CS 2451
Database Systems:
Entity-Relationship (ER) Model

http://www.seas.gwu.edu/~bhagiweb/cs2541
Spring 2020
Instructor: Dr. Bhagi Narahari

1

Course Summary….

- Relational Data Model
- Formal query languages
  - Relational algebra and Relational Calculus
- SQL
  - DDL to define schema and constraints
  - Query component…basic SQL + non-RA operators (GroupBy etc.)
- Experience working with commercial DBMS and developing DB applications
  - MySQL, PHP
- Next - Database schema design: how to design a "good" schema, how to measure "good"?
  - Normal Forms (3NF, BCNF)
- Detour (this class): Conceptual Level Database design
  - Entity-Relationship (ER) Model

2

Database Design

- The ability to design databases and associated applications is critical to the success of the modern enterprise.
- Database design requires understanding both the operational and business requirements of an organization as well as the ability to model and realize those requirements in a database.
- Developing database and information systems is performed using a ***development lifecycle***, which consists of a series of steps.

3

The Importance of Database Design

- Just as proper design is critical for developing large applications, success of database projects is determined by the effectiveness of database design.
- Some statistics on software projects:
  - 80 - 90% do not meet their performance goals
  - 80% delivered late and over budget
  - 40% fail or abandoned
  - 10 - 20% meet all their criteria for success
  - Have you been on a project that failed? Yes ? No ?
- The primary reasons for failure are improper requirements specifications, development methodologies, and design techniques.

4

How Does One Build a Database?

- Requirements Analysis: what data, apps, critical operations
  - Get from "client"
    - Typically expressed in some natural language
  - May require going back to the client for resolving questions

  - Query and app development depends on client specifications

5

Building Database Applications: Steps

1. Start with a conceptual model
   - "On paper" using certain techniques
     E-R Model
   - ignore low-level details – focus on logical representation
   - "step-wise refinement" of design with client input
2. Design & implement schema
   - Design and codify (in SQL) the relations/tables
   - Refine the schema – *normalization*
   - Do physical layout – indexes, etc.
3. Import the data
4. Write applications using DBMS and other tools
   - Many of the hard problems are taken care of by other people (DBMS, API writers, library authors, web server, etc.)
   - DBMS takes care of Query Optimization, Efficiency, etc.
5. Test!!

6

Conceptual Model- Why ?

- Convey database design and properties in simple but precise manner
  - Interpreted by any type of user
    Does not need to know anything about CS
  - Capture the business rules of the application

- Picture is worth a thousand words

7

Conceptual Database Design

- ***Conceptual database design*** involves modeling the collected information at a high-level of abstraction without using a particular data model or DBMS.

- Since conceptual database design occurs independently from a particular DBMS or data model, we need high-level modeling languages to perform conceptual design.

- The entity-relationship (ER) model was originally proposed by Peter Chen in 1976 for conceptual design.
  - Can also do ER modeling using Unified Modeling Language (UML) syntax.

8

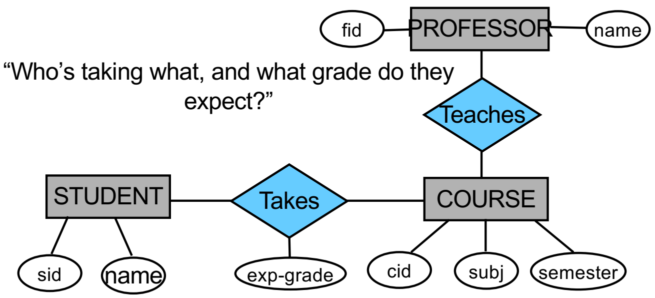
An Example: “mini” banner

- Database containing information about
  - Students
  - Faculty
  - Courses
- Students take courses
- Faculty teach courses
- How to ‘define’ student/faculty/course ?
  - What data is needed ?

9

Example: ER Design for mini-banner:

“Who’s taking what, and what grade do they expect?”



One picture provides info on what your system stores and models

10

Entity-Relationship Modeling

- ***Entity-relationship modeling*** is a top-down approach to database design that models the data as entities, attributes, and relationships.
- The ER model refines entities and relationships by including properties of entities and relationships called *attributes*, and by defining *constraints* on entities, relationships, and attributes.
- The ER model conveys knowledge at a high-level (conceptual level) which is suitable for interaction with technical and non-technical users.
- Since the ER model is data model independent, it can later be converted into the desired logical model (e.g. relational model).

