COSC 3380 - 19717
Design of Database Systems

Introduction

August 26, 2025

COSC 3380-19717 – Design of Database Systems – Fall Semester 2025

Instructor: Dr. Uma Ramamurthy (uramamur@bcm.edu)

Time: Tuesdays & Thursdays – 5:30 pm to 7:00 pm

Location: SEC 104

Text Book: Fundamentals of Database Systems, 7th Edition,
by Ramez Elmasri, and Shamkant B. Navathe

Topics to be covered:

Introduction to Databases
Relational Data Model
Structured Query Language (SQL)
Conceptual Modeling & Database Design
Database Design Theory and Normalization
Database Security
Distributed databases and NOSQL Systems

Important Dates:

Exam 1 – Thursday, October 9, 2025, 5:30 pm – 7 pm
Exam 2 – Thursday, December 4, 2025, 5:30 pm – 7 pm

Grading:

COSC 3380
Team Project & Presentations – 40%
Exam 1 - 20%
Exam 2 - 30%
Class Participation – 10%*

Class Participation

- Attendance does **NOT** equate to participation
- You are required to ask questions, answer questions and participate in class discussions
- You will be provided a participation grade after the Exam-1 for the first half of the semester
- Your total participation score will be given as part of the final grade
- IMPORTANT: Start early and participate regularly!

Team Project: First task

- Send me an email this week from your CougarNet email address to uramamur@bcm.edu:
  - Subject line: 'COSC-3380 19717: My programming/Database experience'
  - Tell me about your programming background and experience
    - Which programming language(s) you know/have worked in?
  - Have you used databases or built databases?
    - If yes, which ones?

Team Project: Second task

- Start your groundwork on how to build a web application
  - Resources document will be posted tonight on Canvas
  - Use the internet
  - NOT going to build an API
  - A web application
    - With a user interface
    - A data source

Definitions

- **Database**
  - A collection of related data
  - Known facts that can be recorded
  - Some part of the real world about which data are stored in a database
  - Logically coherent collection of data with an implicit meaning
  - Built for a specific purpose
  - Example: a university database -- student demographics, courses, grades and transcripts

Definitions

- **Database Management System**
  - Software package used to define, create, use and maintain a database
  - Consists of several software modules

- **Database system**
  - DBMS software together with the data (in a database)
  - Often applications are included

Applications of Database Technology

- Storage and retrieval of traditional numeric and alphanumeric data in an inventory application
- Multimedia applications (YouTube, Spotify)
- Biometric applications (fingerprints, retina scans)
- Wearable applications (FitBit, Apple Watch)
- Geographical Information Systems (GIS) applications (Google Maps)
- Sensor applications (nuclear reactor)
- Big Data applications (Genome sequencing, Weather forecasting)
- Internet of Things (IoT) applications (Smart homes)
- Data Warehouses

COSC 3380 - 19717
Design of Database Systems

Introduction

August 28, 2025

Recent developments

- Social Networks - capture a lot of information about people and communications among people - posts, tweets, photos, videos
  - All of the above constitute data
- Search Engines - collect their own repository of web pages for search purposes
- New Technologies to manage vast amounts of data generated on the web:
  - Big data storage systems involving large clusters of distributed computers (Bigtable)
  - NOSQL (Not Only SQL) data systems
  - A large amount of data now resides in the “cloud”!

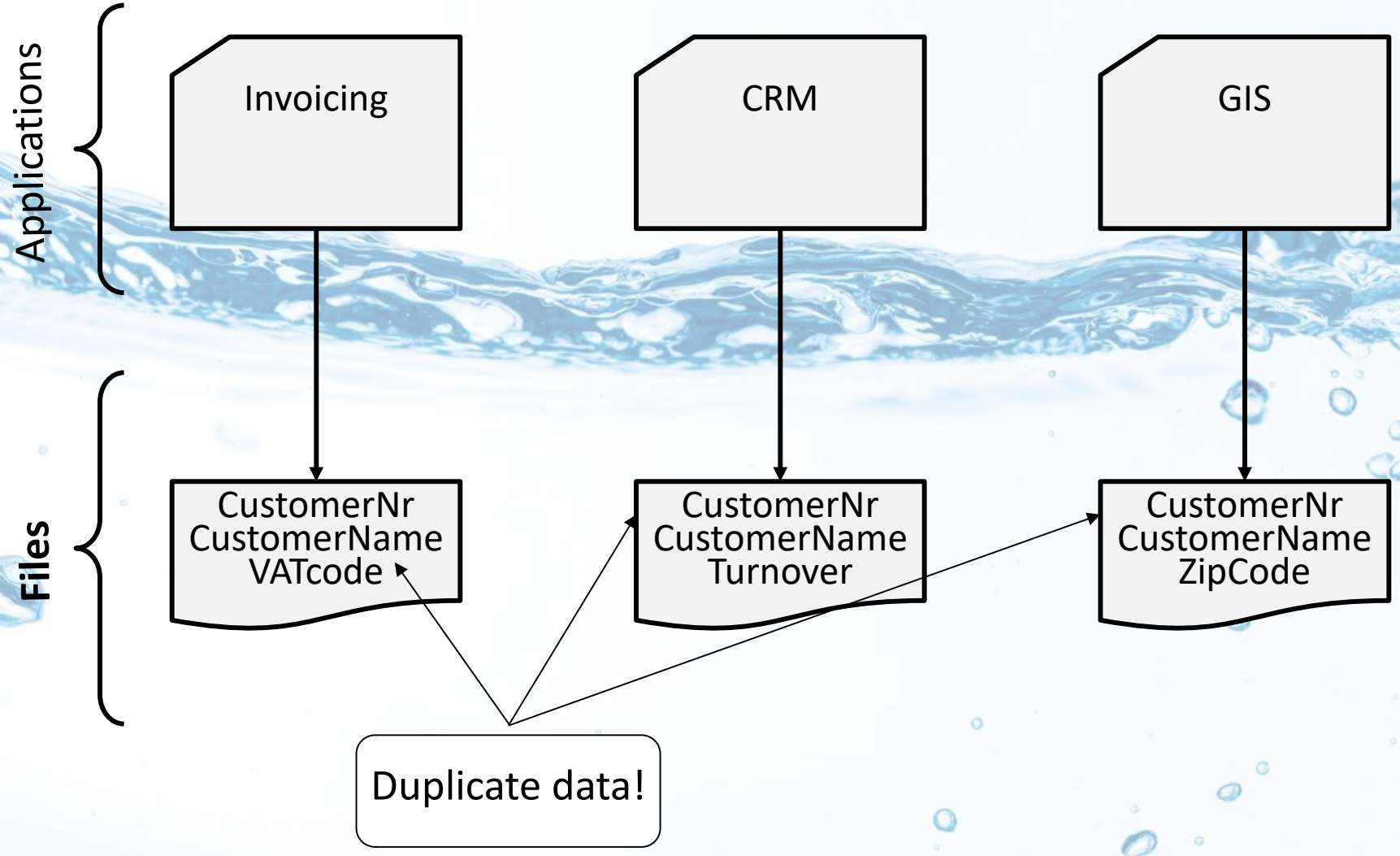
Impact of Databases & Database Technology

- **Businesses:** Banking, Insurance, Retail, Transportation, Healthcare, Manufacturing
- **Service Industries:** Financial, Real-estate, Legal, Electronic Commerce, Small businesses
- **Education:** Resources for Content and Delivery
- **Recent developments:** Social Networks, Environmental and Scientific Applications, Medicine and Genetics
- **Personalized Applications:** Based on smart, mobile devices

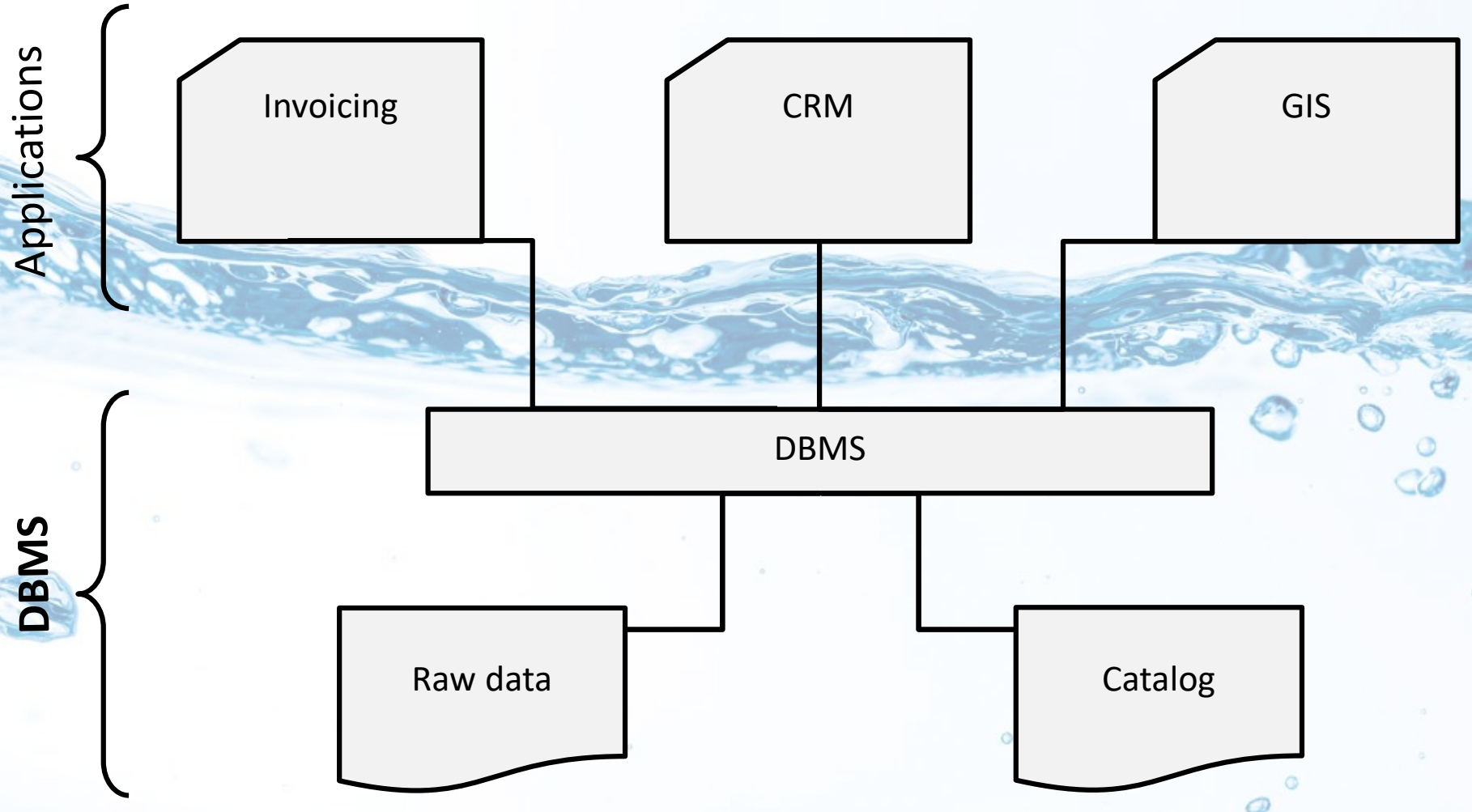
Discussion question

- Why not use files to store and manage data as we did in the 1970s?
  - File approach vs. database approach for data management
  - Text files, Excel, etc.

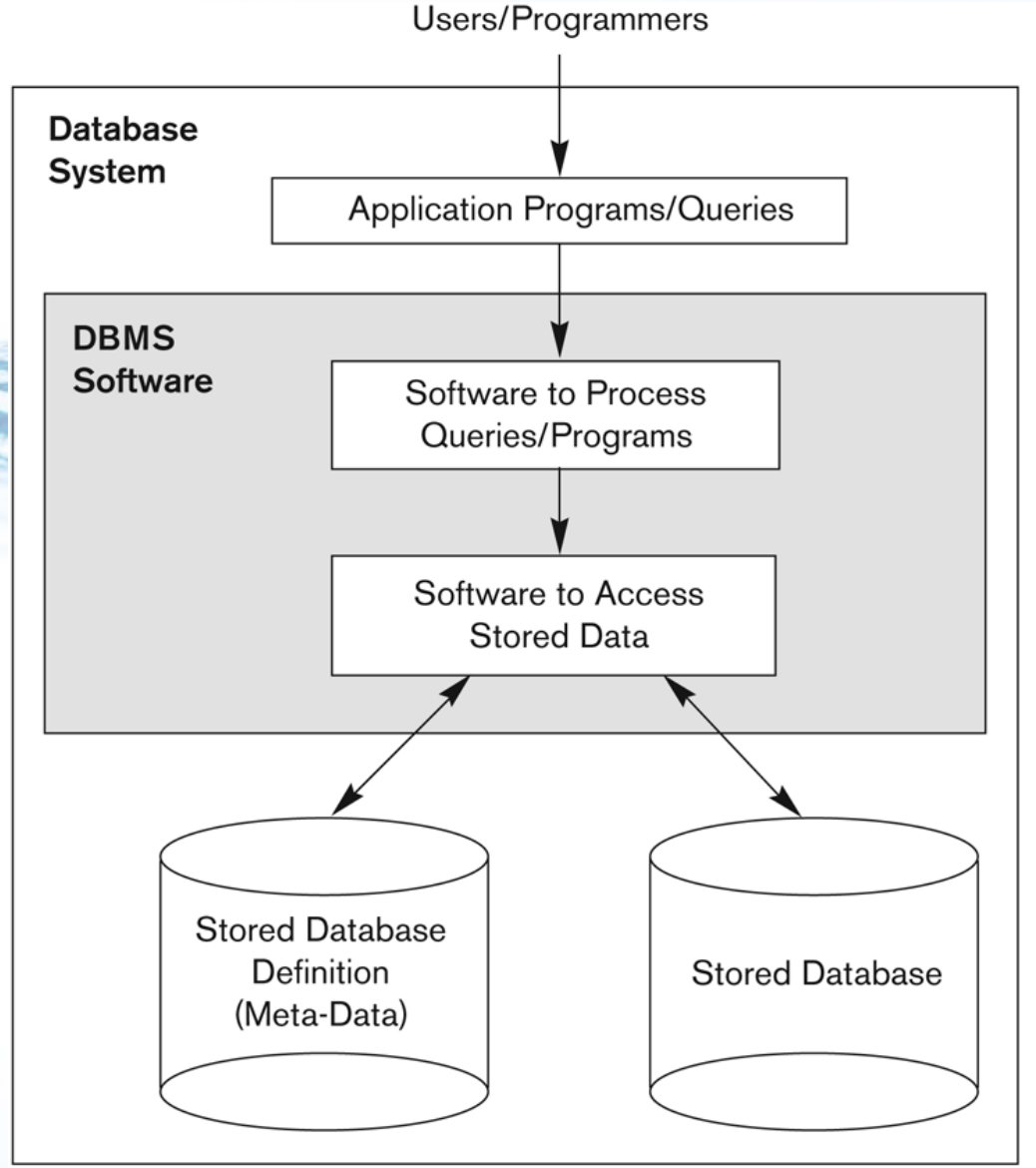
Discussion question



Discussion question



Simplified database system environment



Typical DBMS Functionality

- ***Define*** a particular database in terms of its data types, structures, and constraints on the data to be stored
- ***Meta-data***
  - Database definition or descriptive information
  - Stored by the DBMS in the form of a database catalog or dictionary
- ***Construct*** or ***Load*** the initial database contents on a secondary storage medium
- ***Manipulating*** the database:
  - Modification: Insertions, deletions and updates to its content
  - Retrieval: Querying, generating reports
  - Accessing the database through Web applications
- ***Processing*** and ***Sharing*** by a set of concurrent users and application programs – yet, keeping all data valid and consistent

Additional DBMS Functionality

- DBMS may also provide –
  - **Protection** or **Security measures** to prevent unauthorized access
  - **Active processing** to take internal actions on data

COSC 3380 - 19717
Design of Database Systems

Introduction

September 2, 2025

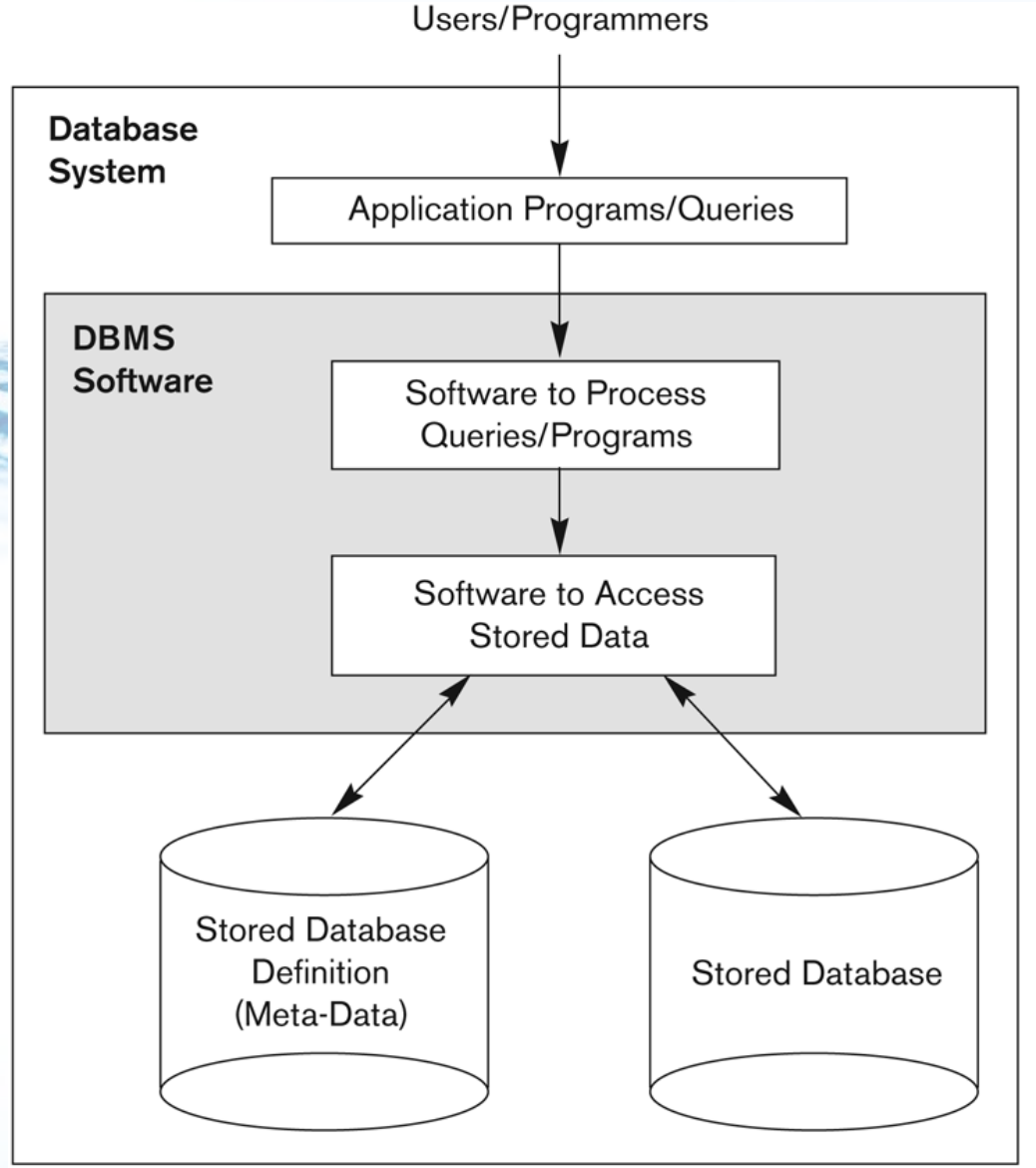
Additional DBMS Functionality

- DBMS may also provide –
  - **Protection** or **Security measures** to prevent unauthorized access
  - **Active processing** to take internal actions on data
  - **Presentation** and **Visualization** of data
  - **Maintenance** of the database and associated programs over the lifetime of the database application

Application activities against a Database

- Applications interact with a database by generating
  - **Queries**: to access different parts of data and formulate the result of a request
  - **Transactions**: may read some data; 'update' certain values or generate new data, and store that in the database
- Applications must not allow unauthorized users to access data
- Applications must keep up with changing user requirements against the database

Simplified database system environment



Example of a Database

Mini-world:

- Part of a UNIVERSITY environment
- Some mini-world ***entities***:
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (Academic) DEPARTMENTs
  - INSTRUCTORs

Mini-world *relationships*:

- SECTIONs *are for specific* COURSEs
- STUDENTs *take* SECTIONs
- COURSEs *have prerequisite* COURSEs
- INSTRUCTORs *teach* SECTIONs
- COURSEs *are offered by* DEPARTMENTs
- STUDENTs *major in* DEPARTMENTs

Snapshot of the UNIVERSITY 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

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

Characteristics of the Database Approach

- **Insulation between programs and data**
  - Program-data independence
  - Allows changing data structures and storage organization without having to change the DBMS access programs

Characteristics of the Database Approach

- **Data Abstraction**
  - A **data model** is used to hide storage details and present the users with a conceptual view of the database.
  - Programs refer to the data model constructs rather than data storage details
- **Support of multiple views of the data**
  - Each user may see a different view of the database, which describes **only** the data of interest to that user.

Characteristics of the Database Approach

- **Sharing of data and multi-user transaction processing:**
  - Allowing a set of **concurrent users** to retrieve from and to update the database.
  - *Concurrency control* within the DBMS guarantees that each **transaction** is correctly executed or aborted
  - *Recovery* subsystem ensures each completed transaction has its effect permanently recorded in the database
  - **OLTP** (Online Transaction Processing) is a major part of database applications. This allows hundreds of concurrent transactions to execute per second.

Advantages of the Database Approach

- Controlling redundancy in data storage, in development and maintenance efforts
- Sharing of data among multiple users
- Restricting unauthorized access to data. Only the DBA staff uses privileged commands and facilities.
- Providing Storage Structures (e.g. indexes) for efficient Query Processing
- Persistent storage for program Objects
  - Object-oriented DBMSs make program objects persistent

COSC 3380 - 19717
Design of Database Systems

Introduction

September 4, 2025

Advantages of the Database Approach

- Optimization of queries for efficient processing
- Providing backup and recovery services
- Providing multiple interfaces to different classes of users
- Representing complex relationships among data
- Enforcing integrity constraints on the database
- Drawing inferences and actions from the stored data using deductive and active rules / triggers

Implications of the Database Approach

- Potential for enforcing standards
  - Very crucial for the success of database applications in large organizations
  - **Standards** refer to data item names, display formats, screens, report structures, meta-data, Web page layouts, etc.
- Reduced application development time
  - Incremental time to add each new application is reduced.

Implications of Database Approach

- Flexibility to change data structures
  - Database structure may evolve as new requirements are defined.
- Availability of current information
  - Extremely important for on-line transaction systems such as shopping, airline, hotel, car reservations.
- Economies of scale
  - Wasteful overlap of resources and personnel can be avoided by consolidating data and applications across departments.

Database Users

- Two groups of users
  - Actors on the Scene –
    - Those who actually use and control the database content, and those who design, develop and maintain database applications
  - Workers behind the Scene
    - Those who design and develop the DBMS software and related tools, and the computer systems operators

Database Users – Actors on the Scene

Database administrators:

- Responsible for authorizing access to the database
- For coordinating and monitoring database use
- Acquiring software and hardware resources, controlling its use and monitoring efficiency of operations

Database Users – Actors on the Scene

Database Designers:

- Responsible to define the content, the structure, the constraints, and functions or transactions against the database
- They must communicate with the end-users and understand their needs

Database Users – Actors on the Scene

End Users

- They use the data for queries, reports and some update the database content.
- End-users can be categorized into:
  - **Casual**: access database occasionally when needed
  - **Naïve** or **Parametric**: they make up a large section of the end-user population.
  - They use previously well-defined functions in the form of "canned transactions" against the database.

