

CS4.301 Data & Applications

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



pk.profgiri



/in/ponguru



@ponguru



Ponnurangam.kumaraguru

Data Models

Data Model:

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

Data Model Structure and Constraints:

Constructs are used to define the database structure

Constructs typically include ***elements*** (and their ***data types***) as well as groups of elements (e.g. ***entity, record, table***), and ***relationships*** among such groups

Constraints specify some restrictions on valid data; these constraints must be enforced at all times

Data Models (continued)

Data Model Operations:

These operations are used for specifying database *retrievals* and *updates* by referring to the constructs of the data model.

Operations on the data model may include ***basic model operations*** (e.g. generic insert, delete, update) and ***user-defined operations*** (e.g. compute_student_gpa, update_inventory)

Categories of Data Models

Conceptual (high-level, semantic) data models:

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

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

Physical (low-level, internal) data models:

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

Implementation (representational) data models:

Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).

Self-Describing Data Models:

Combine the description of data with the data values. Examples include XML, key-value stores and some NOSQL systems.

Schemas versus Instances

Database Schema:

The ***description*** of a database.

Includes descriptions of the database structure, data types, and the constraints on the database.

Schema Diagram:

An ***illustrative*** display of (most aspects of) a database schema.

Schema Construct:

A ***component*** of the schema or an object within the schema, e.g., STUDENT, COURSE.

Database Schema vs. Database State

Database State:

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

Initial Database State:

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

Valid State:

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

Distinction

The **database schema** changes very infrequently.

The **database state** changes every time the database is updated.

Example of a Database Schema

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
----------------	--------------------	-------

Figure 2.1

Schema diagram for the database in Figure 1.2.

Example of a database state

COURSE

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

SECTION

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

GRADE_REPORT

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

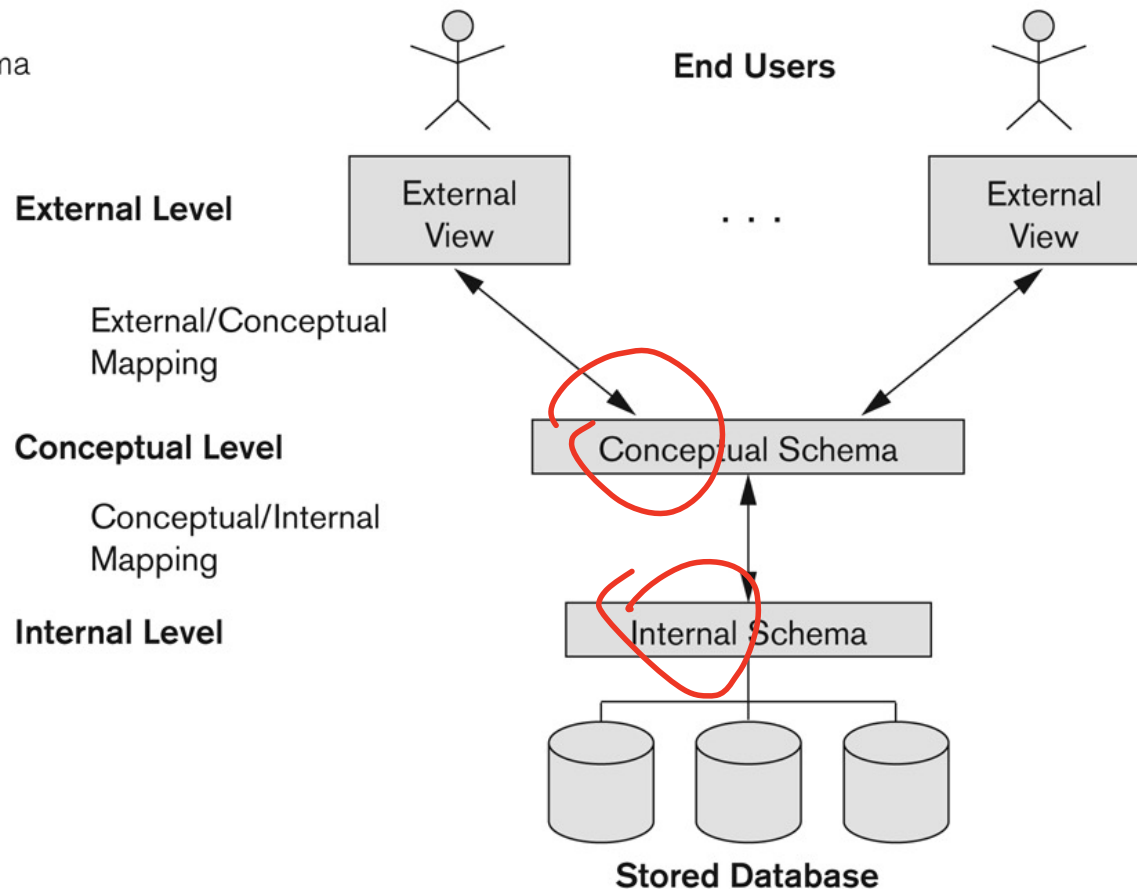
PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2

A database that stores student and course information.

Figure 2.2
The three-schema architecture.



The three-
schema
architecture

DBMS Languages

Data Definition Language (DDL):

Used by the DBA and database designers to specify the conceptual schema of a database.

In many DBMSs, the DDL is also used to define internal and external schemas (views).

In some DBMSs, separate **storage definition language (SDL)** and **view definition language (VDL)** are used to define internal and external schemas.

SDL is typically realized via DBMS commands provided to the DBA and database designers

DBMS Languages

Data Manipulation Language (DML):

Used to specify database retrievals and updates

DML commands (data sublanguage) can be *embedded* in a general-purpose programming language (host language), such as COBOL, C, C++, or Java.

A library of functions can also be provided to access the DBMS from a programming language. Alternatively, stand-alone DML commands can be applied directly (called a *query language*).

Types of DML

High Level or Non-procedural Language:

For example, the SQL relational language

Are “set”-oriented and specify what data to retrieve rather than how to retrieve it.

Also called **declarative** languages.

Low Level or Procedural Language:

Retrieve data one record-at-a-time;

Constructs such as looping are needed to retrieve multiple records, along with positioning pointers.