11

Entity Relationship Model

- Based on collection of real world objects or concept called *entities;* ex: employee, student
  - *attribute* represents properties of entity; s.s.num
- *relationship* represents interaction between entities
- overall logical structure represented by ER diagram representing entity sets, relationships, attributes
- *Conceptual design*:
  - What are the *entities* and *relationships* in the enterprise?
  - What information about these entities and relationships should we store in the database?
  - What are the *integrity constraints* or *business rules* that hold?
- Can map an ER diagram into a relational schema.

12

ER Model Definitions

- *Entity*: Real-world object distinguishable from other objects.
  - An entity is described (in DB) using a set of *attributes*.
- *Entity Set*: A collection of similar entities. E.g., all employees.
  - All entities in an entity set have the same set of attributes. (Until we consider ISA hierarchies, anyway!)
  - Each entity set has a *key*.
  - Each attribute has a *domain*.
- An *entity instance* is a particular example or occurrence of an entity type…eg: Employee John Doe
- Representation/Syntax:
  - **Entity** set represented by **rectangle**
  - **Attribute** represented by **Oval**
    - Key attribute underlined
    - **Composite Attribute**: when it has multiple fields (ex: address)

13

ER Model Basics (Contd.)

- *Relationship*: Association among two or more entities. E.g., Dan takes Database Course; Attishoo works in Pharmacy department.
  - Relationship can also have attributes (that appear only for this relationship set)
- Representation/Syntax: a Diamond symbol
  - Attributes represented by Oval (same as before)
- *Relationship Set*: Collection of similar relationships.
  - An n-ary relationship set R relates n entity sets E1 ... En; each relationship in R involves entities e1 ∈ E1, ..., en ∈ En
    - Same entity set could participate in different relationship sets, or in different “roles” in same set.

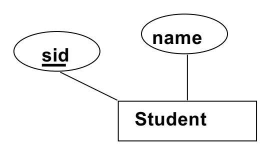
14

Conceptual Design Process

- What are the entities being represented? STUDENTS
- What are the relationships? Takes
- What info (attributes) do we store about each? name sid exp-grade
- What keys & integrity constraints do we have?

15

Student Entity



16

Example of a composite attribute

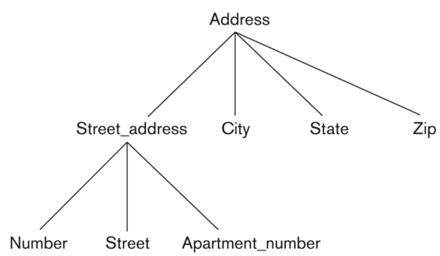
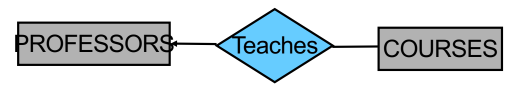


Figure 3.4
A hierarchy of composite attributes.

17

Connectivity in the E-R Diagram?

- Attributes can *only* be connected to entities or relationships
- Entities can *only* be connected via relationships
- As for the edges, let's consider kinds of relationships and integrity constraints…

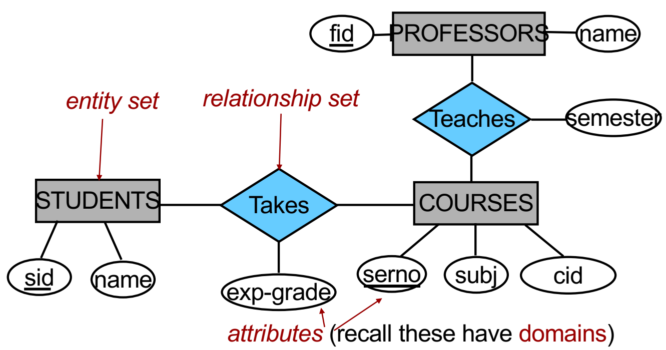


(warning: different ER implementations have slightly different notation here!)