Examples:

- Users of Mobile Apps
- Bank-tellers or reservation clerks are parametric users who do this activity for an entire shift of operations.
- Social Media Users post and read information from those sites

Database Users – Actors on the Scene

End Users

- **Sophisticated:**
  - These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities.
  - Many use tools in the form of software packages that work closely with the stored database.
- **Stand-alone:**
  - Mostly maintain personal databases using ready-to-use packaged applications.

  Examples:
  - user of a tax program that creates its own internal database.
  - user that maintains a database of personal photos and videos.

Database Users – Actors on the Scene

- **System Analysts and Application Developers**
  - **System Analysts**
    - Understand the user requirements of naïve and sophisticated users
    - Design applications including canned transactions to meet those requirements.
  - **Application Programmers**
    - Implement the specifications developed by analysts
    - Test and debug them before deployment.
  - **Business Analysts**
    - Analyze vast amounts of business data and real-time data ('Big Data') for better decision making related to planning, advertising, marketing etc.

Database Users – Actors behind the Scene

- **Tool Developers**
  - Design and implement software systems
    - Modeling and designing databases
    - Prototyping
    - Test data generation
    - User interface creation
    - Simulation
    - Performance monitoring
  - Facilitate building of applications and allow using database effectively

Database Users – Actors behind the Scene

- **System Designers and Implementors**
  - Design and implement DBMS packages in the form of modules and interfaces, test and debug them.
  - The DBMS must interface with applications, programming language compilers, operating system components, etc.
- **Operators and Maintenance Personnel**
  - They manage the actual running and maintenance of the database system hardware and software environment.

COSC 3380 - 19717
Design of Database Systems

Introduction

September 9, 2025

Team Project – Today's task

- Find your teammates
- Choose a team leader
- Review the topics for the team project and select your top 3 topics
- Send an email to uramamur@bcm.edu with your team's top 3 project topics, so your team gets assigned a topic based on first-come-first-served basis
- Discuss technologies to be used in your team project

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 not to use a DBMS

- 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

When not to use a DBMS

- When no DBMS may suffice:
  - If there are stringent real-time requirements that may not be met because of DBMS overhead (e.g., telephone switching systems)
  - If the database system is not able to handle the complexity of data because of modeling limitations (e.g., complex genome and protein databases)
  - If the database users need special operations not supported by the DBMS (e.g., GIS and location based services).


Database System
Concepts and Architecture

Data Models

- **Data Model**
  - A set of concepts to describe
    - the ***structure*** of a database
    - the ***operations*** for manipulating these structures
    - 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

- **Data Model Operations**
  - Operations used for specifying database *retrievals* and *updates* by referring to the constructs of the data model.
  - Operations on the data model
    - ***Basic model operations***
      - generic insert, delete, update
    - ***User-defined operations***
      - compute_student_gpa
      - update_inventory
      - notify_manager

Categories of Data Models

- **Conceptual (high-level, semantic) data models**
  - Provide concepts that are close to the way many users perceive data
    - ***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
  - Usually specified in an ad-hoc manner through DBMS design and administration manuals

Categories of Data Models

- **Implementation (representational) data models**
  - Provide concepts that fall between conceptual and physical models
  - Used by many commercial DBMS implementations
  - Example: 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 NOSQL systems

Schema

- Database Schema
  - ***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

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

Instances

- Database State
  - The actual data stored in a database at a ***particular moment in time***
  - 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 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

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

Database Schema vs. Database State

- Distinction
  - The ***database schema*** changes very infrequently
  - The ***database state*** changes every time the data in the database are updated

- **Schema → intension**
- **State → extension**

COSC 3380 - 19717
Design of Database Systems

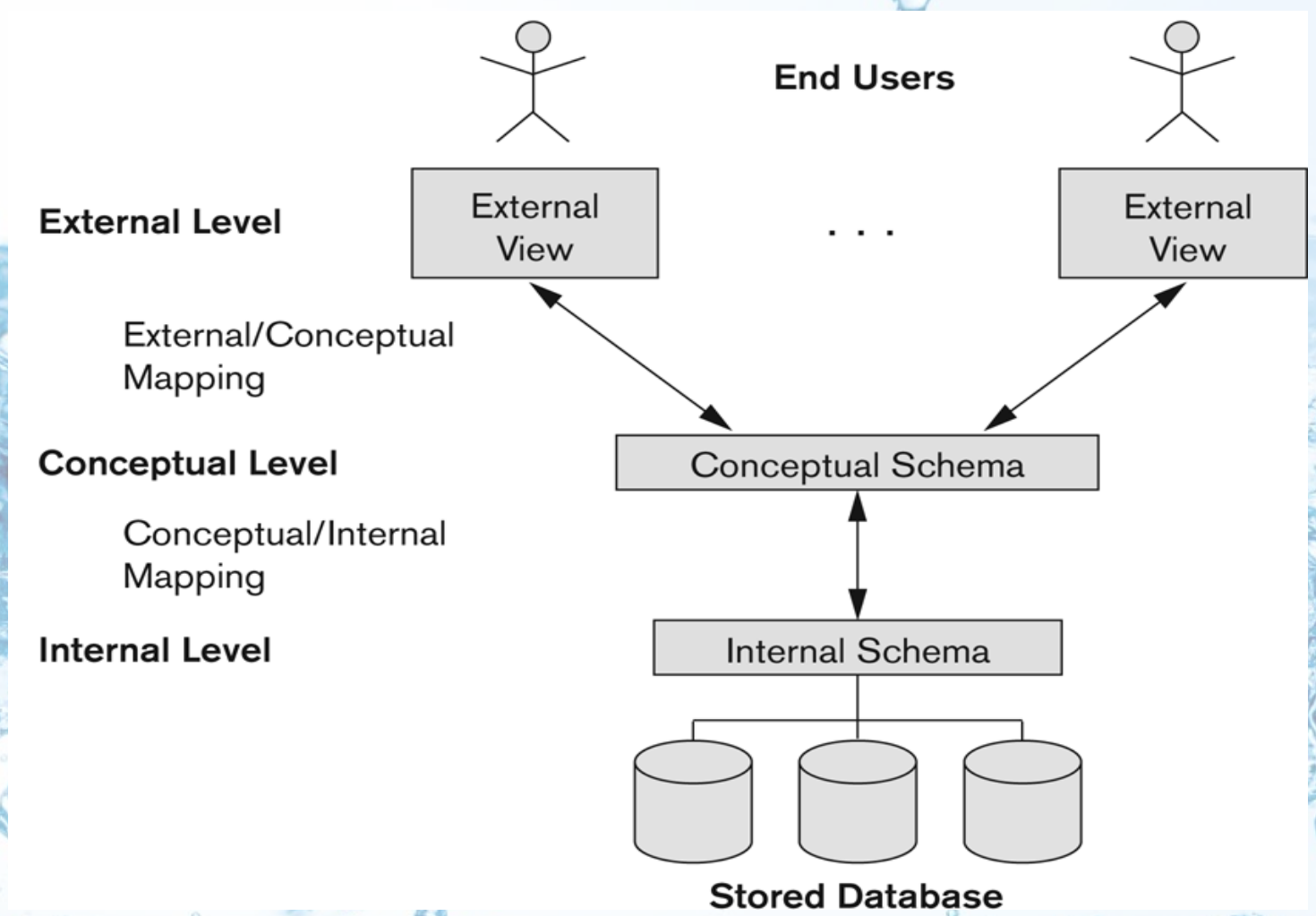
Database System Concepts and Architecture

September 11, 2025

Three-Schema Architecture

- To support DBMS characteristics of
  - **Program-data independence**
  - Support of **multiple views** of the data
- Defines DBMS schemas at ***three*** levels:
  - **Internal schema** - to describe physical storage structures and access paths (eg. indexes).
    - Typically uses a **physical** data model.
  - **Conceptual schema** - to describe the structure and constraints for the whole database for a community of users.
    - Uses a **conceptual** or an **implementation** data model.
  - **External schema** - to describe the various user views.
    - Usually uses the same data model as the conceptual schema.

Three-Schema Architecture



Three-Schema Architecture

- Mappings among schema levels are needed to transform requests and data.
  - Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution
  - Data extracted from the internal DBMS level is reformatted to match the user's external view
  - Example: formatting the results of an SQL query for display in a Web page

Data Independence

- **Logical Data Independence**:
  - The capacity to change the conceptual schema without having to change the external schemas and their associated application programs
  - Example: Adding data items, change constraints
- **Physical Data Independence**:
  - The capacity to change the internal schema without having to change the conceptual schema
  - Example: Reorganizing file structures, creating new indexes to improve database performance
- **Which one is easier to achieve?**

Data Independence

- When a schema at a lower level is changed, only the **mappings** between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence
- The higher-level schemas themselves are **unchanged**.
  - The application programs need not be changed since they refer to the external schemas.

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

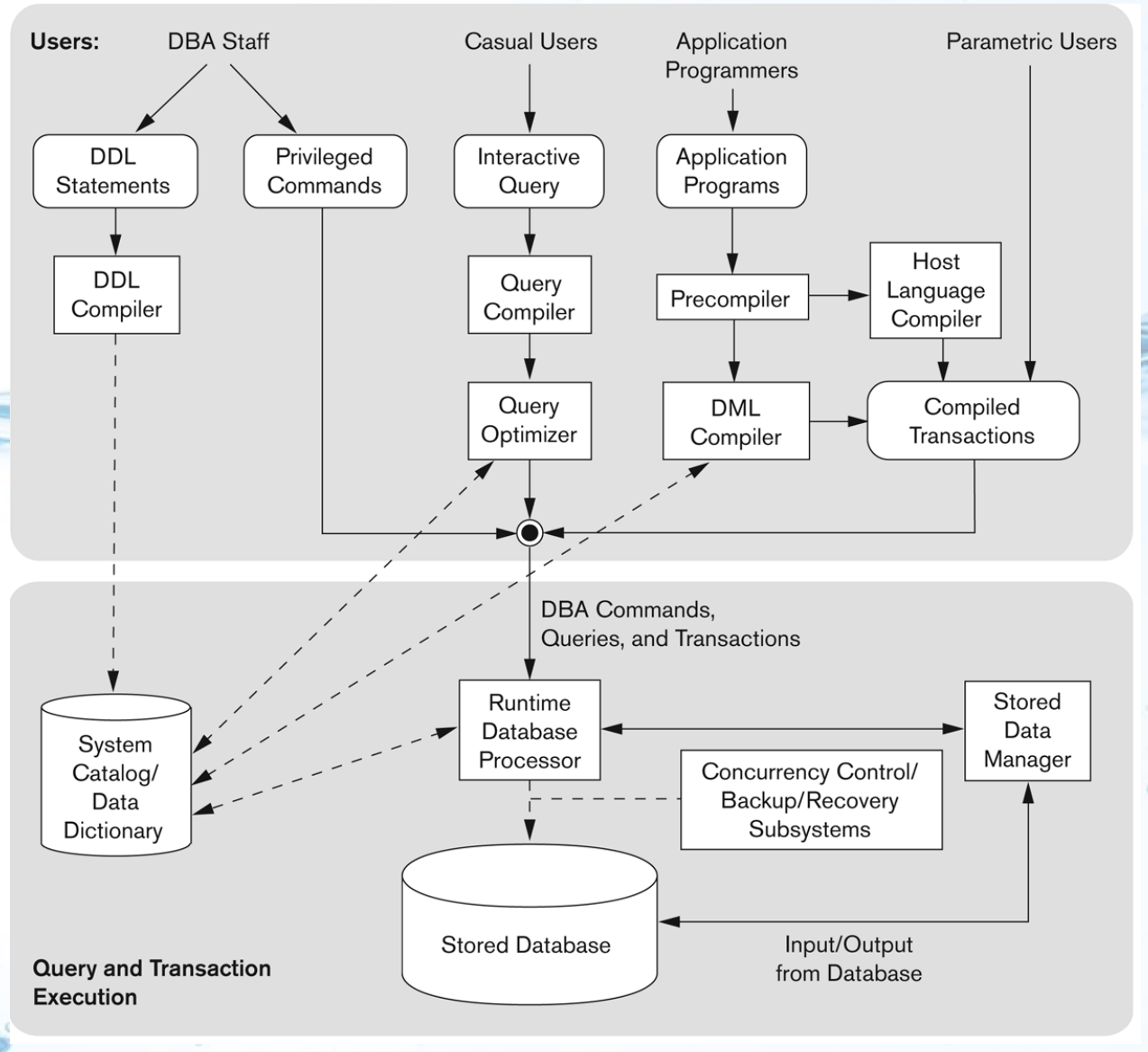
DBMS Languages

- **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 C, 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 (*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
  - A **declarative** language
- **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

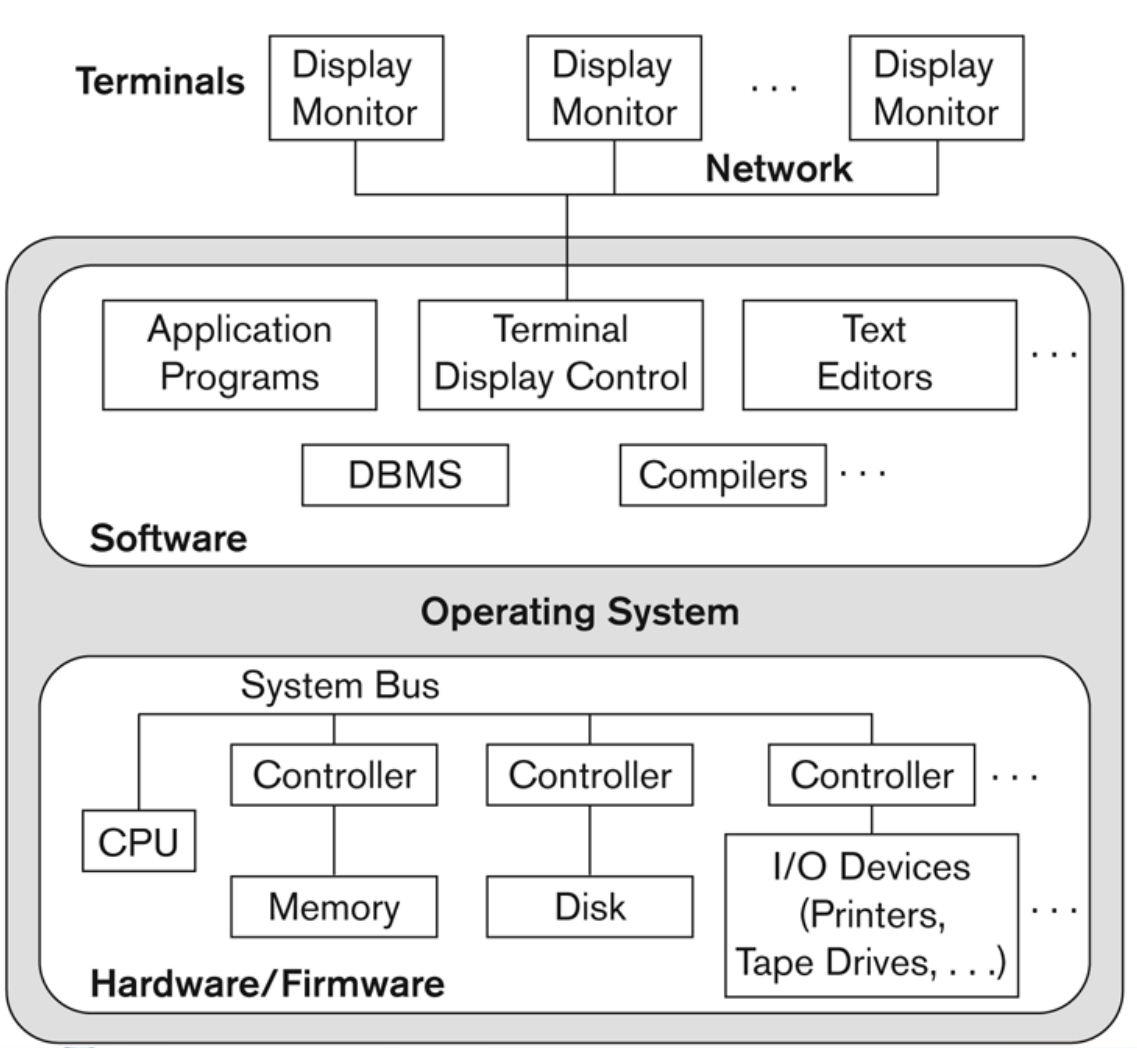
Typical DBMS Modules



DBMS Architectures

- Centralized DBMS
  - Combines everything into single system including
    - DBMS software
    - Hardware,
    - Application programs
    - User interface processing software
  - User can still connect through a remote terminal
    - All processing is done at central system

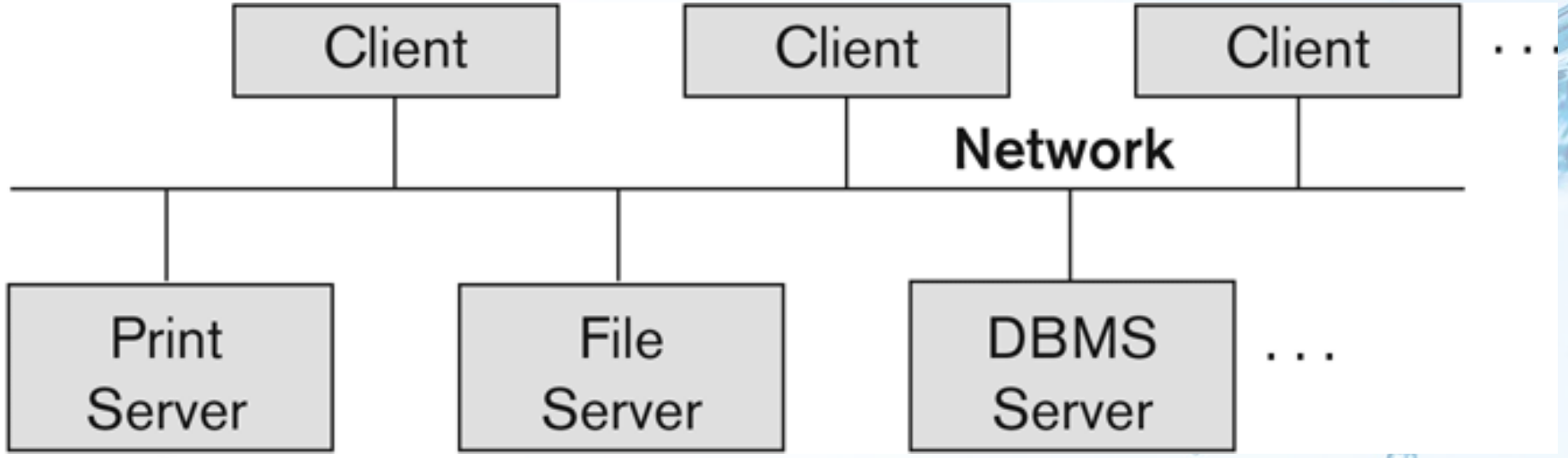
Centralized Architecture



Basic 2-tier Client-Server Architectures

- Specialized Servers with specialized functions
  - Print server
  - File server
  - DBMS server
  - Web server
  - Email server
- Clients can access the specialized servers as needed

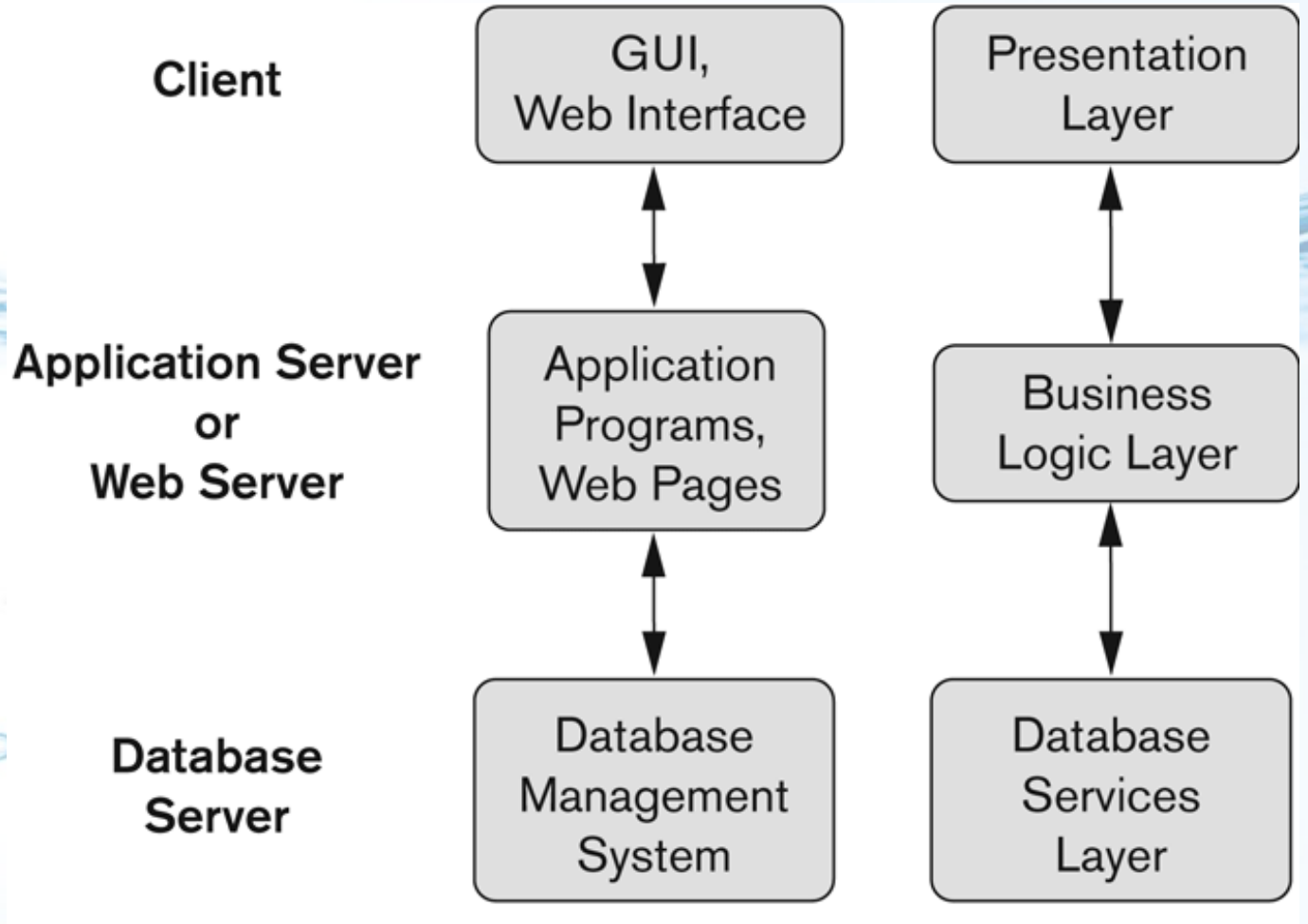
Logical two-tier Client Server Architecture