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

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

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

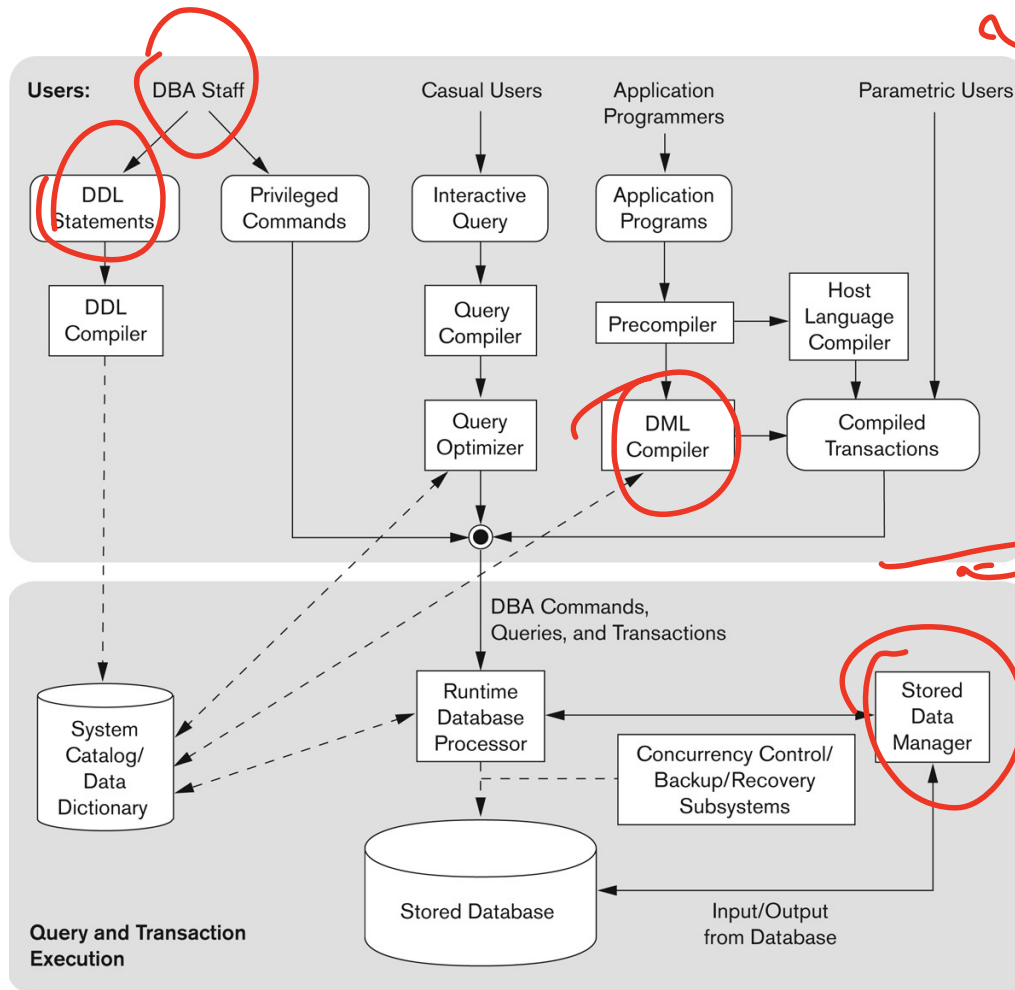


Figure 2.3
Component modules of a DBMS and their interactions.

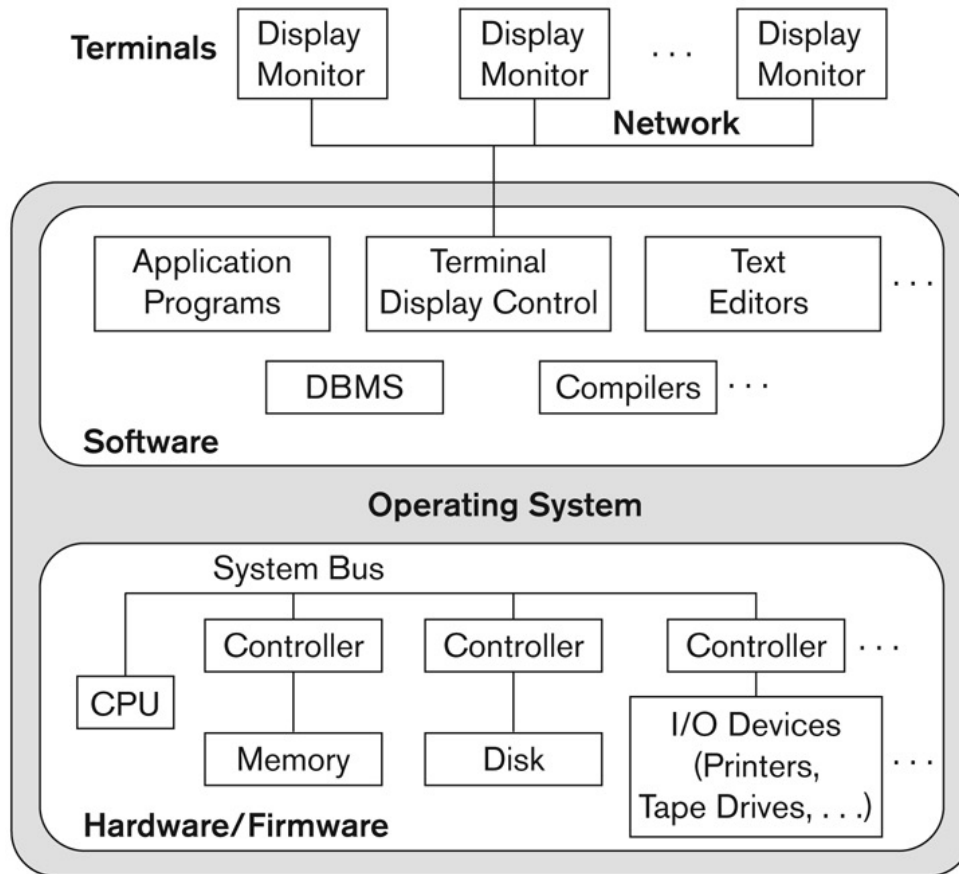
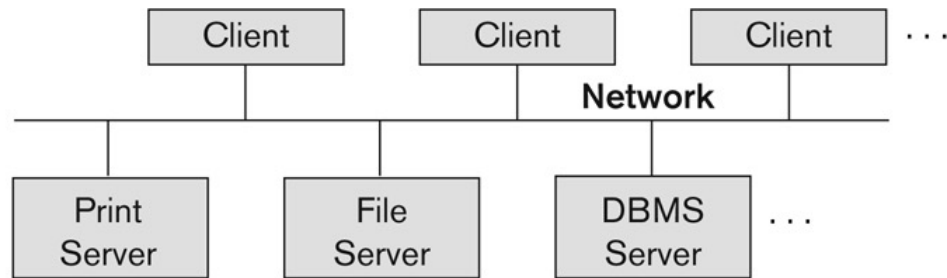


Figure 2.4
A physical centralized architecture.

A Physical
Centralized
Architecture

Logical two-tier client server architecture

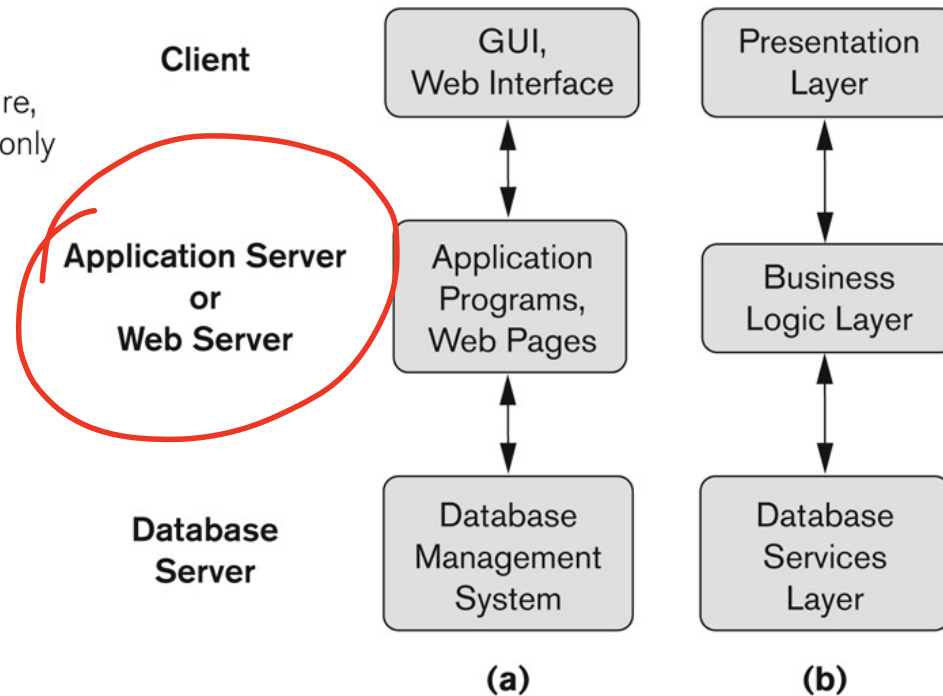
Figure 2.5
Logical two-tier
client/server
architecture.



Three-tier client-server architecture

Figure 2.7

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



Classification of DBMSs

Based on the data model used

Legacy: Network, Hierarchical.

Currently Used: Relational, Object-oriented, Object-relational

Recent Technologies: Key-value storage systems, NOSQL systems: document based, column-based, graph-based and key-value based. Native XML DBMSs.

Other classifications

Single-user (typically used with personal computers)
vs. multi-user (most DBMSs).

