

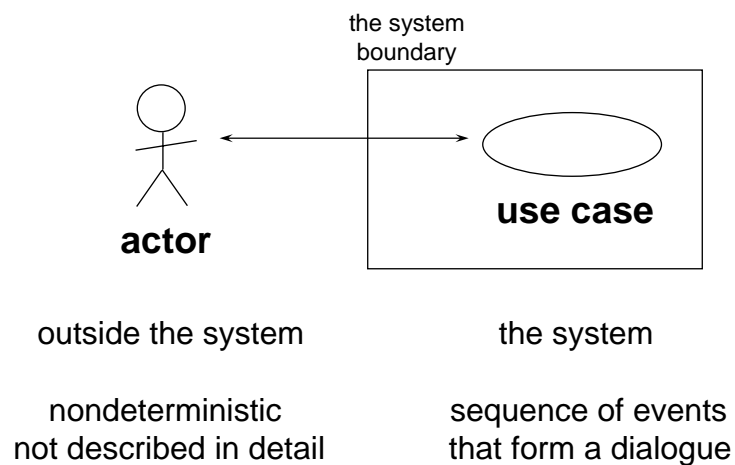
Lecture 4

- Use Cases
 - Model
 - Scope
 - Actors
 - Examples
 - Exceptions
 - User Interfaces
 - Refined Diagrams
- ER Diagrams
 - Notation
 - Cardinality Constraints
 - Weak Entities
 - Modeling Options
 - [Converting to Tables]

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission 4/25/2006

Slide 1
Lecture 4

Use Case Model



CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission 4/25/2006

Slide 2
Lecture 4

Setting the Scope

Describe the purpose of the System
with a name & short text annotation

Set the System Boundary
What's in, what's out

Identify outside or existing systems:
What's out
(decide what you will implement vs.
what you will interface to)

Setting the Scope of the “System”

Identify what's inside the “System”

ATM
machine for
a bank

or is it a network of ATMs for a bank?
or is it a distributed, full-service bank system?

Actors for Use Cases

actor - anything that needs to exchange information with the system. actor can be a system.

One user or user group may play one or more roles (that is may fill the role of one or more actors)

Identify all actors - to help identify the scope of the system

Describe the purpose for each actor (role)

Possible Kinds of Actors

Prototypical user (plays a certain role)

External system (provides service)

DBA

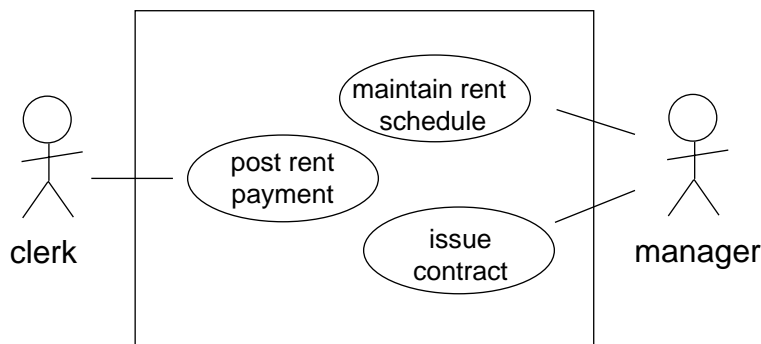
- system startup
- account creation
- operations
- systems maintenance
- termination

Identify All Potential Actors

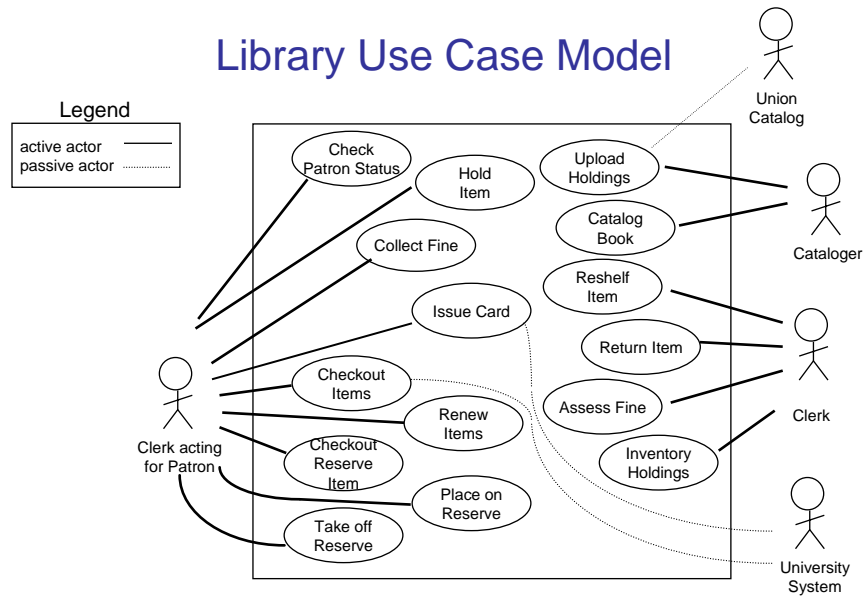
An *active* actor initiates the use case with a request to the system. A *passive* actor just responds to requests from the system.
Are your actors passive or active? human?