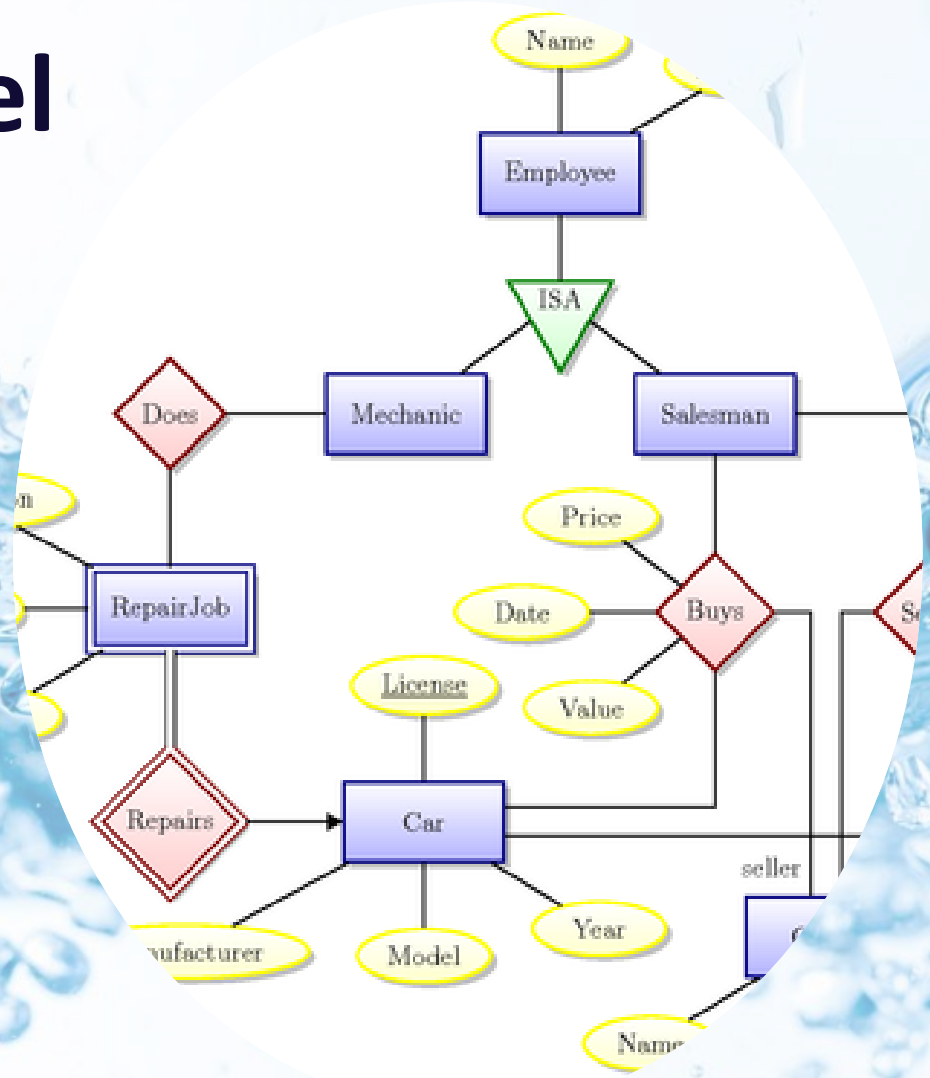
Three Tier Client-Server Architecture

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

Three-tier client-server architecture



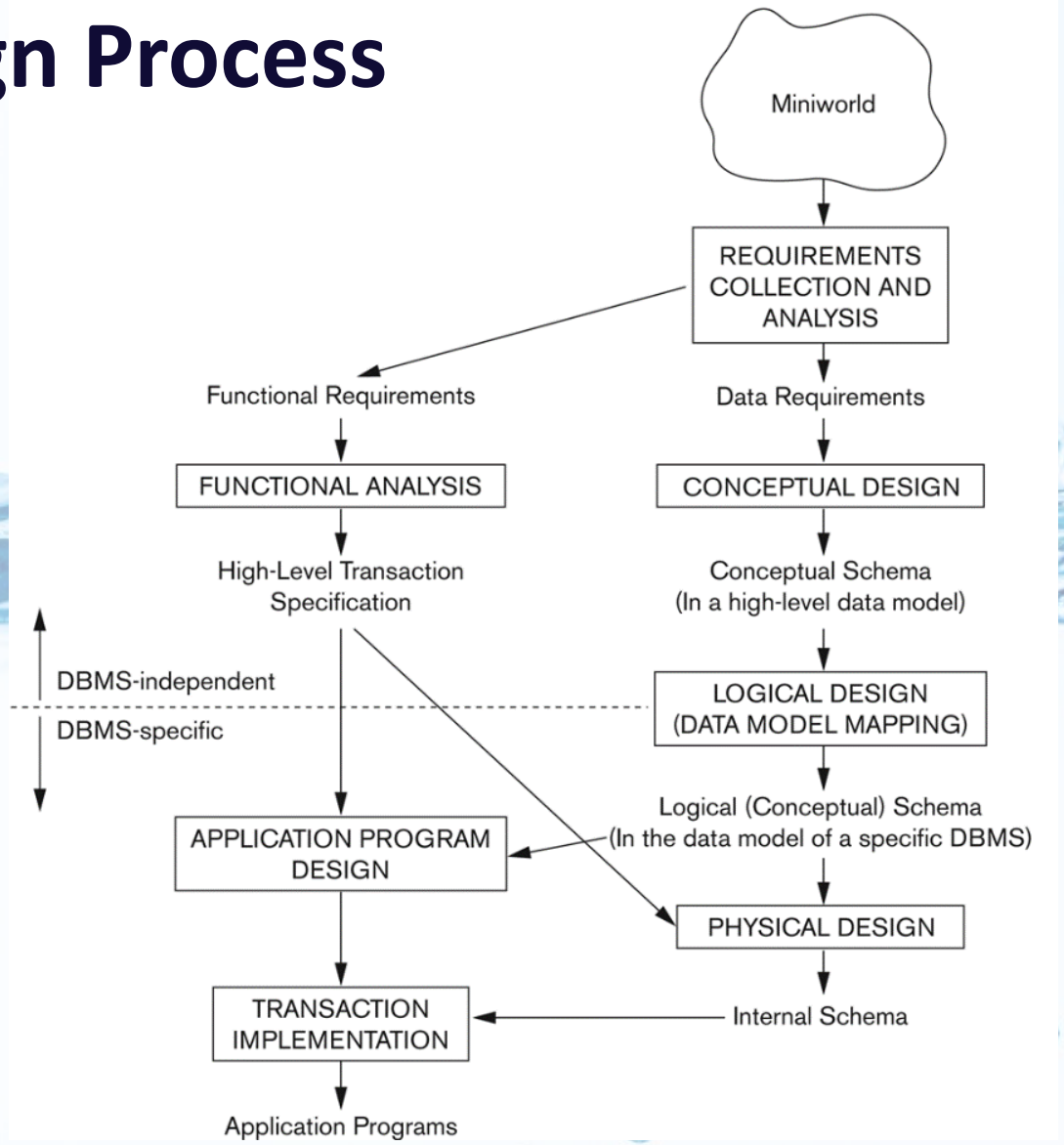
Data Modeling with Entity-Relationship (ER) Model



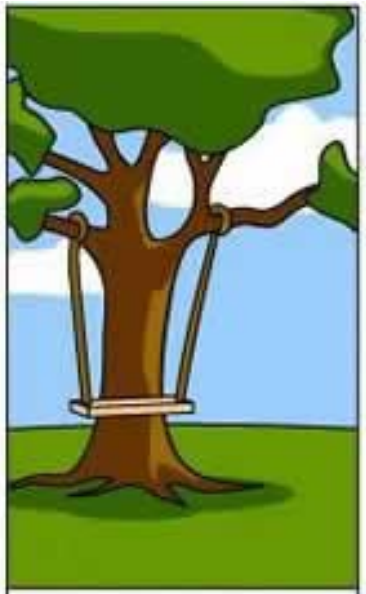
Overview of Database Design Process

- Two main activities
  - Database design
  - Applications design
- Conceptual database design
  - To design the conceptual schema for a database application
- Applications design
  - Focus on the programs and interfaces that access the database
  - Considered as part of software engineering

Overview of Database Design Process

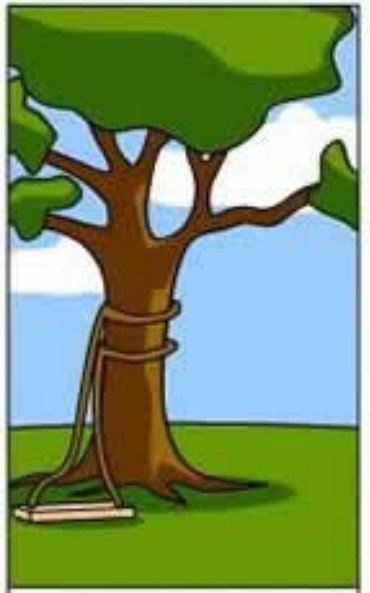


How the customer explained it



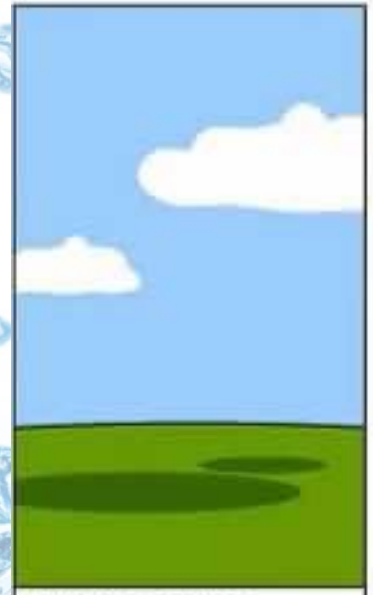
How the Project Leader understood it

How the Analyst designed it

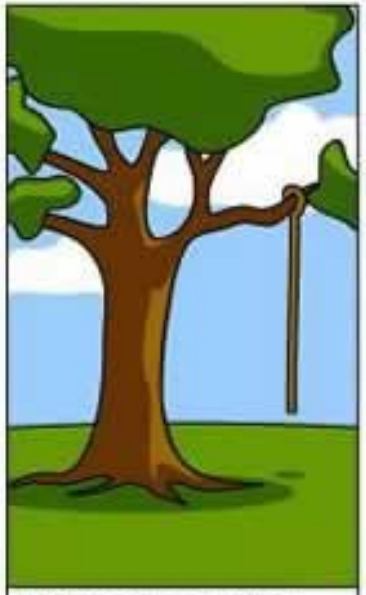


How the Programmer wrote it

How the Business Consultant described it

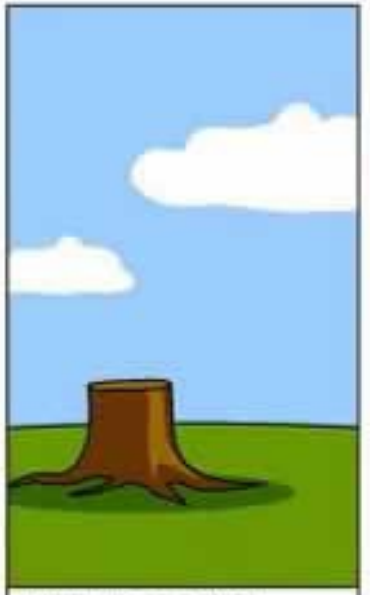


How the project was documented



What operations installed

How the customer was billed



How it was supported

What the customer really needed

COSC 3380 - 19717
Design of Database Systems

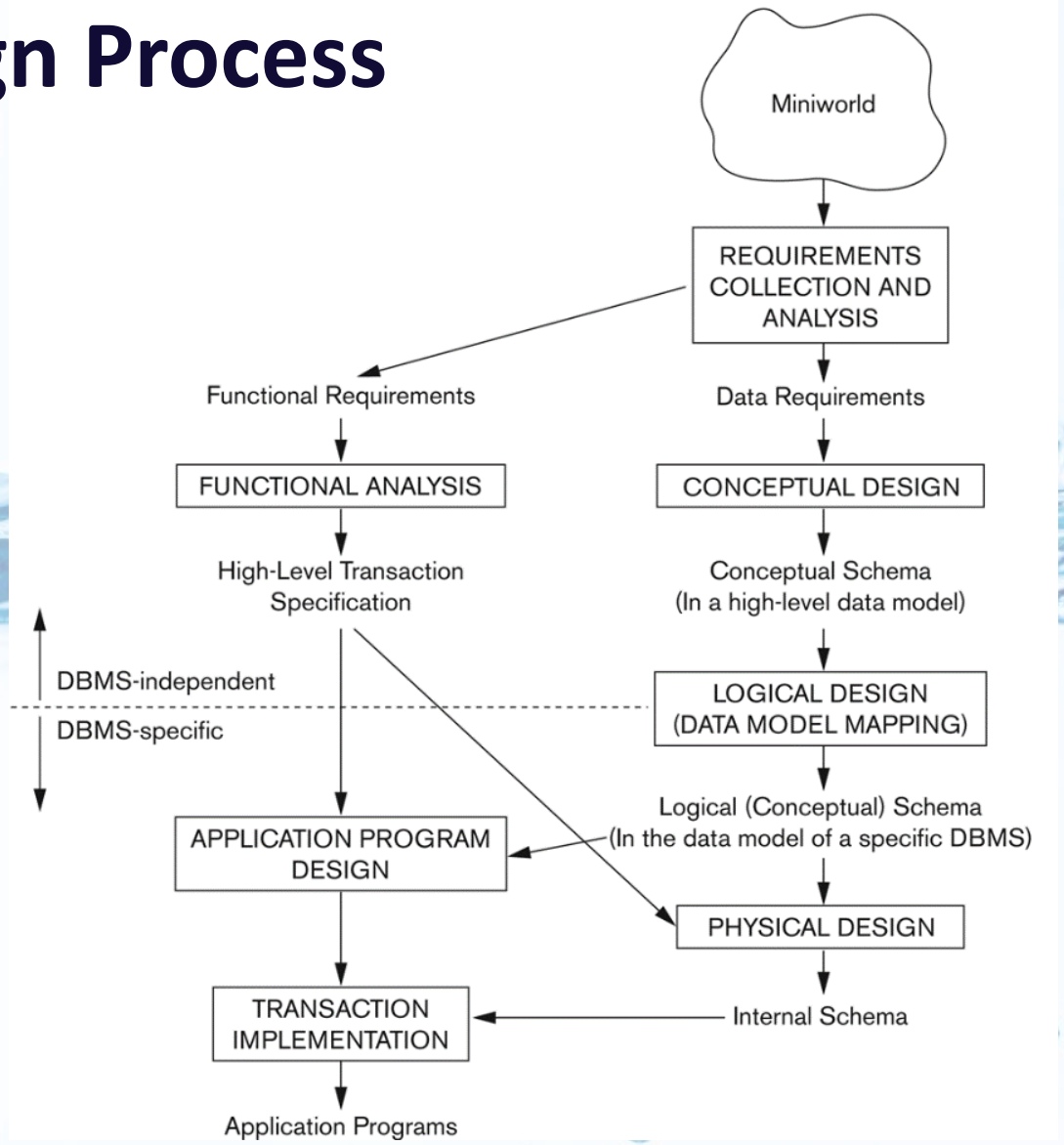
Data Modeling with Entity-Relationship (ER) Model

September 16, 2025

Overview of Database Design Process

Two main activities

- Database design
- Applications design



Example: COMPANY Database

- Create a database schema 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

- The database will store each EMPLOYEE's social security number, address, salary, sex, 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
  - Also 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, sex, 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
    - Eg.: the EMPLOYEE - John Smith, the Research DEPARTMENT, the ProductX PROJECT
  - Attributes are properties used to describe an entity
    - Eg.: an EMPLOYEE entity may have the attributes of Name, SSN, Address, Gender, BirthDate

ER Model Concepts

- **Entities and Attributes**
  - A specific entity will have a value for each of its attributes
    - Eg.: 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 – integer, string/char, date, or enumerated type

Types of Attributes

- **Simple**
  - Each entity has a single atomic value for the attribute
  - Eg.: SSN, Gender
- **Composite**
  - The attribute may be composed of several components
    - Address (Apt#, House#, Street, City, State, ZipCode, Country)
    - Name(FirstName, MiddleName, LastName)
    - Composition may form a hierarchy where some components are themselves composite

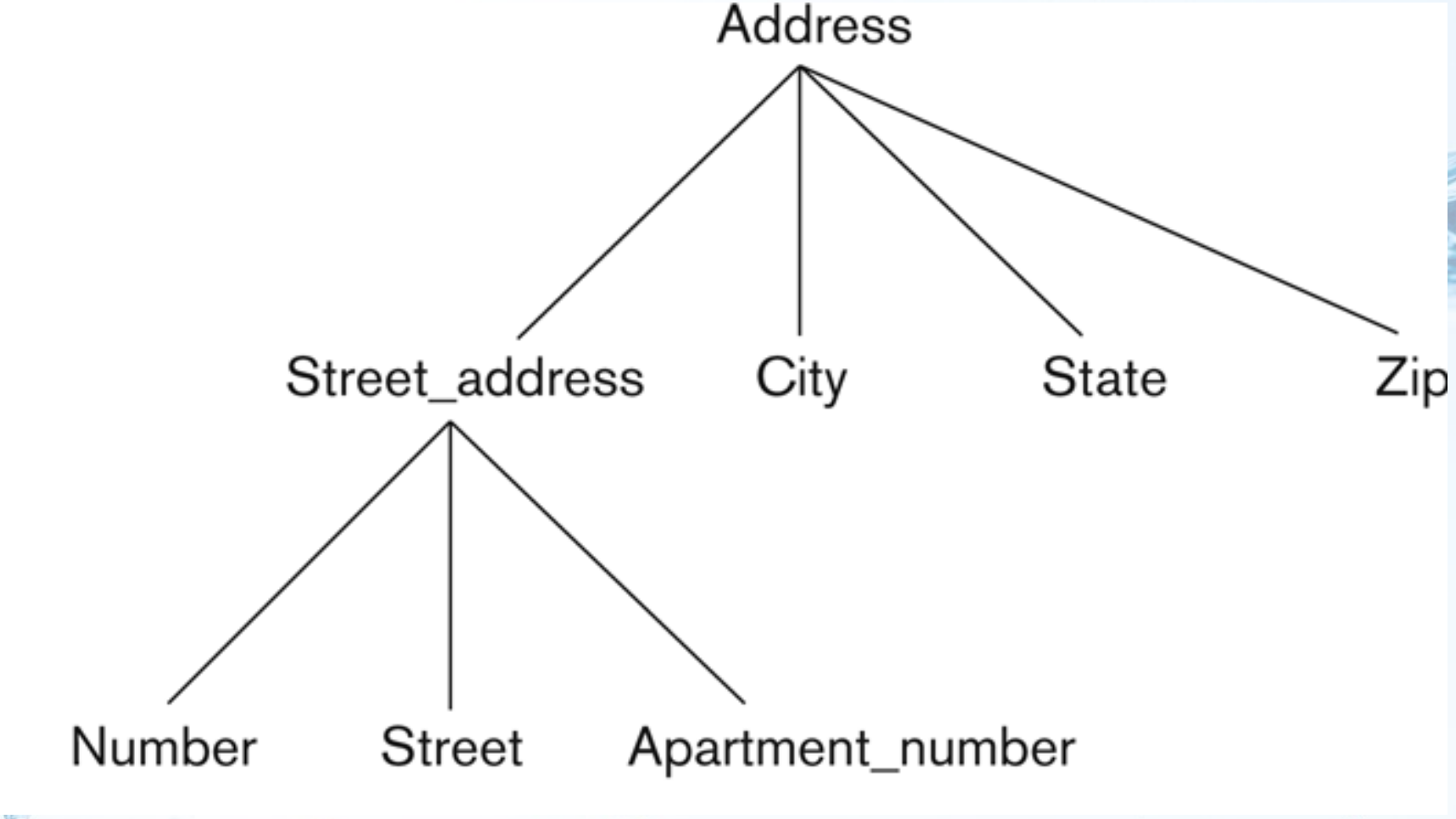
Types of Attributes

- **Multi-valued**
  - An entity may have multiple values for that attribute.
    - *Color* of a CAR
    - *Previous_Degrees* of a STUDENT
    - Denoted as {Color} or {Previous_Degrees}

Types of Attributes

- Composite and multi-valued attributes may be nested arbitrarily to any number of levels
  - Previous_Degrees of a STUDENT - composite multi-valued attribute
    - {Previous_Degrees (College, Year, Degree, Field)}
  - Multiple Previous_Degrees values can exist
  - Each has four subcomponent attributes:
    - College, Year, Degree, Field

Example of a composite attribute



Entity Types and Key Attributes

- 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.

