CS4.301 Data & Applications

Ponnurangam Kumaraguru (“PK”)
#ProfGiri @ IIIT Hyderabad

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

Example of a simple database

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.

Example of a simplified database catalog

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 enumerared type with all known majors. XXXXNNNN is used to define a type with four alpha characters followed by four digits

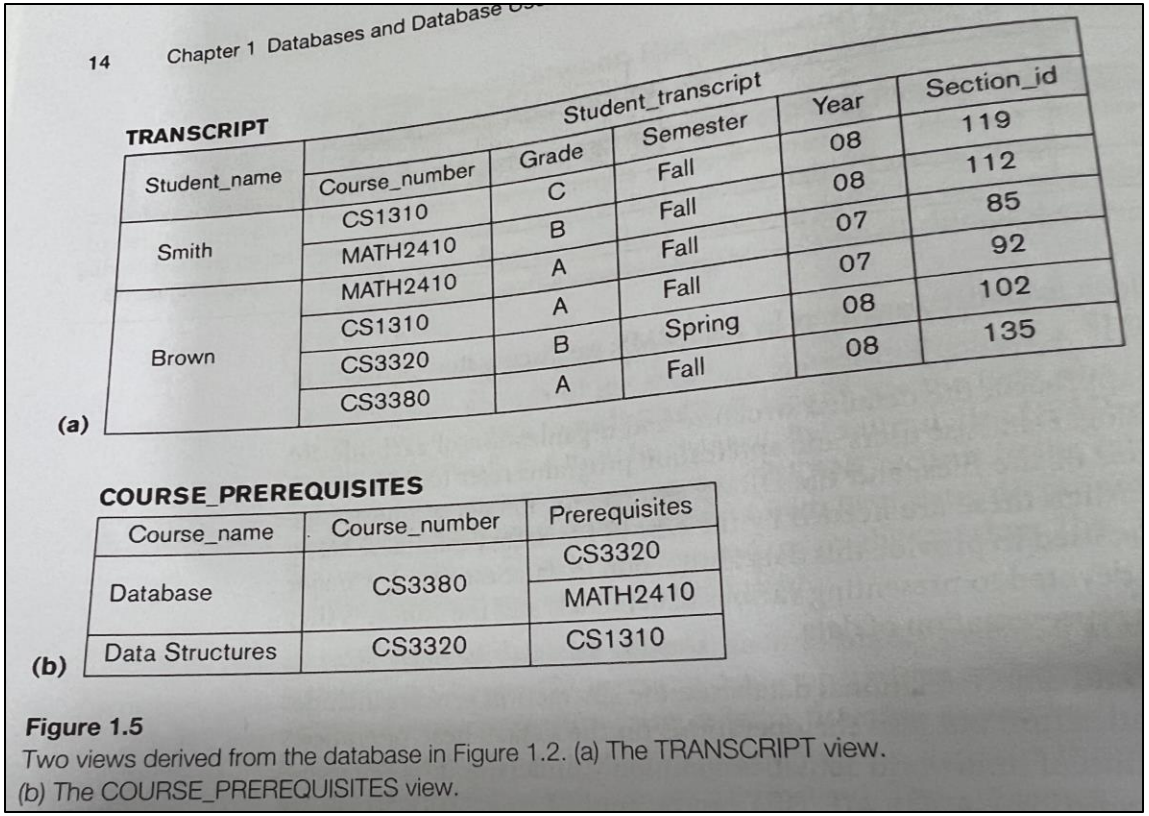
Figure 1.3
An example of a database catalog for the database in Figure 1.2.

Views

Many users to DB

Each users may require a different view

View may be a subset or virtual data derived



TRANSCRIPT

Student_name	Student_transcript				
	Course_number	Grade	Semester	Year	Section_id
Smith	CS1310	C	Fall	08	119
	MATH2410	B	Fall	08	112
Brown	MATH2410	A	Fall	07	85
	CS1310	A	Fall	07	92
	CS3320	B	Spring	08	102
	CS3380	A	Fall	08	135

(a)

COURSE_PREREQUISITES

Course_name	Course_number	Prerequisites
Database	CS3380	CS3320
		MATH2410
Data Structures	CS3320	CS1310

(b)

Figure 1.5

Two views derived from the database in Figure 1.2. (a) The TRANSCRIPT view. (b) The COURSE_PREREQUISITES view.

Online Transaction Processing (OLTP)

Multiuser DB

Concurrency control

Flight ticket booking, seats available

Transaction

Executing program or process that includes one or more database accesses, reading or updating of database records

Properties [ACID]

- Atomicity: either all are executed or none are executed [A/c A → A/c B]
- Consistency: any data written to a DB must be valid according to the defined rules [telephone number]
- Isolation: each transaction appears to execute in isolation, even though 100s may be executing at the same time [updating the seat preference]
- Durability: guarantees that once a transaction has been committed, it will remain committed even in the case of a system failure

Actors on the Scene: Day-to-Day use of DB

Database administrators

- authorizing access to DB, coordinating & monitoring its use, accountable for security breaches & response time

Database designers

- responsible for identifying the data to be stored in the DB, interact with potential group of users and develop *views* of the DB

End Users: Casual, naïve / parametric, sophisticated, stand-alone users

- Casual: occasional users, typically middle or high-level managers
- Naïve / parametric: constantly updating the db using *canned transaction,* done using mobile apps
  - bank tellers checking balances post withdrawals & deposits
  - reservation agents checking for availability
  - social media users post and read items on platforms

Actors on the Scene: Day-to-Day use of DB

End Users: Casual, naïve, sophisticated, stand-alone users

- sophisticated: thoroughly familiarize themselves with all facilities of DBMS, implement their own, complex requirements
- stand-alone: maintain personal DB using ready-made programs; TALLY

System analysts & application programmers

- determine the requirements of end-users, including naïve, develop specifications for canned transactions
- map implement above specifications as programs, they test – debug maintain these canned transactions
- software developers / engineers play these roles sometimes

Actors Behind the Scene: Maintain the DB

DBMS designers & implementers

design and implement the DBMS modules; complex modules like query language processing, interface processing, controlling concurrency, handling data recovery & security

Tool developers

design & implement tools; optional packages that are often purchased separately; facilitate DB modeling & design, system design, and improved performance

Operators & maintenance personnel

responsible for running & maintenance of the hardware & software environment for DB