Tenant System Use Case Model



three "use cases" have been identified so far, each use case just has a name



CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission 4/25/2006

Slide 9
Lecture 4

Use Case Models vs. Use Cases

- We've been considering use case models, and names of use cases.
- Models are a graphical representation of a number of use cases.

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission 4/25/2006

Slide 10
Lecture 4

Club Information System Example

- Software to support the student-run clubs at a university
- Clubs have officers and members. Clubs schedule and sponsor events, which are optionally open to the public. Clubs have meetings with agenda and minutes. A club can have a mailing list.
- What are the boundaries of the system?
 - Room reservation?
 - Membership roster?
 - Mailing list creation, management, posting, archive?

Example Use Case – Search Event

Actor: A member of the public (MP)

Use case: The MP is searching for club events on a particular date.

Preconditions: The MP is at the CIS home page, but not logged in as a member.

Scenario A:

1. MP selects "Search Events" on MP home page
2. System presents a page with choice of dates for the current month
3. MP selects a date from among the choices
4. System presents a page with events for that date, giving time and club name
5. MP selects an event
6. System presents a page with details of that event, including location, description and cost

Exceptions to Normal Events

- Can be listed at the **end** of the use case or, if it's not too complex, within the use case using control flow operators.
- Example:
 - 4a. If there are no events for the selected date, System presents a page saying that there are no events for the selected date

Variants of Scenario

Slightly different sequence of events that doesn't merit a separate scenario or use case

Example:

Alternative Scenario A1:

- 3alt. MP selects a different month
- 4alt. System presents a page with choice of dates for the current month

Use of Use Cases

Use cases have many purposes

- Get user feedback about system capabilities
- Determine what functions and components the system needs
- Identify data requirements

Finding Use Cases

- Identified through actors
(identify actors first)
- Consider each actor, in turn;
decide on use cases
- Read the requirements, from the
point of view of each actor

Questions to help discover use cases:

- What are the main tasks of each actor?
- Will the actor read, write or change system information?
- Does the actor inform the system of outside changes?
- Does the actor wish to be informed of unexpected changes?
- Is this an *active* or a *passive* actor?

User Interface Sketches

- We need to understand how the user interface affects the use cases
- Appropriate to do during the use case model (using available techniques)
- Good way to engage users; get use cases right
- Detailed user interface **design** is a separate activity from developing the use case model

Sketch the User Interface

to encourage brainstorming about functions
(should be user interface technology-neutral)

Enter Event Information

Club **Sailing** Date: **5 June 2006**

Location SH 143 NH 281	Event Title Training for regatta	Description Mandatory meeting for all members serving as officials in July regatta
------------------------------------------------------------------------------------------	-------------------------------------	---------------------------------------------------------------------------------------

ok
cancel
 Public event

How much detail should we put in a use case?

- **Should we find all use cases?**
 80-20 rule...try to get the most important use cases. Try to get a range of use cases (e.g., several from each actor) initially
- **Should we write all exceptions?**
 Not necessarily, especially early on
- **Should an exception or alternative be a use case in its own right?**
 If the behavior is significantly different – write another use case... otherwise simply list at the end

Use Case Model in Two Styles

- Original use cases

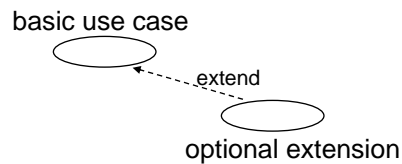
each scenario = one meaningful interaction
no concern about relationship among use cases

- Refined Use Case Model

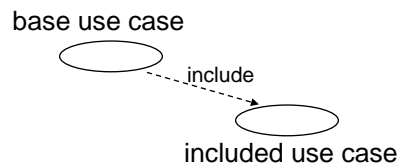
identify common parts; connect the use cases with
<<include>> and <<extend>> links