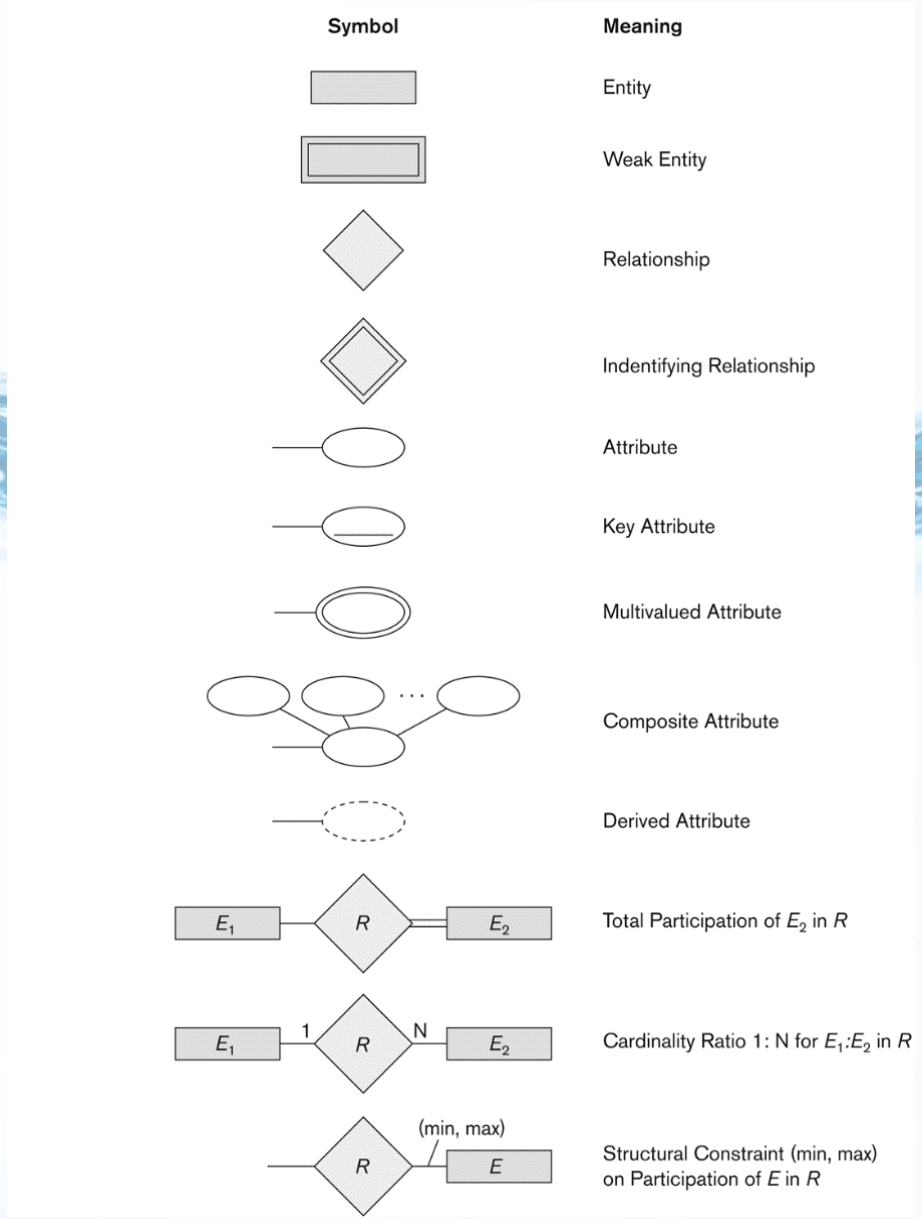
Entity Types and Key Attributes

- 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
  - This is different from the relational schema where only one “primary key” is underlined

Value Sets (Domains) of Attributes

- Each simple attribute is associated with a value set
  - ***Lastname*** has a value which is a character string of up to 30 characters
  - ***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
  - Value sets are similar to data types in most programming languages
  - Integer, character (n), real, bit

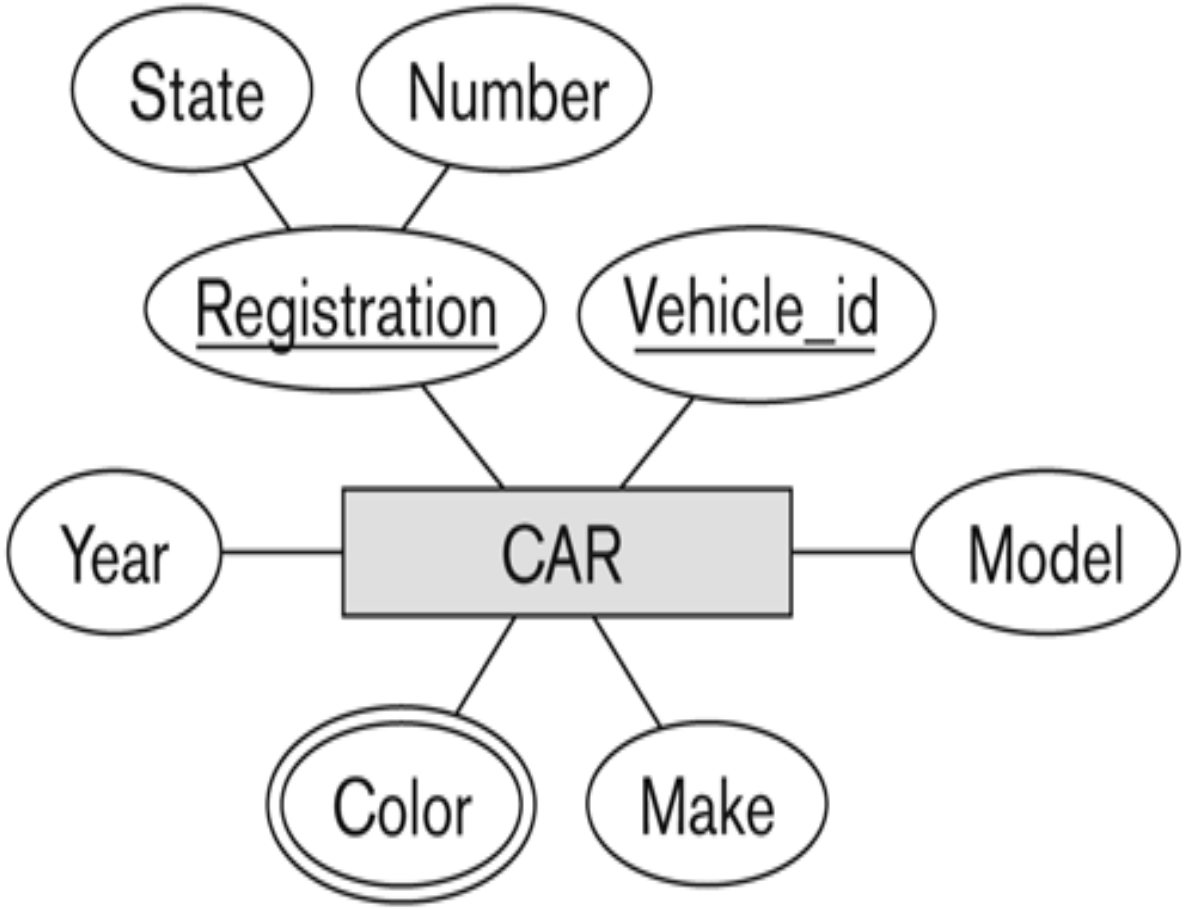
Notation for ER diagrams



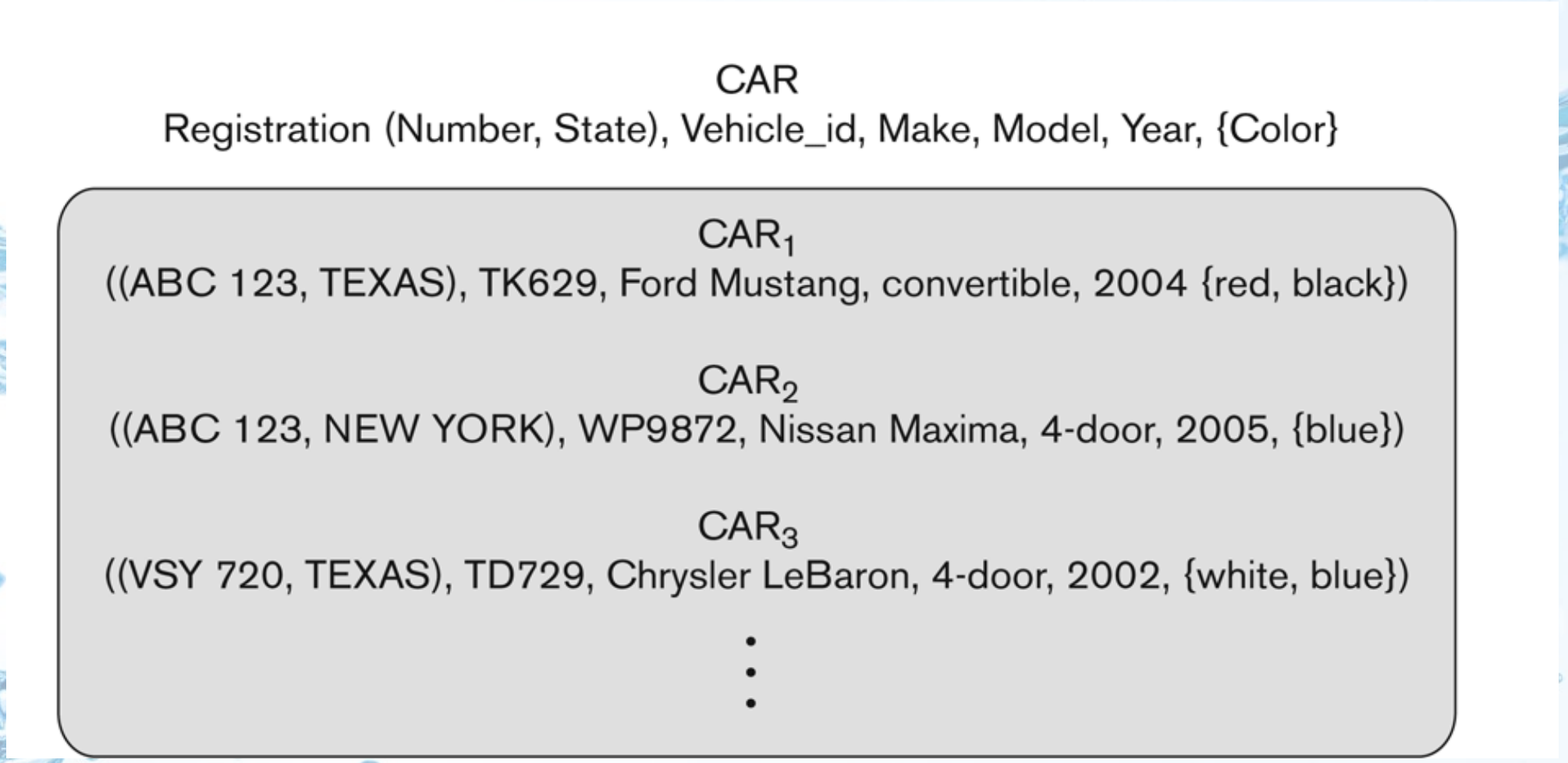
Displaying an Entity type

- 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

Entity Type CAR



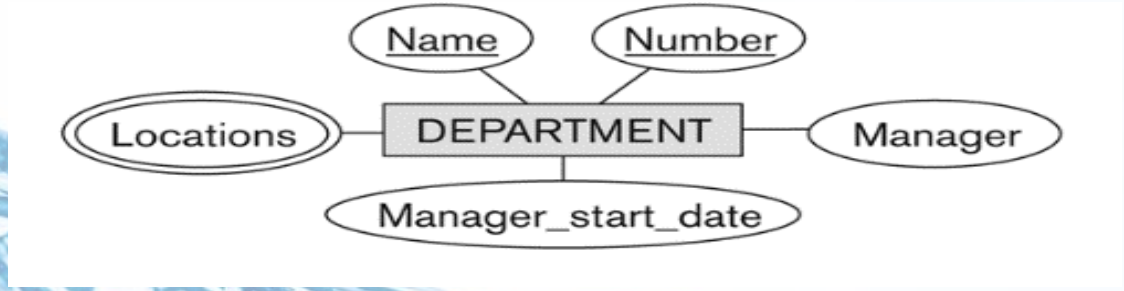
Entity Set for CAR

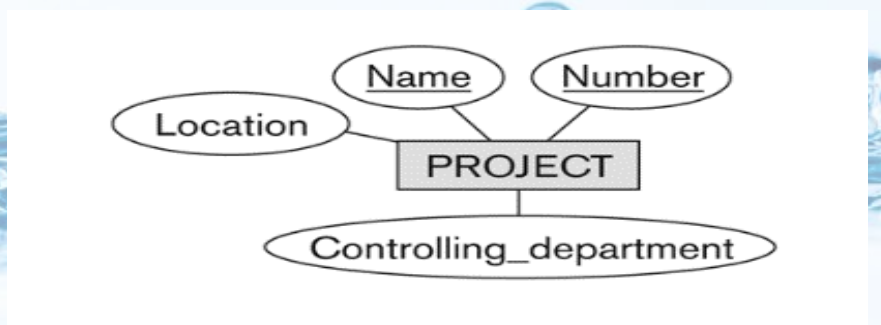


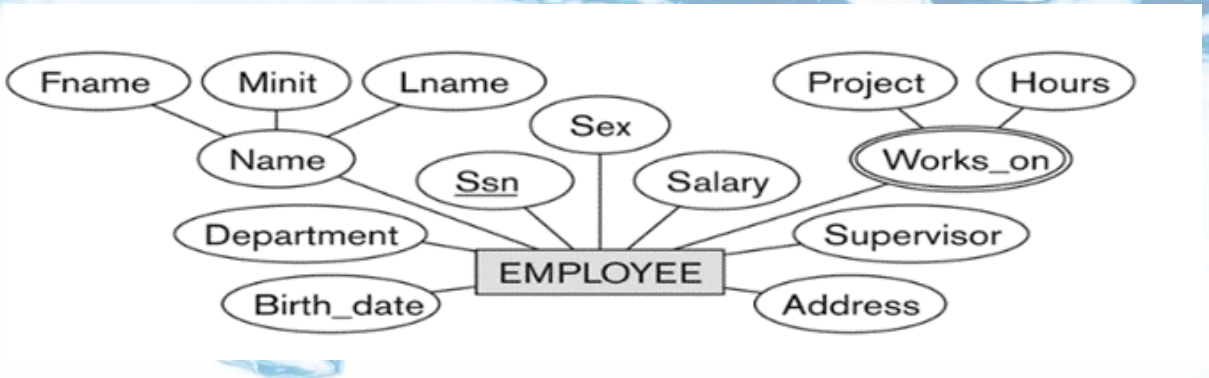
Conceptual Design of Entity Types for COMPANY

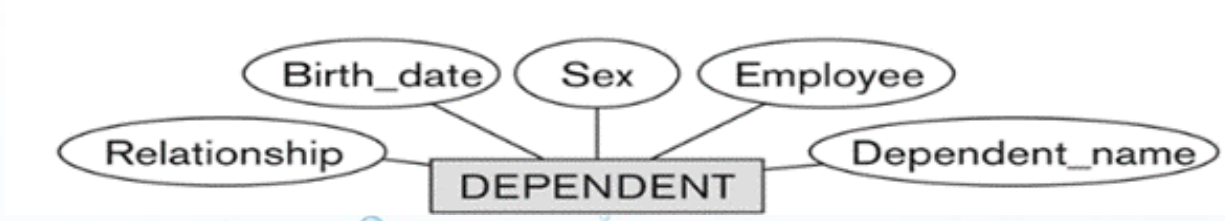
- Four initial entity types in the COMPANY database:
  - DEPARTMENT
  - PROJECT
  - EMPLOYEE
  - DEPENDENT

Initial Design of Entity Types









COSC 3380 - 19717
Design of Database Systems

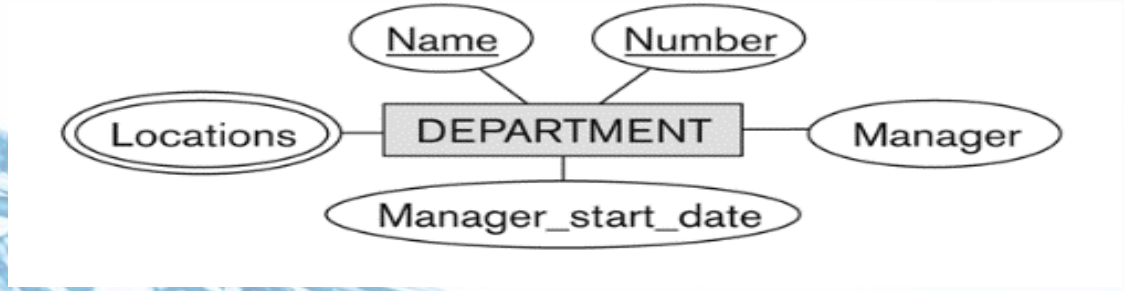
Data Modeling with Entity-Relationship (ER) Model

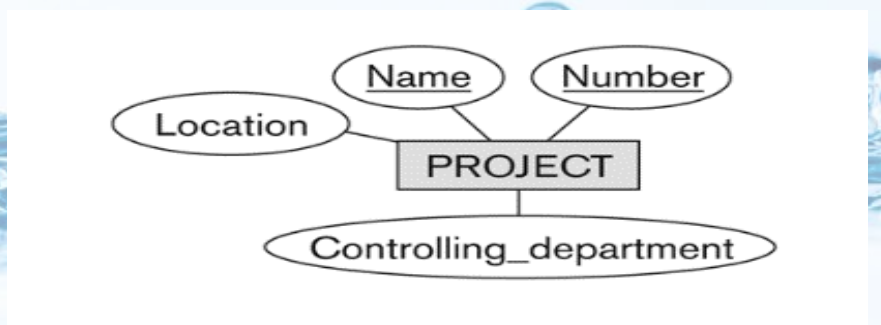
September 18, 2025

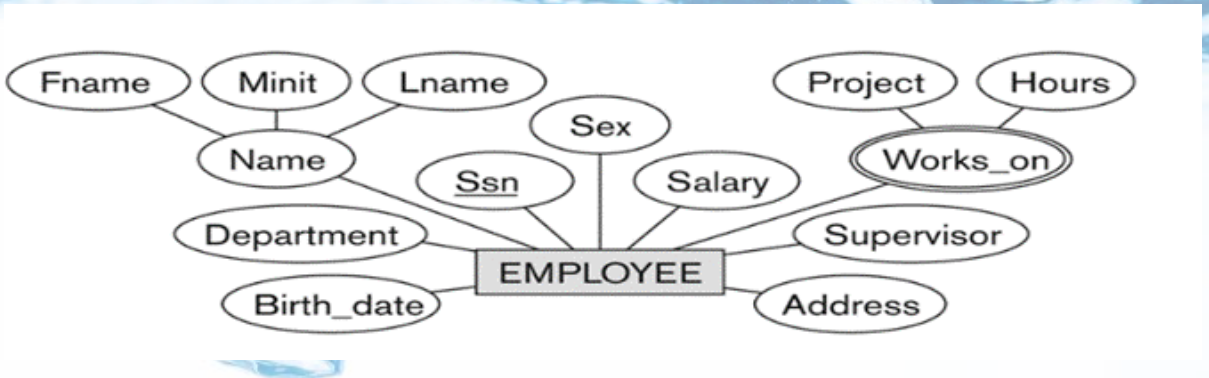
Conceptual Design of Entity Types for COMPANY

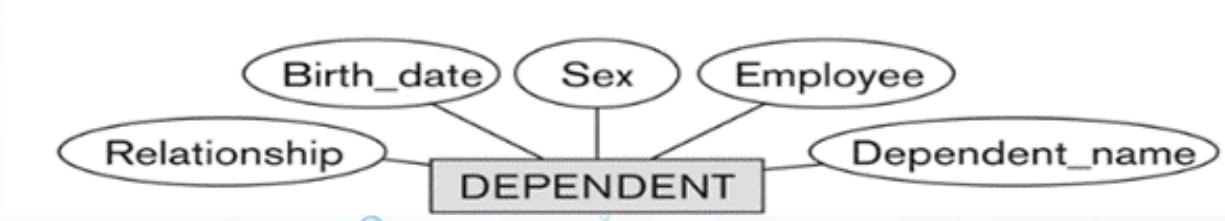
- Four initial entity types in the COMPANY database:
  - DEPARTMENT
  - PROJECT
  - EMPLOYEE
  - DEPENDENT

Initial Design of Entity Types





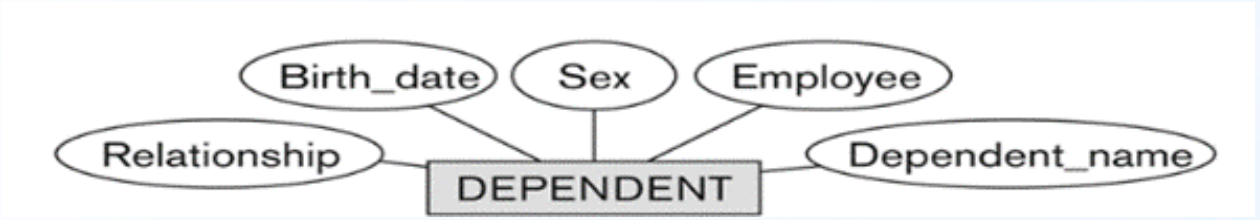




Weak Entity Types

- An entity that does not have a key attribute
  - identification-dependent on another entity type
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Weak 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

Weak Entity Types



- A DEPENDENT entity is identified by the dependent's first name
- 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

Refine the initial design

- The initial design is typically not complete
- Some aspects in the requirements will be represented as **relationships**
- ER model has three main concepts:
  - Entities (entity types and entity sets)
  - Attributes (simple, composite, multivalued)
  - Relationships (relationship types and relationship sets)

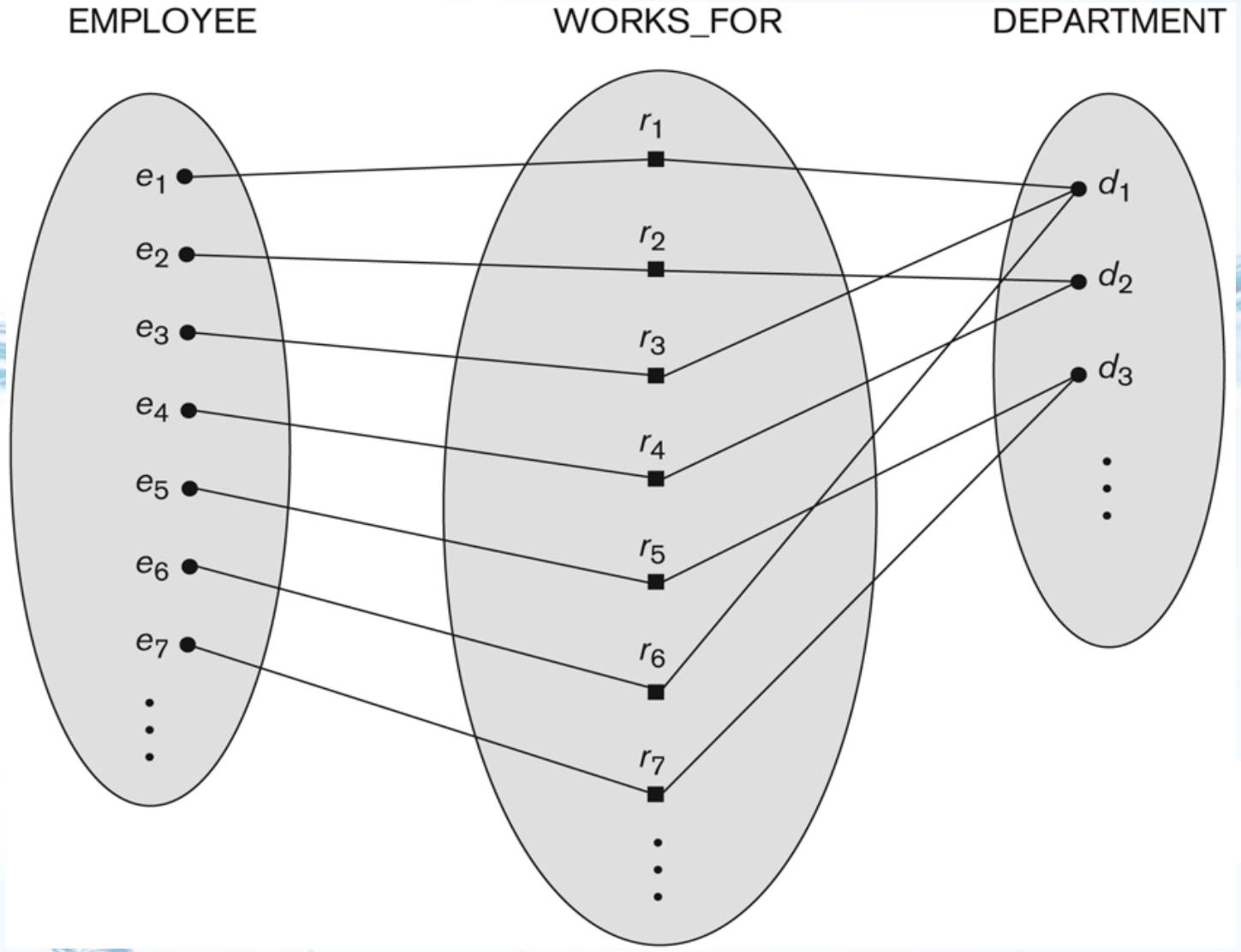
Relationships and Relationship Types

- A **relationship** relates two or more distinct entities with a specific meaning.
  - EMPLOYEE John Smith *works on* the ProductX PROJECT
  - EMPLOYEE Franklin Wong *manages* the Research DEPARTMENT.