Advantages of using DBMS approach

Controlling redundancy

redundancy in storing the same data multiple times e.g. student details in university maintained by acad & finance office separately

duplication of efforts, storage space, inconsistent data

ideally student details in only one place, *data normalization*

keeping all needed data together, *denormalization*

Advantages of using DBMS approach

Restricting unauthorized access

> your grades accessible to only some; my salary and personal details only to some [hopefully ☺]

Providing storage structures and search techniques for efficient query processing

> efficiently executing queries & updates; creating *indexes* and maintaining it; *buffering* & *caching* modules

Advantages of using DBMS approach

Providing backup & recovery

- provide facilities for recovering from hardware & software failures
- complex updates, should not crash; if crash what state to recover

Providing multiple user interfaces

- apps for mobile users; query language for causal; programming language for application programmers; forms / command codes for parametric; menu & natural language interfaces for standalone

Advantages of using DBMS approach

Representing Complex relationship among data

DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise, and to retrieve / update related data easily & efficiently

Enforcing integrity constraints

student name: 30 alphabetic characters; record in one file must be related to records in other files [e.g. every SECTION record must be related to a COURSE record] *referential integrity*

uniqueness on data item values [e.g. every COURSE record must have a unique value for COURSE_NUMBER] *key or uniqueness constraint*

Advantages of using DBMS approach

Permitting inferencing and actions using rules and triggers

- triggers associated with tables; trigger is a rule activated by updates to the table results in performing some addition operations to other tables, sending messages, etc.
- stored procedures are invoked appropriately when some conditions are met

Historical Development of Database Technology

Early Database Applications:

- The Hierarchical and Network Models were introduced in mid 1960s and dominated during the seventies.
- A bulk of the worldwide database processing still occurs using these models, particularly, the hierarchical model using IBM's IMS system.

Relational Model based Systems:

- Relational model was originally introduced in 1970, was heavily researched and experimented within IBM Research and several universities.
- Relational DBMS Products emerged in the early 1980s.

Object-oriented and emerging applications:

- Object-Oriented Database Management Systems (OODBMSs) were introduced in late 1980s and early 1990s to cater to the need of complex data processing in CAD and other applications.
  - Their use has not taken off much.
- Many relational DBMSs have incorporated object database concepts, leading to a new category called *object-relationa*l DBMSs (ORDBMSs)
- *Extended relational* systems add further capabilities (e.g. for multimedia data, text, XML, and other data types)

Historical Development of Database Technology (continued)

Web contains data in HTML (Hypertext markup language) with links among pages.

This has given rise to a new set of applications and E-commerce is using new standards like XML (eXtended Markup Language). (see Ch. 13).

Script programming languages such as PHP and JavaScript allow generation of dynamic Web pages that are partially generated from a database (see Ch. 11).

- Also allow database updates through Web pages

Social Media platforms such as Facebook and Twitter are generating millions of transactions a day and businesses are interested to tap into this data to “understand” the users

Cloud Storage and Backup is making unlimited amount of storage available to users and applications

“Big Data”, Hadoop, Mapreduce, Spark based technology.

NOSQL -- systems have been designed for rapid search and retrieval from documents, processing of huge graphs occurring on social networks, and other forms of unstructured data with flexible models of transaction processing (Chapter 24).

When not to use a DBMS

Main inhibitors (costs) of using a DBMS:

- High initial investment and possible need for additional hardware.
- Overhead for providing generality, security, concurrency control, recovery, and integrity functions.

When a DBMS may be unnecessary:

- If the database and applications are simple, well defined, and not expected to change.
- If access to data by multiple users is not required.

When a DBMS may be infeasible:

- In embedded systems where a general purpose DBMS may not fit in available storage

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

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.

Schemas

Database State:

- The actual data stored in a database at a ***particular moment in time***. This includes the collection of all the data in the database.
- Also called database instance (or occurrence or snapshot).
  - The term *instance* is also applied to individual database components, e.g. *record instance, table instance, entity instance*

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

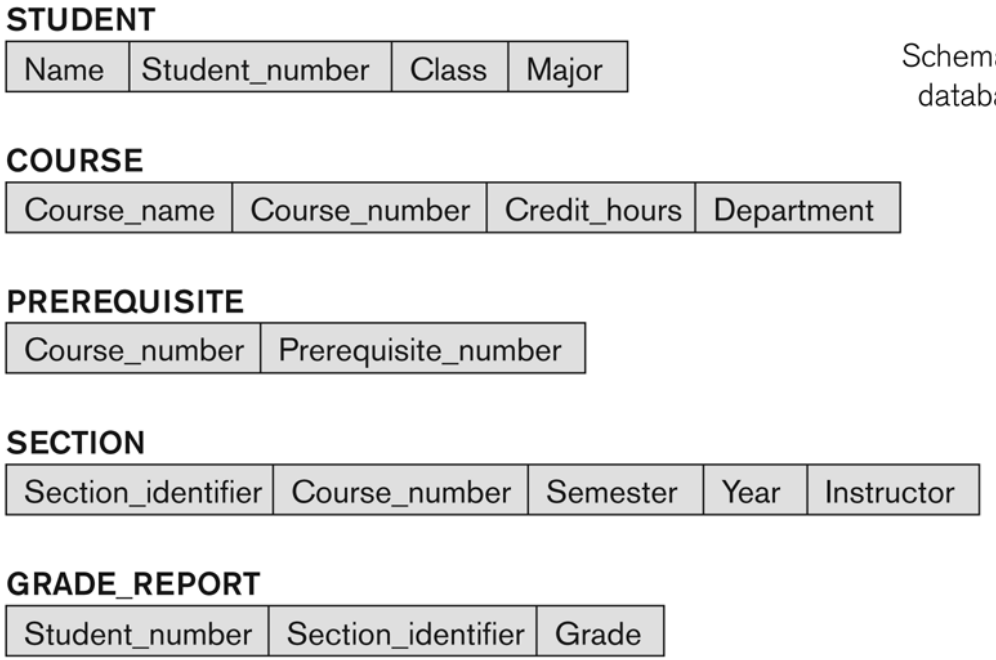


Figure 2.1
Schema diagram for the database in Figure 1.2.

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.