Centralized (uses a single computer with one database) vs. distributed (multiple computers, multiple DBs)

Variations of Distributed DBMSs (DDBMSs)

Homogeneous DDBMS

Heterogeneous DDBMS

Federated or Multidatabase Systems

Participating Databases are loosely coupled with high degree of autonomy.

Distributed Database Systems have now come to be known as client-server based database systems because:

They do not support a totally distributed environment, but rather a set of database servers supporting a set of clients.

Cost considerations for DBMSs

Cost Range: from free open-source systems to configurations costing millions of dollars

Examples of free relational DBMSs: MySQL, PostgreSQL, others

Commercial DBMS offer additional specialized modules, e.g. time-series module, spatial data module, document module, XML module

- These offer additional specialized functionality when purchased separately

- Sometimes called cartridges (e.g., in Oracle) or blades

Different licensing options: site license, maximum number of concurrent users (seat license), single user, etc.

This Lecture

Administrativa

Tutorial: 1st Oct 1130 – 1300hrs, H205 – How many attended? How did it go?

1st Quiz: 10th Oct

Email ID for any questions: dna-m22-tas@googlegroups.com

Data Modeling Using the Entity-Relationship (ER) Model

What we will cover?

Overview of Database Design Process

Example Database Application (COMPANY)

ER Model Concepts

- Entities and Attributes

- Entity Types, Value Sets, and Key Attributes

- Relationships and Relationship Types

- Weak Entity Types

- Roles and Attributes in Relationship Types

ER Diagrams - Notation

ER Diagram for COMPANY Schema

Alternative Notations – UML class diagrams

Overview of Database Design Process

Two main activities:

- Database design

- Applications design

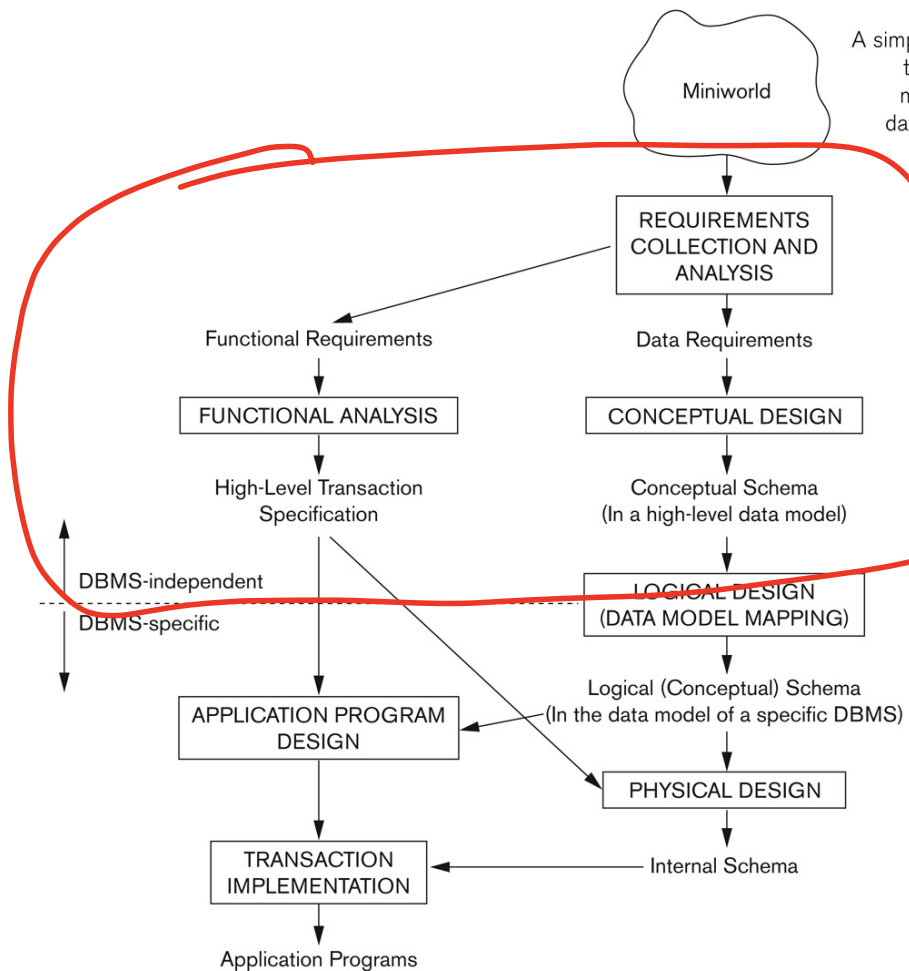
Focus in this chapter on conceptual database design

- To design the conceptual schema for a database application

Applications design focuses on the programs and interfaces that access the database

- Generally considered part of software engineering

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



Overview of Database Design Process

Methodologies for Conceptual Design

Entity Relationship (ER) Diagrams (This Chapter)

Enhanced Entity Relationship (EER) Diagrams (Chapter 4)

Use of Design Tools in industry for designing and documenting large scale designs

The UML (Unified Modeling Language) Class Diagrams are popular in industry to document conceptual database designs

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).

Composition may form a hierarchy where some components are themselves composite.

Multi-valued

An entity may have multiple values for that attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT.

Denoted as {Color} or {PreviousDegrees}.

Types of Attributes (2)

In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels, although this is rare.

For example, PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees (College, Year, Degree, Field)}

Multiple PreviousDegrees values can exist

Each has four subcomponent attributes:

College, Year, Degree, Field

Example of a composite attribute

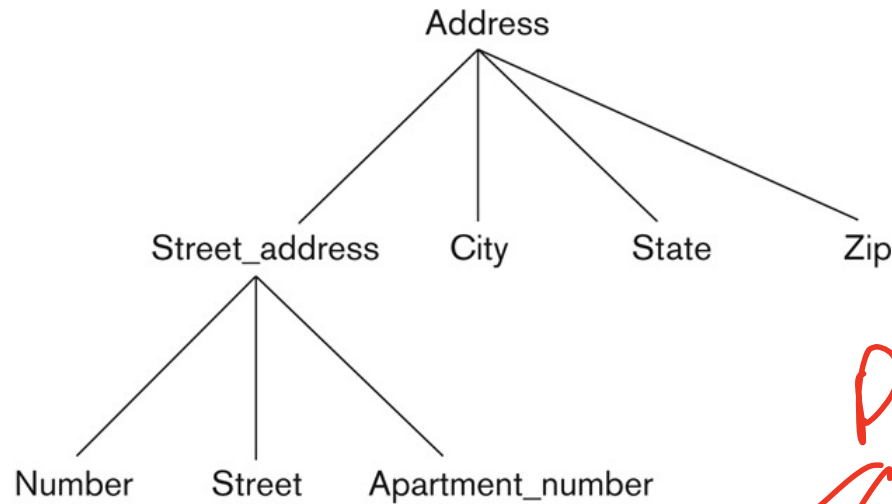


Figure 3.4
A hierarchy of
composite attributes.

Degeer

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.

Each key is underlined

Entity Set

Each entity type will have a collection of entities stored in the database

Called the **entity set** or sometimes **entity collection**

Previous slide shows three CAR entity instances in the entity set for CAR

Same name (CAR) used to refer to both the entity type and the entity set

However, entity type and entity set may be given different names

Entity set is the current *state* of the entities of that type that are stored in the database

Value Sets (Domains) of Attributes

Each simple attribute is associated with a value set

E.g., Lastname has a value which is a character string of upto 15 characters, say

Date has a value consisting of MM-DD-YYYY where each letter is an integer

A **value set** specifies the set of values associated with an attribute

Attributes and Value Sets

Value sets are similar to data types in most programming languages – e.g., integer, character (n), real, bit

Mathematically, an attribute A for an entity type E whose value set is V is defined as a function

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

Where $P(V)$ indicates a power set (which means all possible subsets) of V. The above definition covers simple and multivalued attributes.

We refer to the value of attribute A for entity e as $A(e)$.