Relationships and Relationship Types

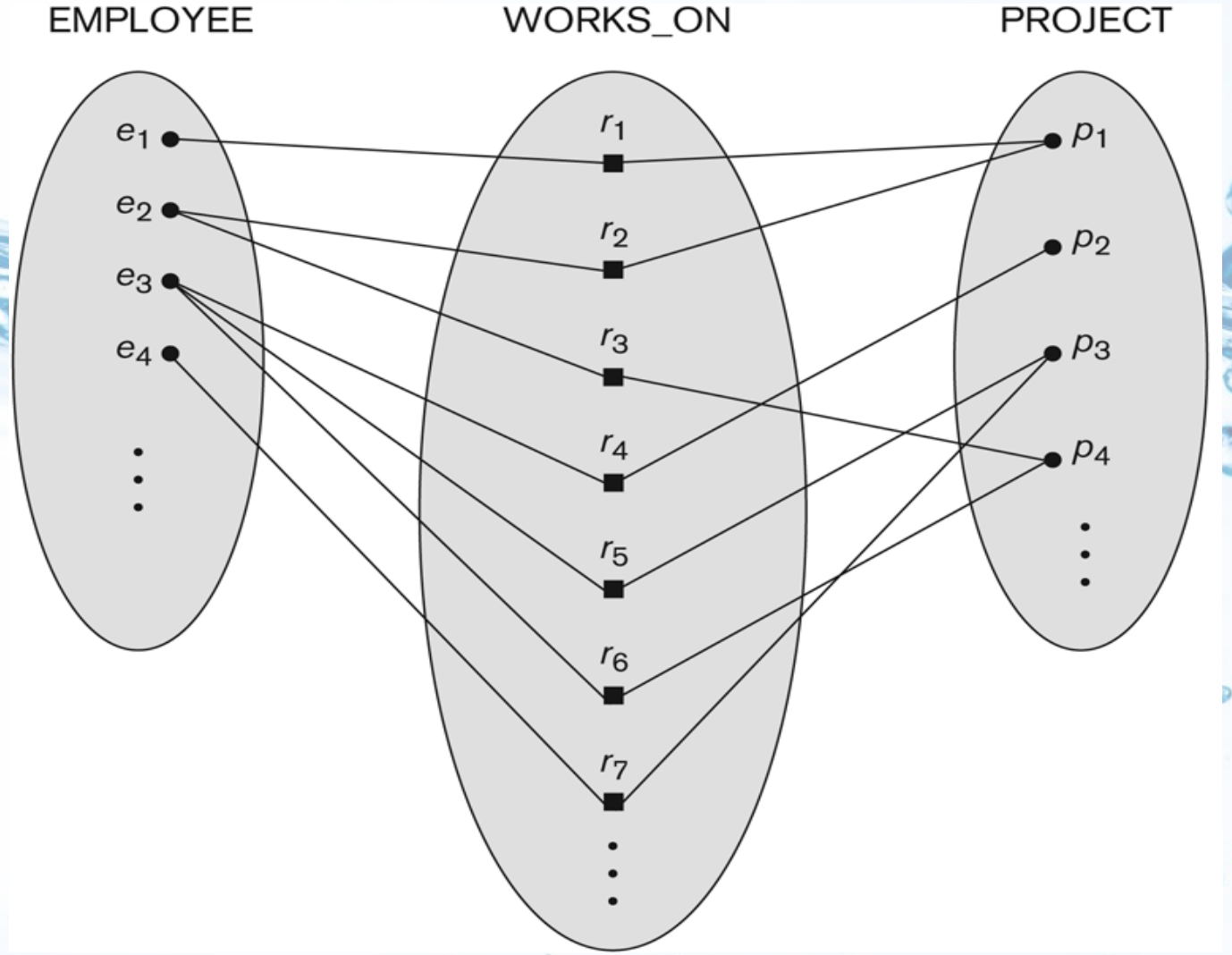
- Relationships of the same type are grouped or typed into a **relationship type**
  - WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate
  - MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate
- The degree of a relationship type is the number of participating entity types
  - Both MANAGES and WORKS_ON are *binary* relationships

WORKS_FOR relationship



N:1

WORKS_ON relationship



M:N

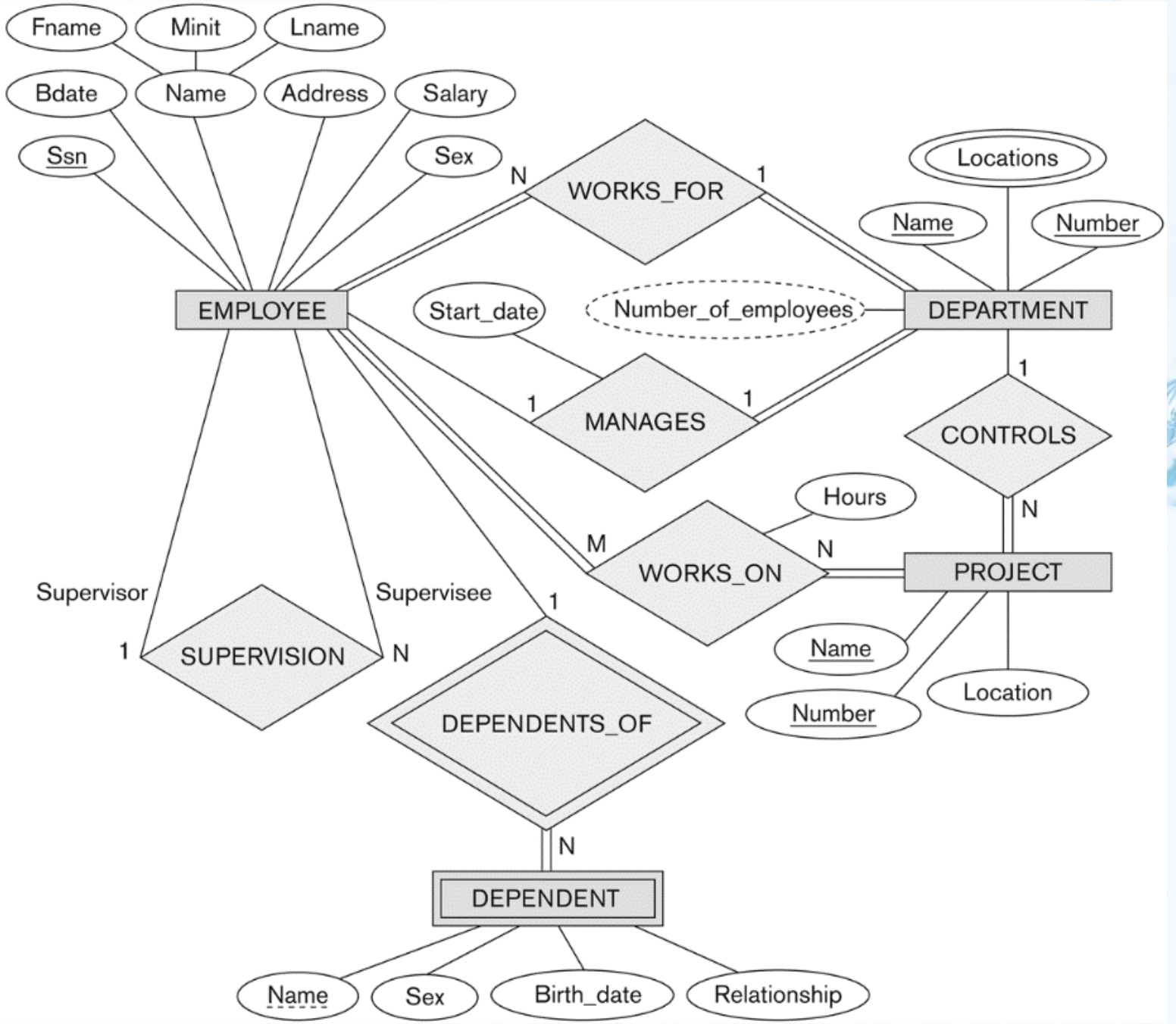
Relationship type vs. relationship set

- 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

Relationships in COMPANY database schema

- By examining the requirements, six relationship types exist
- All are *binary* relationships (degree of 2)
- Relationships 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



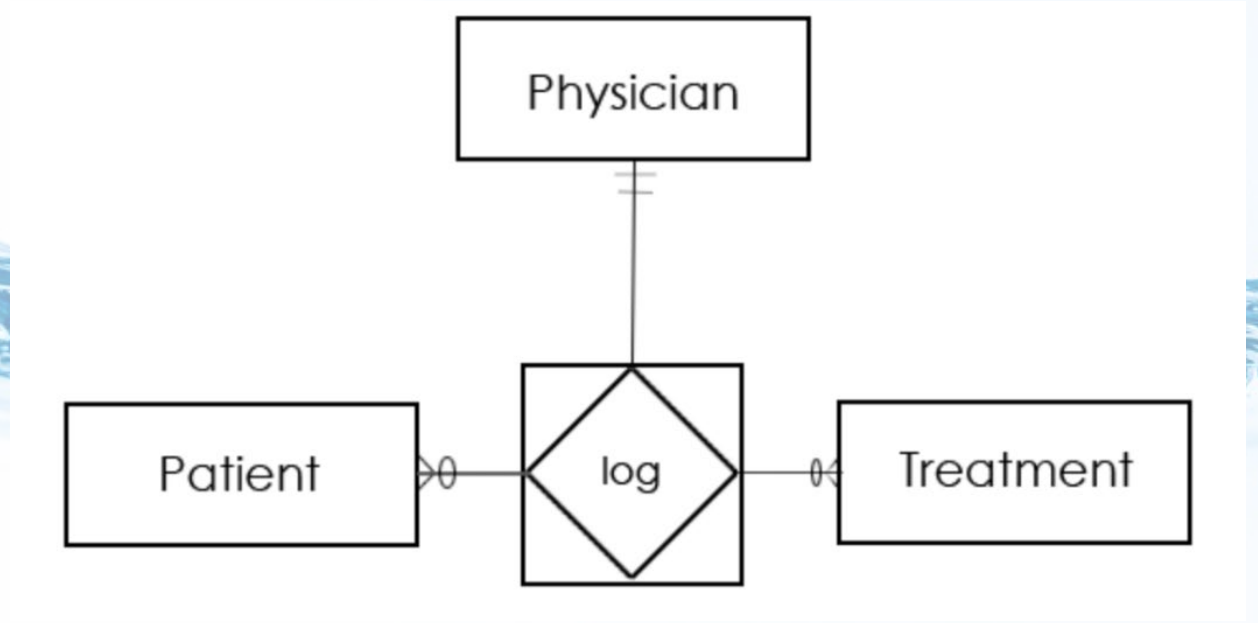
Relationship Types

- 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
- 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)

Example of a ternary relationship



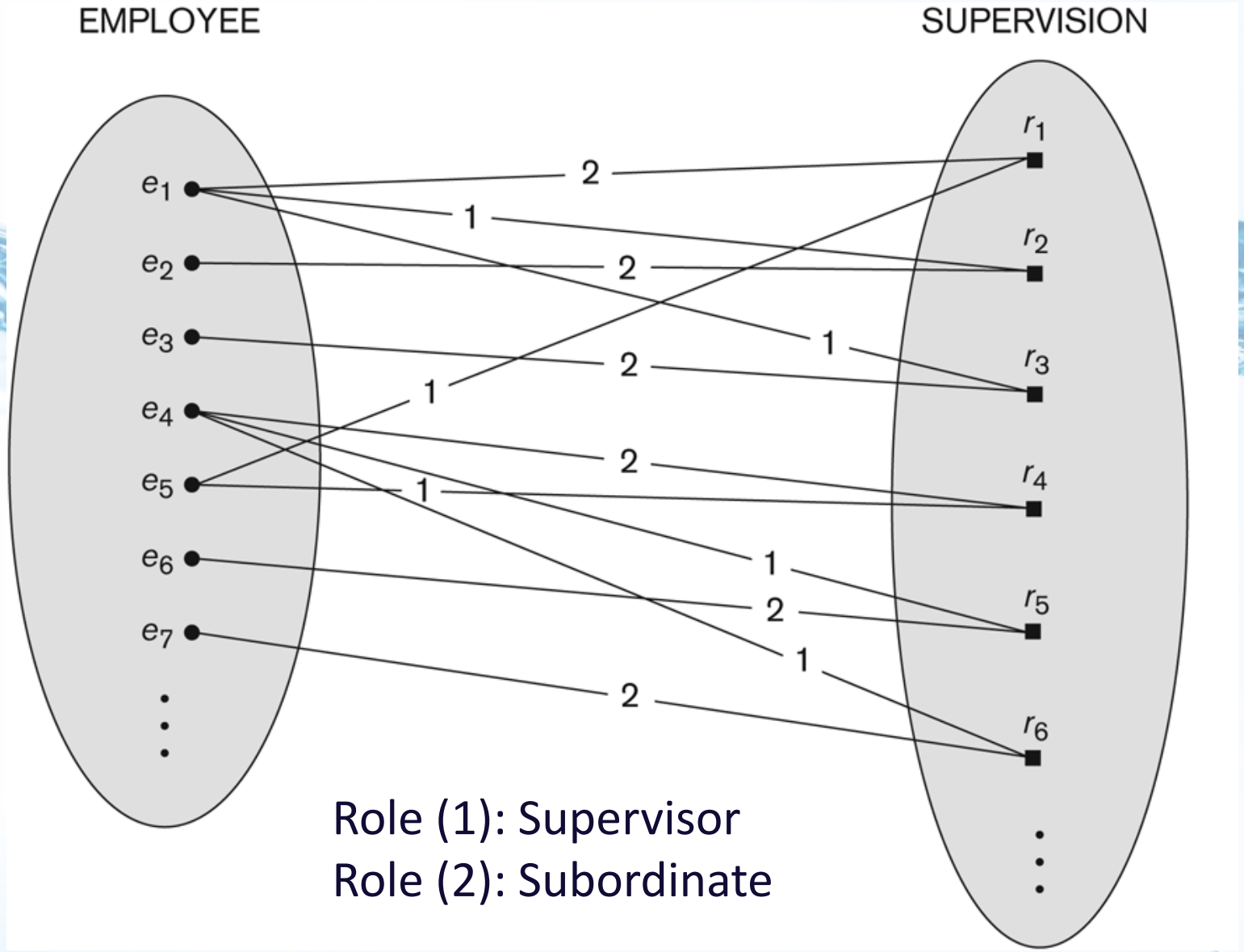
- 1 Physician with 1 Patient can log **M** Treatments
- 1 Physician logs 1 Treatment for **N** Patients
- 1 Patient logged 1 Treatment by **1** Physician

'log' is a M:N:1 relationship between participating entities, Treatment – Patient – Physician

Recursive Relationship Type

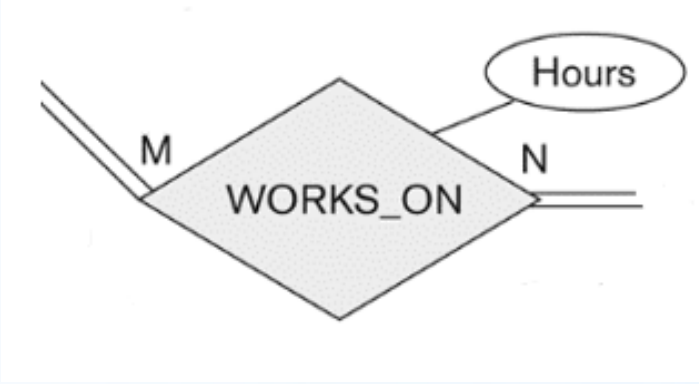
- A relationship type between the same participating entity type in **distinct roles**
- A **self-referencing** relationship type
- Example: the SUPERVISION relationship
- EMPLOYEE participates twice in two distinct roles:
  - supervisor (or boss) role
  - supervisee (or subordinate) role
- Each relationship instance relates two distinct EMPLOYEE entities:
  - One employee in *supervisor* role
  - One employee in *supervisee* role

Recursive Relationship - Supervision

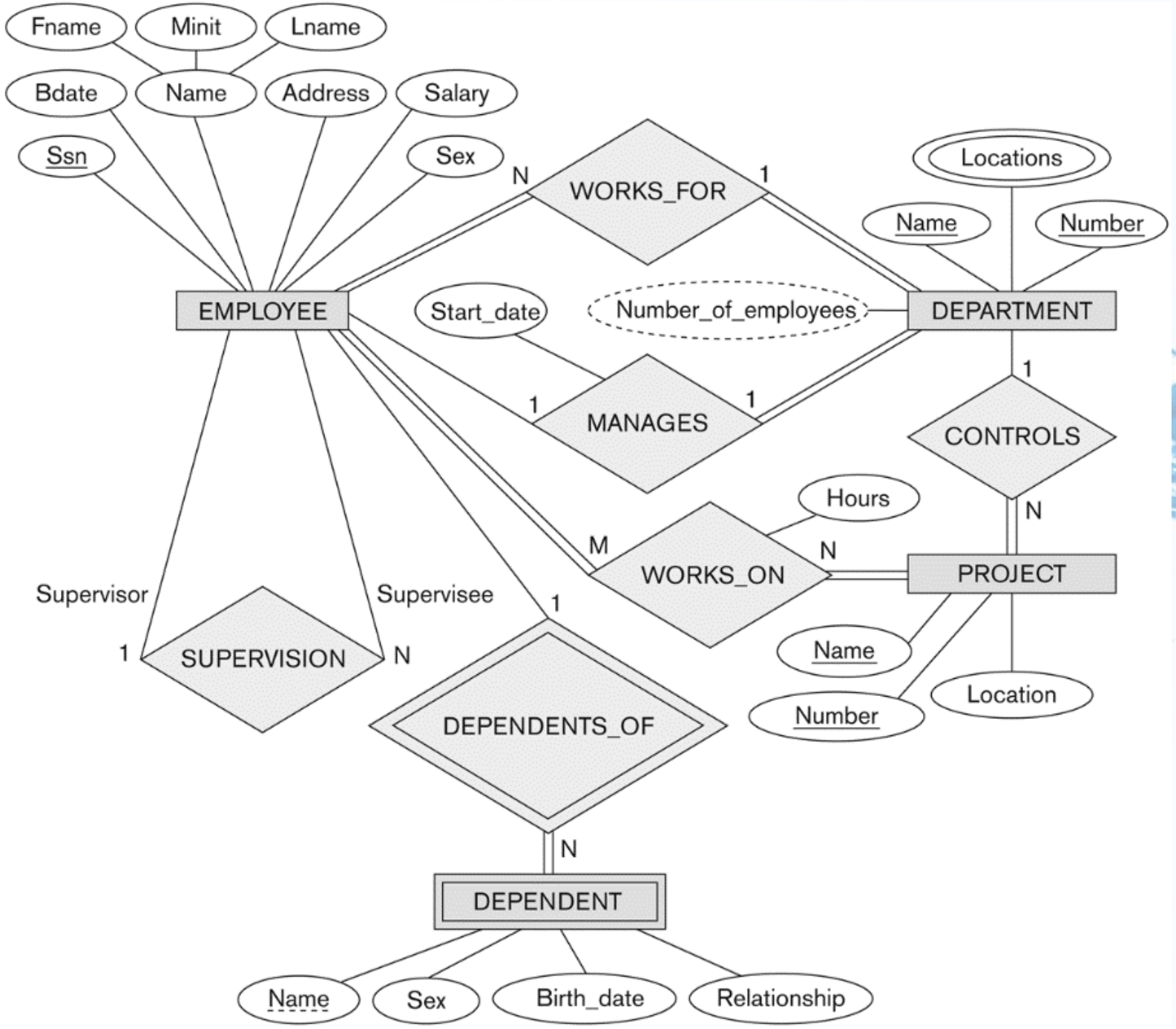


Role (1): Supervisor
Role (2): Subordinate

Attributes of Relationship types



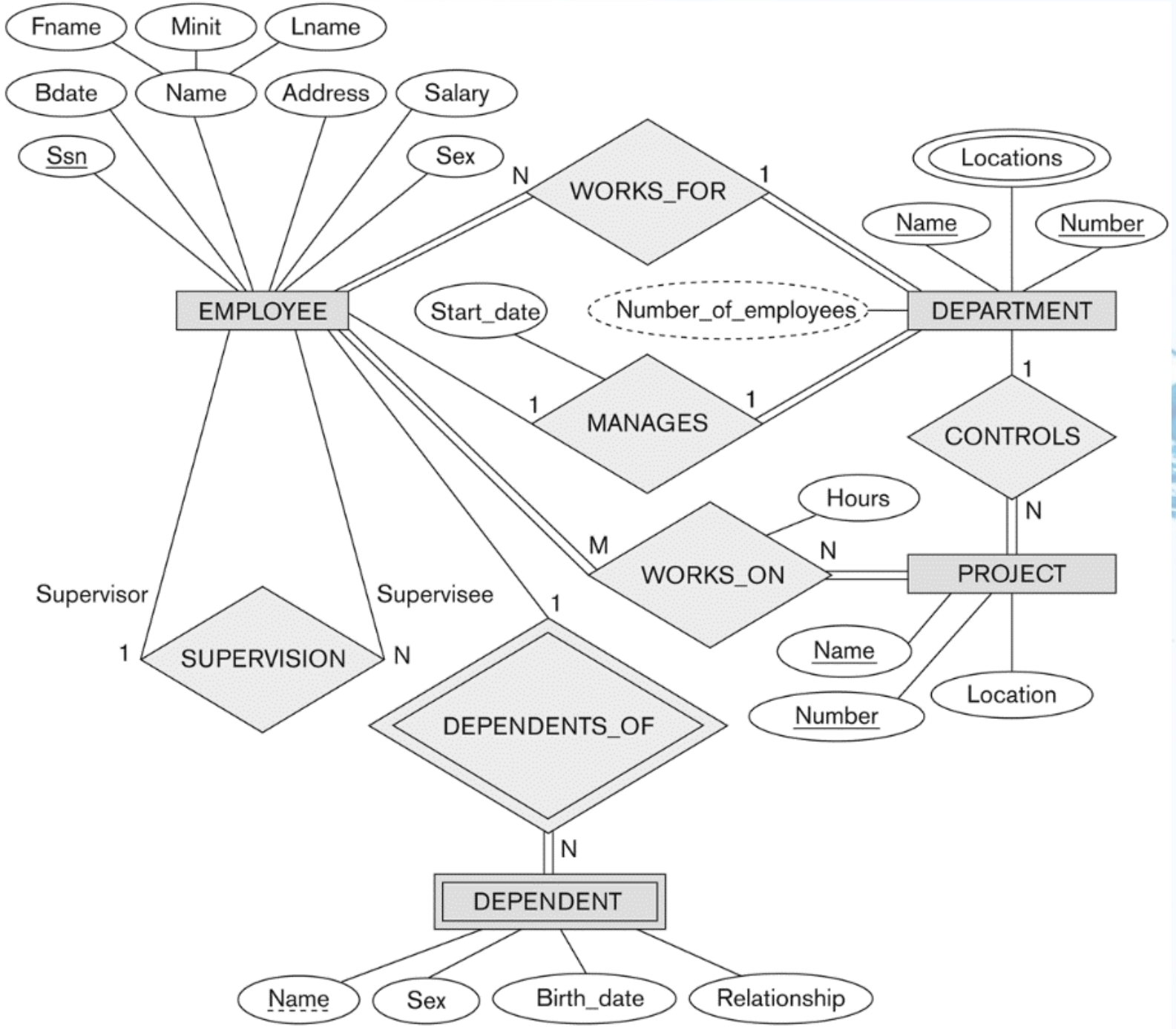
- A relationship type can have attributes
  - For example, HoursPerWeek of WORKS_ON
  - Value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.
    - A value of HoursPerWeek depends on a particular (employee, project) combination
  - Most relationship 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


Fname
Minit
Lname
Name
Bdate
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
N
WORKS_ON
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