18

Entity-Relationship Diagram for the Example

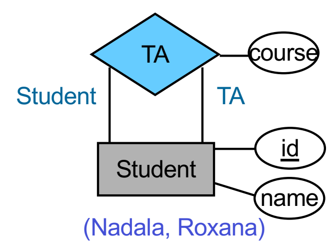
Underlined attributes are keys



19

Roles: Labeled Edges

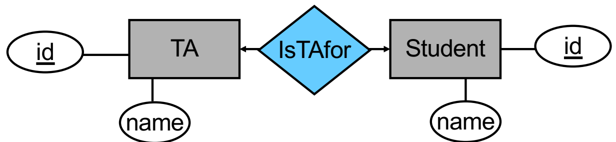
Sometimes a relationship connects the same entity, and the entity has more than one role:



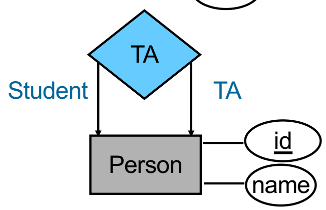
This often indicates the need for recursive queries

20

Roles vs. Separate Entities



What is the difference between these two representations?



21

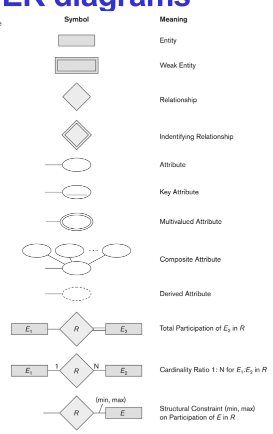
Weak Entity Sets

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
  - Weak entity set must have total participation in this *identifying* relationship set.
  - If Student is deleted, then we MUST delete the Parent
  - Syntax: Bold face rectangles, Double lined rectangles,…

22

NOTATION for ER diagrams

Figure 3.14
Summary of the notation for ER diagrams.



23

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

24

UML Diagrams – Alternate Syntax for ER Diagrams

- Unified Modeling Language (UML)
- Read on your own
- You've seen an example on the lab slides!

25

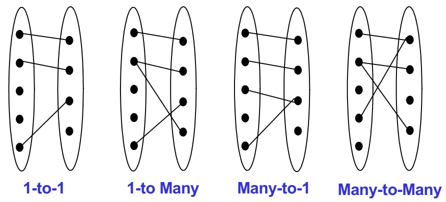
Defining Constraints in ER Model

- Contraints capture properties of the relationship and entities
  - Convey the business rules of the application
- Every entity set has a key attribute..similar to Rel. Model
  - No two elements can have the same value on this attribute
    - Example: Student ID
- How many elements in entity set are associated with another entity in the relationship ?
  - Can a student take more than one course ?
- Does every element in the entity set appear/participate in the relationship ?
  - Must every student take a course ?
- Define constraints based on properties of the mapping/relation between entity sets

26

Properties of relations

- Binary relationships can be classified as one-to-one, many-to-one, one-to-many, many-to-many
- What is the type of mapping/relation



27

Example: the Teaches relationship

- Want to model the info that each course is taught by one faculty.
  - Type of mapping ???
  - 1-to-1
    - Note: This is a Mapping and not a function!
- A student can take more than one course
  - 1 to Many
- Every course must have an instructor
  - Each element in the Course entity set must participate/appear in the Teaches relationship
- A faculty may teach zero or more courses

28

Example: the Takes Course Relationship

- Student can be enrolled in many courses and each Course can have many students
  - Type of mapping:
    Many to Many
- Want to model the condition that every student must take at least one course
  - Each student must appear in Takes relationship
- How many courses can a student take ?
  - Do we want to specify a limit ?
- How many students must be enrolled in a course ?
  - Is there a minimum size for a class ?

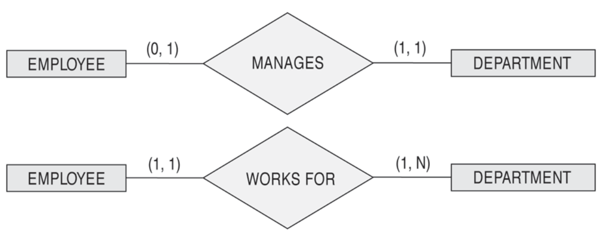
29

Mapping Cardinality, Participation Constraints, Structural constraints

- Type of mapping (**cardinality**)
  - 1-1, 1-many, many-many, many-1
  - Provides some information on relationship sets
- Participation constraints
  - *Total vs Partial*
    Total: Every student sid must appear in Takes relationship
    Partial: All faculty need not appear in Teaches relationship
- Structural constraints:
  - Minimum and maximum times they can appear in relationship
  - Syntax ??