Refined Use Case Diagram

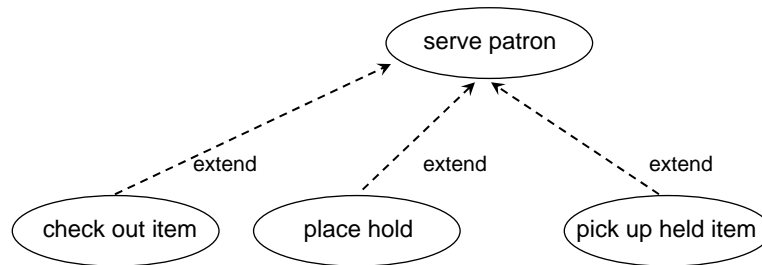
UML notation:
when a use case is
optionally invoked
(a use case *extension*
needs *insertion point*)



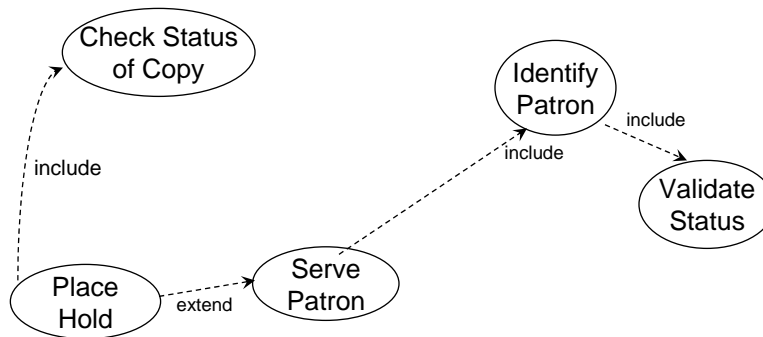
UML notation:
when a use case is
always invoked, like
a procedure
(an *included* use case)