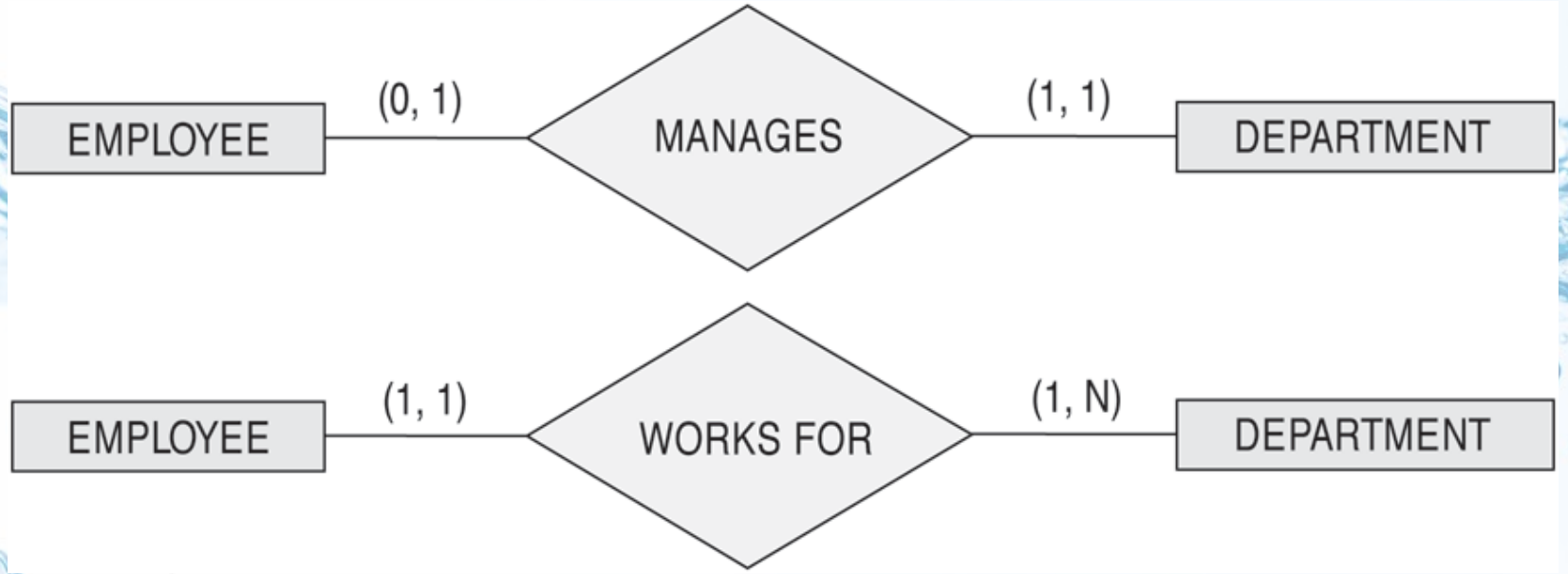
COSC 3380 - 19717
Design of Database Systems

Data Modeling with Entity-Relationship (ER) Model

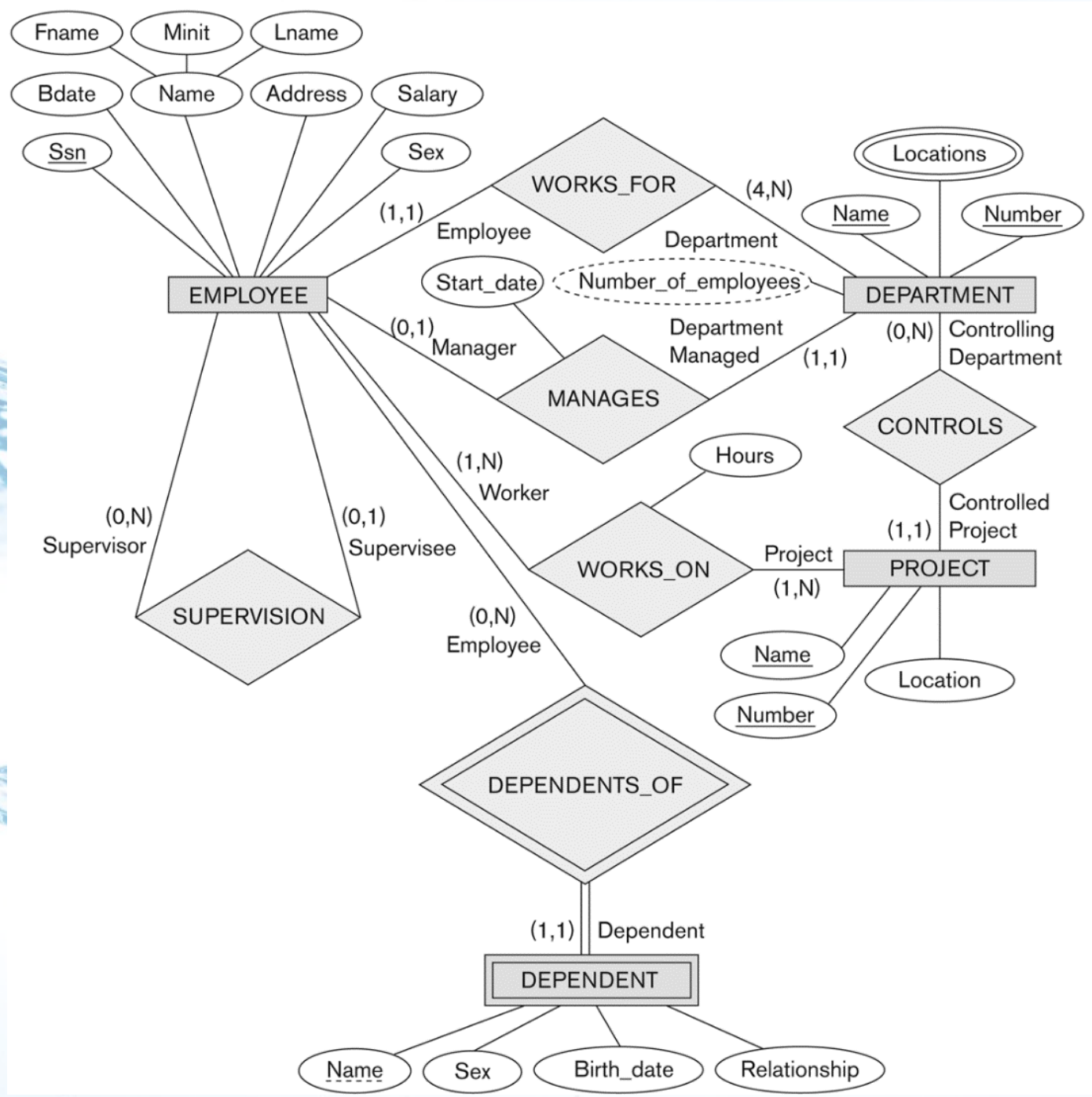
September 23, 2025


Fname
Minit
Lname
Name
Bdate
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
N
WORKS_ON
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

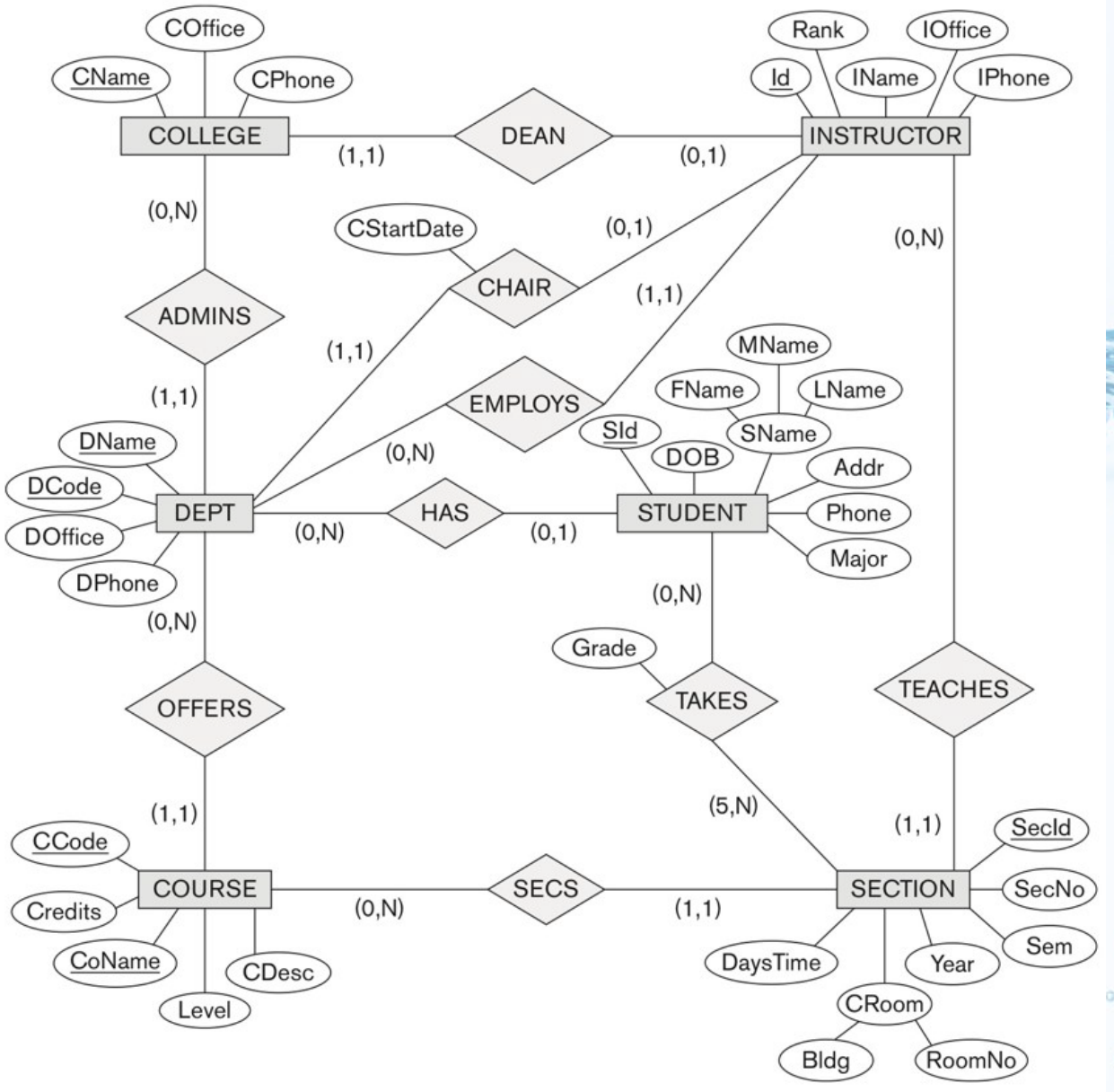
(min,max) notation for relationship constraints



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


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
WORKS_FOR
(4,N)
(1,1)
Employee
Department
Locations
Name
Number
Start_date
Number_of_employees
EMPLOYEE
DEPARTMENT
(0,1)
Manager
Department
Managed
(1,1)
(0,N)
Controlling
Department
MANAGES
CONTROLS
Hours
(1,N)
Worker
Controlled
Project
(1,1)
(0,N)
Supervisor
(0,1)
Supervisee
WORKS_ON
Project
(1,N)
PROJECT
SUPERVISION
(0,N)
Employee
Name
Location
Number
DEPENDENTS_OF
(1,1)
Dependent
DEPENDENT
Name
Sex
Birth_date
Relationship

UNIVERSITY database conceptual schema



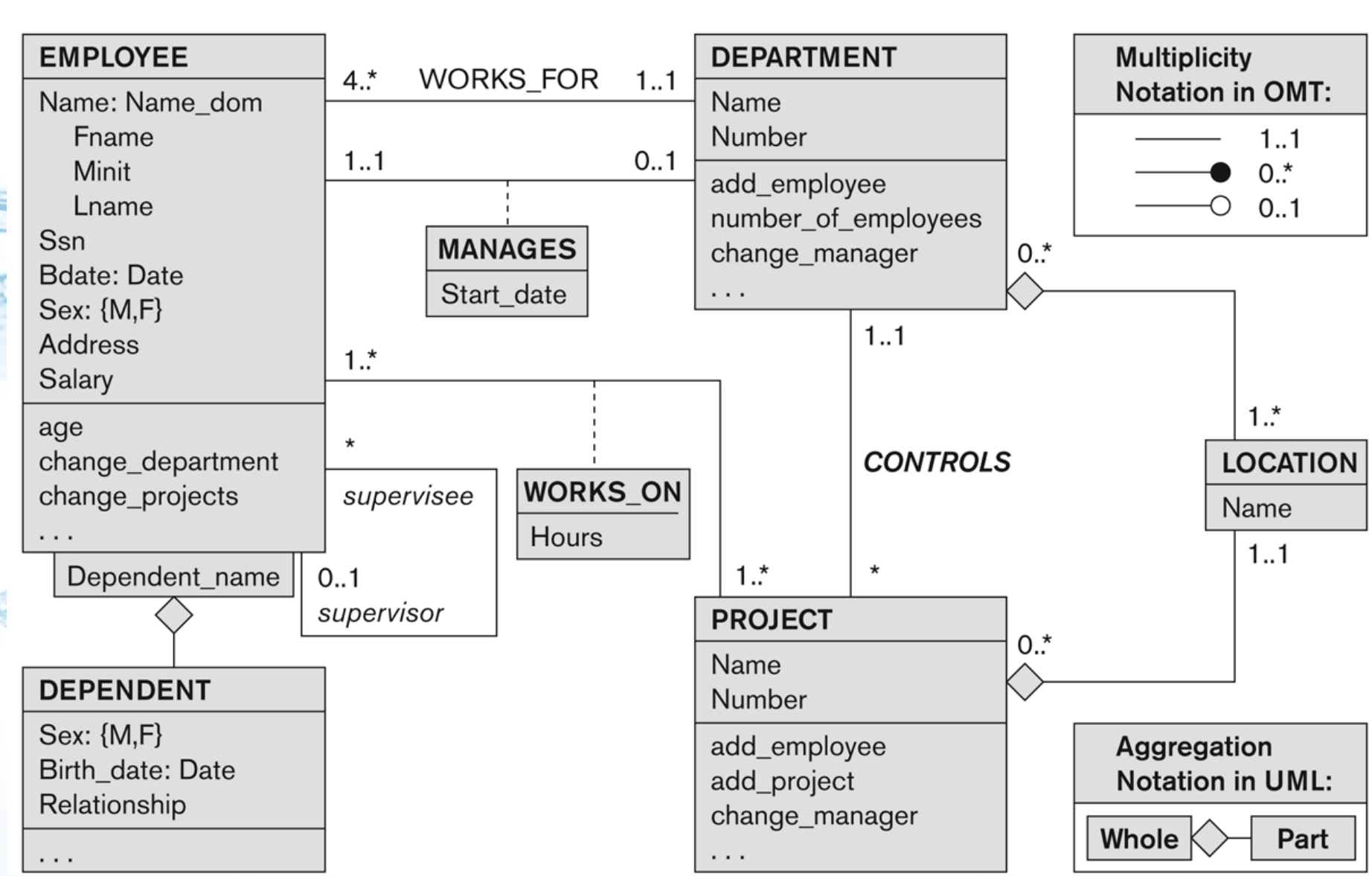
Alternative diagrammatic notation

- ER diagrams is one popular example for displaying database schemas
- Many other notations exist
- UML class diagrams are used in commercial/software design tools

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



Database Catalog

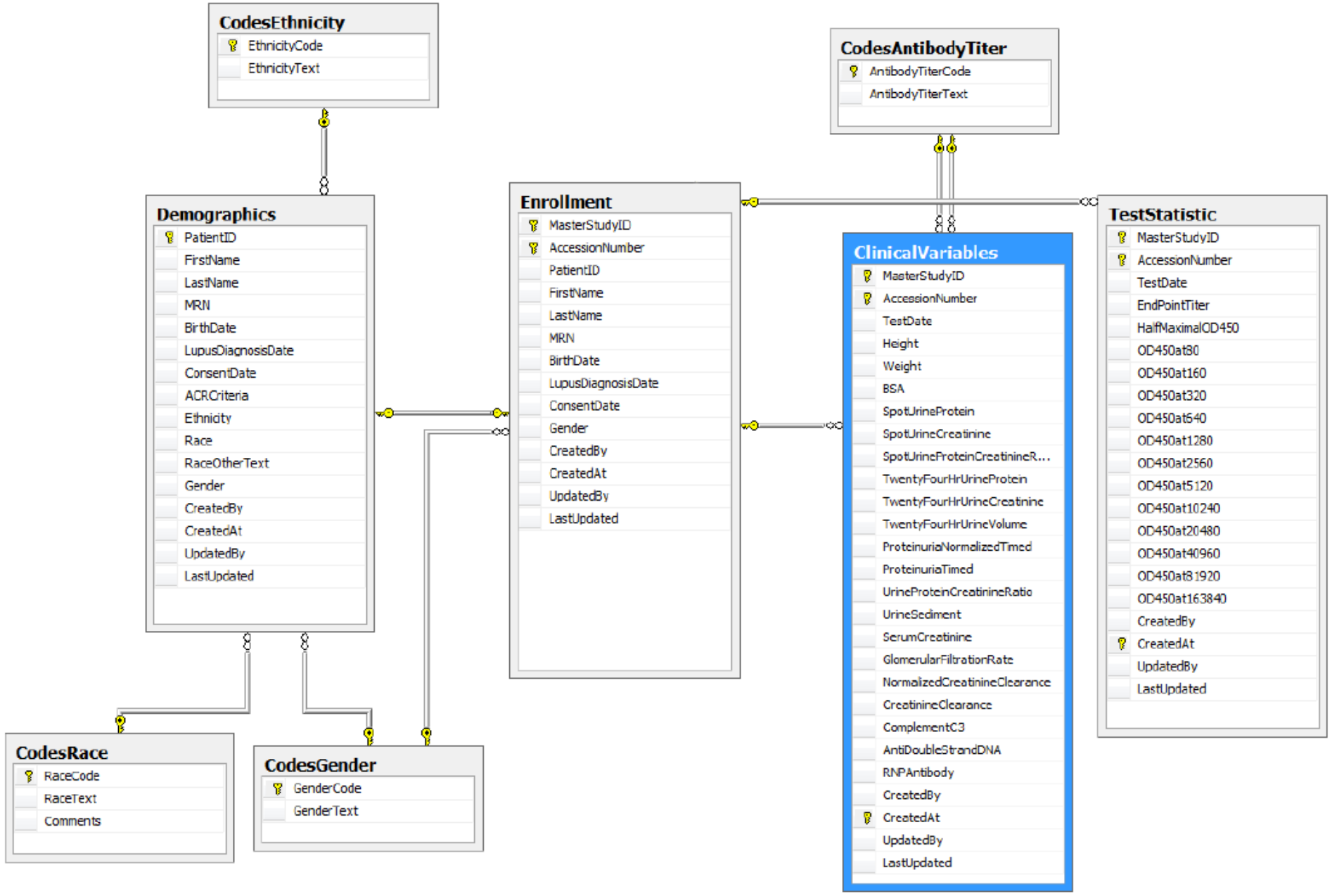
Demographics

Column Name	Data Type	Allow Nulls
PatientID	int	☐
FirstName	nvarchar(50)	☑
LastName	nvarchar(50)	☑
MRN	nvarchar(50)	☑
BirthDate	date	☑
LupusDiagnosisDate	date	☑
ConsentDate	date	☑
ACRCriteria	nvarchar(50)	☑
Ethnicity	smallint	☑
Race	smallint	☑
RaceOtherText	nvarchar(250)	☑
Gender	smallint	☑
CreatedBy	nvarchar(50)	☐
CreatedAt	datetime	☐
UpdatedBy	nvarchar(50)	☐
LastUpdated	datetime	☐
		☐

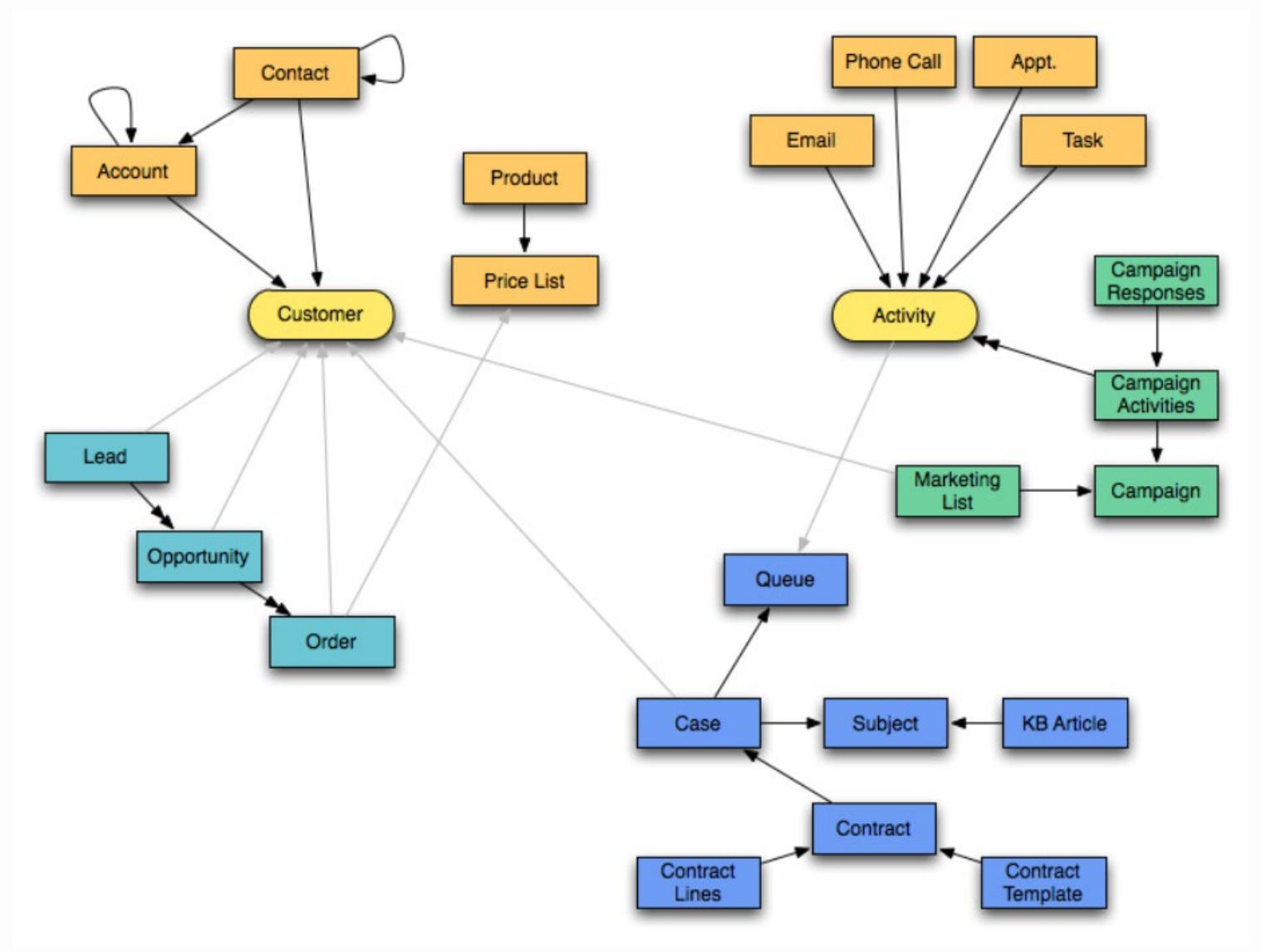
Enrollment

Column Name	Data Type	Allow Nulls
MasterStudyID	smallint	☐
AccessionNumber	int	☐
PatientID	int	☐
FirstName	nvarchar(50)	☑
LastName	nvarchar(50)	☑
MRN	nvarchar(50)	☑
BirthDate	date	☑
LupusDiagnosisDate	date	☑
ConsentDate	date	☑
Gender	smallint	☑
CreatedBy	nvarchar(50)	☐
CreatedAt	datetime	☐
UpdatedBy	nvarchar(50)	☐
LastUpdated	datetime	☐
		☐

Database Diagram



Relational Data Model and Relational Database Constraints



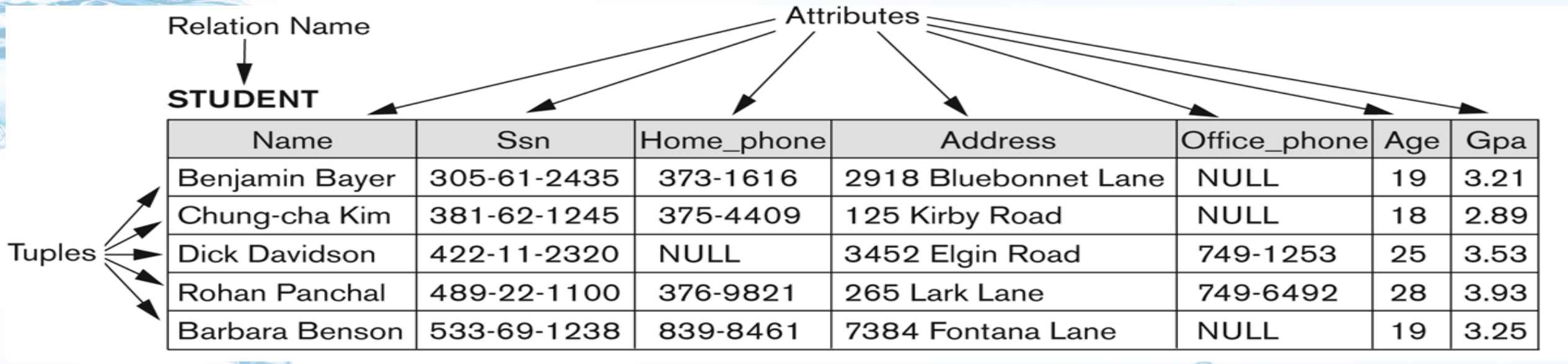
Relational Model Concepts

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

Informal Definitions

- 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



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

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
  - In the STUDENT table, SSN is the key
  - Sometimes row-ids or sequential/random numbers are assigned as keys to identify the rows in a table
    - Called ***artificial key*** or ***surrogate key***
    - ***Why?***

Formal Definitions - Schema

- The **Schema** (or description) of a Relation:
  - Denoted by $R(A_1, A_2, ...A_n)$
  - R is the **name** of the relation
  - The **attributes** of the relation are $A_1, A_2, ..., A_n$
- **Degree** (or **arity**) of a relation
  - Number of attributes *n* of its relation schema
- CUSTOMER (Cust-id, Cust-name, Address, Phone#)
  - CUSTOMER is the relation name (Degree of 4)
  - Defined over the four attributes: Cust-id, Cust-name, Address, Phone#
- Each attribute has a **domain** or a set of valid values.
  - For example, the domain of Cust-id is 6 digit numbers.