30

The (min,max) notation for relationship constraints



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

31

COMPANY ER Schema Diagram using (min, max) notation

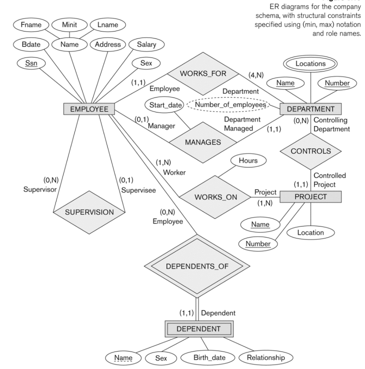


Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

32

Conceptual Design Using the ER Model

- Design choices:
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: constraints, type, participation
- Constraints in the ER Model:
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams.

33

Summary of Conceptual Design

- *Conceptual design* follows *requirements analysis*,
  - Yields a high-level description of data to be stored
  - Visual language – the diagram is the syntax!
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
  - There are additional constructs in a “real” ER model based tools.
- Can automate mapping of ER model to relational tables!

34

A detailed example: The Company Database

- COMPANY database keeps track of Employees and Departments
  - Employees identified by SSN, Name, Location
  - Department specified byDepartment ID (did), Name, Budget
- Each department has a unique manager
  - Database must keep track of starting date
- Each employee works in a department
  - Database must keep track of starting date

35

Initial Conceptual Design of Entity Types for the COMPANY Database Schema

- Based on the requirements, we can identify four initial entity types in the COMPANY database:
  - DEPARTMENT
  - PROJECT
  - EMPLOYEE
  - DEPENDENT
- Their initial conceptual design is shown on the following slide
- The initial attributes shown are derived from the requirements description

36

Initial Design of Entity Types: EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

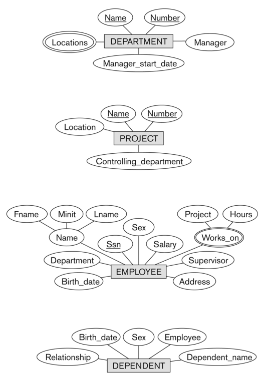


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

37

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

38

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.

39

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

40

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

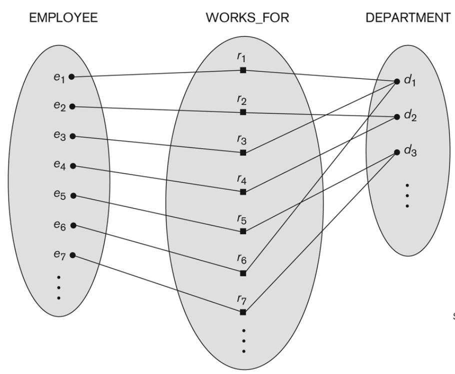


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

41

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

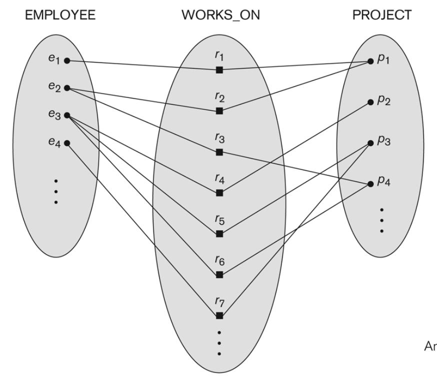


Figure 3.13
An M:N relationship, WORKS_ON.

42

Relationship type vs. relationship set (2)

- Previous figures displayed the relationship sets
- Each instance in the set relates individual participating entities – one from each participating entity type
- In ER diagrams, we represent the *relationship type* as follows:
  - Diamond-shaped box is used to display a relationship type
  - Connected to the participating entity types via straight lines
  - Note that the relationship type is not shown with an arrow. The name should be typically be readable from left to right and top to bottom.

43

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)

44

ER DIAGRAM – Relationship Types are:

WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF

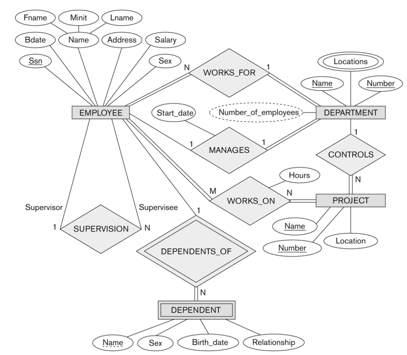


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

45

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.