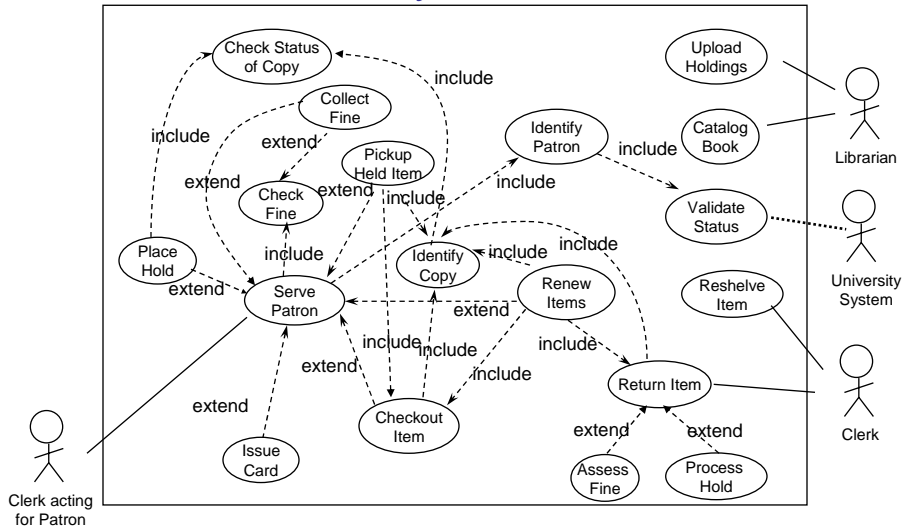
Example of Use Case Extension



Example of Included Use Case



Refined Library Use Case Model



Conceptual Database Design Using the Entity-Relationship (ER) Model

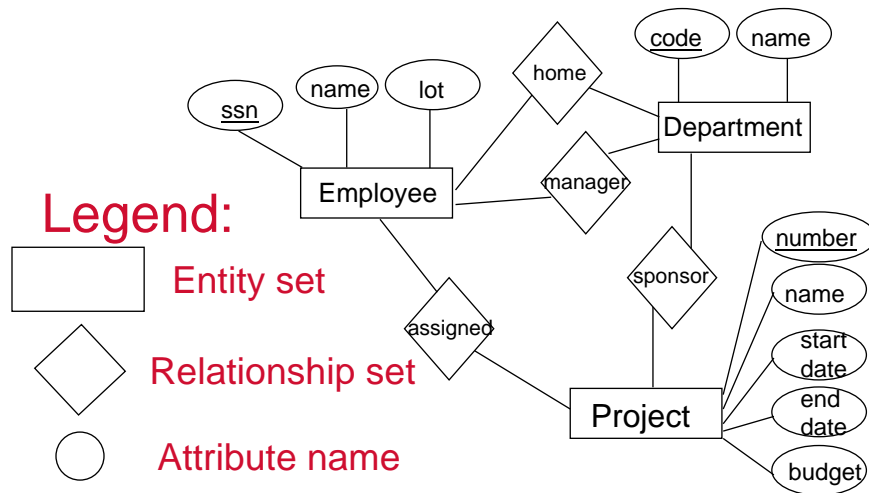
Overview of Database Design

- **Conceptual design:** (ER Model is used for this.)
 - What are the **entities** and **relationships** we need?
- **Logical design:**
 - Transform ER design to Relational Schema
- **Schema Refinement:** (Normalization)
 - Check relational schema for redundancies and related anomalies.
- **Physical Database Design and Tuning:** *We'll do these later*
 - Consider typical workloads; (sometimes) modify the database design; select file types and indexes.

Entity-Relationship Model is a different model than the Relational Model

- **Relation model** has:
 - **tables** (relations) with attributes, keys, foreign keys, domain definitions for attributes
- **Entity-Relationship model** has:
 - **Entities and entity sets** with attributes, keys, and domain definitions for attributes
 - **Relationships among entities and relationship sets** with cardinality constraints (in the book they refer to key constraints and participation constraints)

Entity-Relationship Diagram (original syntax)



CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission 4/25/2006

Slide 29
Lecture 4

Definitions

- **Entity:** Real-world object distinguishable from other objects.
 - An entity is described using a set of *attributes*.
- **Entity Set:** A collection of similar entities. E.g., all employees. (often referred to as just entity)

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission 4/25/2006

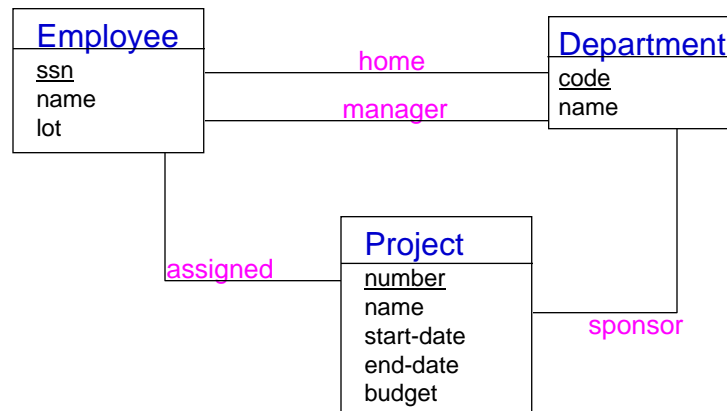
Slide 30
Lecture 4

Definitions

- Relationship: Association among 2 or more entities.
E.g., Attishoo's home department is Pharmacy.
- Relationship Set: Collection of similar relationships.
E.g., Home (often referred to as just relationship).

UML version of the same E-R Diagram

UML: Unified Modeling Language – for OO Design



Practice 1

- A club has a name, office and phone; it is uniquely identified by their names.
- Clubs sponsor events. Each event has one main sponsor, and may have co-sponsors.
- An event has a title, date, location and description; it is uniquely identified by the title and date.

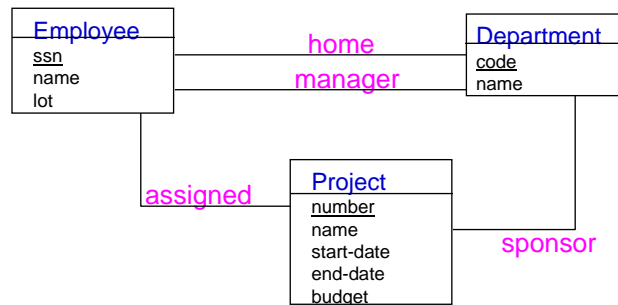
Draw the Entity-Relationship Diagram (ERD).

Practice 2

- Let's raw the UML version of the previous ERD.

