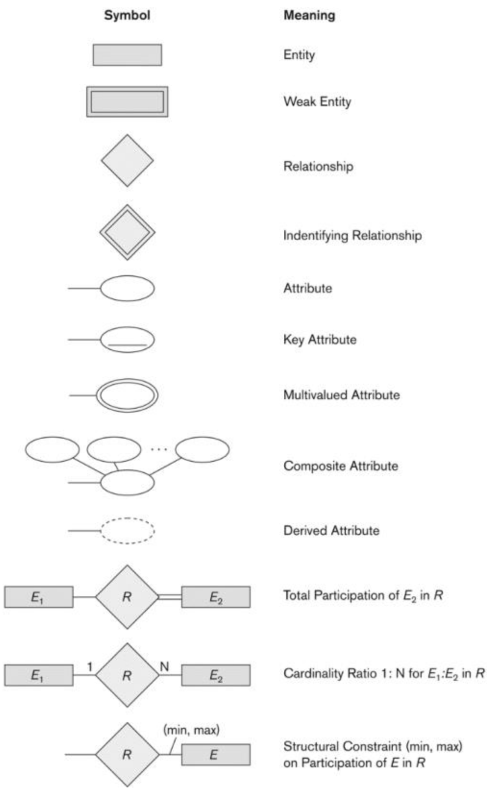
ER diagrams is one popular example for displaying database schemas

Many other notations exist in the literature and in various database design and modeling tools

UML class diagrams is representative of another way of displaying ER concepts that is used in several commercial design tools

Summary of notation for ER diagrams

Figure 3.14
Summary of the notation for ER diagrams.



UML class diagrams

Represent classes (similar to entity types) as large rounded boxes with three sections:

- Top section includes entity type (class) name
- Second section includes attributes
- Third section includes class operations (operations are not in basic ER model)

Relationships (called associations) represented as lines connecting the classes

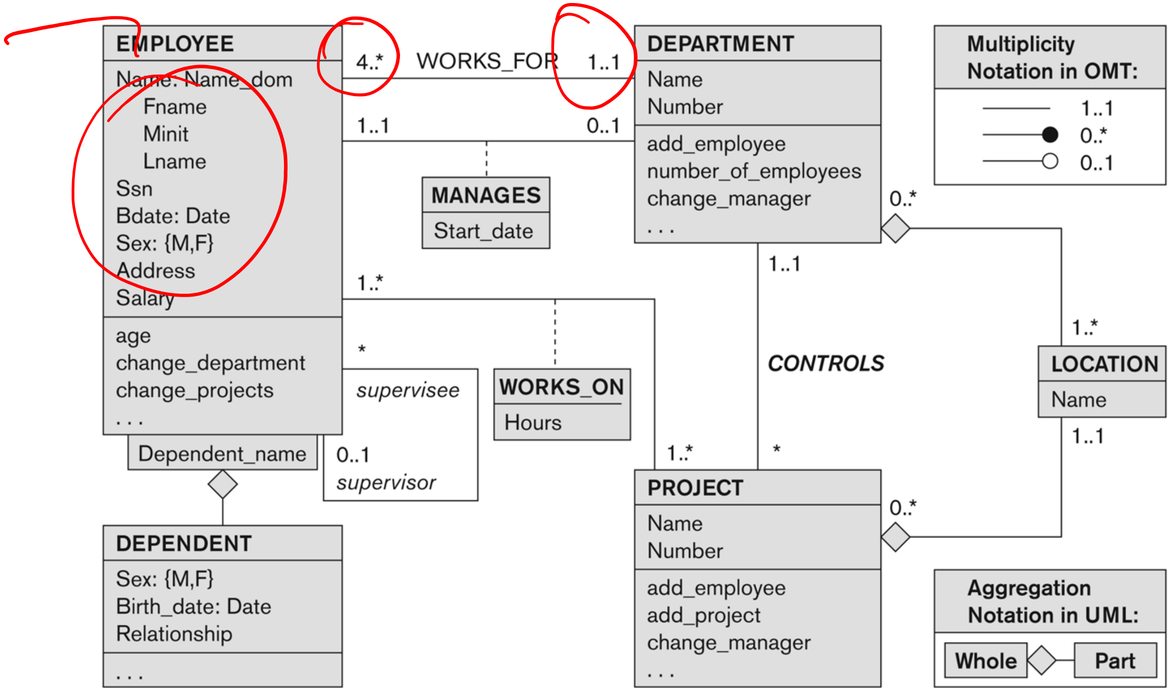
- Other UML terminology also differs from ER terminology