46

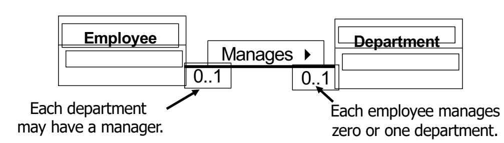
Constraints on Relationships

- 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)

47

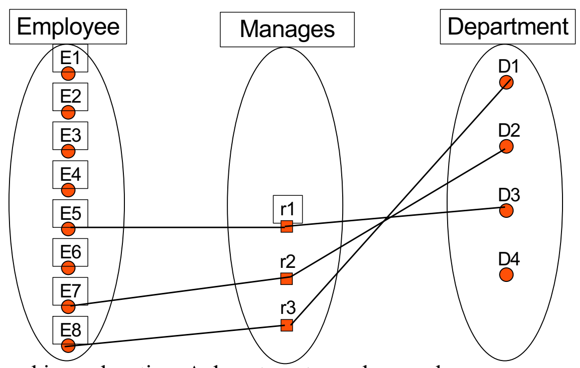
One-to-One Relationships

- In a one-to-one relationship, each instance of an entity class E1 can be associated with ***at most one*** instance of another entity class E2 and vice versa.
- Example: A department may have only one manager, and a manager may manage only one department.



48

One-to-One Relationship Example



Relationship explanation: A department may have only one manager. A manager (employee) may manage only one department.

49

Many-to-one (N:1) Relationship

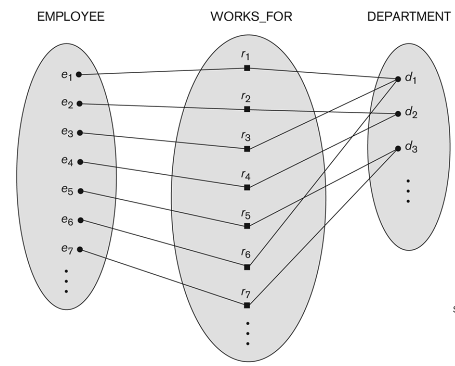
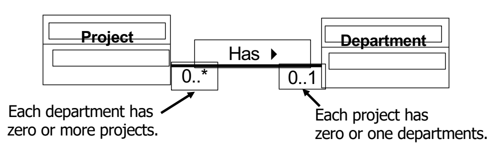


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

50

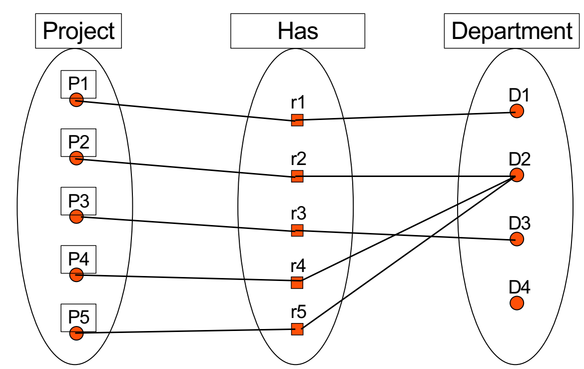
One-to-Many Relationships

- In a one-to-many relationship, each instance of an entity class E1 can be associated with ***more than one*** instance of another entity class E2. However, E2 can only be associated with ***at most one*** instance of entity class E1.
- Example: A department may have multiple projects, but a project may have only one department.



51

One-to-Many Relationship Example

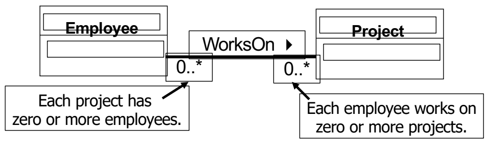


Relationship: One-to-many relationship between department and project.

52

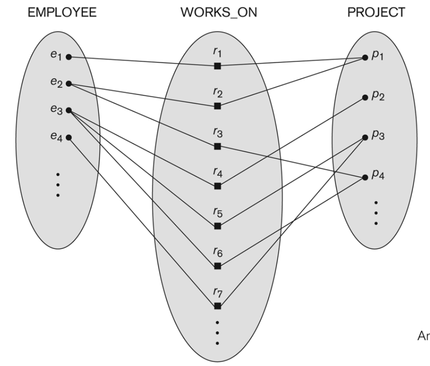
Many-to-Many Relationships

- In a many-to-many relationship, each instance of an entity class E1 can be associated with ***more than one*** instance of another entity class E2 and vice versa.
- Example: An employee may work on multiple projects, and a project may have multiple employees working on it.



53

Many-to-many (M:N) Relationship



An M:N relationship, WORKS_ON.