Displaying an Entity type

In ER diagrams, an entity type is displayed in a rectangular box

Attributes are displayed in ovals

Each attribute is connected to its entity type



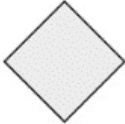



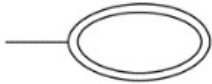
Components of a composite attribute are connected to the oval representing the composite attribute

Each key attribute is underlined

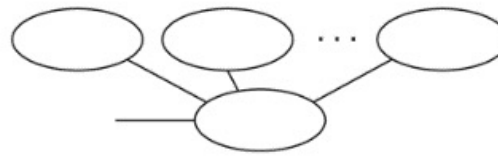
Multivalued attributes displayed in double ovals

Figure 3.14

Summary of the notation for ER diagrams.

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute

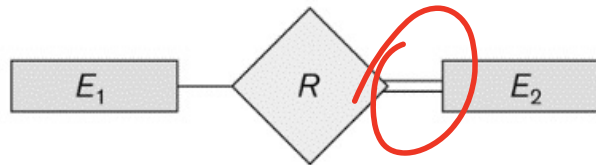




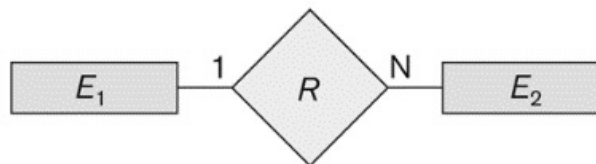
Composite Attribute



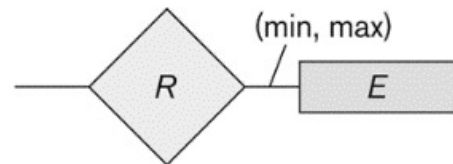
Derived Attribute



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

Entity Type CAR with two keys and a corresponding Entity Set

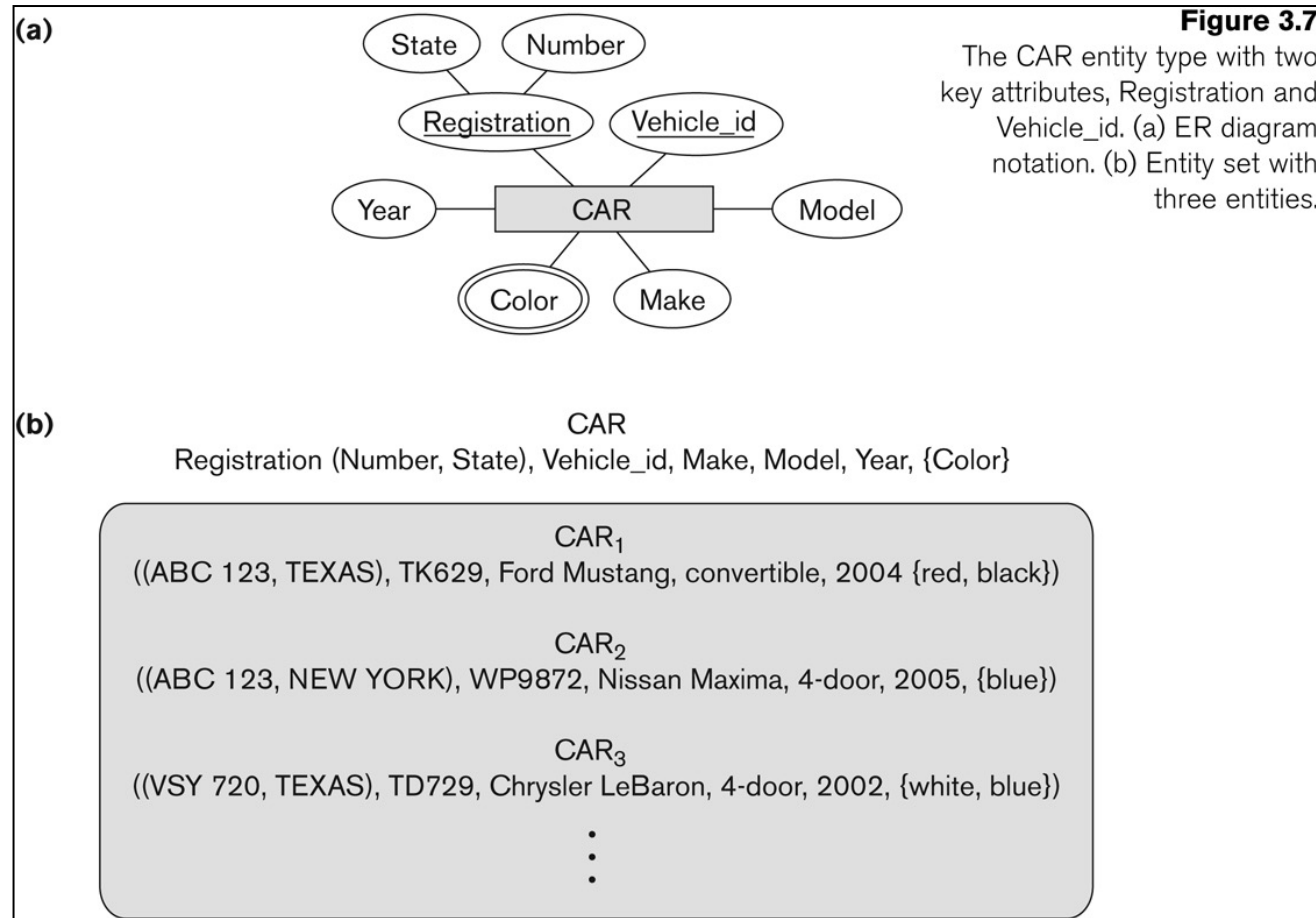


Figure 3.7

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

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

Initial Design of Entity Types:

EMPLOYEE,
DEPARTMENT, PROJECT,
DEPENDENT

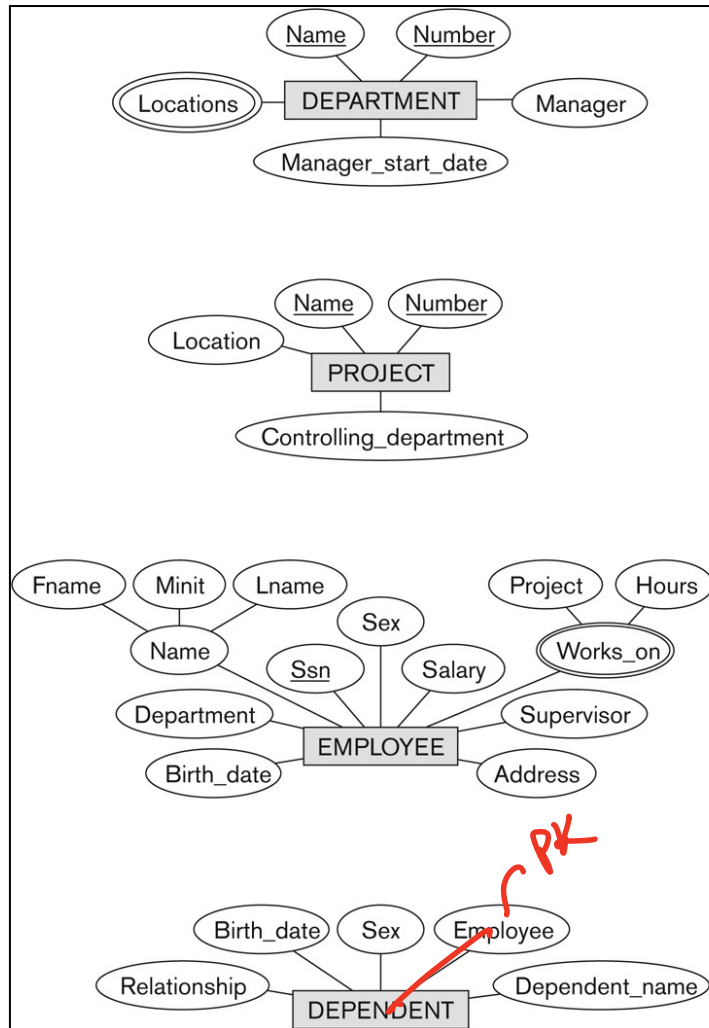


Figure 3.8
Preliminary design of entity
types for the COMPANY
database. Some of the
shown attributes will be
refined into relationships.