Equivalent Relational Schema

Employee (ssn, name, lot, home-dept)
 Project-team(ssn, number)
 Department (id, name, manager)
 Project (number, name, start-date, end-date, budget, sponsor)



CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
 Some slides adapted from R. Ramakrishnan, with permission 4/25/2006 Slide 35
 Lecture 4

Equivalent Relational Schema - with foreign keys shown

Employee (ssn, name, lot, home-dept)

Project-team(ssn, number)

Department (id, name, manager)

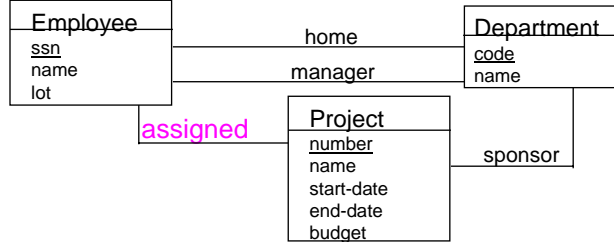
Project (number, name, start-date, end-date, budget, sponsor)

Notice that the many-to-many relationship set must be represented in a (new) table.

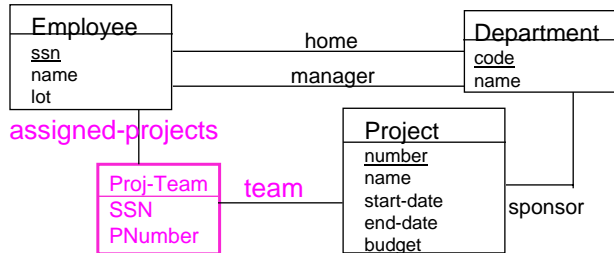
CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
 Some slides adapted from R. Ramakrishnan, with permission 4/25/2006 Slide 36
 Lecture 4

Many-to-many relationship sets

ERD



Relational
DB Diagram



CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 37
Lecture 4

What data do we need to record a relationship?



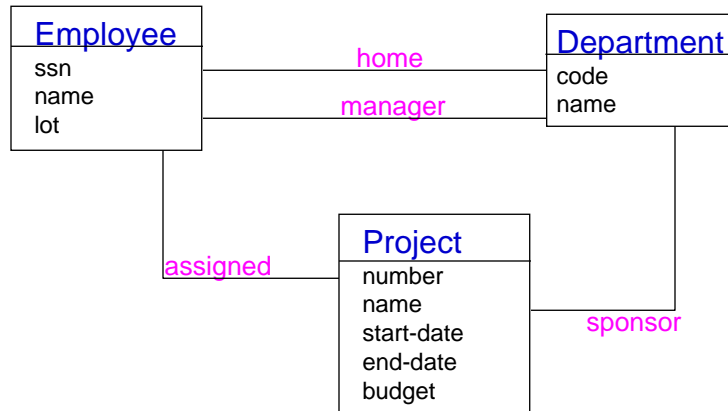
We must indicate which employee and which department we want to have connected (for each relationship).

We need the key value for an employee and the key value for the department – stored together – to represent the relationship.

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 38
Lecture 4

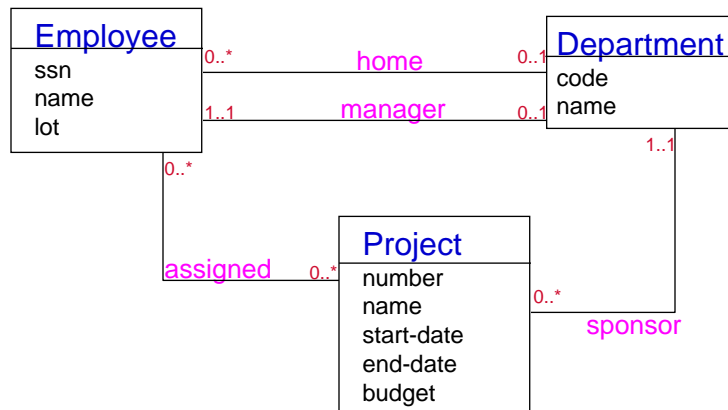
Cardinality Constraints on Relationship sets: How many entities can participate?



CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission 4/25/2006

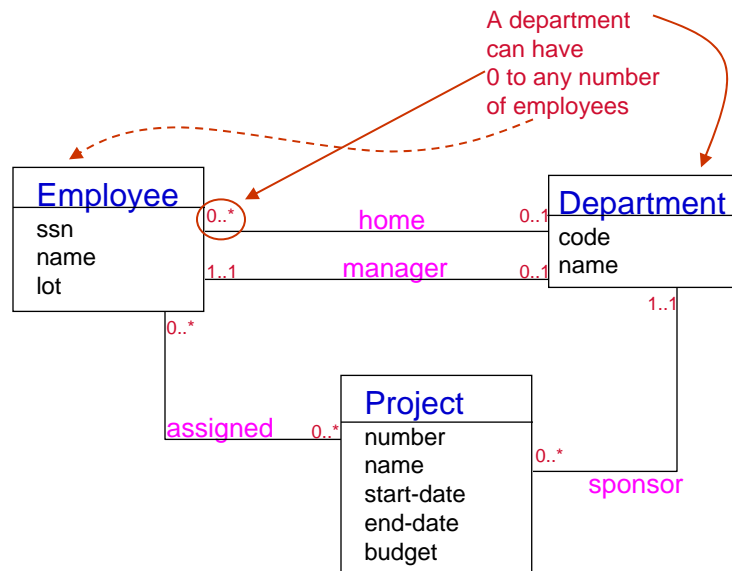
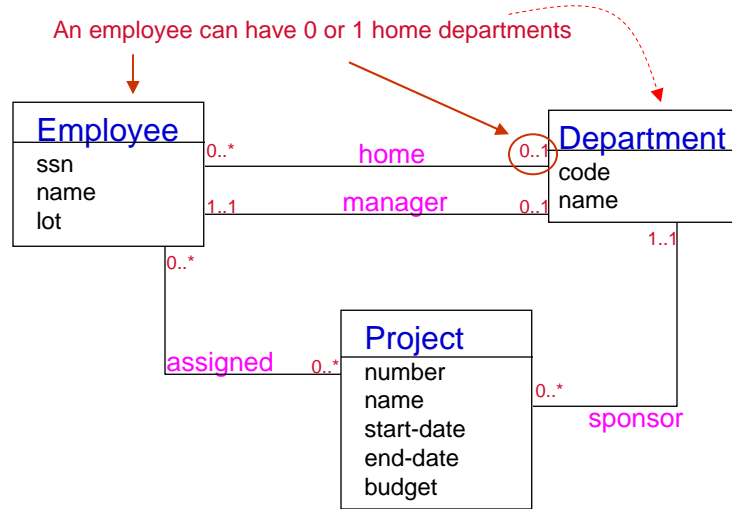
Slide 39
Lecture 4

Cardinality Constraints on Relationship sets How many entities can participate?

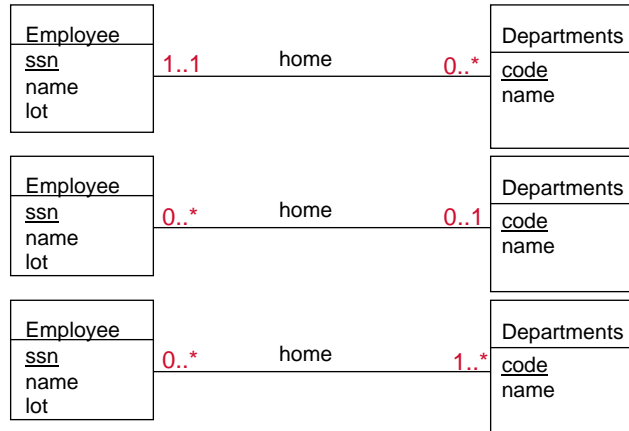


CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission 4/25/2006

Slide 40
Lecture 4

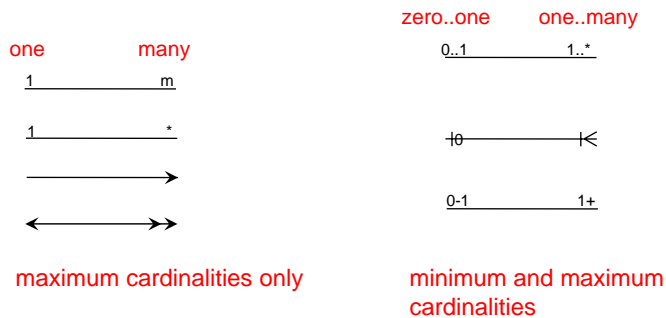


Some Alternative Constraints (in UML)

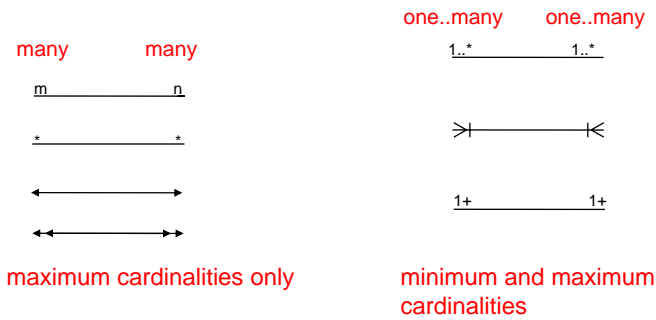


Which one is right? We must discover the semantics of the application!