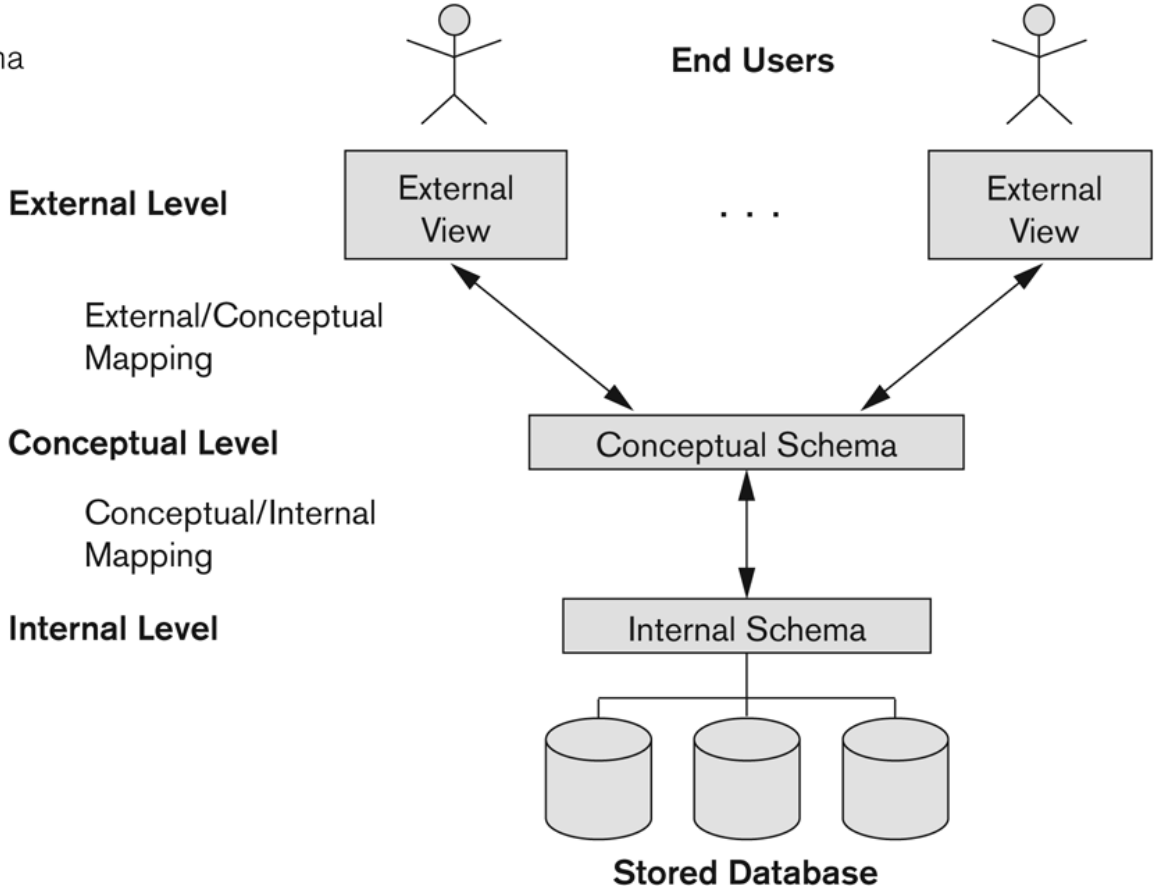
Example of a database state

Three-Schema Architecture

Defines DBMS schemas at ***three*** levels:

- **Internal schema** at the internal level to describe physical storage structures and access paths (e.g indexes).
  - Typically uses a **physical** data model.
- **Conceptual schema** at the conceptual level to describe the structure and constraints for the whole database for a community of users.
  - Uses a **conceptual** or an **implementation** data model.
- **External schemas** at the external level to describe the various user views.
  - Usually uses the same data model as the conceptual schema.

Figure 2.2
The three-schema architecture.



The three-schema architecture

Typical DBMS Component Modules

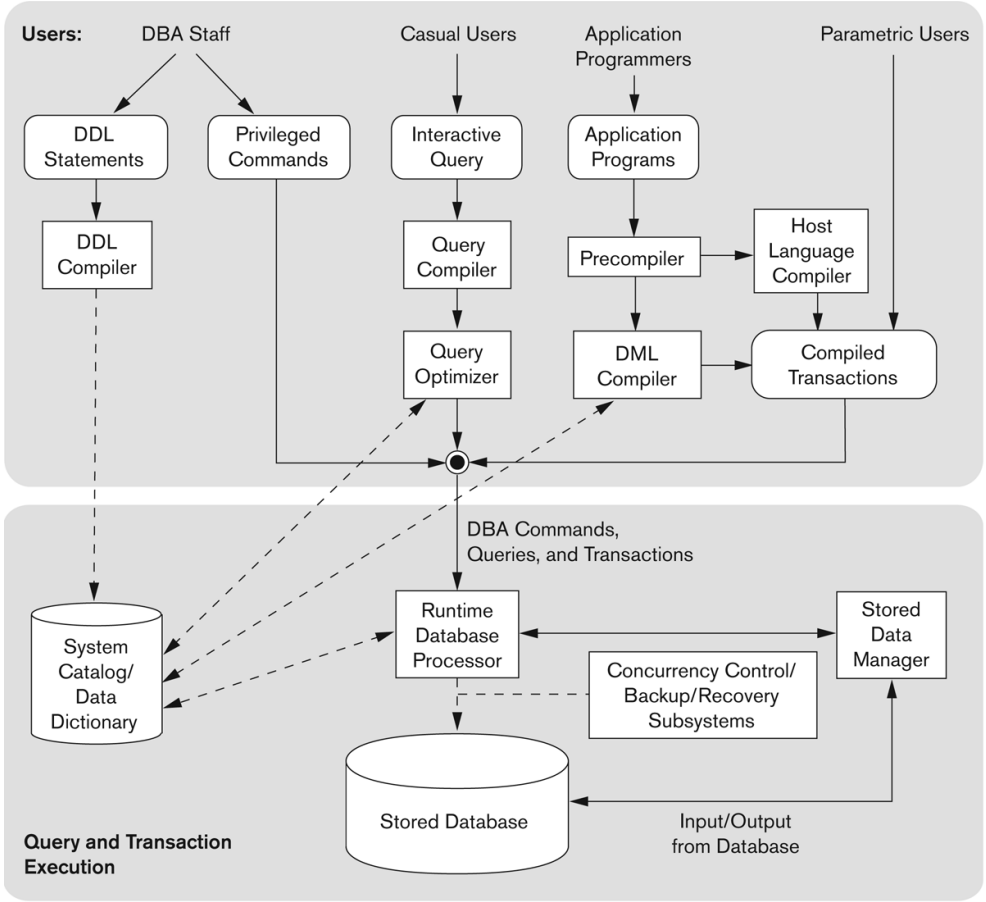


Figure 2.3
Component modules of a DBMS and their interactions.