Refining the initial design by introducing **relationships**

The initial design is typically not complete

Some aspects in the requirements will be represented as **relationships**

ER model has three main concepts:

- Entities (and their entity types and entity sets)

- Attributes (simple, composite, multivalued)

- Relationships (and their relationship types and relationship sets)

We introduce relationship concepts next

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.

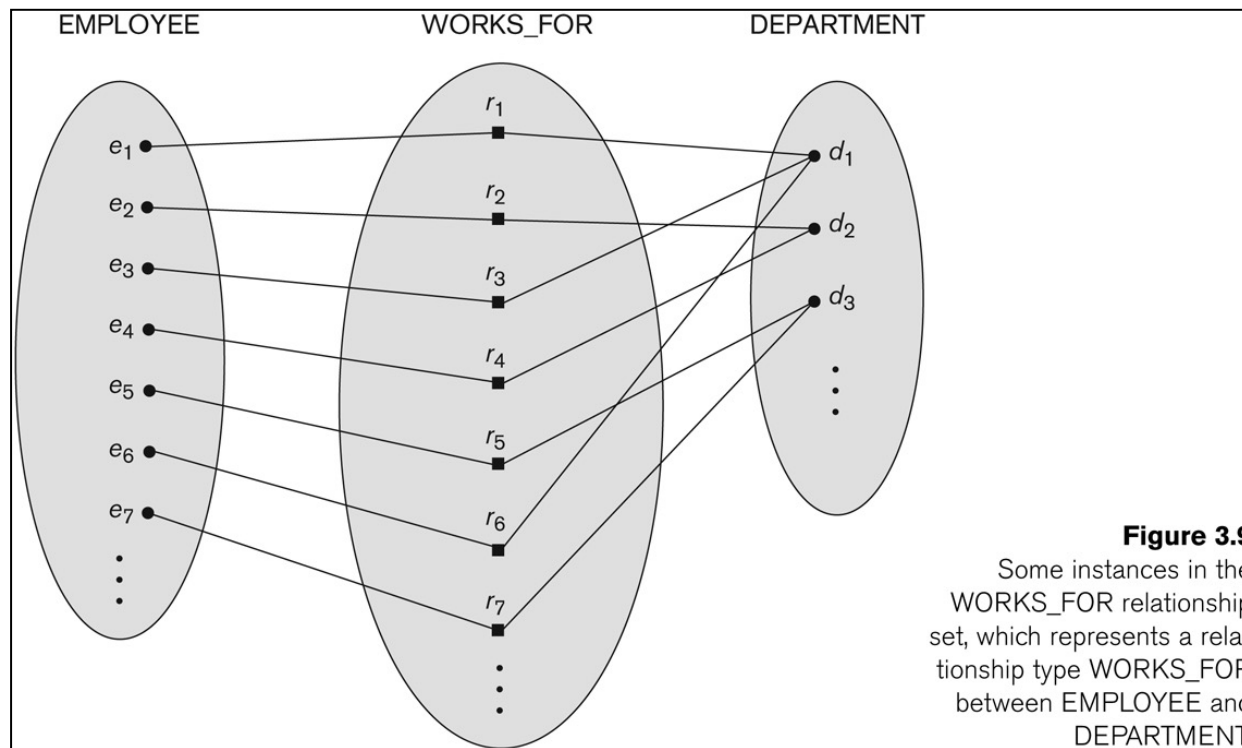
Relationships of the **same type** are grouped or typed into a **relationship type**.

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

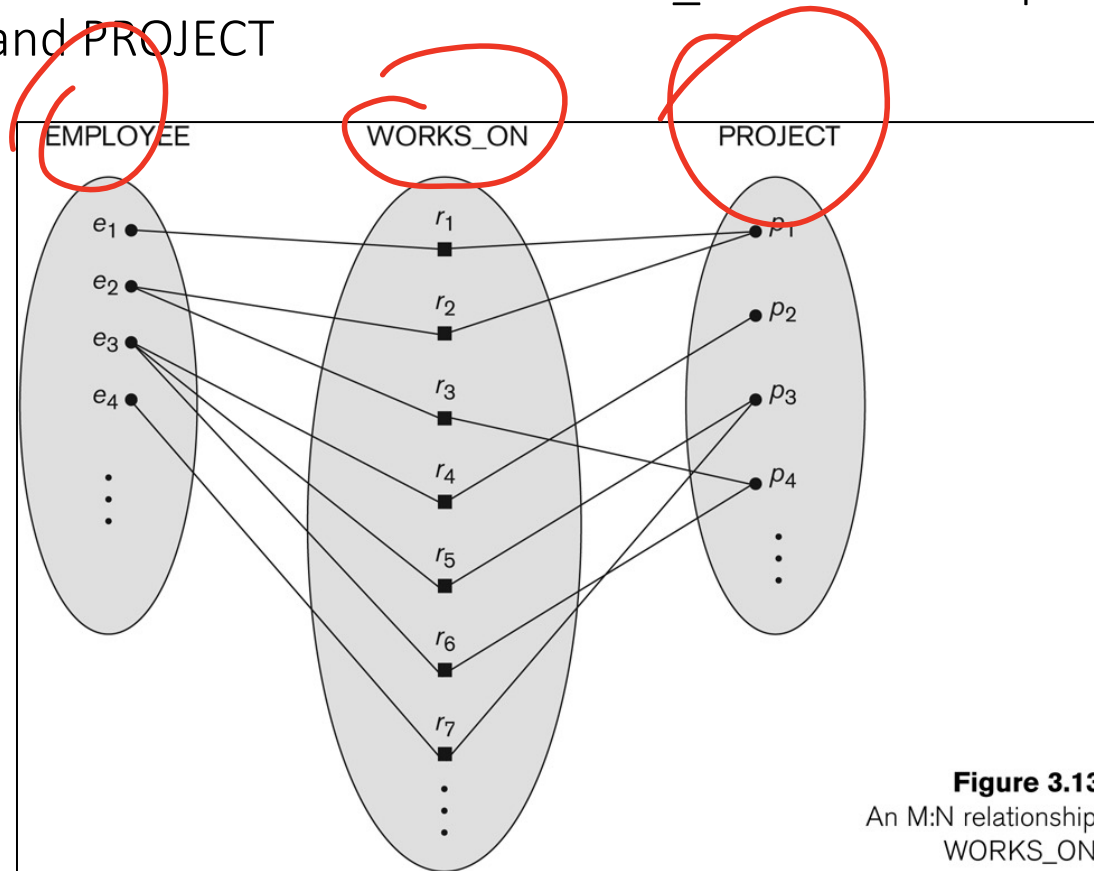
The degree of a relationship type is the number of participating entity types.

Both MANAGES and WORKS_ON are *binary* relationships.

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



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



Relationship type vs. relationship set (1)

Relationship Type:

- Is the schema description of a relationship

- Identifies the relationship name and the participating entity types

- Also identifies certain relationship constraints

Relationship Set:

- The current set of relationship instances represented in the database

- The current *state* of a relationship type

Relationship type vs. relationship set (2)

Previous figures displayed the relationship sets

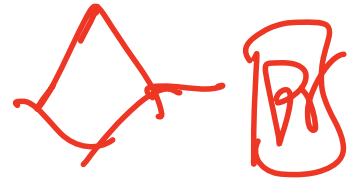
Each instance in the set relates individual participating entities – one from each participating entity type

In ER diagrams, we represent the *relationship type* as follows:

- Diamond-shaped box is used to display a relationship type

- Connected to the participating entity types via straight lines

- Note that the relationship type is not shown with an arrow. The name should be typically be readable from left to right and top to bottom.