54

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

55

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.

56

A Recursive Relationship Supervision`

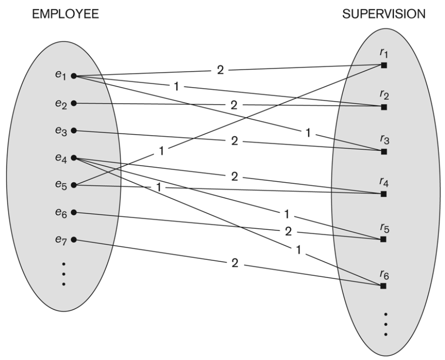


Figure 3.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

57

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

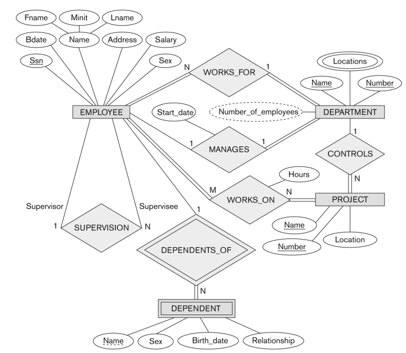


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

58

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

59

Attributes of Relationship types

- A relationship type can have attributes:
  - For example, HoursPerWeek of WORKS_ON
  - Its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.
    - A value of HoursPerWeek depends on a particular (employee, project) combination
  - Most relationship attributes are used with M:N relationships
    - In 1:N relationships, they can be transferred to the entity type on the N-side of the relationship

60

Example Attribute of a Relationship Type: Hours of WORKS_ON

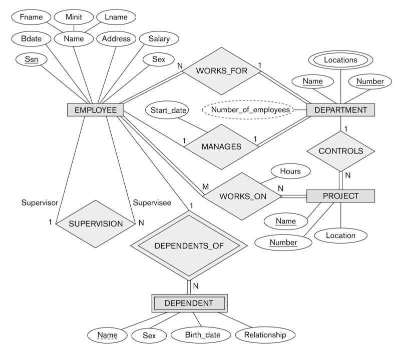


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

61

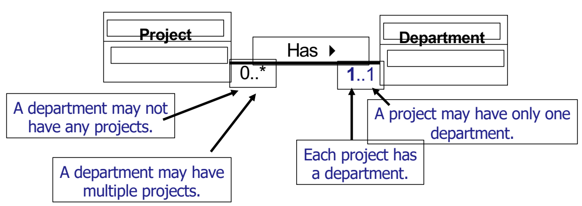
Participation Constraints

- ***Cardinality*** is the *maximum* number of relationship instances for an entity participating in a relationship type.
- ***Participation*** is the *minimum* number of relationship instances for an entity participating in a relationship type.
  - Participation can be *optional* (zero) or *mandatory (1 or more)*.
- If an entity's participation in a relationship is mandatory (also called *total* participation), then the entity's existence depends on the relationship.
  - Called an *existence dependency*.

62

Participation Constraints Example

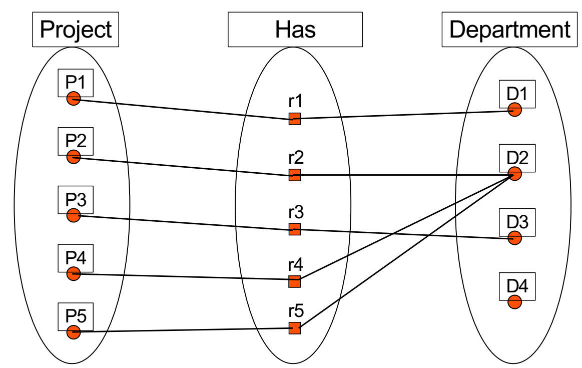
- Example: A project is associated with one department, and a department may have zero or more projects.



Note: Every project must participate in the relationship (mandatory).

63

One-to-Many Participation Relationship Example

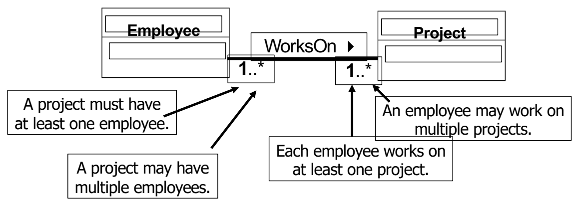


Relationship explanation: A project must be associated with one department. A department may have zero or more projects.