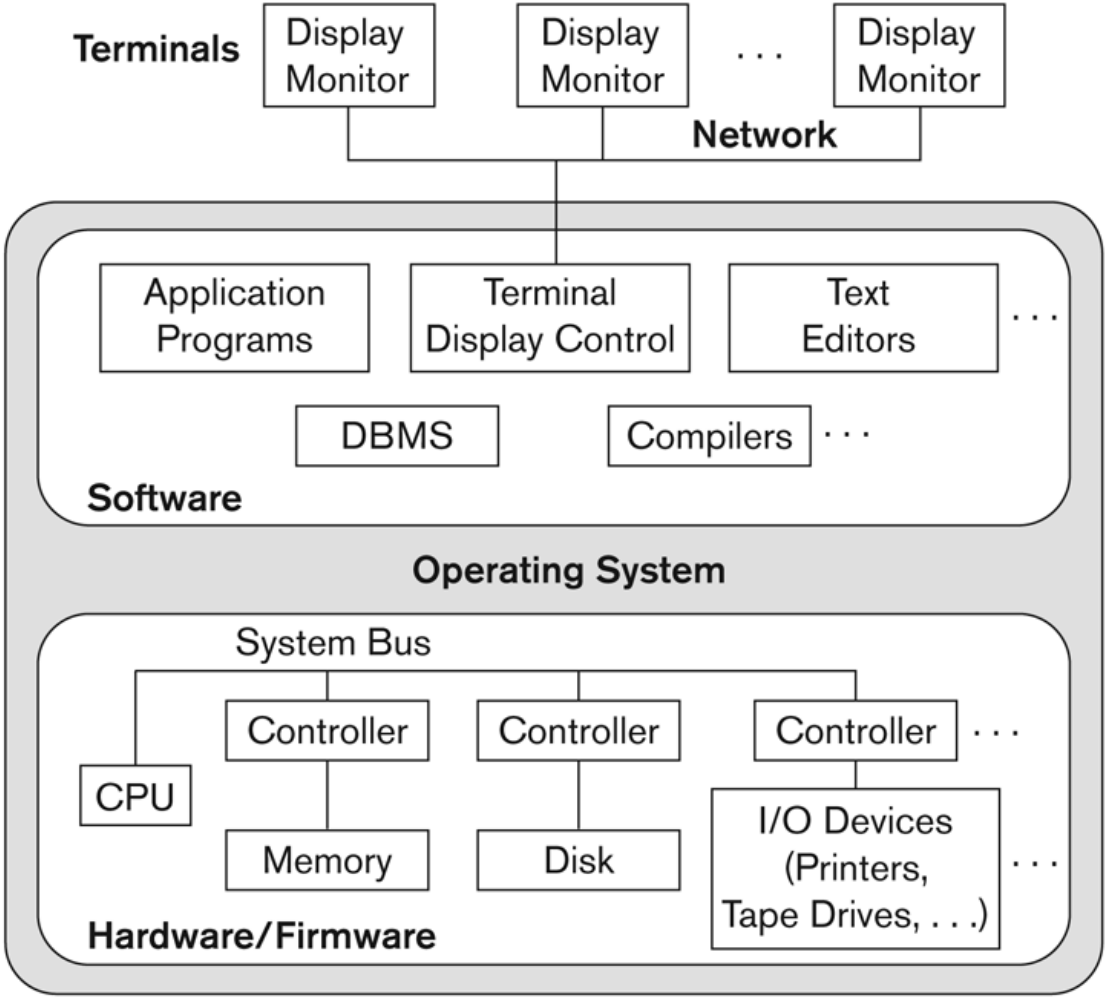


Figure 2.4
A physical centralized architecture.

Logical two-tier client server architecture

Figure 2.5
Logical two-tier client/server architecture.

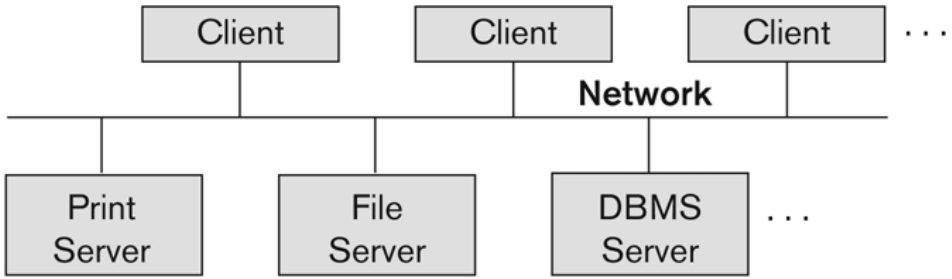
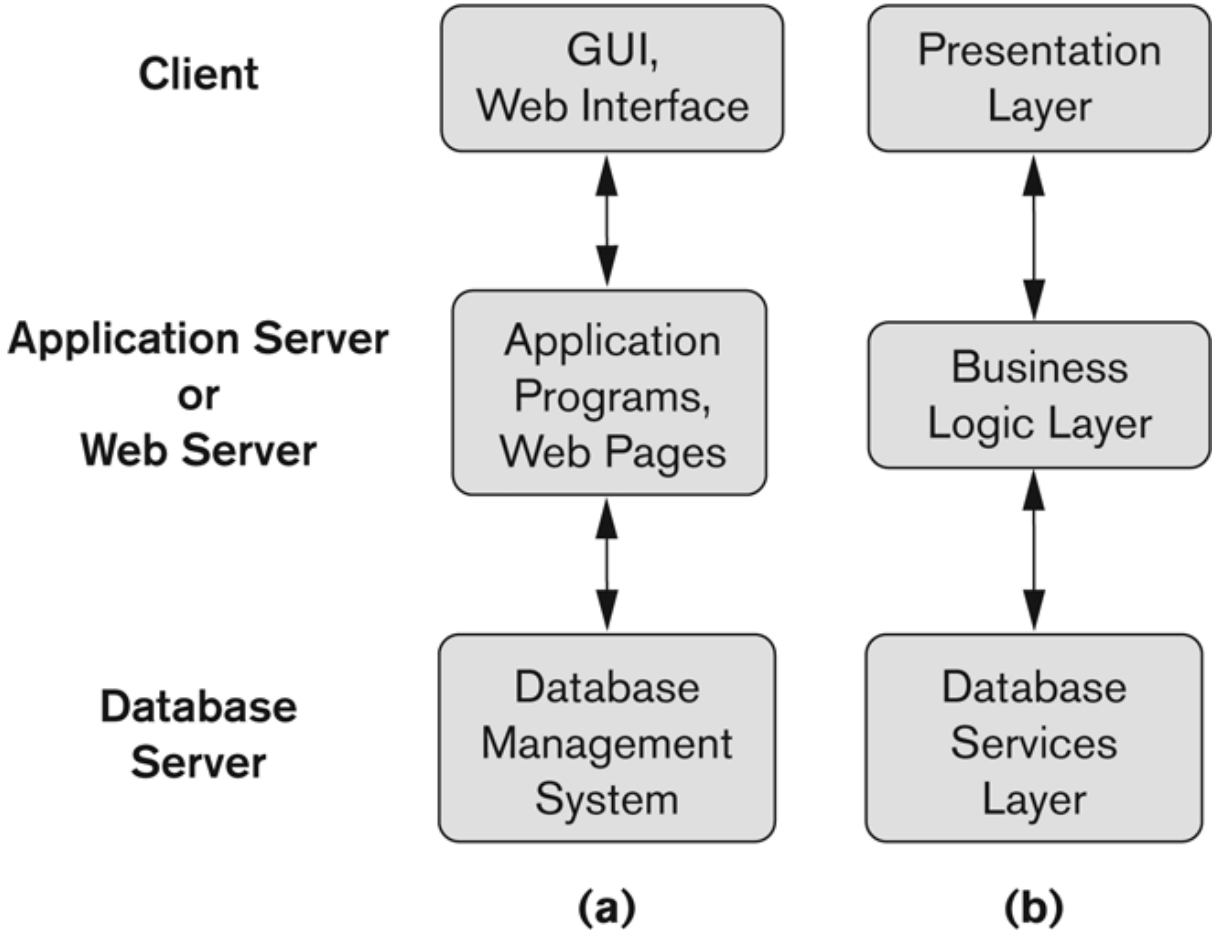


Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.



DBMS Languages

Data Definition Language (DDL)

Data Manipulation Language (DML)

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.
- QBE – Query By Example

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.
- COBOL / FORTRAN

This Lecture

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 is 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

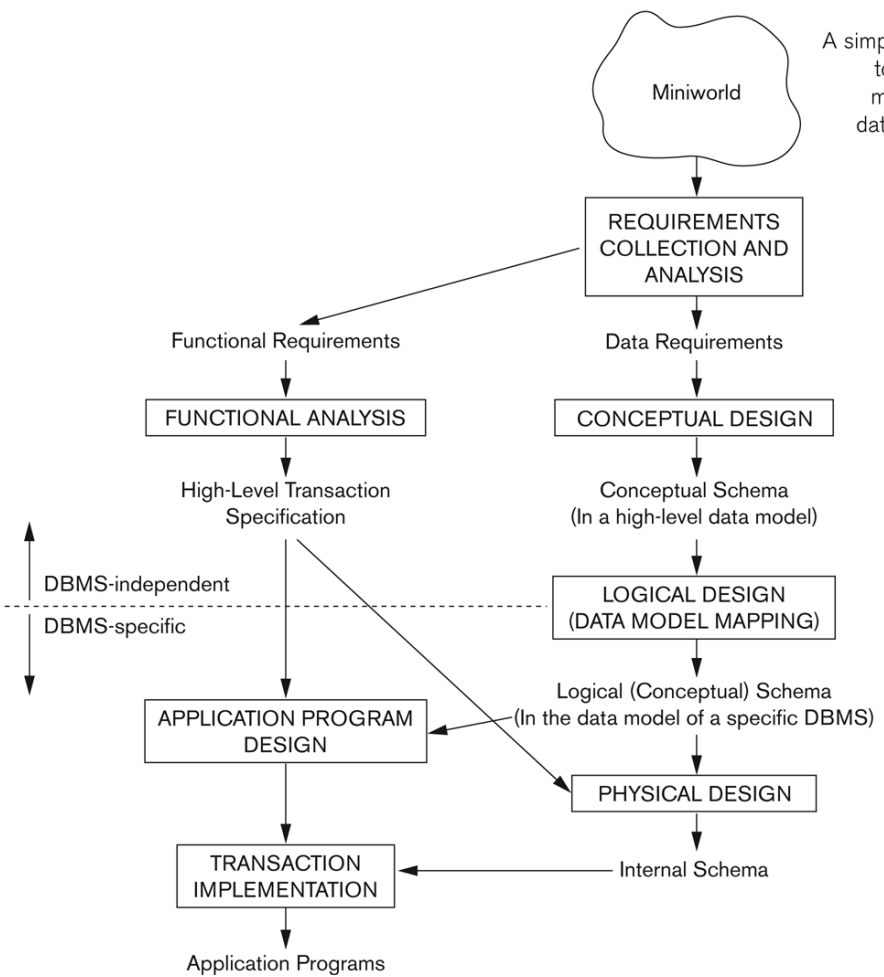


Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Overview of Database Design Process

Methodologies for Conceptual Design

Entity Relationship (ER) Diagrams

Enhanced Entity Relationship (EER) Diagrams

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).

Multi-valued

Multiple values for the 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

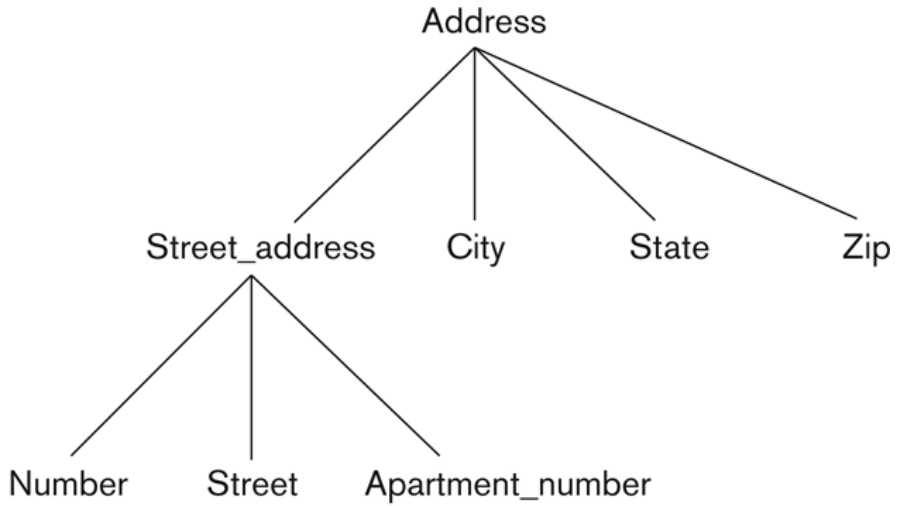


Figure 3.4
A hierarchy of composite attributes.