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)

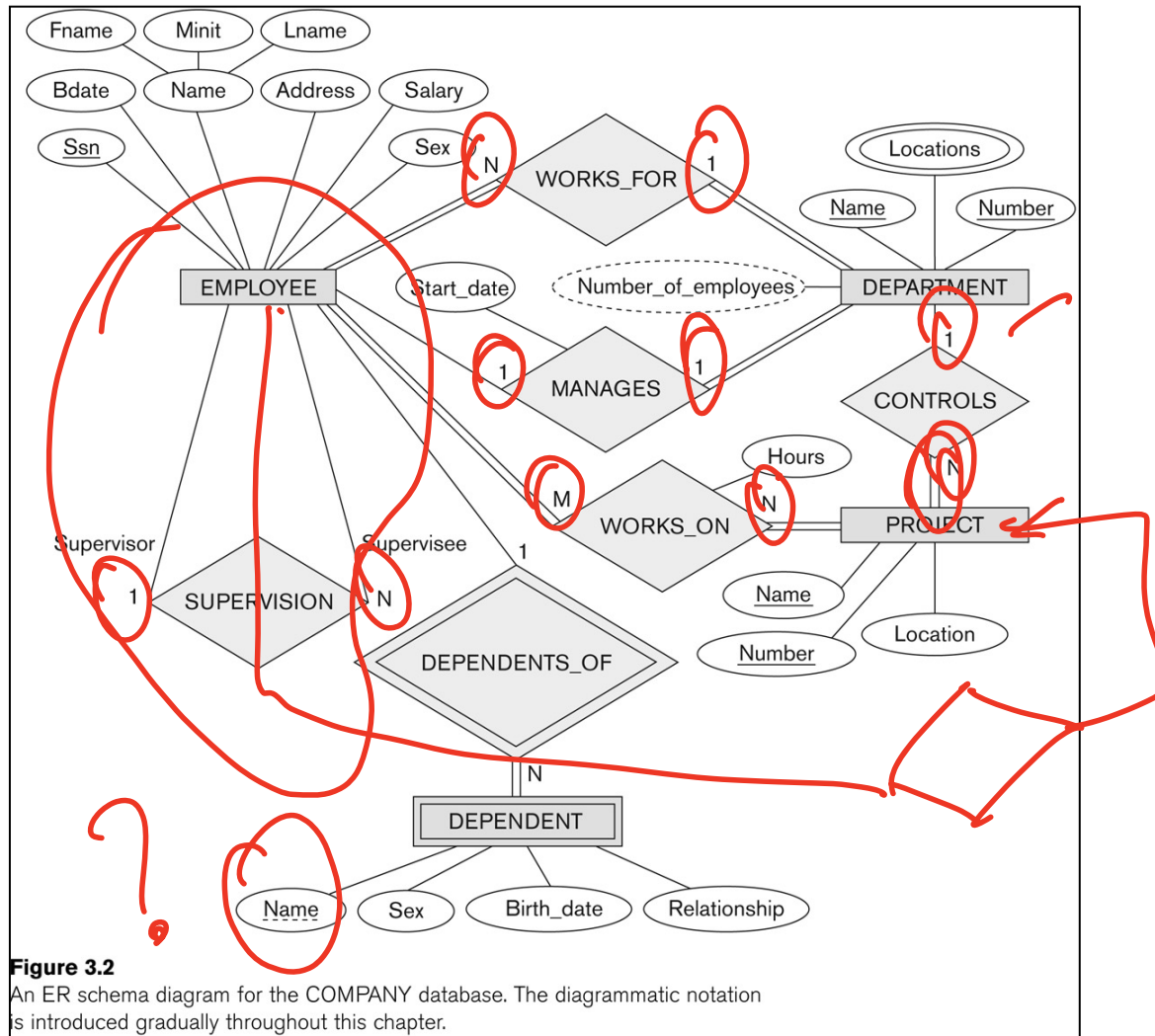
- MANAGES (also between EMPLOYEE, DEPARTMENT)

- CONTROLS (between DEPARTMENT, PROJECT)

- WORKS_ON (between EMPLOYEE, PROJECT)

- SUPERVISION (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))

- DEPENDENTS_OF (between EMPLOYEE, DEPENDENT)



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

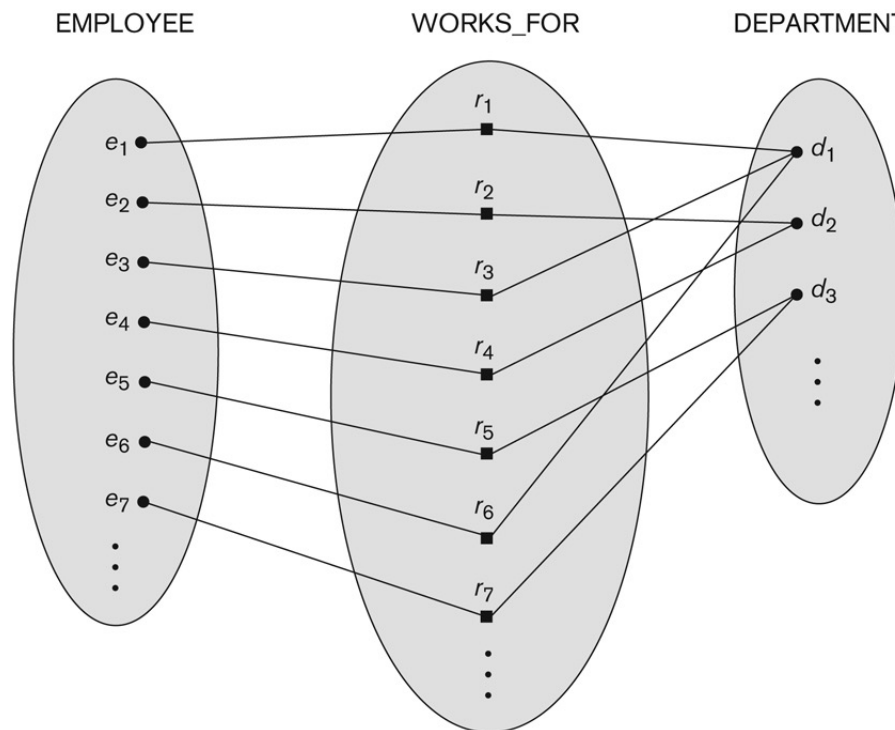


Figure 3.9

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

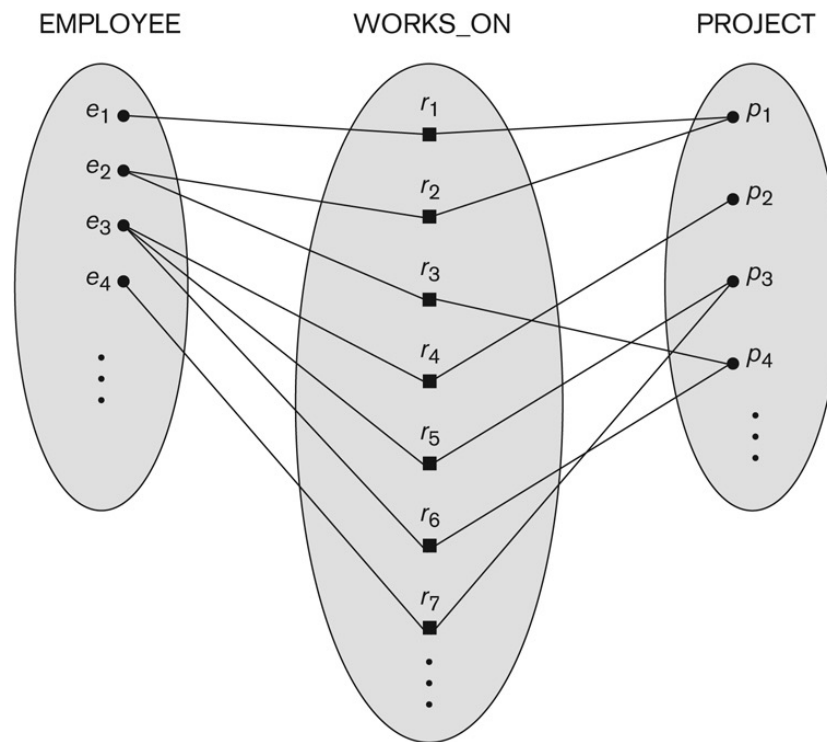


Figure 3.13
An M:N relationship,
WORKS_ON.