Used in database design and object-oriented software design

UML has many other types of diagrams for software design

UML class diagram for COMPANY database schema

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



Other alternative diagrammatic notations

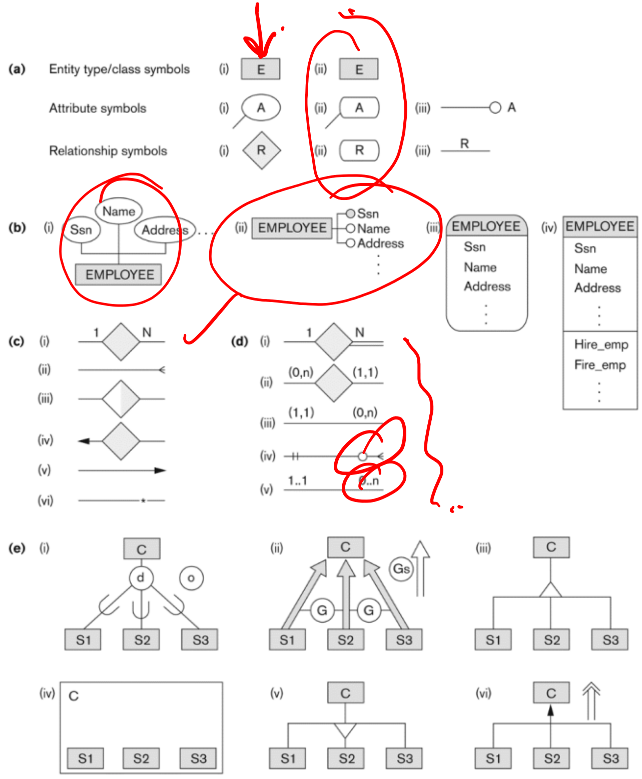


Figure A.1
Alternative notations. (a) Symbols for entity type/class, attribute, and relationship. (b) Displaying attributes. (c) Displaying cardinality ratios. (d) Various (min, max) notations. (e) Notations for displaying specialization/generalization.

Some of the Automated Database Design Tools (Note: Not all may be on the market now)

COMPANY	TOOL	FUNCTIONALITY
Embarcadero Technologies	ER Studio	Database Modeling in ER and IDEF1X
	DB Artisan	Database administration, space and security management
Oracle	Developer 2000/Designer 2000	Database modeling, application development
Popkin Software	System Architect 2001	Data modeling, object modeling, process modeling, structured analysis/design
Platinum (Computer Associates)	Enterprise Modeling Suite: Erwin, BPWin, Paradigm Plus	Data, process, and business component modeling
Persistence Inc.	Pwertier	Mapping from O-O to relational model
Rational (IBM)	Rational Rose	UML Modeling & application generation in C++/JAVA
Resolution Ltd.	Xcase	Conceptual modeling up to code maintenance
Sybase	Enterprise Application Suite	Data modeling, business logic modeling
Visio	Visio Enterprise	Data modeling, design/reengineering Visual Basic/C++

DBMS Interfaces

- Stand-alone query language interfaces
  - Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. SQL*Plus in ORACLE)
- Programmer interfaces for embedding DML in programming languages
- User-friendly interfaces
  - Menu-based, forms-based, graphics-based, etc.
- Mobile Interfaces: interfaces allowing users to perform transactions using mobile apps

DBMS Programming Language Interfaces

Programmer interfaces for embedding DML in a programming languages:

- **Embedded Approach**: e.g embedded SQL (for C, C++, etc.), SQLJ (for Java)
- **Procedure Call Approach**: e.g. JDBC for Java, ODBC (Open Databse Connectivity) for other programming languages as API's (application programming interfaces)
- **Database Programming Language Approach**: e.g. ORACLE has PL/SQL, a programming language based on SQL; language incorporates SQL and its data types as integral components
- **Scripting Languages:** PHP (client-side scripting) and Python (server-side scripting) are used to write database programs.