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**

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) – Name (Employee)

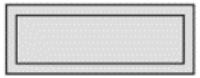
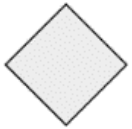
Displaying an Entity type

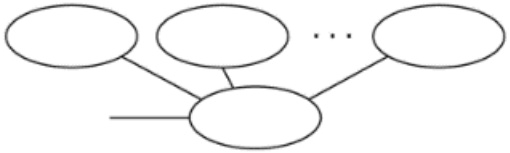
In ER diagrams, an entity type is displayed in a rectangular box

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

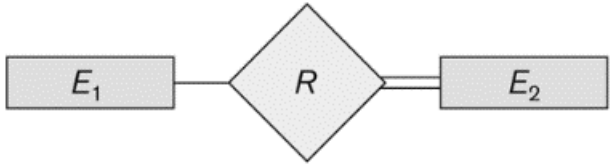
Figure 3.14
Summary of the notation for ER diagrams.

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

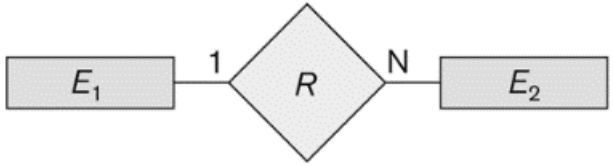


Composite Attribute

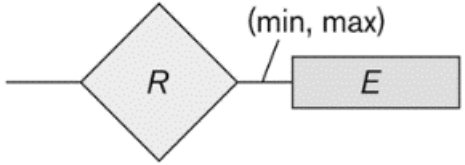
Derived Attribute



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

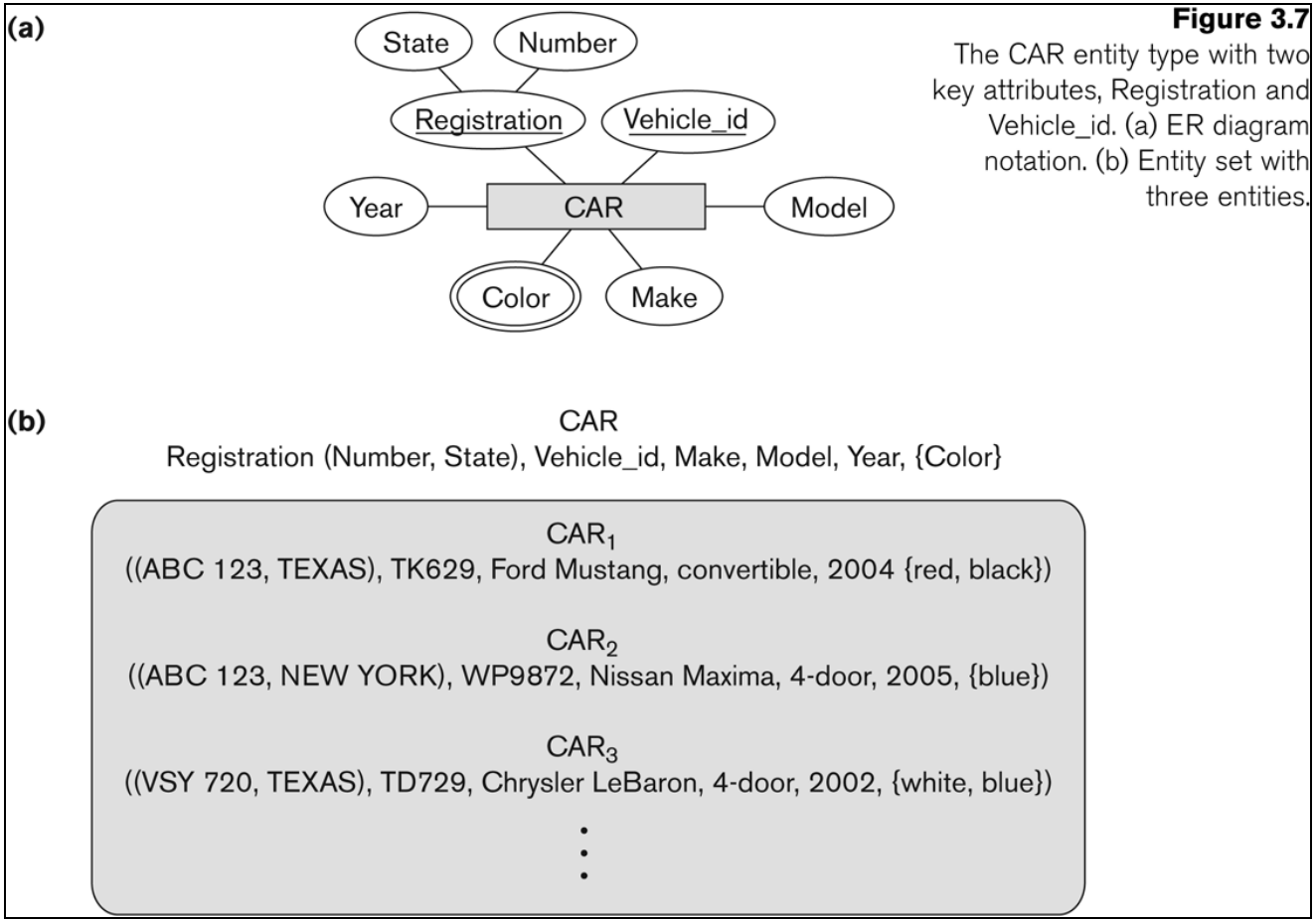


Figure 3.7
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

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 in the next slide

The initial attributes shown are derived from the requirements description

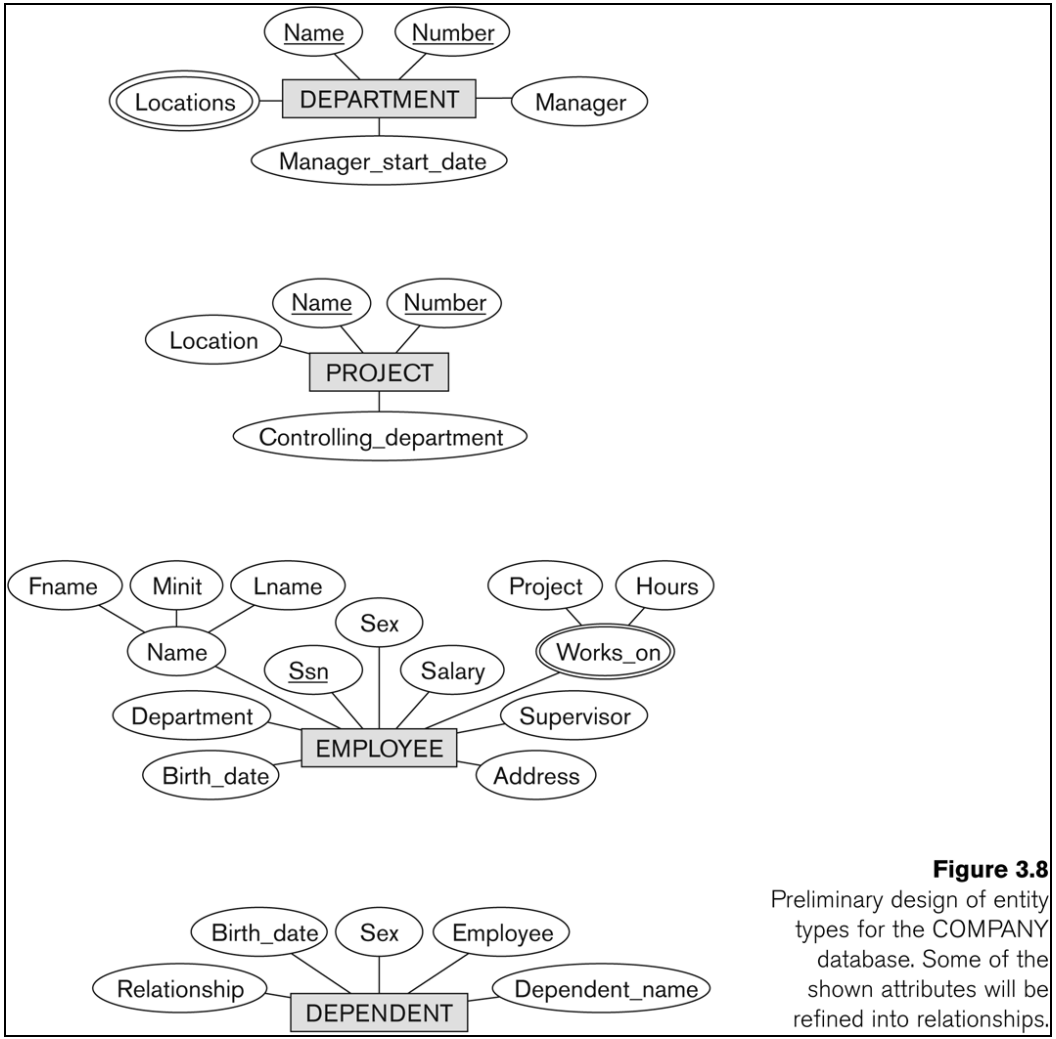


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Initial Design of Entity Types:

EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

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

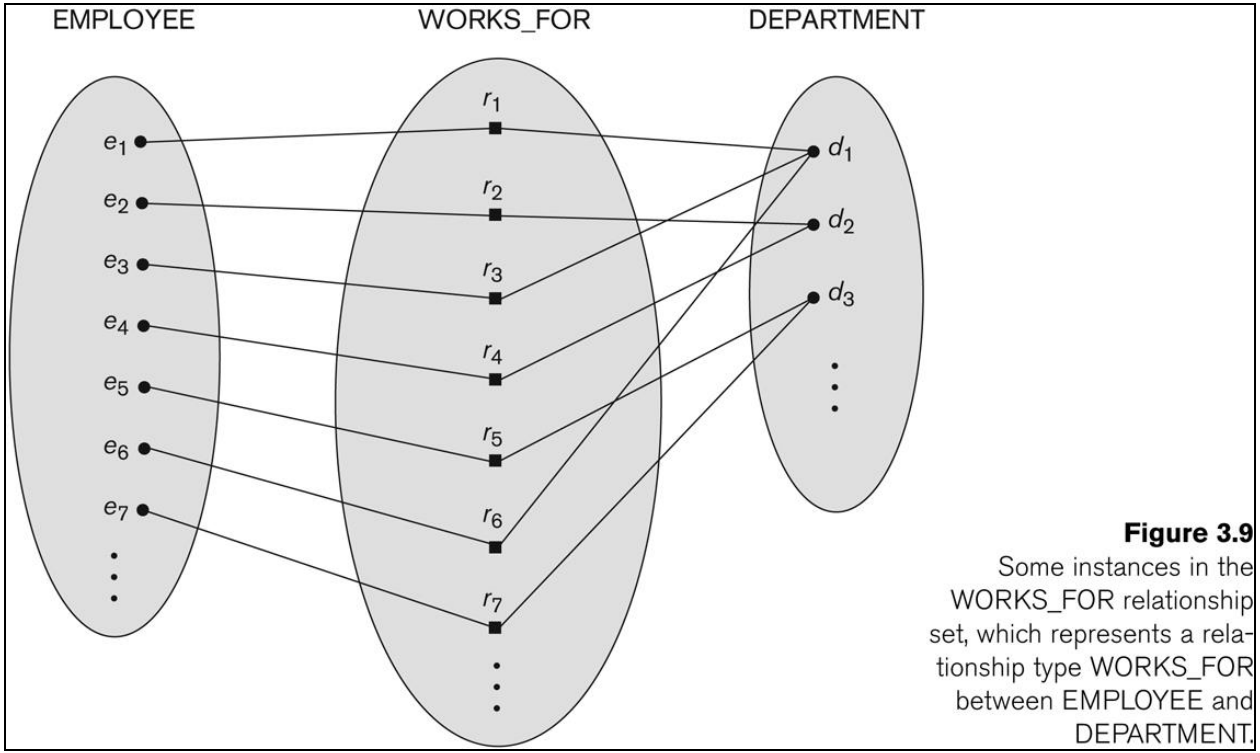


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

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

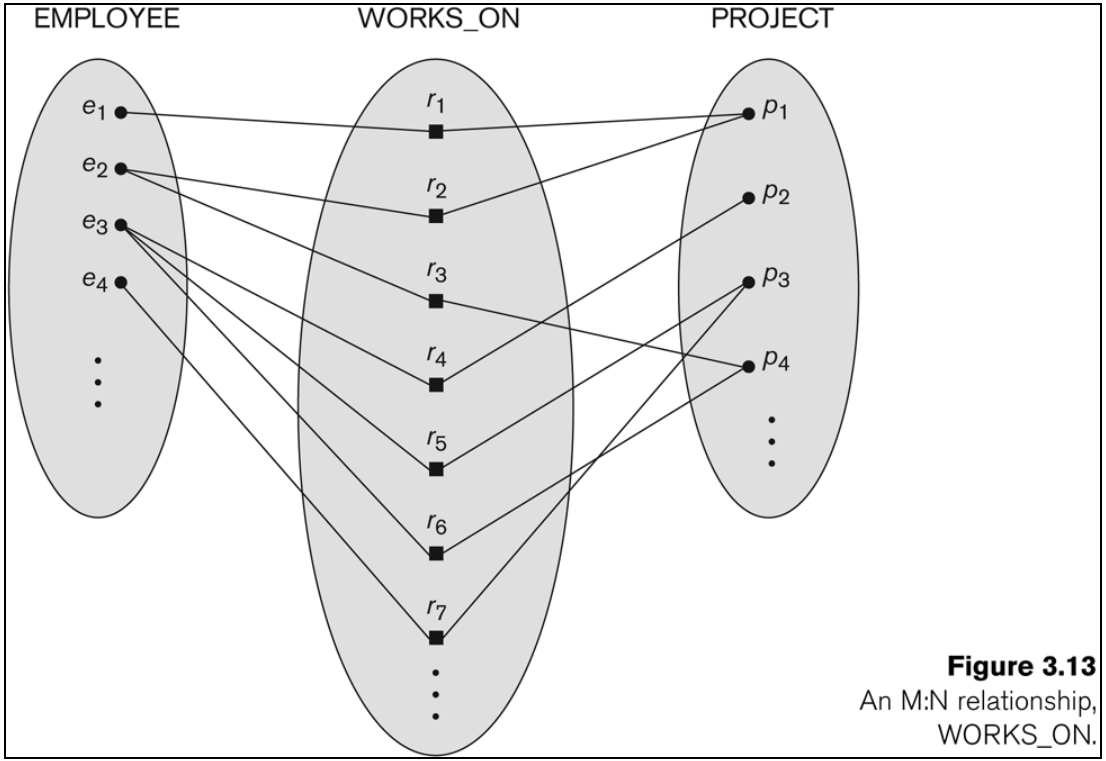


Figure 3.13
An M:N relationship, WORKS_ON.

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)

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)

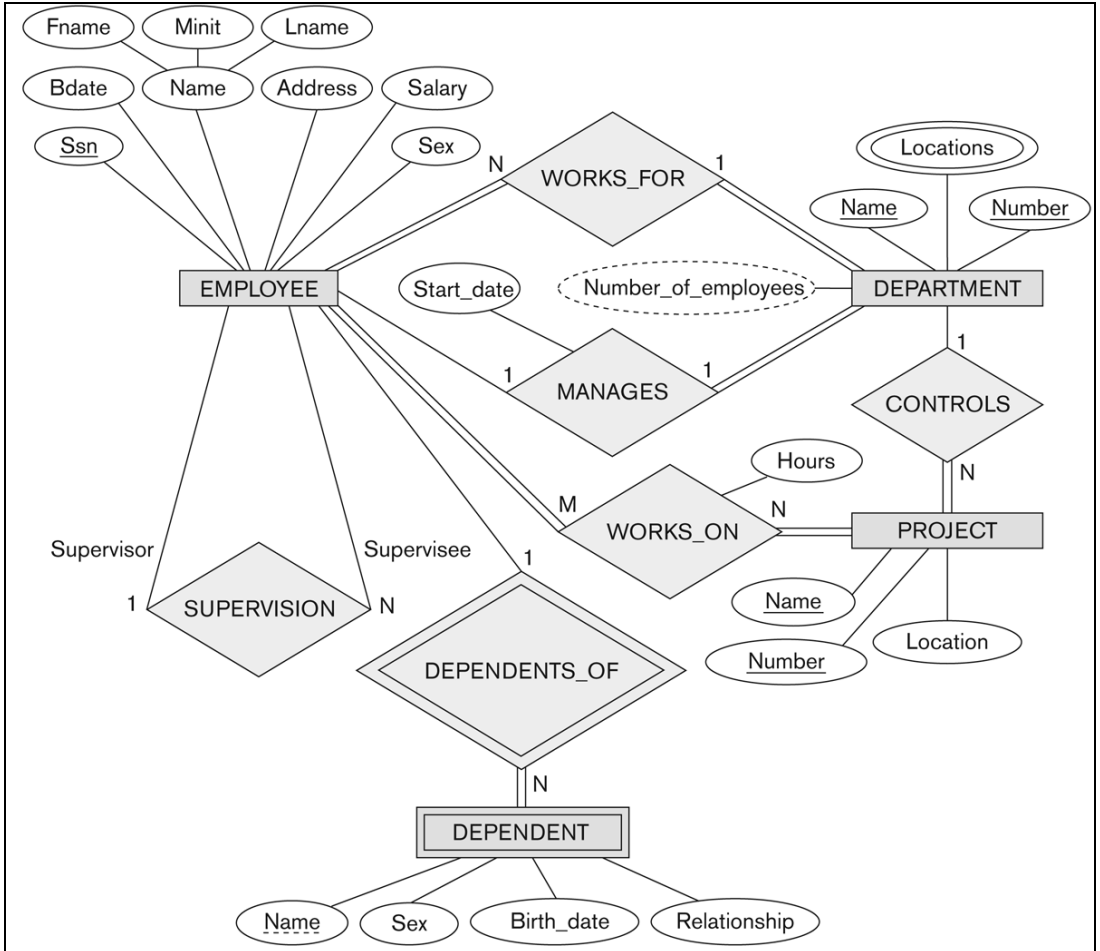


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

ER DIAGRAM – Relationship Types are:

WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF

Activity

Create requirements for Infinium

Have at least 4 Entities, and develop ER diagram for the same

Be creative in terms of attributes, relationships, and entities!

Any questions?

Bibliography / Acknowledgements

Instructor materials from Elmasri & Navathe 7e


pk.profgiri


Ponnurangam.kumaraguru


/in/ponguru


ponguru

Thank you for attending the class!!!


pk.guru@iiit.ac.in