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`

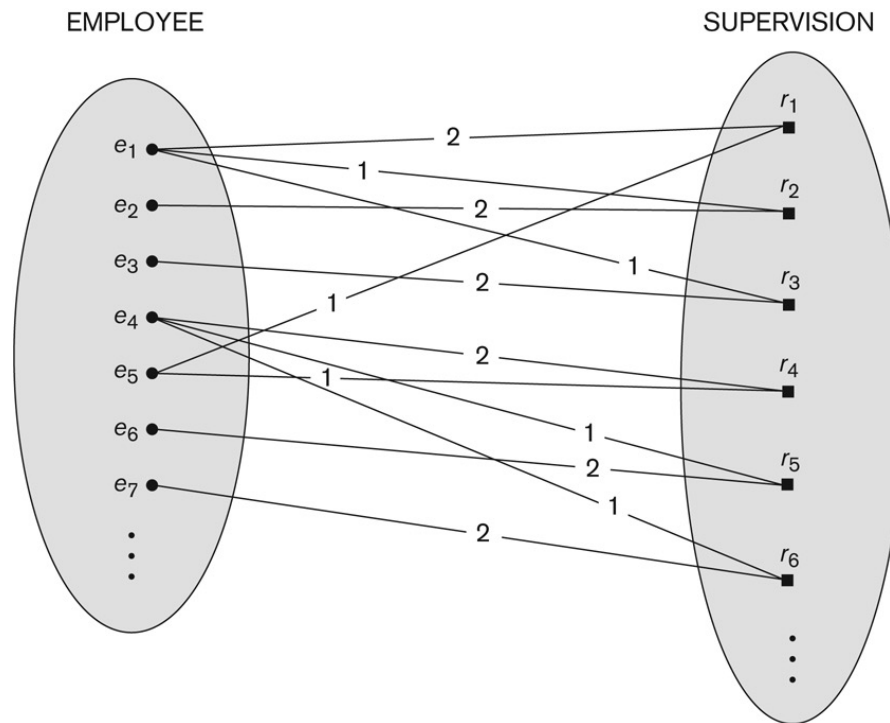


Figure 3.11

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

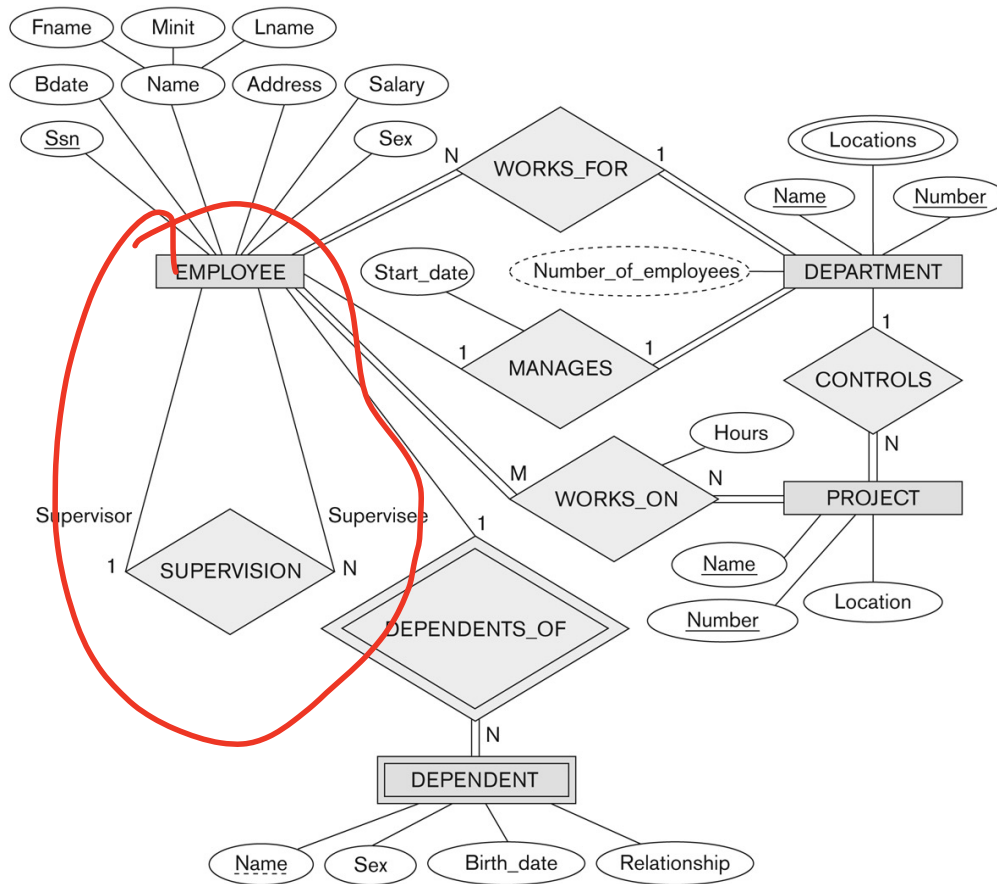


Figure 3.2

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

In 1:N relationships, they can be transferred to the entity type on the N-side of the relationship

Example Attribute of a Relationship Type: Hours of WORKS_ON

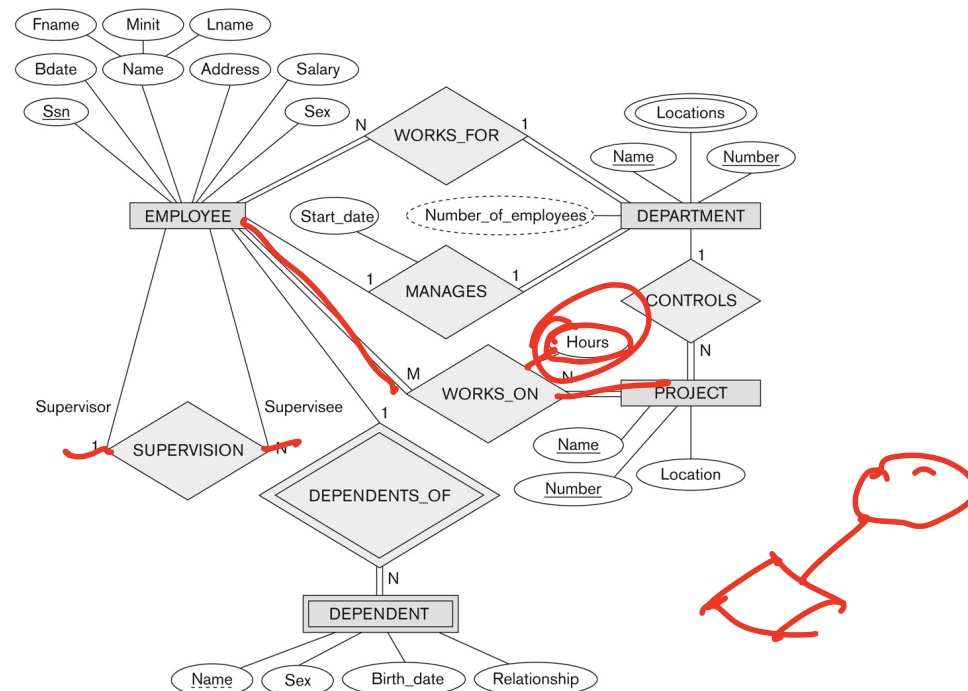


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

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.

NOTE: These are easy to specify for Binary Relationship Types.