User-Friendly DBMS Interfaces

- Menu-based (Web-based), popular for browsing on the web
- Forms-based, designed for naïve users used to filling in entries on a form
- Graphics-based
  - Point and Click, Drag and Drop, etc.
  - Specifying a query on a schema diagram
- Natural language: requests in written English
- Combinations of the above:
  - For example, both menus and forms used extensively in Web database interfaces

Other DBMS Interfaces

- Natural language: free text as a query
- Speech: Input query and Output response
- Web Browser with keyword search
- Parametric interfaces, e.g., bank tellers using function keys.
- Interfaces for the DBA:
  - Creating user accounts, granting authorizations
  - Setting system parameters
  - Changing schemas or access paths

Database System Utilities

To perform certain functions such as:

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

Typical DBMS Component Modules

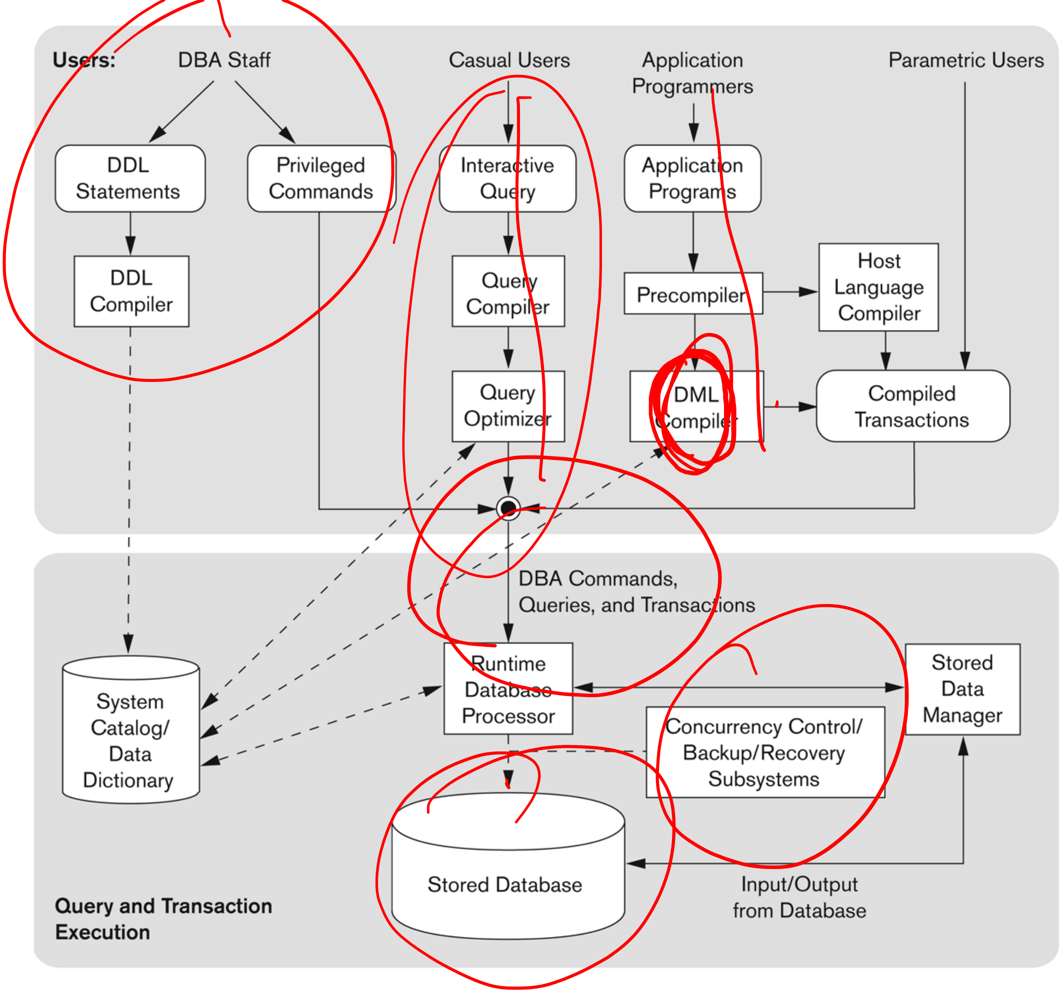


Figure 2.3
Component modules of a DBMS and their interactions.

Centralized and Client-Server DBMS Architectures

Centralized DBMS:

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

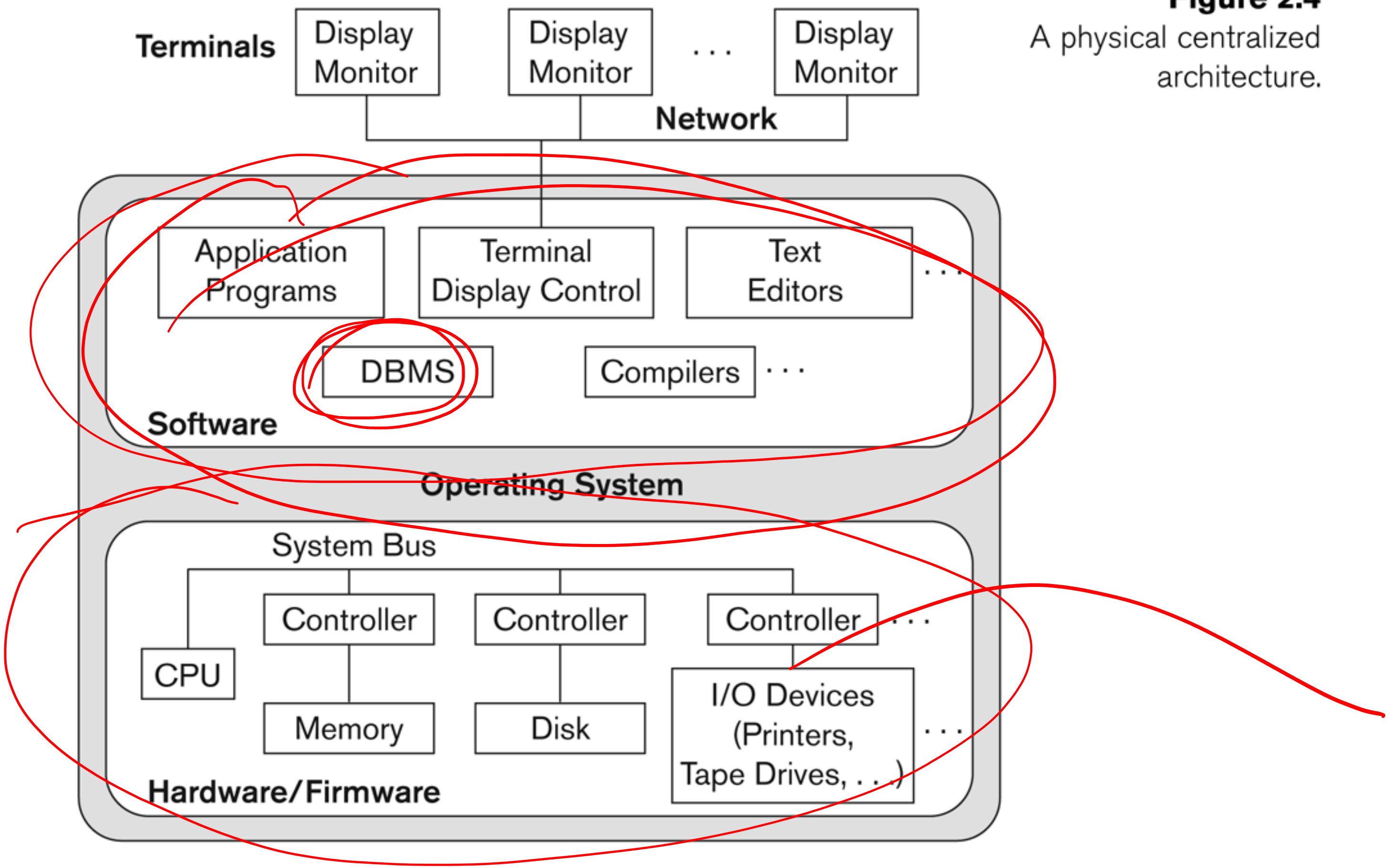
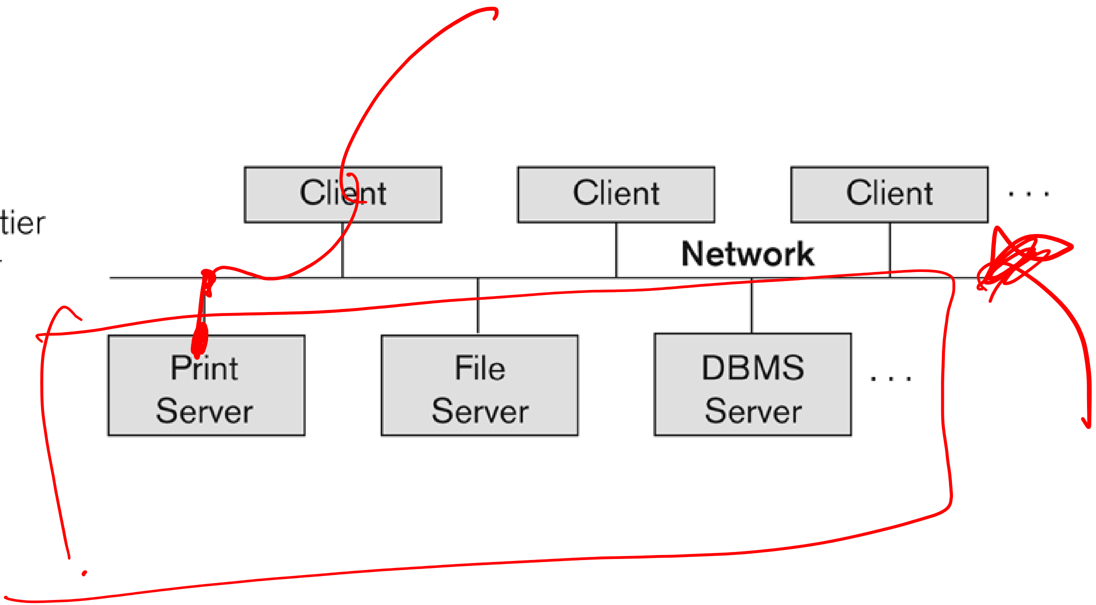