Attributes and Relations Examples

- A Relation of degree seven for university students
  - STUDENT (Name, SSN, Address, Home_phone, Office_Phone, Age, GPA)
- Using the data type of each attribute
  - STUDENT (Name: string, SSN:string, Address: string, Home_phone: string, Office_Phone: string, Age: integer, GPA: real)

Domains

- **Domain** D
  - Set of atomic values
- **Atomic**
  - Each value indivisible
- Specifying a domain
  - **Data type** specified for each domain

Domain Examples

- Usa_phone_numbers. The set of ten-digit phone numbers valid in the United States.
- Local_phone_numbers. The set of seven-digit phone numbers valid within a particular area code in the United States. The use of local phone numbers is quickly becoming obsolete, being replaced by standard ten-digit numbers.
- Social_security_numbers. The set of valid nine-digit Social Security numbers. (This is a unique identifier assigned to each person in the United States for employment, tax, and benefits purposes.)
- Names: The set of character strings that represent names of persons.
- Grade_point_averages. Possible values of computed grade point averages; each must be a real (floating-point) number between 0 and 4.
- Employee_ages. Possible ages of employees in a company; each must be an integer value between 15 and 80.
- Academic_department_names. The set of academic department names in a university, such as Computer Science, Economics, and Physics.
- Academic_department_codes. The set of academic department codes, such as 'CS', 'ECON', and 'PHYS'.

Domains, Attributes, Tuples, Relations

- **Relation** (or **relation state**)
  - Set of ***n*-tuples** $r = \{t_1, t_2, ..., t_n\}$
  - Each n-tuple t
    - Ordered list of n values $t = <v_1, v_2, ..., v_n>$
    - Each value v_i, $1 \leq i \leq n$, is an element of $dom(A_i)$ or is a special NULL value

STUDENT

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

Domains, Attributes, Tuples, Relations

- **Cardinality**
  - Total number of values in a domain
- **Current relation state**
  - Relation state at a given time
  - Reflects only the valid tuples that represent a particular state of the real world
  - Change in state of real world changes the relation state
    - What happens to schema R?

Domains, Attributes, Tuples, Relations

- **Attribute names**
  - Indicate different **roles**, or interpretations, for the domain
  - Keep it meaningful
  - *USA_phone_numbers* for different telephone numbers in the Student relation

Definition Summary

Informal Terms	Formal Terms
Table	Relation
Column Header	Attribute
All possible Column Values	Domain
Row	Tuple
Table Definition	Schema of a Relation
Populated Table	State of the Relation

Characteristics of Relations

- Ordering of tuples in a relation
  - Relation defined as a set of tuples
  - Elements have no order among them
- Ordering of values within a tuple and an alternative definition of a relation
  - Order of attributes and values is not that important
  - As long as correspondence between attributes and values maintained

STUDENT

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

Characteristics of Relations

$t = <$ (Address, 3452 Elgin Road),(Name, Dick Davidson),(Ssn, 422-11-2320),(Age, 25),
(Office_phone, (817)749-1253),(Gpa, 3.53),(Home_phone, NULL)>

$t = <$ (Name, Dick Davidson),(Ssn, 422-11-2320),(Home_phone, NULL),(Address, 3452 Elgin Road),
(Office_phone, (817)749-1253),(Age, 25),(Gpa, 3.53)>

Characteristics of Relations

STUDENT

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

- Values and NULLs in tuples
  - Each value in a tuple is atomic
  - **Flat relational model**
    - Composite and multivalued attributes not allowed
  - Multivalued attributes
    - Must be represented by separate relations

COSC 3380 - 19717
Design of Database Systems

Relational Data Model
and
Relational Database Constraints

September 25, 2025

Characteristics of Relations

STUDENT

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

- Values and NULLs in tuples
  - Each value in a tuple is atomic
  - **Flat relational model**
    - Composite and multivalued attributes not allowed
  - Multivalued attributes
    - Must be represented by separate relations

Characteristics of Relations

- NULL values
  - Represent the values of attributes that may be unknown or may not apply to a tuple
  - Meanings for NULL values
    - *Value unknown*
    - *Value exists but is not available*
    - *Attribute does not apply to this tuple (also known as 'value undefined')*

Relational Model Notation

- Relation schema R of degree n
  - Denoted by $R(A_1, A_2, ..., A_n)$
- Uppercase letters Q, R, S
  - Denote relation names
- Lowercase letters q, r, s
  - Denote relation states
- Letters t, u, v
  - Denote tuples

Relational Model Notation

- Name of a relation schema: STUDENT
  - Indicates the current set of tuples in that relation
- Notation: STUDENT(Name, Ssn, ...)
  - Refers only to relation schema
- Attribute *A* can be qualified with the relation name *R* to which it belongs
  - Using the dot notation *R.A*
  - STUDENT.Name, STUDENT.Ssn, STUDENT.Gpa

Relational Model Constraints

- **Constraints**
  - Restrictions on the actual values in a database state
  - Derived from the rules of the mini-world that the database represents

- **Inherent model-based constraints** or **implicit constraints**
  - Inherent in the data model
  - Eg. relational model does not allow a list as a value for any attribute; a relation cannot have duplicate tuples

Relational Model Constraints

- **Schema-based constraints** or **explicit constraints**
  - Can be directly expressed in schemas of the data model
  - Domain constraints, key constraints, constraints on NULLs, entity integrity constraints, referential integrity constraints
- **Application-based** or **semantic constraints** or **business rules**
  - Cannot be directly expressed in schema
  - Expressed and enforced by application program
  - Examples?

Schema-based: Domain Constraints

- Typically include:
  - Numeric data types for integers and real numbers
  - Characters
  - Boolean
  - Fixed-length strings
  - Variable-length strings
  - Date, time, timestamp
  - Money
  - Other special data types

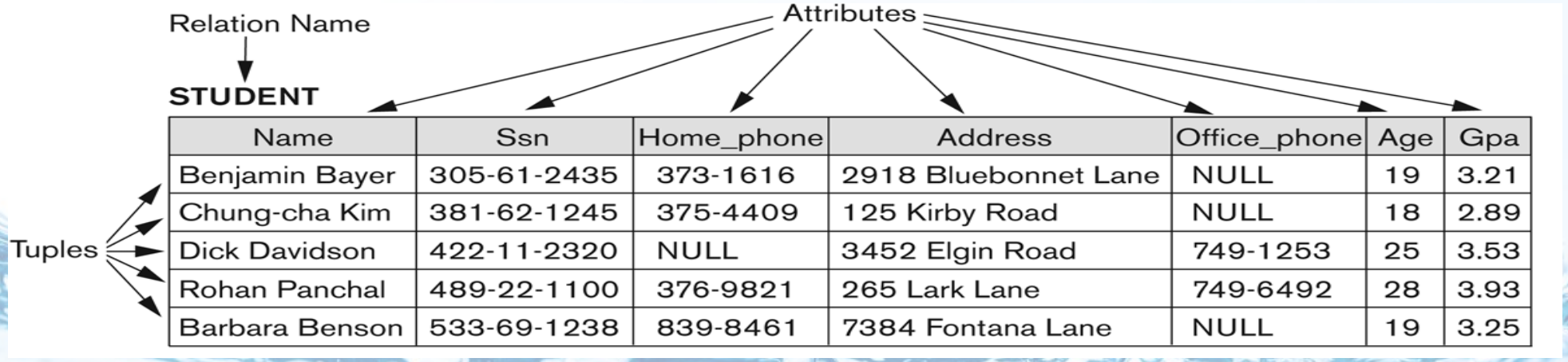
Key Constraints

- No two tuples can have the same combination of values for all their attributes.
- **Superkey**
  - No two distinct tuples in any state *r* of *R* can have same values for a subset of attributes, SK
  - Such a subset SK is the Superkey of R
  - Specifies a ***uniqueness constraint***
  - Every relation has at least one default superkey
  - Superkey can have redundant attributes

Key Constraints

- **Key**
  - Superkey of R
  - Property: Removing any attribute *A* from key, *K* leaves a set of attributes *K'* that is not a superkey of *R* any more
  - No redundancy
  - Determined from the meaning of attributes
  - The property is ***time-invariant***

Key Constraints



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

- SSN is a key of STUDENT
- Any set of attributes including SSN is a Superkey
  - {SSN, Name, Age}
- That superkey is not a key of STUDENT. Why?
- Any superkey formed from a single attribute is also a key.
- A key with multiple attributes must have all its attributes together for uniqueness

COSC 3380 - 19717
Design of Database Systems

Relational Data Model
and
Relational Database Constraints

September 30, 2025

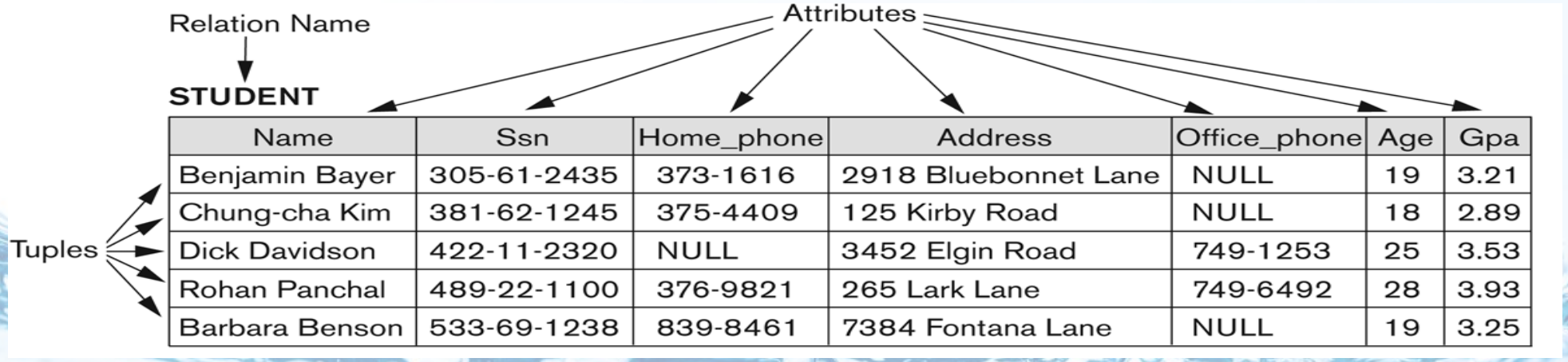
Relational Model Constraints

- **Inherent model-based constraints** or **implicit constraints**
  - Inherent in the data model
- **Schema-based constraints** or **explicit constraints**
  - Can be directly expressed in schemas of the data model
- **Application-based** or **semantic constraints** or **business rules**

Schema-based Constraints

- Domain Constraints
- Key Constraints
  - Superkey
  - Key

Key Constraints



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

- SSN is a key of STUDENT
- Any set of attributes including SSN is a Superkey
  - {SSN, Name, Age}
- That superkey is not a key of STUDENT. Why?
- Any superkey formed from a single attribute is also a key.
- A key with multiple attributes must have all its attributes together for uniqueness

Key Constraints

- Key satisfies two properties:
  - Two distinct tuples in any state of relation cannot have identical values for (all) attributes in a key
  - Minimal superkey
    - Cannot remove any attributes and still have uniqueness constraint hold
    - Minimality property is required for a key, though optional for a superkey
- A key is superkey, but may not be the other way around, depending on minimality!

Key Constraints & Constraints on NULL Values

- **Candidate keys**
  - Relation schema may have more than one key
- **Primary key** of the relation
  - Chosen from the candidate keys
  - A single attribute or a small number of attributes
  - Underline attribute(s)
- Other candidate keys are designated as **unique keys**
- NOT NULL constraint

Relational Databases and Schemas

- **Relational database schema** *S*
  - Set of relation schemas $S = \{R_1, R_2, ..., R_m\}$
  - Set of integrity constraints IC
- **Relational database state**
  - Set of relation states $DB = \{r_1, r_2, ..., r_m\}$
  - Each r_i is a state of R_i
  - The r_i relation states satisfy integrity constraints specified in IC

Relational Databases and Schemas

- **Invalid state**
  - Does not obey all the integrity constraints
- **Valid state**
  - Satisfies all the constraints in the defined set of integrity constraints IC

Relational Databases and Schemas

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Integrity, Referential Integrity, Foreign Keys

- **Entity integrity constraint**
  - No primary key value can be NULL
    - Used for identification of individual tuples in a relation
  - Ensures ability to distinguish tuples when referenced from other relations

Integrity, Referential Integrity, Foreign Keys

- **Referential integrity constraint**
  - Specified between two relations
  - Maintains consistency among tuples in two relations
  - A tuple in one relation that refers to another relation must refer to an existing tuple in that relation

Integrity, Referential Integrity, Foreign Keys

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

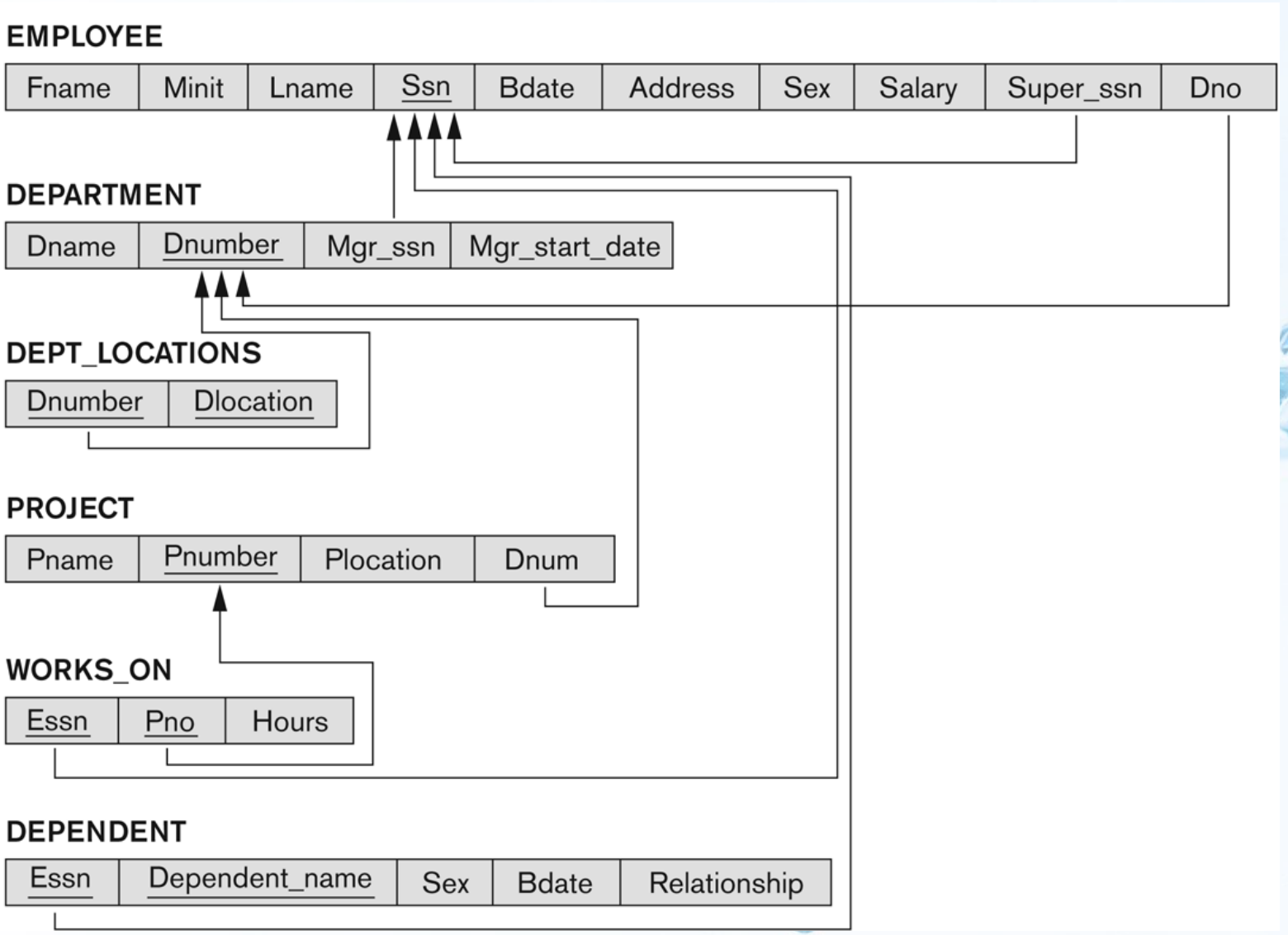
Integrity, Referential Integrity, Foreign Keys

- **Foreign key** rules:
  - A set of attributes, FK in R_1 have the same domain(s) as the primary key attributes PK in R_2
  - Value of FK in a tuple t_1 of the current state $r_1(R_1)$ either occurs as a value of PK for some tuple t_2 in the current state $r_2(R_2)$ or is NULL
    - Tuple t_1 references or refers to tuple t_2
    - R1 – *referencing relation*
    - R2 – *referenced relation*
  - *Referential integrity constraint from R1 to R2 holds*

Integrity, Referential Integrity, Foreign Keys

- Diagrammatically display referential integrity constraints
  - Directed arc from each foreign key to the relation it references
- All integrity constraints should be specified in the relational database schema

Integrity, Referential Integrity, Foreign Keys



Integrity, Referential Integrity, Foreign Keys

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Other Types of Constraints

- **Semantic integrity constraints**
  - May have to be specified and enforced on a relational database
    - 40 - Maximum number of hours an employee can work per week
    - Salary of an employee should not be more than his/her supervisor's salary
  - Use **triggers** and **assertions**
  - More common to check for these types of constraints in the application programs

Other Types of Constraints

- **State constraints**
  - Constraints discussed so far…
  - Define the constraints that a valid state of the database must satisfy

Other Types of Constraints

- **Transition constraints**
  - Defined to deal with state changes in the database
    - Salary of an employee cannot decrease!
  - Enforced by application programs

Other Types of Constraints

- **Functional dependency constraint**
  - Establishes a functional relationship among two sets of attributes *X* and *Y*
  - Value of *X* determines a unique value of *Y*
  - Enforced using ***Validation*** checks

Operations on Relations

- INSERT a tuple
- MODIFY a tuple
- DELETE a tuple
- Integrity constraints should not be violated by these update operations.
- Several update operations may have to be grouped together.
- Updates may ***propagate*** to cause other updates automatically - to maintain integrity constraints.

Possible violations on INSERT

- INSERT may violate any of the constraints:
  - **Domain constraint**
    - if one of the attribute values provided for the new tuple is not of the specified attribute domain
  - **Key constraint**
    - if the value of a key attribute in the new tuple already exists in another tuple in the relation
  - **Referential integrity**
    - if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation
  - **Entity integrity**
    - if the primary key value is null in the new tuple

Possible violations on INSERT

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

- Insert <‘Cecilia’, ‘F’, ‘Kolonsky’, NULL, ‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F, 28000, NULL, 4> into EMPLOYEE
- Insert <‘Alicia’, ‘J’, ‘Zelaya’, ‘999887777’, ‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F, 28000, ‘987654321’, 4> into EMPLOYEE
- Insert <‘Cecilia’, ‘F’, ‘Kolonsky’, ‘677678989’, ‘1960-04-05’, ‘6357 Windswept, Katy, TX’, F, 28000, ‘987654321’, 7> into EMPLOYEE

Possible violations on DELETE

- DELETE may violate referential integrity:
  - If the primary key value of the tuple being deleted is referenced from other tuples in the database
    - Can be remedied by several actions: RESTRICT, CASCADE, SET NULL
      - RESTRICT option: reject the deletion
      - CASCADE option: propagate the new primary key value into the foreign keys of the referencing tuples
      - SET NULL option: set the foreign keys of the referencing tuples to NULL
  - One of the above options must be specified during database design for each foreign key constraint

Possible violations on DELETE

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

- Delete the WORKS_ON tuple with Essn = ‘999887777’ and Pno = 10
- Delete the EMPLOYEE tuple with Ssn = ‘999887777’
- Delete the EMPLOYEE tuple with Ssn = ‘333445555’

COSC 3380 - 19717
Design of Database Systems

Relational Data Model
and
Relational Database Constraints

October 2, 2025

Operations on Relations

- INSERT a tuple
- MODIFY a tuple
- DELETE a tuple
- Integrity constraints should not be violated by these update operations.