64

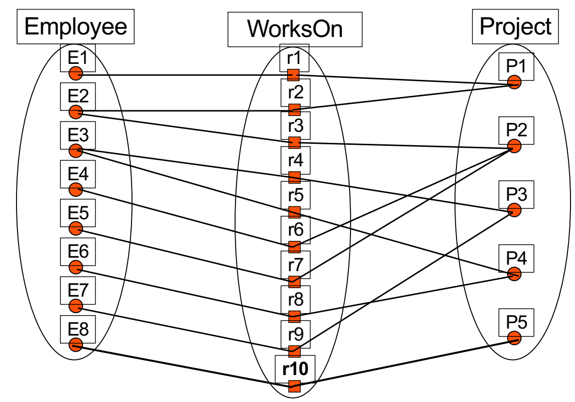
Participation Constraints Example 2

- Example: A project must have one or more employees, and an employee must work on one or more projects.



65

Many-to-Many Relationship Participation Example



66

Notation for Constraints on Relationships

- Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N
  - Shown by placing appropriate numbers on the relationship edges.
- Participation constraint (on each participating entity type): total (called existence dependency) or partial.
  - Total shown by double line, partial by single line.
- NOTE: These are easy to specify for Binary Relationship Types.

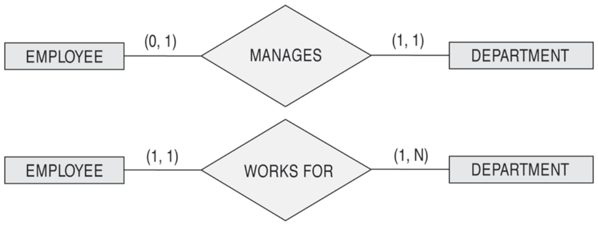
67

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≤max, min≥0, max ≥1
- Derived from the knowledge of mini-world constraints
- Examples:
  - A department has exactly one manager and an employee can manage at most one department.
    - Specify (0,1) for participation of EMPLOYEE in MANAGES
    - Specify (1,1) for participation of DEPARTMENT in MANAGES
  - An employee can work for exactly one department but a department can have any number of employees.
    - Specify (1,1) for participation of EMPLOYEE in WORKS_FOR
    - Specify (0,n) for participation of DEPARTMENT in WORKS_FOR

68

The (min,max) notation for relationship constraints

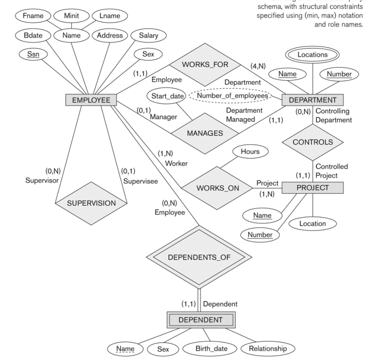


Read the min,max numbers next to the entity type and looking **away from** the entity type

69

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.



70

Alternate “syntax” for ER Model: UML Notation

- If you are familiar with UML, then ER database design can be expressed using Unified Modeling Language (UML) diagrams

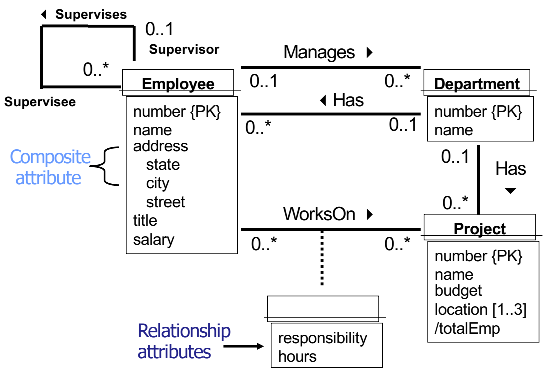
71

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

72

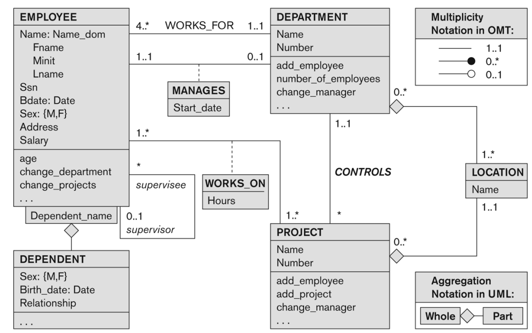
ER Model Example in UML notation



73

UML class diagram for COMPANY database schema

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



74