Figure 2.4
A physical centralized architecture.

A Physical Centralized Architecture

Logical two-tier client server architecture

Figure 2.5
Logical two-tier client/server architecture.



Clients

Provide appropriate interfaces through a client software module to access and utilize the various server resources.

Clients may be diskless machines or PCs or Workstations with disks with only the client software installed.

Connected to the servers via some form of a network.

(LAN: local area network, wireless network, etc.)

DBMS Server

Provides database query and transaction services to the clients

Relational DBMS servers are often called SQL servers, query servers, or transaction servers

Applications running on clients utilize an Application Program Interface (**API**) to access server databases via standard interface such as:

- ODBC: Open Database Connectivity standard
- JDBC: for Java programming access

Two Tier Client-Server Architecture

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

Three Tier Client-Server Architecture

Common for Web applications

Intermediate Layer called Application Server or Web Server:

- Stores the web connectivity software and the business logic part of the application used to access the corresponding data from the database server
- Acts like a conduit for sending partially processed data between the database server and the client.

Three-tier Architecture can enhance security:

- Database server only accessible via middle tier
- Clients cannot directly access database server
- Clients contain user interfaces and Web browsers
- The client is typically a PC or a mobile device connected to the Web

Three-tier client-server architecture

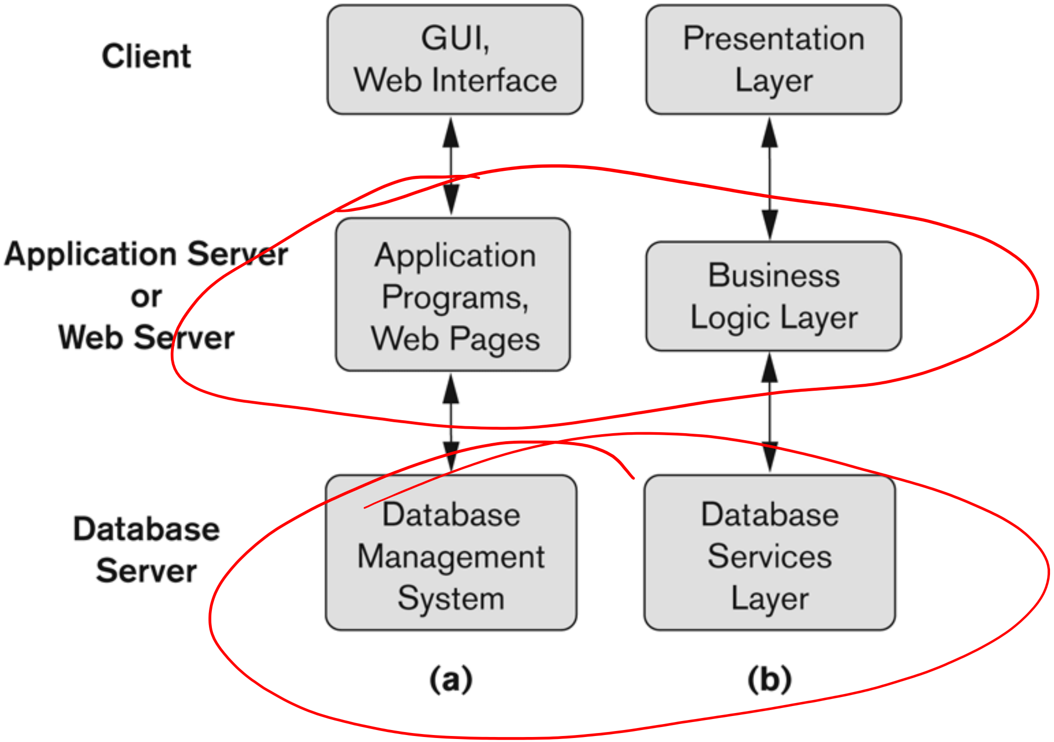


Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

The Relational Data Model and Relational Database Constraints

Relational Model Concepts

The relational Model of Data is based on the concept of a *Relation*

- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations

We review the essentials of the *formal relational model* in this module

In *practice*, there is a *standard model* based on SQL – We will see this as next module

Relational Model Concepts

A Relation is a mathematical concept based on the ideas of sets

The model was first proposed by Dr. E.F. Codd of IBM Research in 1970 in the following paper:

- "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970

The above paper caused a major revolution in the field of database management and earned Dr. Codd the coveted ACM Turing Award

Informal Definitions

Informally, a **relation** looks like a **table** of values.

A relation typically contains a **set of rows**.

The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**

In the formal model, rows are called **tuples**

Each **column** has a column header that gives an indication of the meaning of the data items in that column

In the formal model, the column header is called an **attribute name** (or just **attribute**)

Example of a Relation

Relation Name

Attributes

STUDENT

Tuples

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

Informal Definitions

Key of a Relation:

- Each row has a value of a data item (or set of items) that uniquely identifies that row in the table
  - Called the *key*
- In the STUDENT table, SSN is the key

- Sometimes row-ids or sequential numbers are assigned as keys to identify the rows in a table
  - Called *artificial key* or *surrogate key*

Bibliography / Acknowledgements

Instructor materials from Elmasri & Navathe 7e

pk.profgiri

Ponnurangam.kumaraguru

/in/ponguru

ponguru

pk.guru@iiit.ac.in

Thank you for attending the class!!!