# CS4.301 Data & Applications

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

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

# Discussion on Relationship Types

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

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

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

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

# Constraints on Relationships

Constraints on Relationship Types

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

| Term | Meaning | Example |
|---|---|---|
| **Existence-dependent** | One entity *cannot exist* without another | *Course Section → Course* |
| **Not existence-dependent** | Entity *can exist* independently | *Professor → Department* |

# Many-to-one (N:1) Relationship

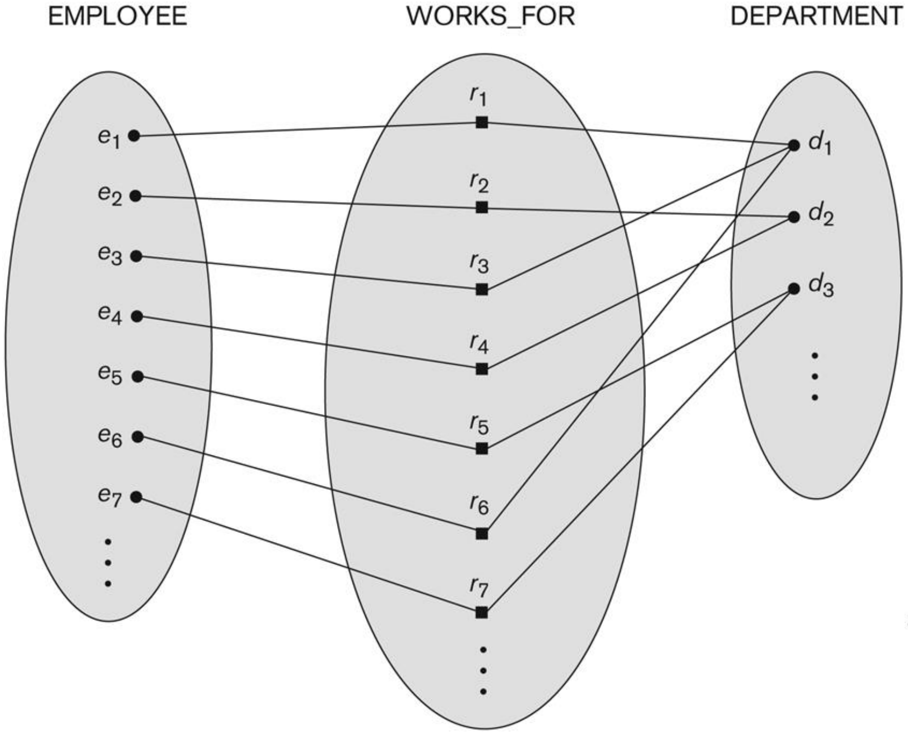



**Figure 3.9**
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

# Many-to-many (M:N) Relationship

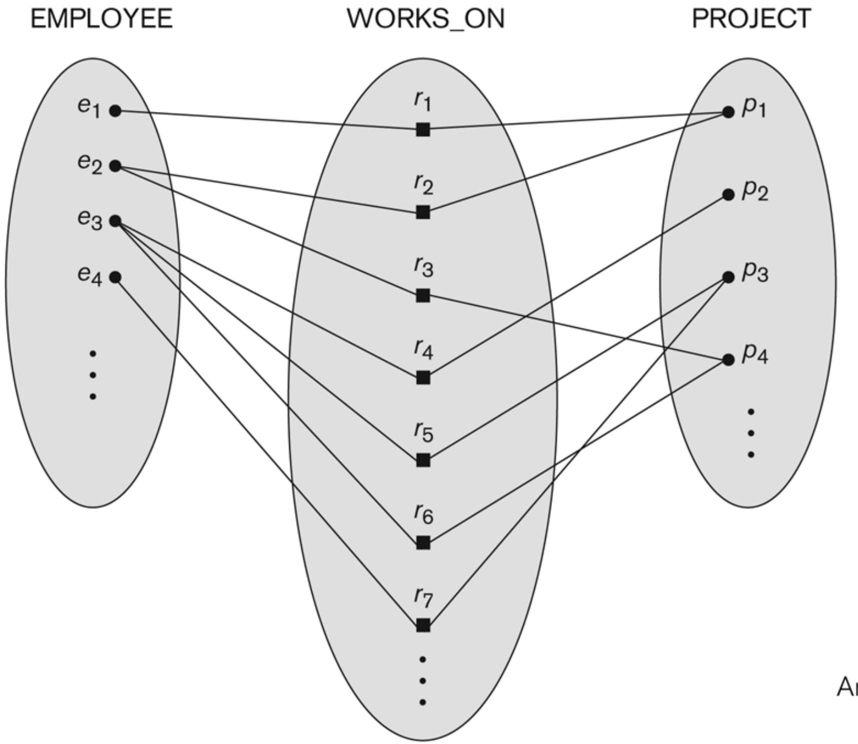



**Figure 3.13**
An M:N relationship, WORKS_ON.

# Recursive Relationship Type

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

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

Example: the SUPERVISION relationship

EMPLOYEE participates twice in two distinct roles:

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

Each relationship instance relates two distinct EMPLOYEE entities:

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

# Displaying a recursive relationship

In a recursive relationship type.

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

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

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

# A Recursive Relationship Supervision`

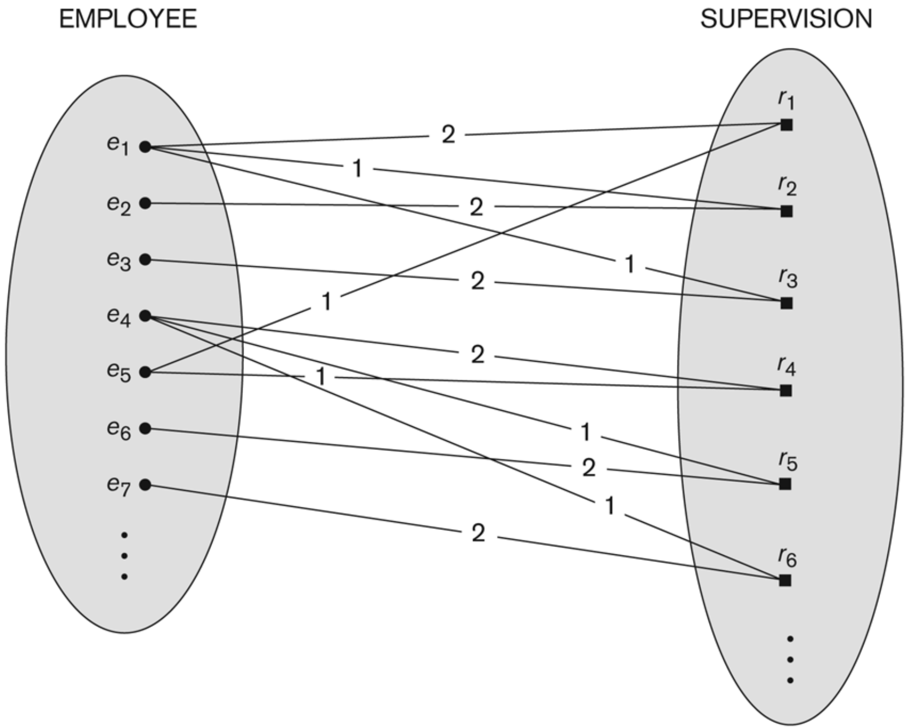



**Figure 3.11**
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

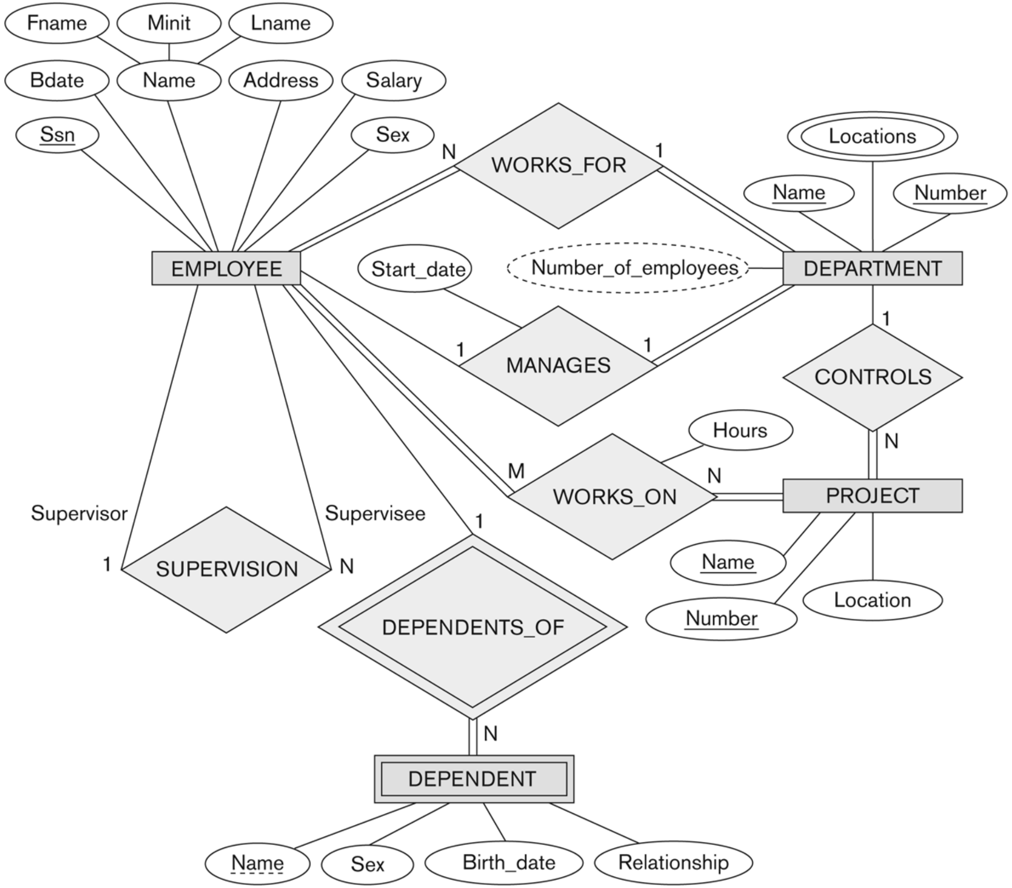



**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

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

# This Lecture

# Weak Entity Types

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

A weak entity must participate in an identifying relationship type with an owner or identifying entity type

Entities are identified by the combination of:

- A partial key of the weak entity type
- The particular entity they are related to in the identifying relationship type

**Example:**

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

# Weak Entity Types

| Concept | Symbol | Meaning |
| --- | --- | --- |
| Strong Entity | Single rectangle | Independent existence |
| Weak Entity | Double rectangle | Depends on strong entity |
| Identifying Relationship | Double diamond | Defines weak entity’s identity |

# Attributes of Relationship types

A relationship type can have attributes:

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

A **relationship type** can have an **attribute** when that attribute **belongs to the association itself** — not to any of the entities individually.

Think of it as information that **only makes sense because of the relationship** between the entities.

# Attributes of Relationship types

Add attributes to a relationship type **only when**:

1. The information applies **to the link itself**, not to either entity.
2. It's **not derivable** from the entities' existing attributes.
3. It helps capture **real-world meaning** of that association.

# Attributes of Relationship types

When an ER relationship has attributes, those attributes are stored as columns in the relationship table (also called a junction or association table) that represents the relationship in the relational database.

Employee(Emp_ID, Name)

Project(Proj_ID, Title)

Works_On(Emp_ID, Proj_ID, hours_per_week)

Relationship attributes become **columns in the relationship (junction) table** when the ER model is converted to the relational model.

# Example Attribute of a Relationship Type: Hours of WORKS_ON

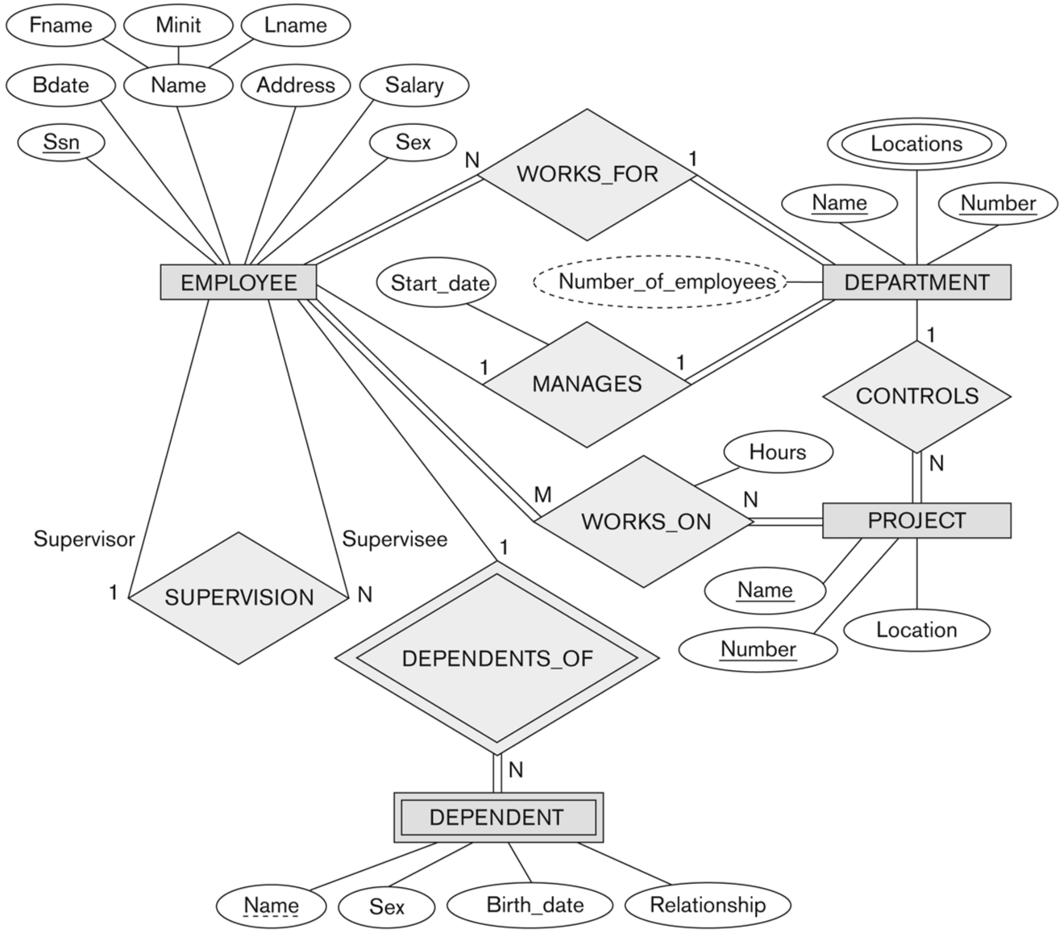



**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# Notation for Constraints on Relationships

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

- Shown by placing appropriate numbers on the relationship edges.

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

- Total shown by double line, partial by single line.

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

Specified on each participation of an entity type E in a relationship type R

Specifies that each entity e in E participates in at least *min* and at most *max* relationship instances in R

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

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

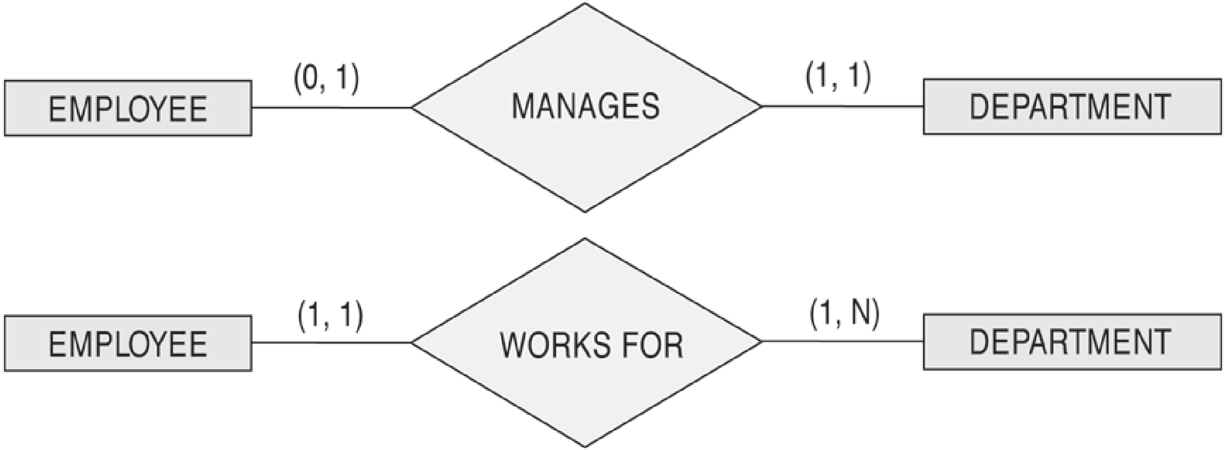
Derived from the knowledge of mini-world constraints

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

Examples:

- A department has exactly one manager and an employee can manage at most one department.
  - Specify (0,1) for participation of EMPLOYEE in MANAGES
  - Specify (1,1) for participation of DEPARTMENT in MANAGES
- An employee can work for exactly one department but a department can have any number of employees.
  - Specify (1,1) for participation of EMPLOYEE in WORKS_FOR
  - Specify (0,n) for participation of DEPARTMENT in WORKS_FOR

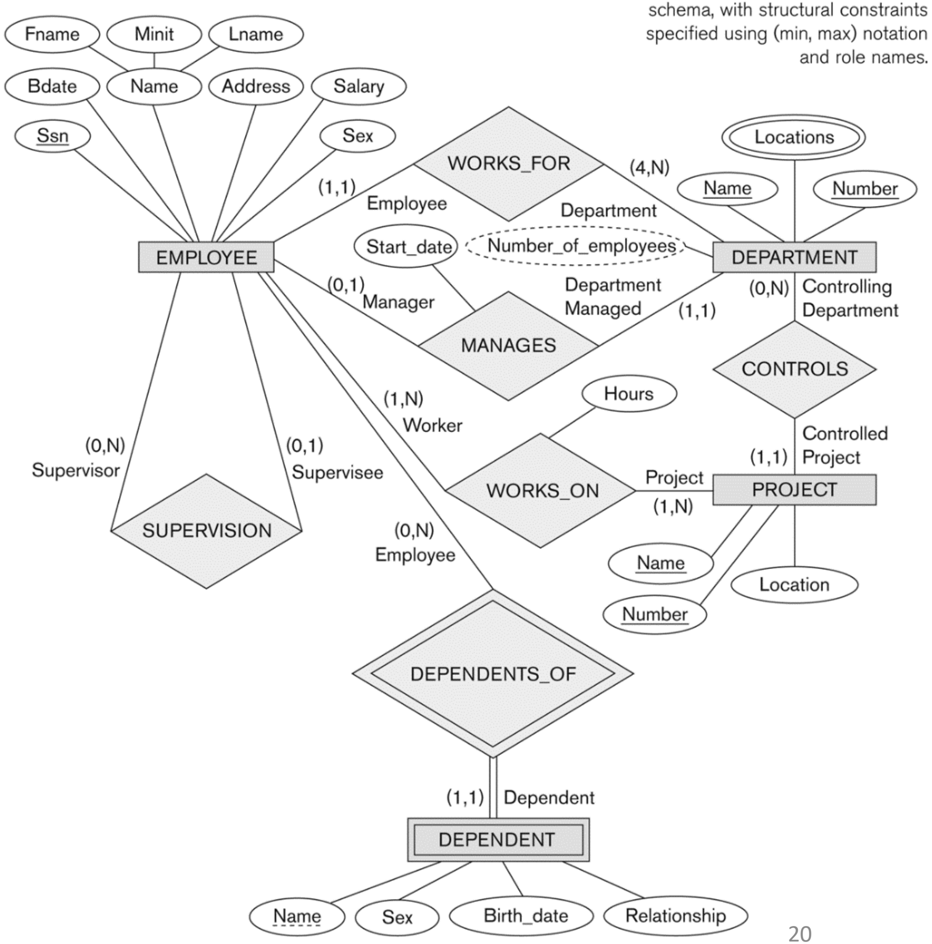
# The (min,max) notation for relationship constraints




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

COMPANY ER Schema Diagram using (min, max) notation

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



# Alternative diagrammatic notation

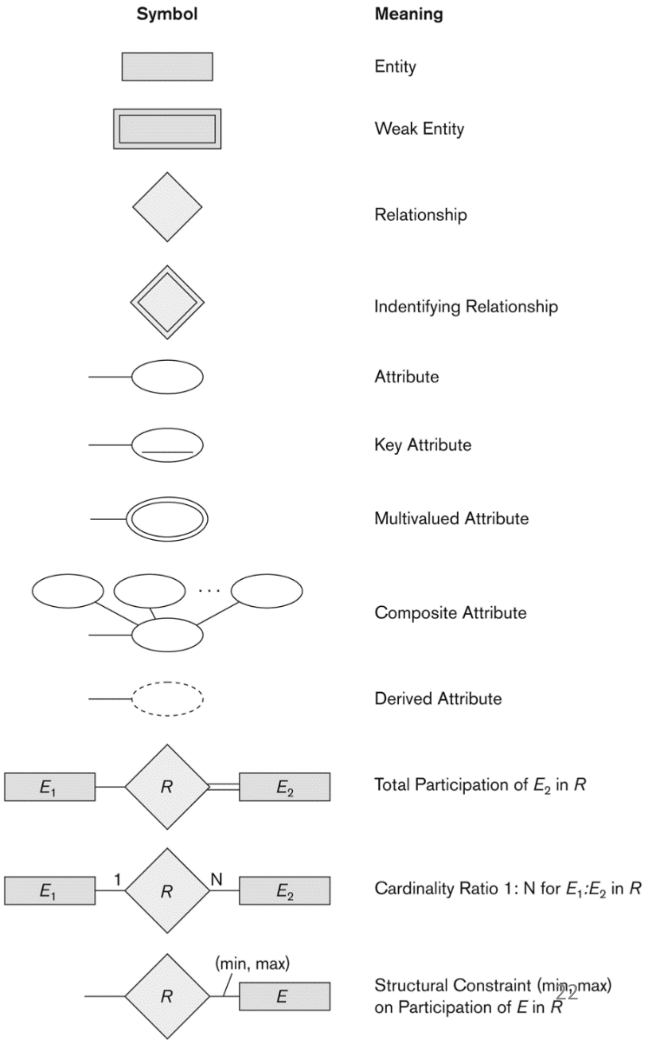
ER diagrams is one popular example for displaying database schemas

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

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

# Summary of notation for ER diagrams

**Figure 3.14**
Summary of the notation for ER diagrams.



# UML class diagrams

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

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

Relationships (called associations) represented as lines connecting the classes

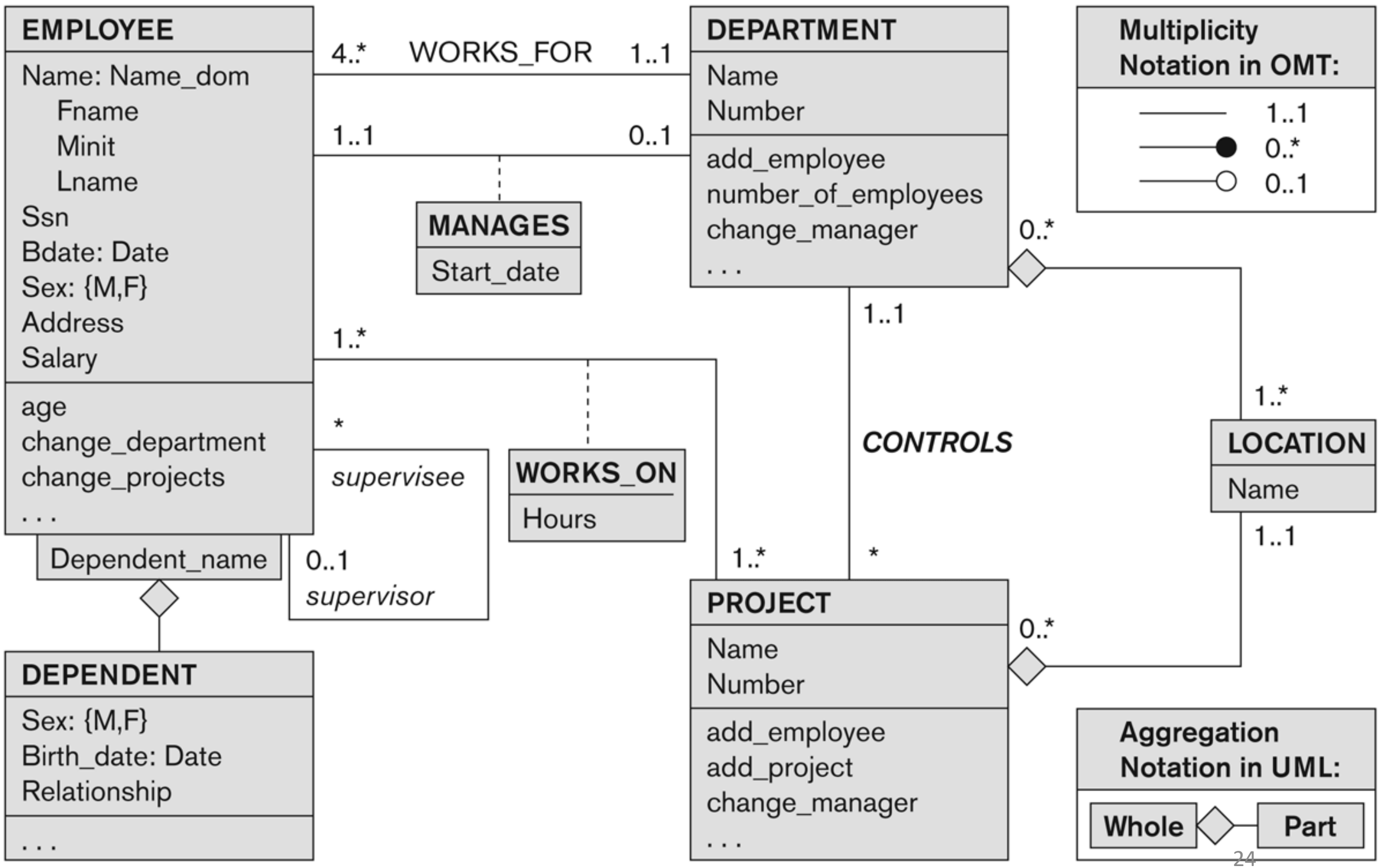
- Other UML terminology also differs from ER terminology

Used in database design and object-oriented software design

UML has many other types of diagrams for software design

UML class diagram for COMPANY database schema

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



# Other alternative diagrammatic notations

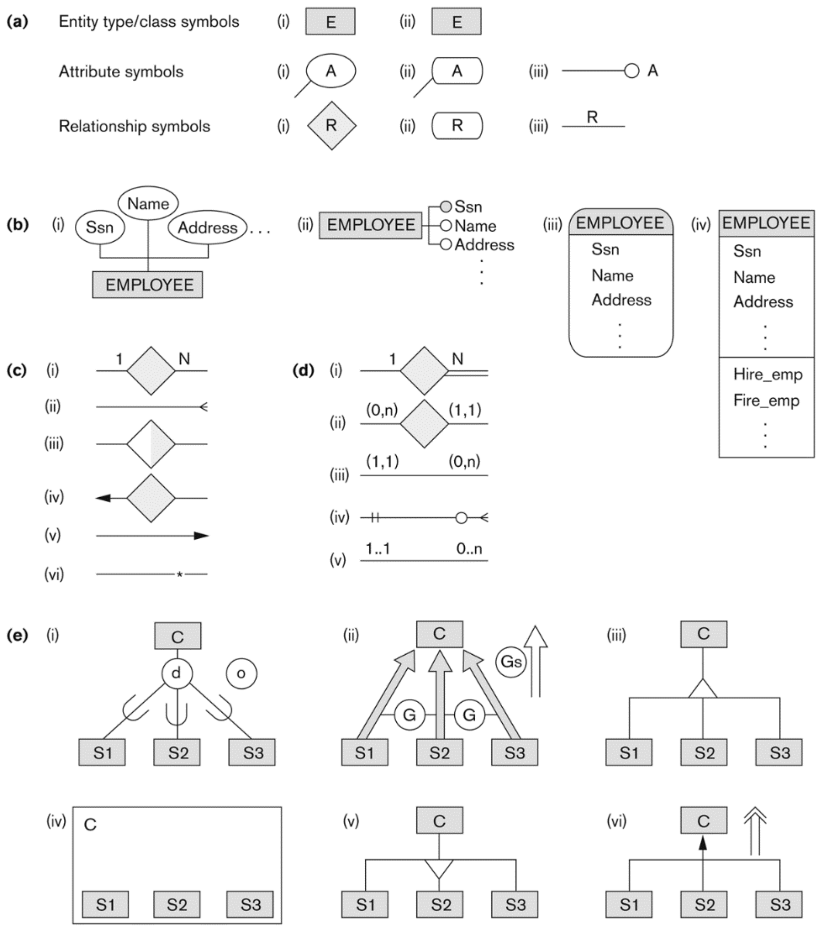



**Figure A.1**
Alternative notations. (a) Symbols for entity type/class, attribute, and relationship. (b) Displaying attributes. (c) Displaying cardinality ratios. (d) Various (min, max) notations. (e) Notations for displaying specialization/generalization.

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

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

# DBMS Interfaces

Stand-alone query language interfaces

- Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. SQL*Plus in ORACLE)

Programmer interfaces for embedding DML in programming languages

User-friendly interfaces

- Menu-based, forms-based, graphics-based, etc.

Mobile Interfaces: interfaces allowing users to perform transactions using mobile apps

# DBMS Programming Language Interfaces

Programmer interfaces for embedding DML in a programming languages:

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

# User-Friendly DBMS Interfaces

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

# Other DBMS Interfaces

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

# Database System Utilities

To perform certain functions such as:

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

# Typical DBMS Component Modules

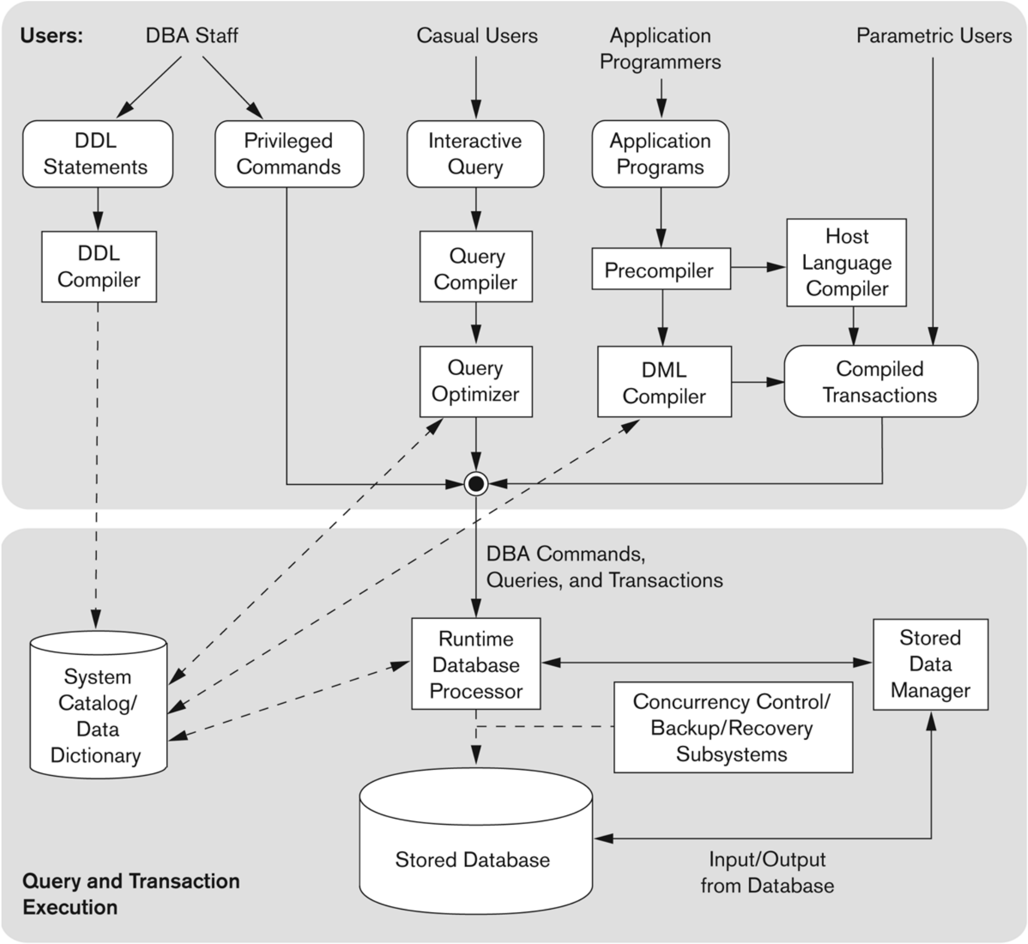



**Figure 2.3**
Component modules of a DBMS and their interactions.

# Centralized and Client-Server DBMS Architectures

Centralized DBMS:

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

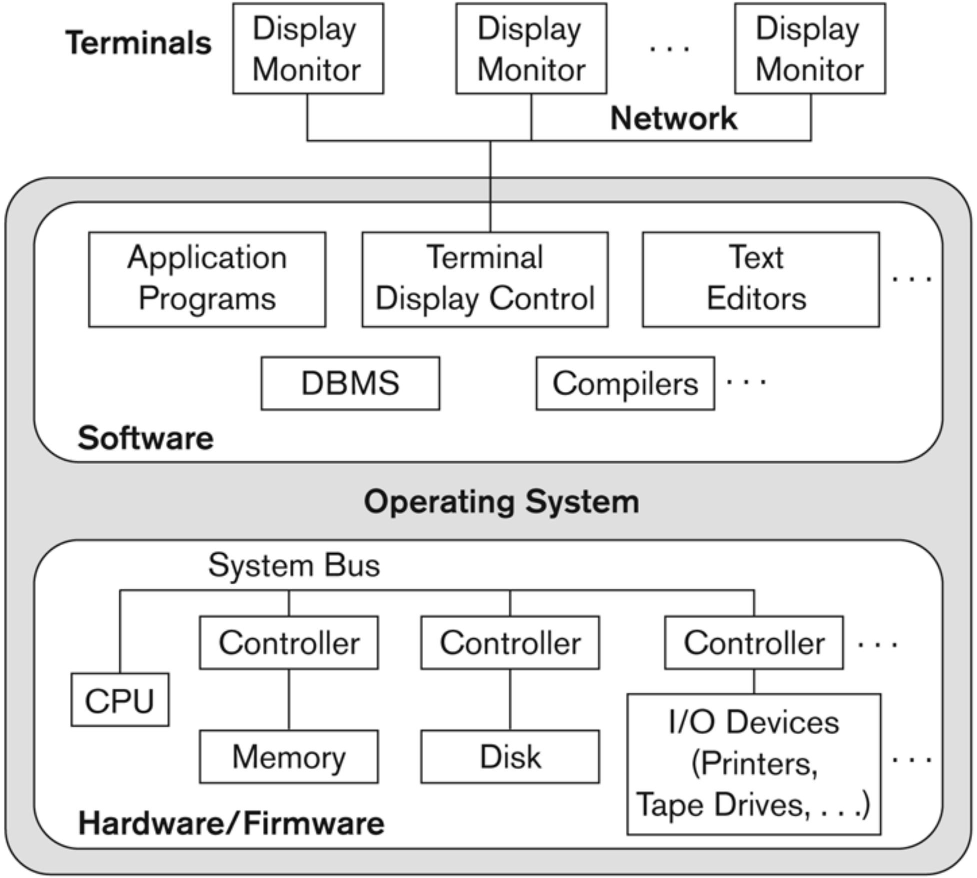
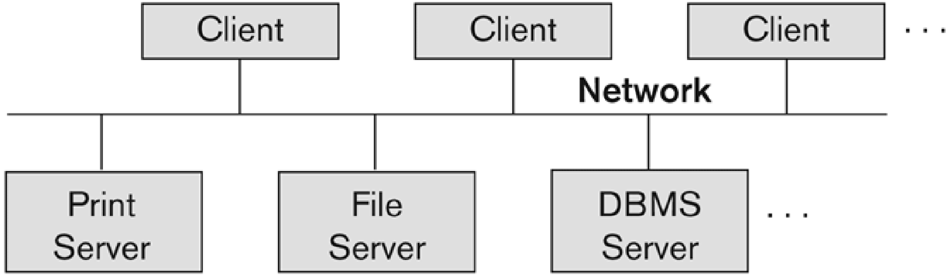



**Figure 2.4**
A physical centralized architecture.

# A Physical Centralized Architecture

# Logical two-tier client server architecture

**Figure 2.5**
Logical two-tier client/server architecture.



# Clients

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

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

Connected to the servers via some form of a network.

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

# DBMS Server

Provides database query and transaction services to the clients

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

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

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

# Two Tier Client-Server Architecture

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

A client program may connect to several DBMSs, sometimes called the data sources.

In general, data sources can be files or other non-DBMS software that manages data.

# Three Tier Client-Server Architecture

Common for Web applications

Intermediate Layer called Application Server or Web Server:

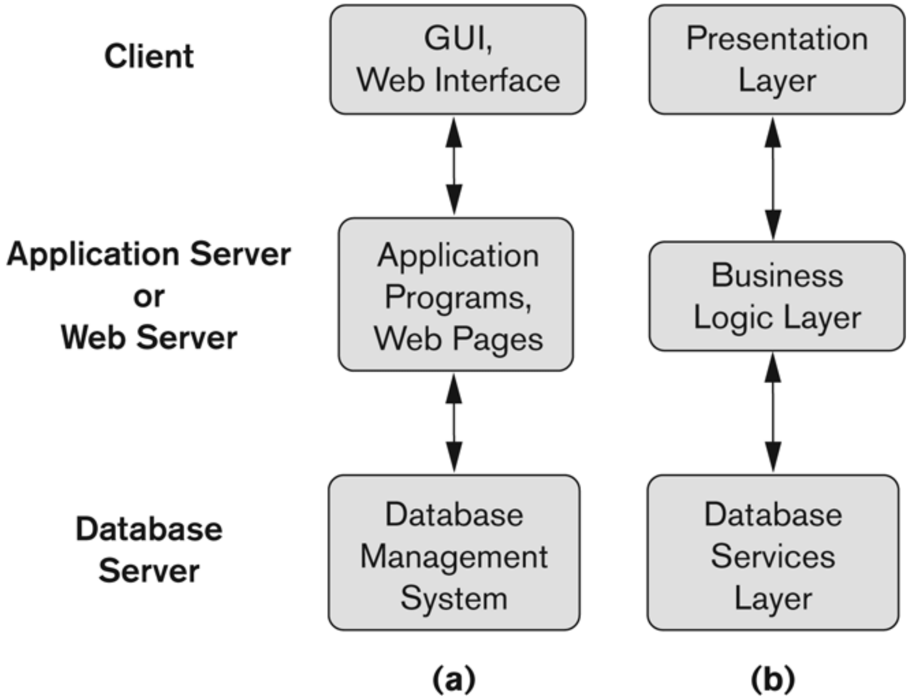
- Stores the web connectivity software and the business logic part of the application used to access the corresponding data from the database server
- Acts like a conduit for sending partially processed data between the database server and the client.

Three-tier Architecture can enhance security:

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

# Three-tier client-server architecture

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



# The Relational Data Model and Relational Database Constraints

# Relational Model Concepts

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

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

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

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

# Relational Model Concepts

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

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

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

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

# Informal Definitions

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

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

The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship** In the formal model, rows are called **tuples**

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

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

# Example of a Relation

Attributes

Relation Name

**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 |

Tuples

**Figure 5.1**
The attributes and tuples of a relation STUDENT.

# Informal Definitions

Key of a Relation:

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

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

# Activity

Create 3 recursive relationship and show them in ER diagram

Create 3 different weak entities in 3 different mini-world and show them in ER diagram

Create 3 different sets of relationships in Infinium showing cardinality / min-max

# Formal Definitions - Schema

The **Schema** (or description) of a Relation:

- Denoted by R(A1, A2, .....An)
- R is the **name** of the relation
- The **attributes** of the relation are A1, A2, ..., An

Example:

CUSTOMER (Cust-id, Cust-name, Address, Phone#)

- CUSTOMER is the relation name
- 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.

# Formal Definitions - Tuple

A **tuple** is an ordered set of values (enclosed in angled brackets ‘< … >’)

Each value is derived from an appropriate *domain*.

A row in the CUSTOMER relation is a 4-tuple and would consist of four values, for example:

- <632895, "John Smith", "101 Main St. Atlanta, GA  30332", "(404) 894-2000">
- This is called a 4-tuple as it has 4 values
- A tuple (row) in the CUSTOMER relation.

A relation is a **set** of such tuples (rows)

# Formal Definitions - Domain

A **domain** has a logical definition:

- Example: “USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.

A domain also has a data-type or a format defined for it.

- The USA_phone_numbers may have a format: (ddd)ddd-dddd where each d is a decimal digit.
- Dates have various formats such as year, month, date formatted as yyyy-mm-dd, or as dd mm,yyyy etc.

The attribute name designates the role played by a domain in a relation:

- Used to interpret the meaning of the data elements corresponding to that attribute
- Example: The domain Date may be used to define two attributes named “Invoice-date” and “Payment-date” with different meanings

# Formal Definitions - State

The **relation state** is a subset of the Cartesian product of the domains of its attributes

each domain contains the set of all possible values the attribute can take.

Example: attribute Cust-name is defined over the domain of character strings of maximum length 25

dom(Cust-name) is varchar(25)

The role these strings play in the CUSTOMER relation is that of the *name of a customer*.

# Formal Definitions - Summary

Formally,

Given R(A1, A2, ........., An)

r(R) $\subset$ dom (A1) X dom (A2) X ....X dom(An)

R(A1, A2, …, An) is the **schema** of the relation

R is the **name** of the relation

A1, A2, …, An are the **attributes** of the relation

r(R): a specific **state** (or "value" or “population”) of relation R – this is a *set of tuples* (rows)

r(R) = {t1, t2, …, tn} where each ti is an n-tuple [All Rows in a table]

ti = <v1, v2, …, vn> where each vj *element-of* dom(Aj) [Single Row in the table]

# Formal Definitions - Example

Let R(A1, A2) be a relation schema:

- Let dom(A1) = {0,1}
- Let dom(A2) = {a,b,c}

Then: dom(A1) X dom(A2) is all possible combinations:

- {<0,a> , <0,b> , <0,c>, <1,a>, <1,b>, <1,c> }

The relation state r(R) $\subset$ dom(A1) X dom(A2)

For example: r(R) could be {<0,a> , <0,b> , <1,c> }

- this is one possible state (or “population” or “extension”) r of the relation R, defined over A1 and A2.
- It has three 2-tuples: <0,a> , <0,b> , <1,c>

# 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 |

# Example – A relation STUDENT

Relation Name

Attributes

**STUDENT**

Tuples

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

**Figure 5.1**
The attributes and tuples of a relation STUDENT.

# Characteristics Of Relations

Ordering of tuples in a relation r(R):

The tuples are *not considered to be ordered*, even though they appear to be in the tabular form.

Ordering of attributes in a relation schema R (and of values within each tuple):

We will consider the attributes in R(A1, A2, ..., An) and the values in t=<v1, v2, ..., vn> to be ordered .

(However, a more general alternative definition of relation does not require this ordering. It includes both the name and the value for each of the attributes ).

Example: t= { <name, “John” >, <SSN, 123456789> }

This representation may be called as “self-describing”.

# Same state as previous Figure (but with different order of tuples)

**Figure 5.2**
The relation STUDENT from Figure 5.1 with a different order of tuples.

**STUDENT**

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

# Bibliography / Acknowledgements

Instructor materials from Elmasri & Navathe 7e

pk.profgiri

Ponnurangam.kumaraguru

/in/ponguru

ponguru

pk.guru@iiit.ac.in

Thank you for attending the class!!!