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 $\min \leq \max$, $\min \geq 0$, $\max \geq 1$

Derived from the knowledge of mini-world constraints

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

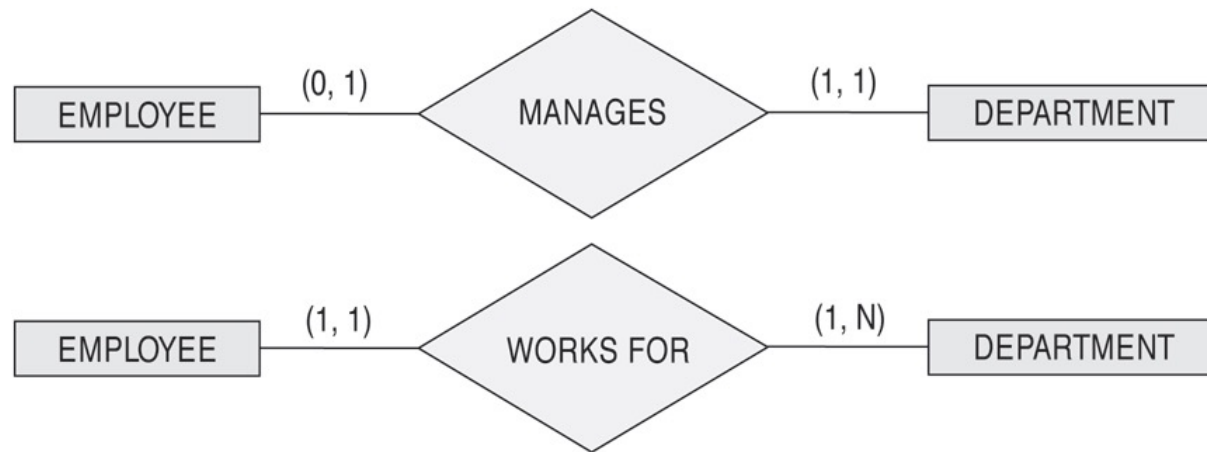
Specify (1,1) for participation of DEPARTMENT in MANAGES

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

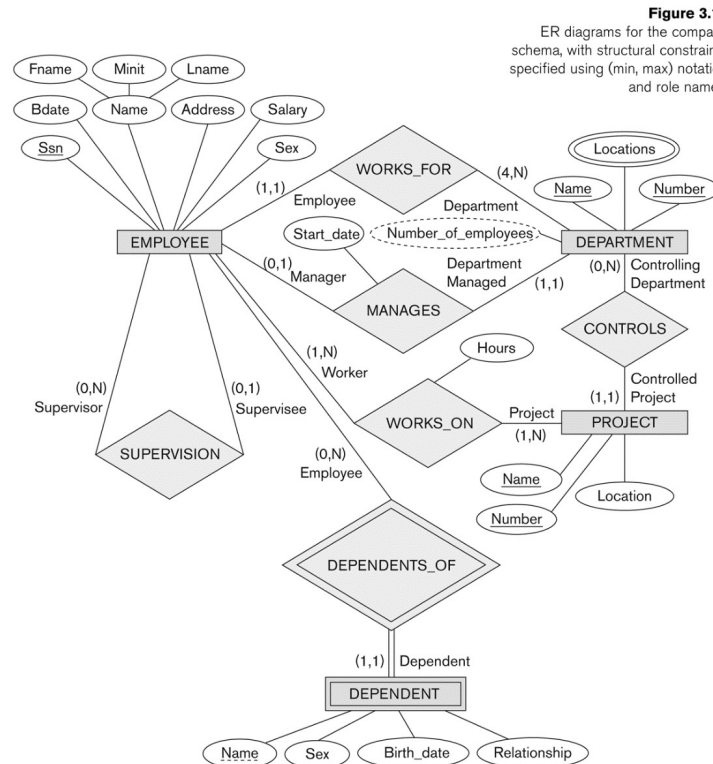
Specify (0,n) for participation of DEPARTMENT in WORKS_FOR

The (min,max) notation for relationship constraints



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



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

Appendix A illustrates some of the alternative notations that have been used

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.

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	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

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

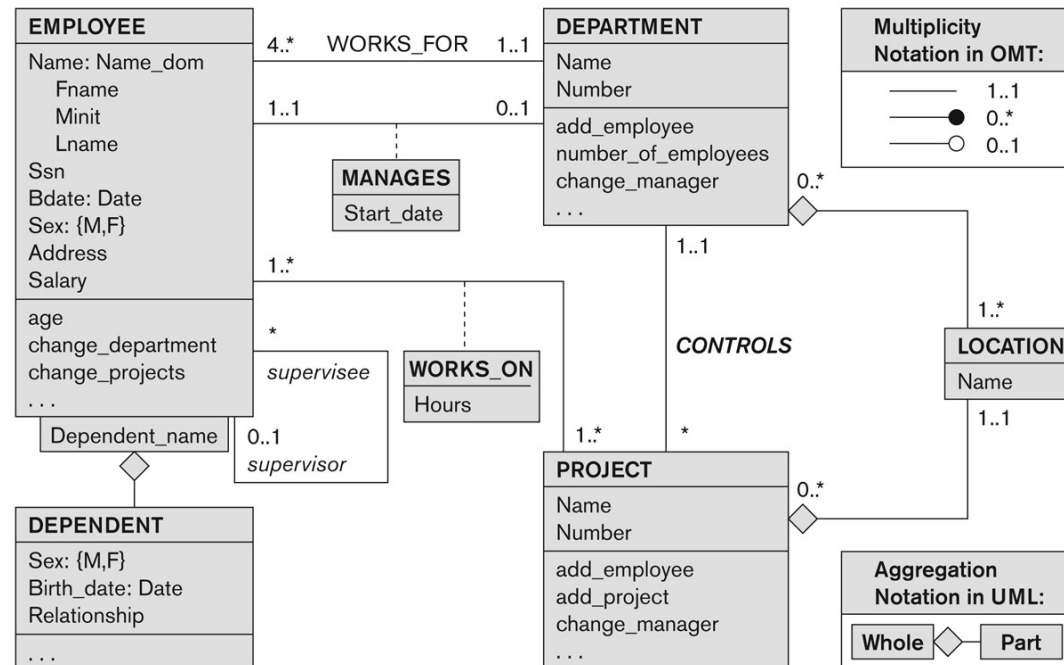
Used in database design and object-oriented software design

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

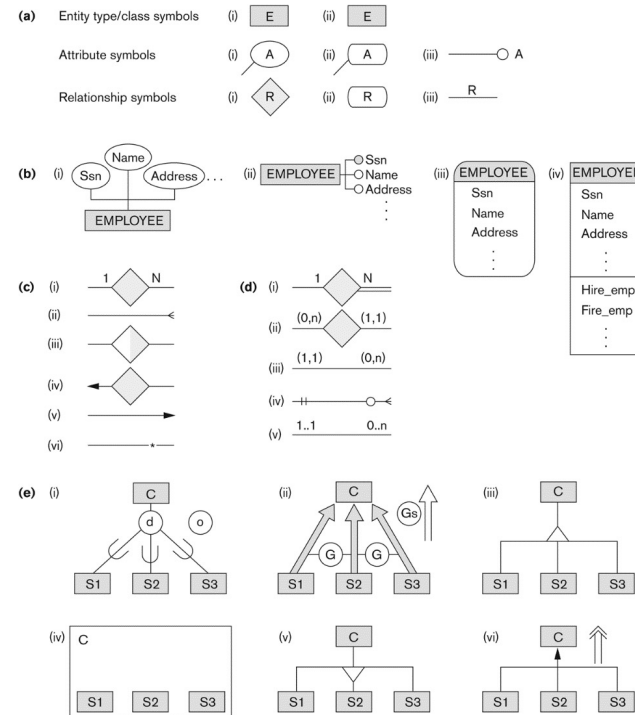


Figure A.1
 Alternative notations. (a) Symbols for entity type/class, attribute, and relationship. (b) Displaying attributes. (c) Displaying cardinality ratios. (d) Various (min, max) notations. (e) Notations for displaying specialization/generalization.

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++

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!!!