Possible violations on INSERT

- INSERT may violate any of the constraints:
  - **Domain constraint**
  - **Key constraint**
  - **Referential integrity**
  - **Entity integrity**

Possible violations on DELETE

- DELETE may violate referential integrity:
  - If the primary key value of the tuple being deleted is referenced from other tuples in the database
    - Can be remedied by several actions: RESTRICT, CASCADE, SET NULL

Possible violations on UPDATE/Modify

- UPDATE may violate domain constraint and NOT NULL constraint on an attribute being modified
- Any of the other constraints may also be violated, depending on the attribute being updated:
  - Updating the primary key (PK):
    - Similar to a DELETE followed by an INSERT
    - Need to specify similar options as DELETE
  - Updating a foreign key (FK):
    - May violate referential integrity
  - Updating an ordinary attribute (neither PK nor FK):
    - Can only violate domain constraints

Update Operations

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation (RESTRICT or REJECT option)
  - Perform the operation but inform the user of the violation
  - Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
  - Execute a user-specified error-correction routine

Possible violations on UPDATE

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

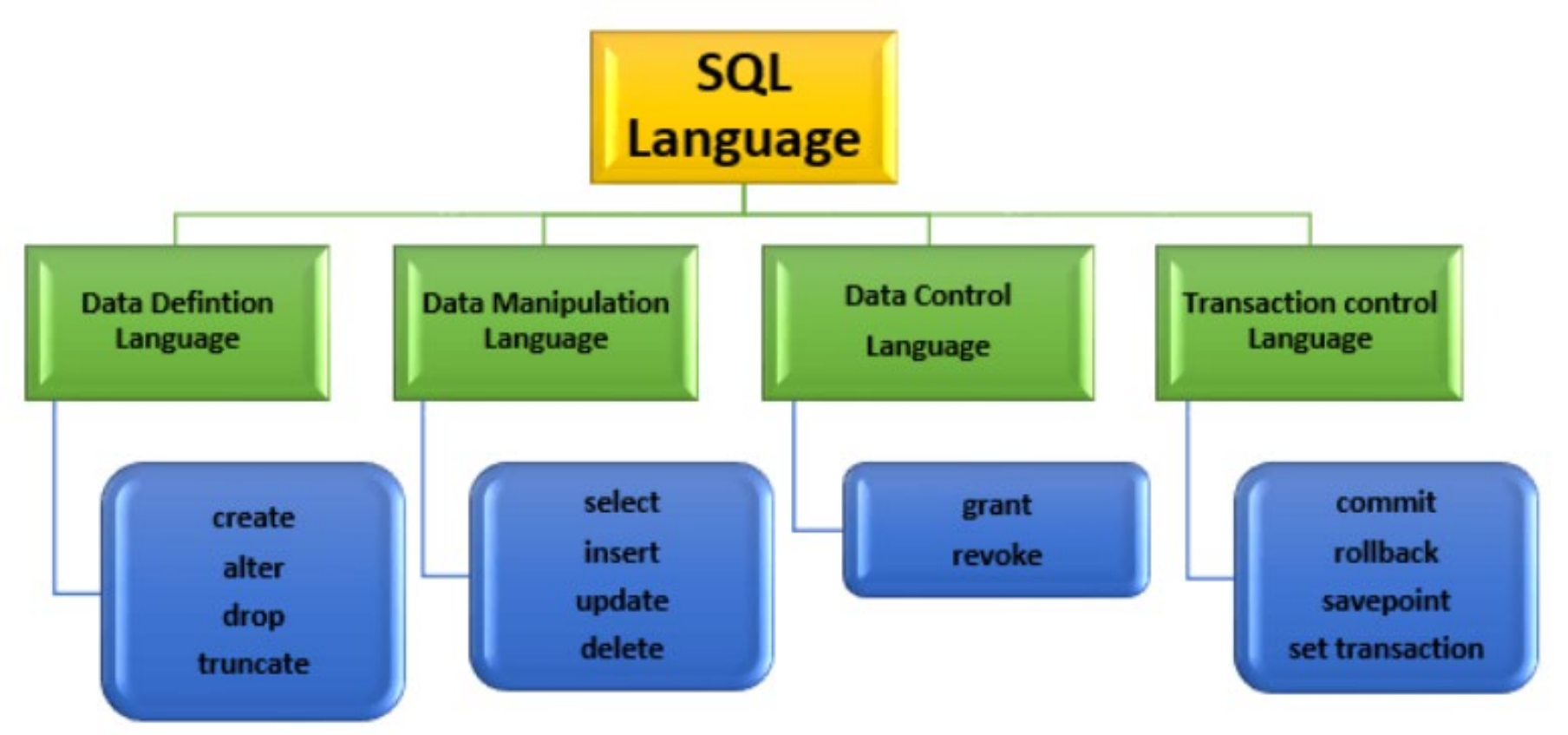
Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

- Update the salary of the EMPLOYEE tuple with Ssn = ‘999887777’ to 28000
- Update the Dno of the EMPLOYEE tuple with Ssn = ‘999887777’ to 7
- Update the Ssn of the EMPLOYEE tuple with Ssn = ‘999887777’ to ‘987654321’
- Update the Dno of the EMPLOYEE tuple with Ssn = ‘999887777’ to 1.

Integrity Constraints and Transactions

- Transaction
  - A program that includes some database operations:
  - reading from the database
  - inserting data
  - deleting data
  - updates to the data in the database
- At the end of a transaction, the database must remain in a valid state
  - Satisfying all constraints specified in the schema

Basic
Structured Query Language (SQL)



Basic SQL

- **SQL language**
  - Considered one of the major reasons for the commercial success of relational databases
- **SQL**
  - **Structured Query Language**
  - Statements for data definitions, queries, and updates (both DDL and DML)
  - **Core specification**
  - Plus specialized **extensions**

Key Characteristics of SQL

- Set-oriented and declarative
- Free form language
- Case insensitive
- Can be used interactively from a command prompt or executed by a program

SQL Data Definition & Data Types

- Terminology
  - **Table**, **row**, and **column** used for relational model terms relation, tuple, and attribute
- CREATE statement
  - Main SQL command for data definition

Schema & Catalog Concepts

- **SQL schema**
  - Identified by a **schema name**
  - Includes an **authorization identifier** and **descriptors** for each element
- Schema **elements** include
  - Tables, constraints, views, domains, and other constructs
- Each statement in SQL ends with a semicolon

Schema and Catalog Concepts

- CREATE SCHEMA statement

  CREATE SCHEMA COMPANY AUTHORIZATION ‘Jsmith’;

  CREATE DATABASE COMPANY AUTHORIZATION ‘Jsmith’;

- **Catalog**
  - Named collection of schemas in an SQL environment
- **SQL environment**
  - Installation of an SQL-compliant RDBMS on a computer system

CREATE TABLE Command

- Specify a new relation
  - Provide name
  - Specify attributes and initial constraints
- Can optionally specify schema:
  - CREATE TABLE COMPANY.EMPLOYEE ...

    or
  - CREATE TABLE EMPLOYEE ...

CREATE TABLE Command

- **Base relations (base tables)**
  - Relation and its tuples are actually created and stored as a file by the DBMS
- **Virtual relations**
  - Created through the CREATE VIEW statement
  - Why do we need views?

Create Table Construct

- A relation is defined using the **create table** command:

  create table *r*

  (A_1 D_1, A_2 D_2, …, A_n D_n,
  (integrity-constraint$_1$), …, (integrity-constraint$_k$))

  - *r* is the name of the relation/table
  - each A_i is an attribute name in the schema of relation *r*
  - D_i is the data type of values in the domain of attribute A_i
- Example:

  create table *instructor* (
  ID **char**(5),
  name **varchar**(20),
  dept_name **varchar**(20),
  salary **numeric**(8,2))

Create Table Construct

- **create table** *student* (
  ID **varchar**(5),
  name **varchar**(20) not null,
  dept_name **varchar**(20),
  tot_cred **numeric**(3,0),
  primary key *(ID),*
  foreign key *(dept_name*) **references** *department*);

- **create table** *takes* (
  ID **varchar**(5),
  course_id **varchar**(8),
  sec_id **varchar**(8),
  semester **varchar**(6),
  year **numeric**(4,0),
  grade **varchar**(2),
  primary key *(ID, course_id, sec_id, semester, year)* ,
  foreign key (*ID*) **references** *student,*
  foreign key (*course_id, sec_id, semester, year*) **references** *section*);

Create Table Construct

- **create table** *course* (
  course_id **varchar**(8),
  title **varchar(**50),
  dept_name **varchar**(20),
  credits **numeric**(2,0),
  primary key *(course_id),*
  foreign key *(dept_name*) **references** *department*);

```
CREATE TABLE EMPLOYEE
        ( Fname          VARCHAR(15)        NOT NULL,
          Minit          CHAR,
          Lname          VARCHAR(15)        NOT NULL,
          Ssn            CHAR(9)            NOT NULL,
          Bdate          DATE,
          Address        VARCHAR(30),
          Sex            CHAR,
          Salary         DECIMAL(10,2),
          Super_ssn      CHAR(9),
          Dno            INT                NOT NULL,
        PRIMARY KEY (Ssn),
        FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE DEPARTMENT
        ( Dname          VARCHAR(15)        NOT NULL,
          Dnumber        INT                NOT NULL,
          Mgr_ssn        CHAR(9)            NOT NULL,
          Mgr_start_date DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
```

```
CREATE TABLE DEPT_LOCATIONS
        ( Dnumber           INT             NOT NULL,
          Dlocation         VARCHAR(15)     NOT NULL,
        PRIMARY KEY (Dnumber, Dlocation),
        FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE PROJECT
        ( Pname             VARCHAR(15)     NOT NULL,
          Pnumber           INT             NOT NULL,
          Plocation         VARCHAR(15),
          Dnum              INT             NOT NULL,
        PRIMARY KEY (Pnumber),
        UNIQUE (Pname),
        FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE WORKS_ON
        ( Essn              CHAR(9)         NOT NULL,
          Pno               INT             NOT NULL,
          Hours             DECIMAL(3,1)    NOT NULL,
        PRIMARY KEY (Essn, Pno),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );
CREATE TABLE DEPENDENT
        ( Essn              CHAR(9)         NOT NULL,
          Dependent_name    VARCHAR(15)     NOT NULL,
          Sex               CHAR,
          Bdate             DATE,
          Relationship      VARCHAR(8),
        PRIMARY KEY (Essn, Dependent_name),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );
```

DDL Concepts using SQL

Data Type	Description
CHAR(n)	Holds a fixed length string with size n
VARCHAR(n)	Holds a variable length string with maximum size n
SMALLINT	Small integer (no decimal) between -32768 to 32767
INT	Integer (no decimal) between -2147483648 to 2147483647
FLOAT(n,d)	Small number with a floating decimal point. The total maximum number of digits is n with a maximum of d digits to the right of the decimal point.
DOUBLE(n,d)	Large number with a floating decimal point. The total maximum number of digits is n with a maximum of d digits to the right of the decimal point.
DATE	Date in format YYYY-MM-DD
DATETIME	Date and time in format YYYY-MM-DD HH:MI:SS
TIME	Time in format HH:MI:SS
BOOLEAN	True or False
BLOB	Binary Large Object (e.g. image, audio, video)

DDL Concepts using SQL

```
CREATE TABLE EMPLOYEE
    ( … ,
      Dno          INT          NOT NULL       DEFAULT 1,
    CONSTRAINT EMPPK
      PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
      FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
                  ON DELETE SET NULL          ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
      FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)
                  ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPARTMENT
    ( … ,
      Mgr_ssn CHAR(9)          NOT NULL       DEFAULT '888665555',
      … ,
    CONSTRAINT DEPTPK
      PRIMARY KEY(Dnumber),
    CONSTRAINT DEPTSK
      UNIQUE (Dname),
    CONSTRAINT DEPTMGRFK
      FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
                  ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPT_LOCATIONS
    ( … ,
    PRIMARY KEY (Dnumber, Dlocation),
    FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)
                  ON DELETE CASCADE           ON UPDATE CASCADE);
```

COSC 3380 - 19717
Design of Database Systems

Basic
Structured Query Language (SQL)

October 7, 2025

DDL Concepts using SQL

```
CREATE TABLE EMPLOYEE
    ( … ,
      Dno          INT          NOT NULL       DEFAULT 1,
    CONSTRAINT EMPPK
      PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
      FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
                  ON DELETE SET NULL          ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
      FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)
                  ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPARTMENT
    ( … ,
      Mgr_ssn CHAR(9)          NOT NULL       DEFAULT '888665555',
      … ,
    CONSTRAINT DEPTPK
      PRIMARY KEY(Dnumber),
    CONSTRAINT DEPTSK
      UNIQUE (Dname),
    CONSTRAINT DEPTMGRFK
      FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
                  ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPT_LOCATIONS
    ( … ,
    PRIMARY KEY (Dnumber, Dlocation),
    FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)
                  ON DELETE CASCADE           ON UPDATE CASCADE);
```

DDL Concepts using SQL

```
CREATE DOMAIN PRODTYPE_DOMAIN AS VARCHAR(10);
CREATE DOMAIN SSN_TYPE AS CHAR(9);
```

Additional constraints on individual tuples within a relation are also possible using **CHECK**

```
CHECK (VALUE IN ('white', 'red', 'rose', 'sparkling'))

Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21);
```

CHECK clauses at the end of a **CREATE TABLE** statement

- **CHECK** (Dept_create_date <= Mgr_start_date);

Apply to each tuple individually

DDL Concepts using SQL

- Column constraints
  - **PRIMARY KEY** constraint defines the primary key of the table
  - **FOREIGN KEY** constraint defines a foreign key of a table
  - **UNIQUE** constraint defines an alternative key of a table
  - **NOT NULL** constraint prohibits NULL values for a column
  - **DEFAULT** constraint sets a default value for a column
  - **CHECK** constraint defines a constraint on the column values

DROP and ALTER Command

- **DROP** command can be used to drop or remove database objects
  - can be combined with CASCADE and RESTRICT
- Examples:

```
DROP SCHEMA PURCHASE CASCADE
DROP SCHEMA PURCHASE RESTRICT
DROP TABLE PRODUCT CASCADE
DROP TABLE PRODUCT RESTRICT
```

DROP and ALTER Command

- **ALTER** statement can be used to modify table column definitions
- Examples:

```
ALTER TABLE PRODUCT ADD PRODIMAGE BLOB
```

```
ALTER TABLE SUPPLIER ALTER SUPSTATUS SET DEFAULT '10'
```

SQL Data Manipulation Language (DML)

- SQL SELECT Statement
- SQL INSERT Statement
- SQL DELETE Statement
- SQL UPDATE Statement

SQL Select Statement

- Basic statement for retrieving information from a database
- Result of SQL SELECT statement is a multiset, and not a set!
- Multiset (bag behavior)
  - Elements are not ordered AND there can be duplicates
- Examples:
  - Set {10, 5, 20} Multiset {10, 5, 10, 20, 5, 10}
- SQL does not eliminate duplicates
  - duplicate elimination is expensive
  - user may want to see duplicate tuples
  - duplicates may be considered by aggregate functions
- Tuple-id may be used as a key

Basic SQL Queries: SELECT-FROM-WHERE Structure

```
SELECT    <attribute list>
FROM      <table list>
WHERE     <condition>;
```

where

- <attribute list> is a list of attribute names whose values are to be retrieved by the query.
- <table list> is a list of the relation names required to process the query.
- <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query.

Basic SQL Queries: SELECT-FROM-WHERE Structure

- Logical comparison operators
  =, <, <=, >, >=, and <>
- **Projection attributes**
  - Attributes whose values are to be retrieved
- **Selection condition**
  - Boolean condition that must be true for any retrieved tuple
  - Selection conditions include join conditions when multiple relations are involved

Product Supplier Database

SUPPLIER(SUPNR, SUPNAME, SUPADDRESS, SUPCITY, SUPSTATUS)

PRODUCT(PRODNR, PRODNAME, PRODTYPE, AVAILABLE_QUANTITY)

SUPPLIES(*SUPNR*, *PRODNR*, PURCHASE_PRICE, DELIV_PERIOD)

PURCHASE_ORDER(PONR, PODATE, *SUPNR*)

PO_LINE(*PONR*, *PRODNR*, QUANTITY)

Product Supplier Database

```
CREATE TABLE SUPPLIER
   (SUPNR CHAR(4) NOT NULL PRIMARY KEY,
    SUPNAME VARCHAR(40) NOT NULL,
    SUPADDRESS VARCHAR(50),
    SUPCITY VARCHAR(20),
    SUPSTATUS SMALLINT);


CREATE TABLE PRODUCT
   (PRODNR CHAR(6) NOT NULL PRIMARY KEY,
    PRODNAME VARCHAR(60) NOT NULL,
    CONSTRAINT UC1 UNIQUE(PRODNAME),
    PRODTYPE VARCHAR(10),
    CONSTRAINT CC1 CHECK(PRODTYPE IN ('white', 'red',
   'rose','sparkling')),
    AVAILABLE_QUANTITY INTEGER);
```

Product Supplier Database

```
CREATE TABLE SUPPLIES
   (SUPNR CHAR(4) NOT NULL,
    PRODNR CHAR(6) NOT NULL,
    PURCHASE_PRICE DOUBLE(8,2)
    COMMENT 'PURCHASE_PRICE IN EUR',
    DELIV_PERIOD TIME
    COMMENT 'DELIV_PERIOD IN DAYS',
    PRIMARY KEY (SUPNR, PRODNR),
    FOREIGN KEY (SUPNR) REFERENCES SUPPLIER (SUPNR)
    ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (PRODNR) REFERENCES PRODUCT (PRODNR)
    ON DELETE CASCADE ON UPDATE CASCADE);
```

Product Supplier Database

```
CREATE TABLE PURCHASE_ORDER
   (PONR CHAR(7) NOT NULL PRIMARY KEY,
    PODATE DATE,
    SUPNR CHAR(4) NOT NULL,
    FOREIGN KEY (SUPNR) REFERENCES SUPPLIER (SUPNR)
    ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE PO_LINE
    (PONR CHAR(7) NOT NULL,
     PRODNR CHAR(6) NOT NULL,
     QUANTITY INTEGER,
     PRIMARY KEY (PONR, PRODNR),
     FOREIGN KEY (PONR) REFERENCES PURCHASE_ORDER (PONR)
     ON DELETE CASCADE ON UPDATE CASCADE,
     FOREIGN KEY (PRODNR) REFERENCES PRODUCT (PRODNR)
     ON DELETE CASCADE ON UPDATE CASCADE);
```

Simple Queries

- SQL statements that retrieve data from only one table

```
Q1: SELECT SUPNR, SUPNAME,SUPADDRESS,
SUPCITY, SUPSTATUS FROM SUPPLIER;
```

```
Q1: SELECT * FROM SUPPLIER;
```

Simple Queries

```
Q1: SELECT SUPNR, SUPNAME,SUPADDRESS, SUPCITY, SUPSTATUS
    FROM SUPPLIER;
```

SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
21	Deliwines	240, Avenue of the Americas	New York	20
32	Best Wines	660, Market Street	San Francisco	90
37	Ad Fundum	82, Wacker Drive	Chicago	95
52	Spirits & co.	928, Strip	Las Vegas	NULL
68	The Wine Depot	132, Montgomery Street	San Francisco	10
69	Vinos del Mundo	4, Collins Avenue	Miami	92

Simple Queries

```
Q2: SELECT SUPNR, SUPNAME FROM SUPPLIER;
```

SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
21	Deliwines	240, Avenue of the Americas	New York	20
32	Best Wines	660, Market Street	San Francisco	90
37	Ad Fundum	82, Wacker Drive	Chicago	95
52	Spirits & co.	928, Strip	Las Vegas	NULL
68	The Wine Depot	132, Montgomery Street	San Francisco	10
69	Vinos del Mundo	4, Collins Avenue	Miami	92

SUPNR	SUPNAME
21	Deliwines
32	Best Wines
37	Ad Fundum
52	Spirits & co.
68	The Wine Depot
69	Vinos del Mundo

Simple Queries

```
Q3: SELECT SUPNR
    FROM PURCHASE_ORDER;
```

SUPNR
32
32
37
37
37
37
37
68
69
94

Simple Queries – using DISTINCT keyword

```
Q4: SELECT DISTINCT SUPNR
    FROM PURCHASE_ORDER;
```

SUPNR
32
37
68
69
94

SUPNR
32
32
37
37
37
37
37
68
69
94

Simple Queries – create a new data item

```
Q5: SELECT SUPNR, PRODNR, DELIV_PERIOD/30 AS
    MONTH_DELIV_PERIOD FROM SUPPLIES;
```

SUPNR	PRODNR	MONTH_DELIV_PERIOD
21	0119	0.0333
21	0178	NULL
21	0289	0.0333
21	0327	0.2000
21	0347	0.0667
21	0384	0.0667
…	…	…

Simple Queries – WHERE clause

```
Q6: SELECT SUPNR, SUPNAME, SUPSTATUS FROM SUPPLIER
    WHERE SUPCITY = 'San Francisco';
```

SUPNR	SUPNAME	SUPSTATUS
32	Best Wines	90
68	The Wine Depot	10

Simple Queries - WHERE clause

```
Q7: SELECT SUPNR, SUPNAME, SUPSTATUS
    FROM SUPPLIER
    WHERE SUPCITY = 'San Francisco' AND
    SUPSTATUS > 80;
```

SUPNR	SUPNAME	SUPSTATUS
32	Best Wines	90

Simple Queries – BETWEEN keyword

```
Q8: SELECT SUPNR, SUPNAME, SUPSTATUS
     FROM SUPPLIER
     WHERE SUPSTATUS BETWEEN 70 AND 80;
```

SUPNR	SUPNAME	SUPSTATUS
94	The Wine Crate	75

Simple Queries – IN keyword

```
Q9:SELECT PRODNR, PRODNAME
    FROM PRODUCT
    WHERE PRODTYPE IN ('WHITE', 'SPARKLING');
```

PRODNR	PRODNAME
0178	Meerdael, Methode Traditionnelle Chardonnay, 2014
0199	Jacques Selosse, Brut Initial, 2012
0212	Billecart-Salmon, Brut Réserve, 2014
0300	Chateau des Rontets, Chardonnay, Birbettes
0494	Veuve-Cliquot, Brut, 2012
0632	Meneghetti, Chardonnay, 2010
…	….

Simple Queries – LIKE keyword

```
Q10: SELECT PRODNR, PRODNAME
     FROM PRODUCT
     WHERE PRODNAME LIKE '%CHARD%';
```

PRODNR	PRODNAME
0300	Chateau des Rontets, Chardonnay, Birbettes
0783	Clos D'Opleeuw, Chardonnay, 2012
0178	Meerdael, Methode Traditionnelle Chardonnay, 2014
0632	Meneghetti, Chardonnay, 2010

Simple Queries – IS NULL check

```
Q11: SELECT SUPNR, SUPNAME, SUPSTATUS
     FROM SUPPLIER
     WHERE SUPSTATUS IS NULL;
```

SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
21	Deliwines	240, Avenue of the Americas	New York	20
32	Best Wines	660, Market Street	San Francisco	90
37	Ad Fundum	82, Wacker Drive	Chicago	95
52	Spirits & co.	928, Strip	Las Vegas	NULL
68	The Wine Depot	132, Montgomery Street	San Francisco	10
69	Vinos del Mundo	4, Collins Avenue	Miami	92

SUPNR	SUPNAME	SUPSTATUS
52	Spirits & Co.	NULL

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Simple Queries

Query 0. Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.

```
Q0:   SELECT   Bdate, Address
      FROM     EMPLOYEE
      WHERE    Fname='John' AND Minit='B' AND Lname='Smith';
```

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

Bdate	Address
1965-01-09	731Fondren, Houston, TX

Simple Queries

Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

```
Q1:   SELECT   Fname, Lname, Address
      FROM     EMPLOYEE, DEPARTMENT
      WHERE    Dname='Research' AND Dnumber=Dno;
```

Fname	Lname	Address
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX