CS4.301 Data & Applications

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

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

Refining the initial design by introducing **relationships**

The initial design is typically not complete

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

ER model has three main concepts:

- Entities (and their entity types and entity sets)
- Attributes (simple, composite, multivalued)
- Relationships (and their relationship types and relationship sets)

We introduce relationship concepts next

Relationships and Relationship Types (1)

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

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

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

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

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

Both MANAGES and WORKS_ON are *binary* relationships.

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

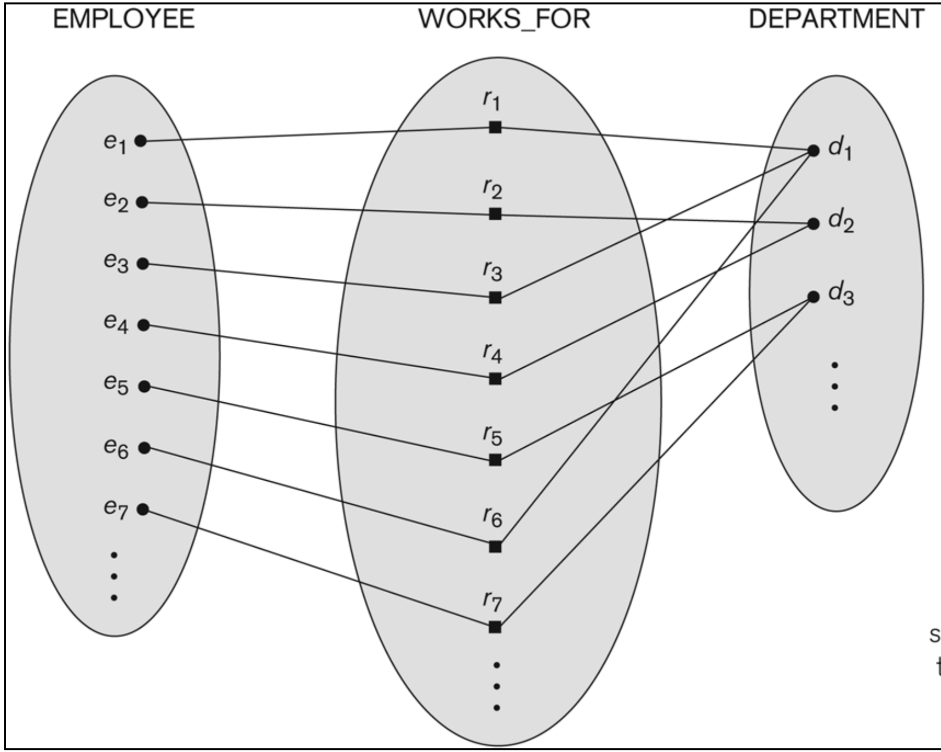


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

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

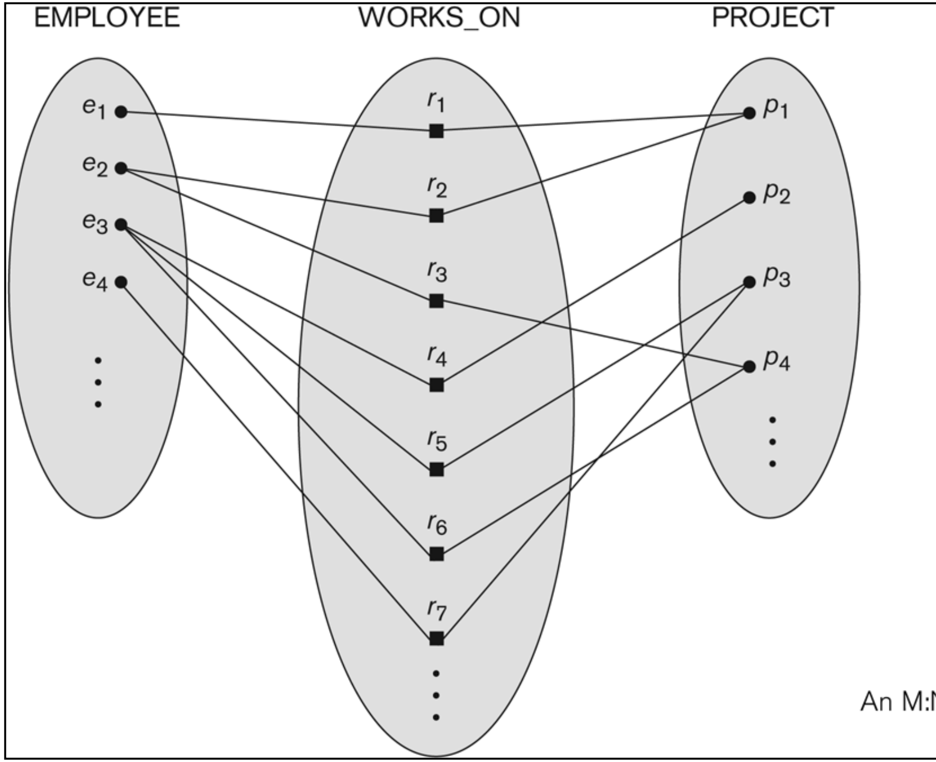


Figure 3.13
An M:N relationship, WORKS_ON.

Relationship type vs. relationship set (1)

Relationship Type:

- Is the schema description of a relationship
- Identifies the relationship name and the participating entity types
- Also identifies certain relationship constraints

Relationship Set:

- The current set of relationship instances represented in the database
- The current *state* of a relationship type

Relationship type vs. relationship set (2)

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

- Diamond-shaped box is used to display a relationship type
- Connected to the participating entity types via straight lines
- Note that the relationship type is not shown with an arrow. The name should be typically be readable from left to right and top to bottom.

Refining the COMPANY database schema by introducing relationships

By examining the requirements, six relationship types are identified

All are *binary* relationships (degree 2)

Listed below with their participating entity types:

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

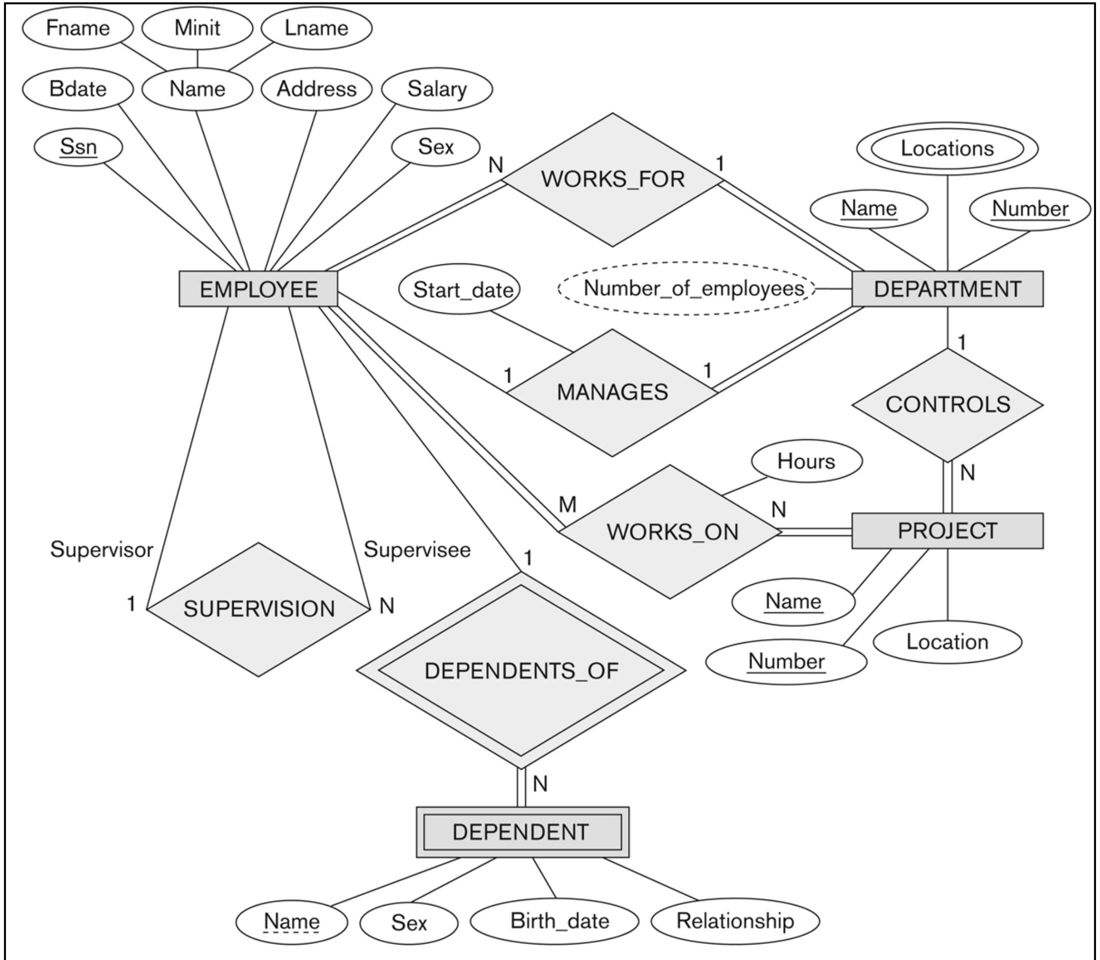


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

ER DIAGRAM – Relationship Types are:

WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF

Constraints on Relationships

Constraints on Relationship Types

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

Many-to-one (N:1) Relationship

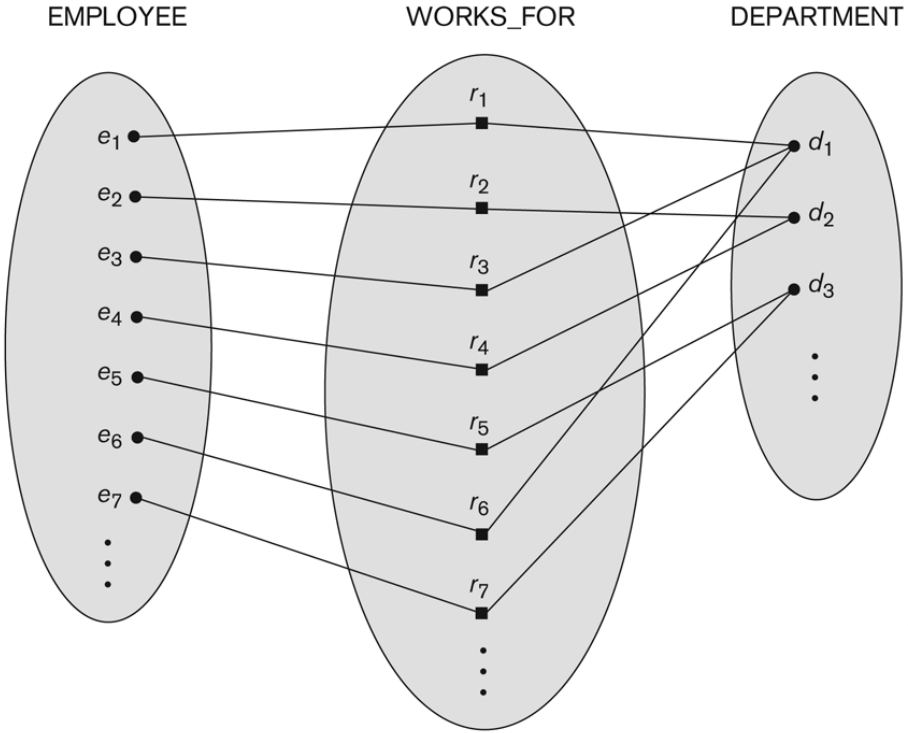


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Many-to-many (M:N) Relationship

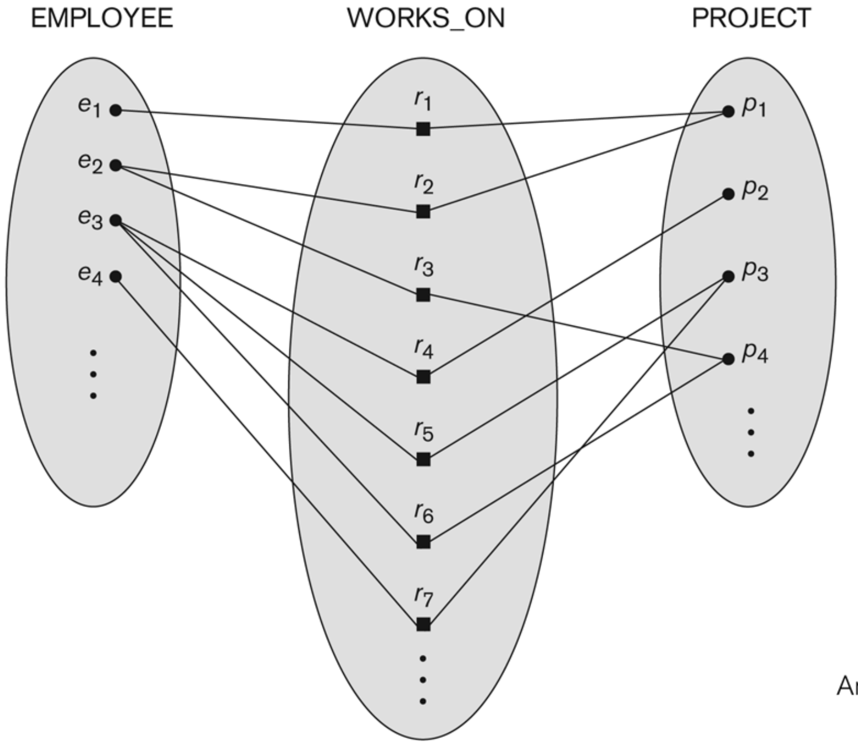


Figure 3.13
An M:N relationship, WORKS_ON.

Recursive Relationship Type

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

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

Example: the SUPERVISION relationship

EMPLOYEE participates twice in two distinct roles:

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

Each relationship instance relates two distinct EMPLOYEE entities:

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

Displaying a recursive relationship

In a recursive relationship type.

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

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

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

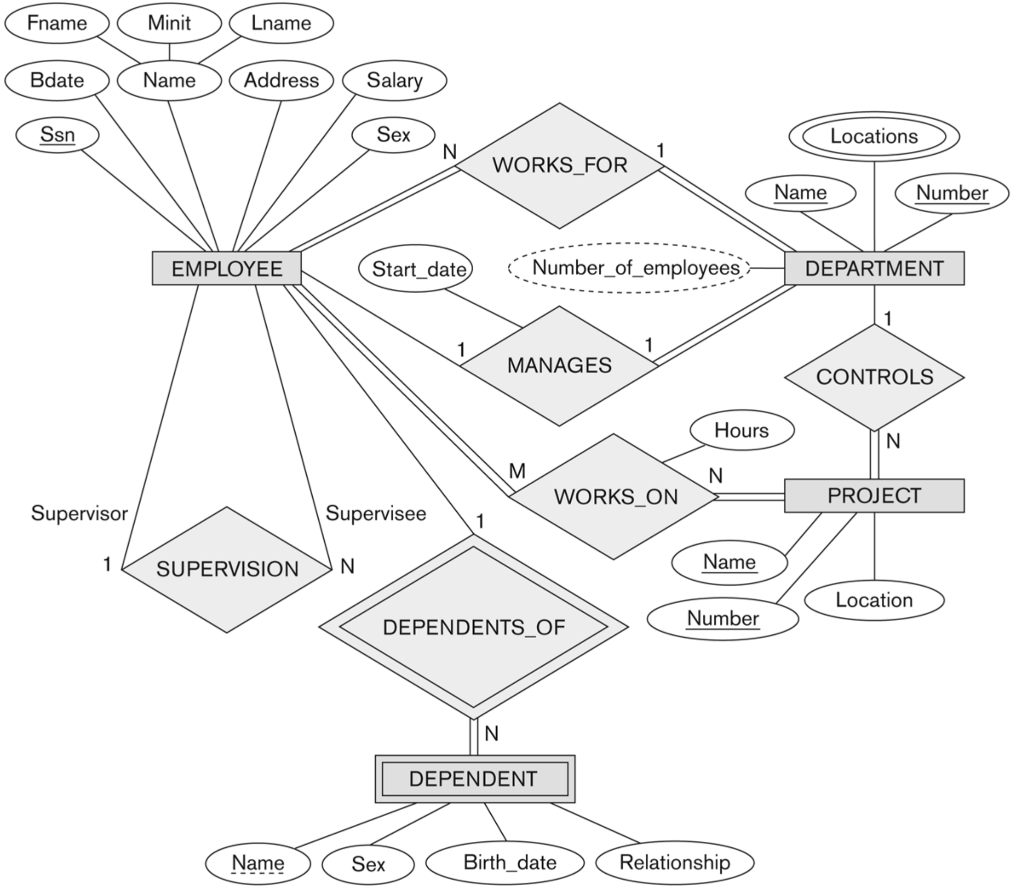


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

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

Attributes of Relationship types

A relationship type can have attributes:

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

Example Attribute of a Relationship Type: Hours of WORKS_ON

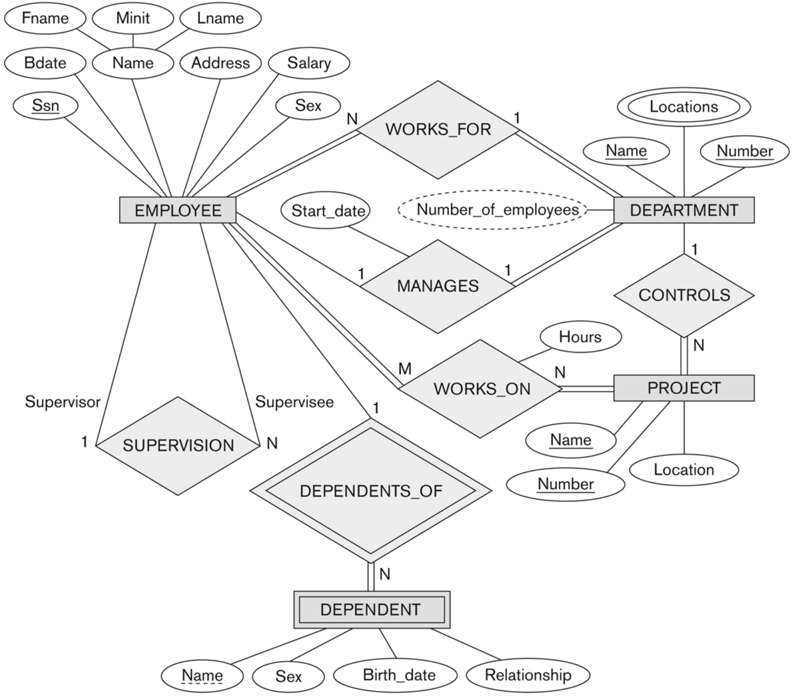


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Notation for Constraints on Relationships

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

Shown by placing appropriate numbers on the relationship edges.

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

Total shown by double line, partial by single line.

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

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

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

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

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

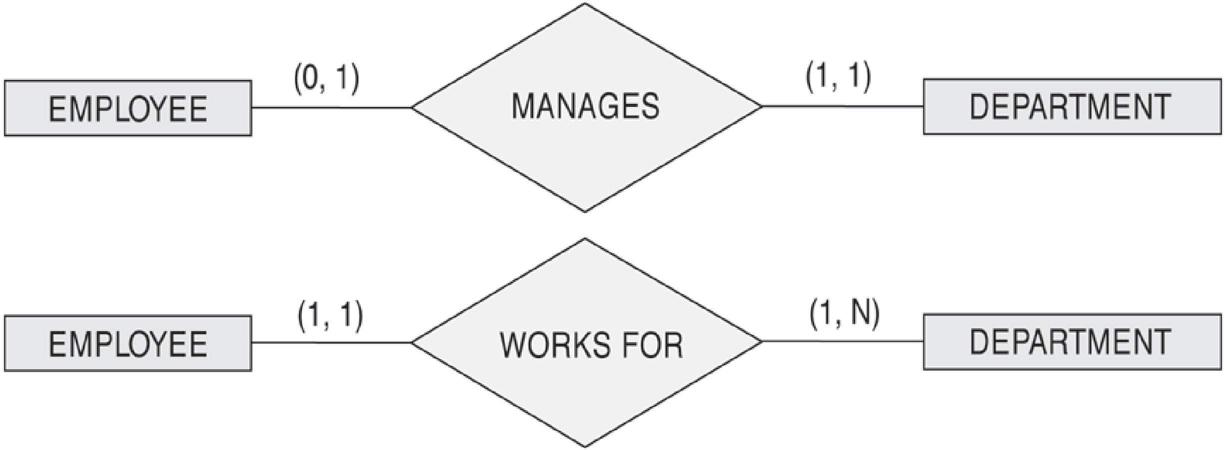
Derived from the knowledge of mini-world constraints

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

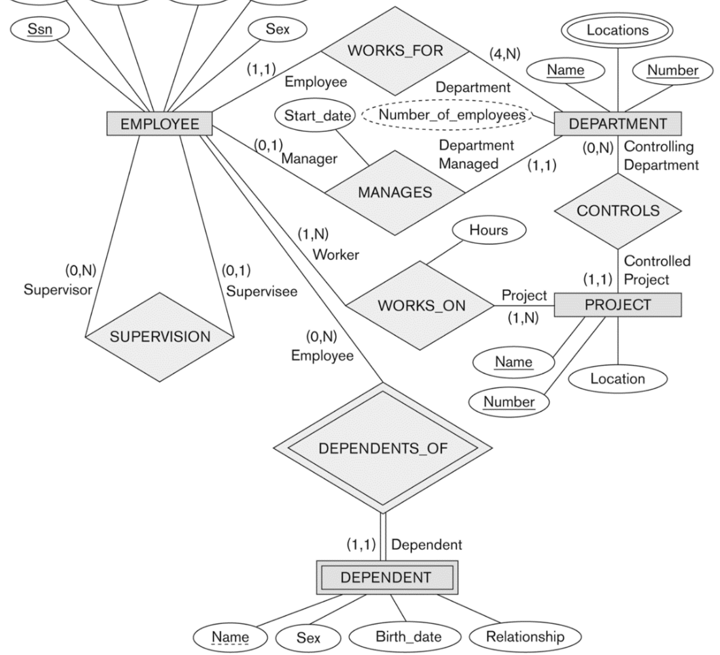


Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

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

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

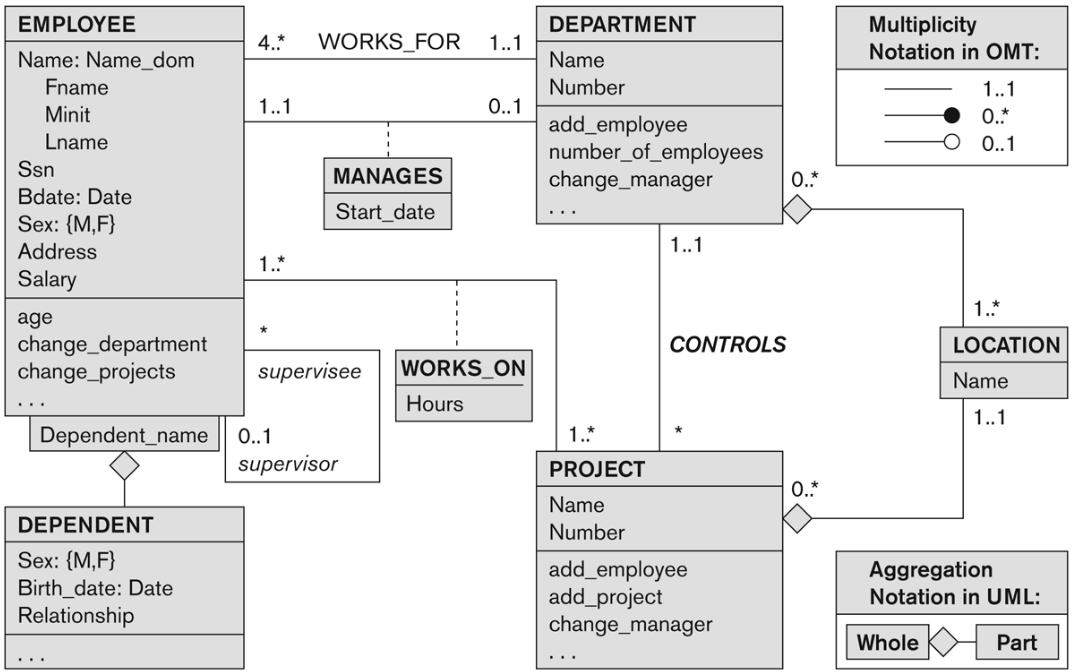
- Other UML terminology also differs from ER terminology

Used in database design and object-oriented software design

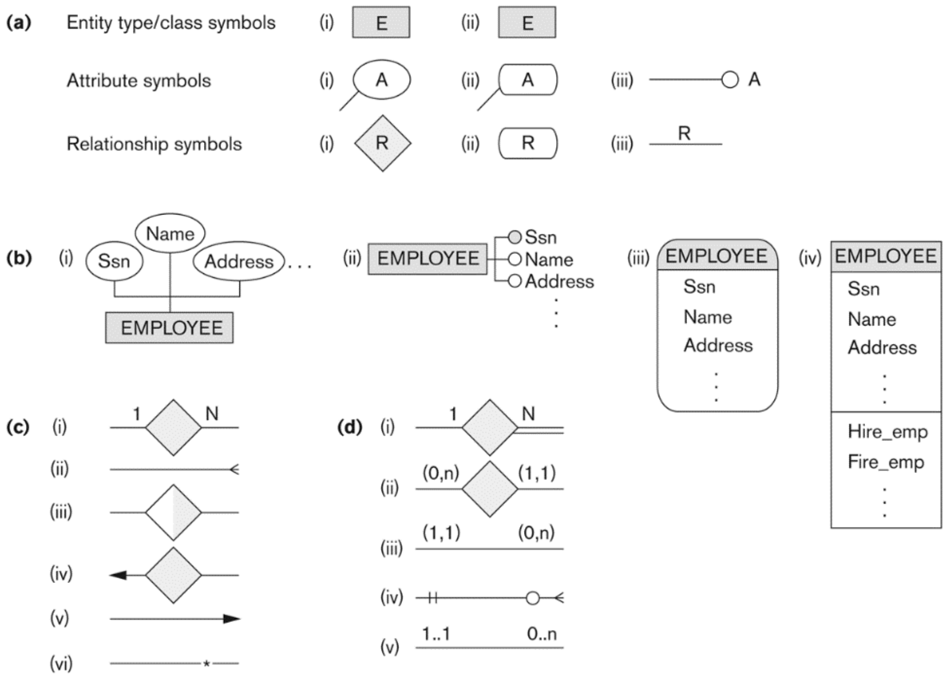
UML has many other types of diagrams for software design

UML class diagram for COMPANY database schema

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



Other alternative diagrammatic notations



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++

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*

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

Characteristics Of Relations

Values in a tuple:

- All values are considered atomic (indivisible).
- Each value in a tuple must be from the domain of the attribute for that column
  - If tuple t = <v1, v2, …, vn> is a tuple (row) in the relation state r of R(A1, A2, …, An)
  - Then each *vi* must be a value from *dom(Ai)*

A special **null** value is used to represent values that are unknown or not available or inapplicable in certain tuples.

Characteristics Of Relations

Notation:

- We refer to **component values** of a tuple t by:
  - t[Ai] or t.Ai
  - This is the value vi of attribute Ai for tuple t
- Similarly, t[Au, Av, ..., Aw] refers to the subtuple of t containing the values of attributes Au, Av, ..., Aw, respectively in t

CONSTRAINTS

Constraints determine which values are permissible and which are not in the database.

They are of three main types:

1. **Inherent or Implicit Constraints**: These are based on the data model itself. (E.g., relational model does not tuples to be duplicates)

2. **Schema-based or Explicit Constraints**: They are expressed in the schema by using the facilities provided by the model. (E.g., max. cardinality ratio constraint in the ER model)

3. **Application based or semantic constraints**: These are beyond the expressive power of the model and must be specified and enforced by the application programs.

Relational Integrity Constraints

Constraints are **conditions** that must hold on **all** valid relation states.

There are three *main types* of (explicit schema-based) constraints that can be expressed in the relational model:

- **Key** constraints
- **Entity integrity** constraints
- **Referential integrity** constraints

Another schema-based constraint is the **domain** constraint

- Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)

Formal Terms
Relation
Attribute
Domain
Tuple
Schema of a Relation
State of the Relation

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!!!