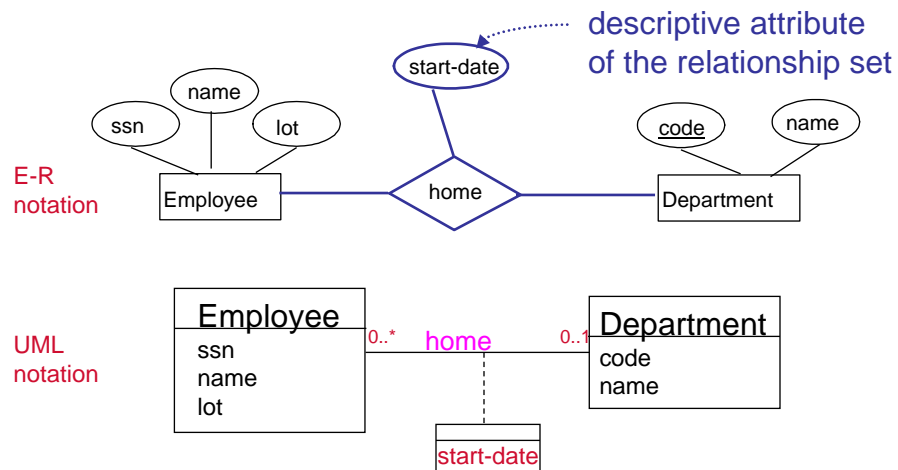
Various notations for “one-to-many”



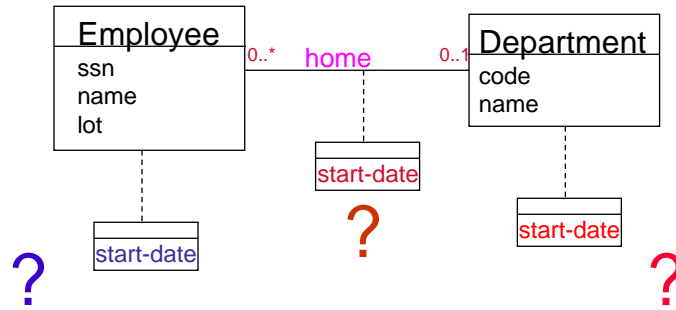
Various notations for “many-to-many”



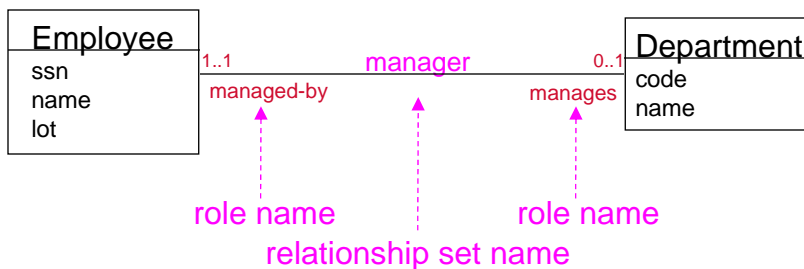
Relationship sets can have attributes



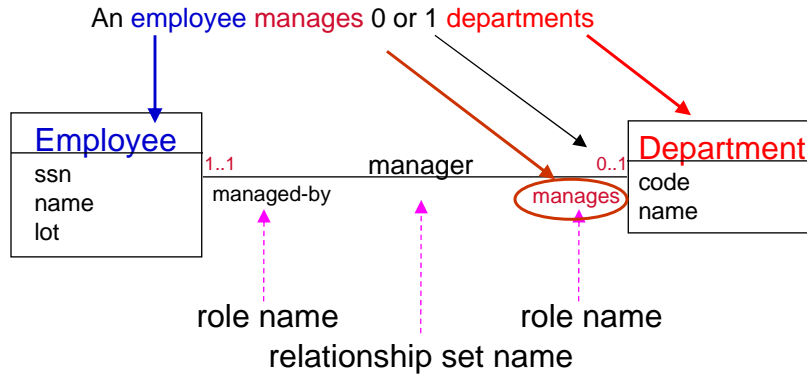
Try all three locations for the attributes:
which one makes sense?



Relationship sets can have **role** names
(in addition to the name of the relationship set)



Example: reading role names

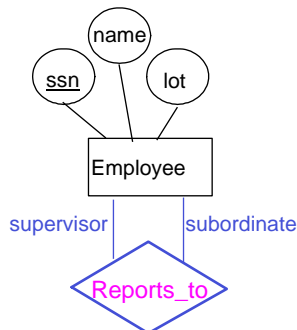


CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission 4/25/2006

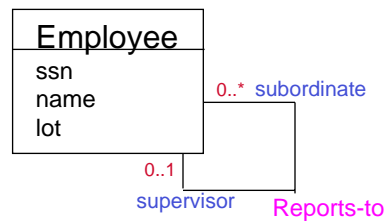
Slide 49
Lecture 4

Same entity sets can participate in different “roles” for the same relationship set

E-R notation



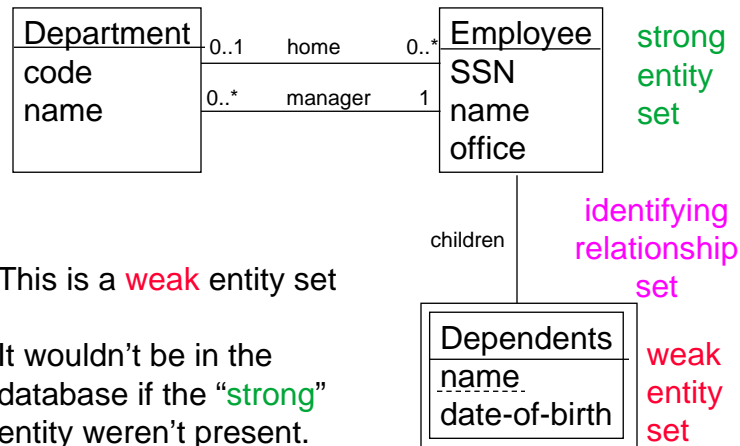
UML notation



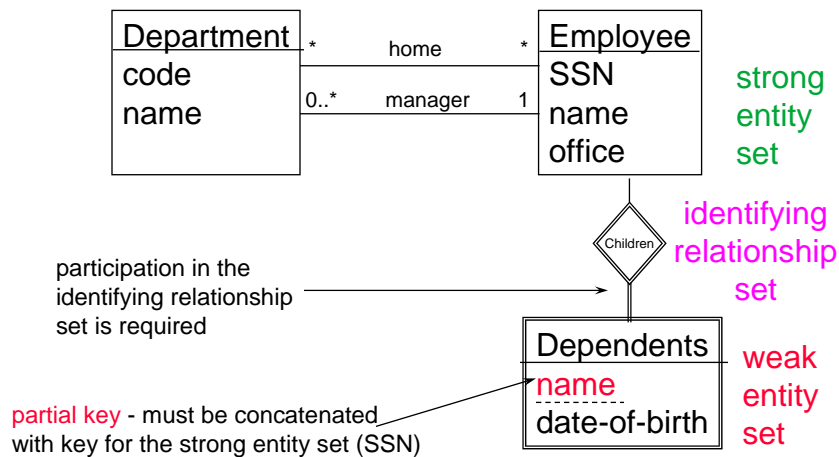
CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission 4/25/2006

Slide 50
Lecture 4

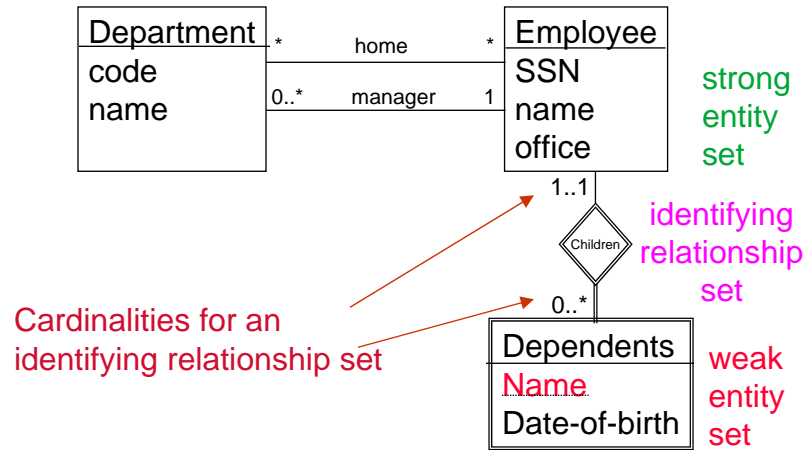
Weak Entity Sets (and Identifying Relationship sets)



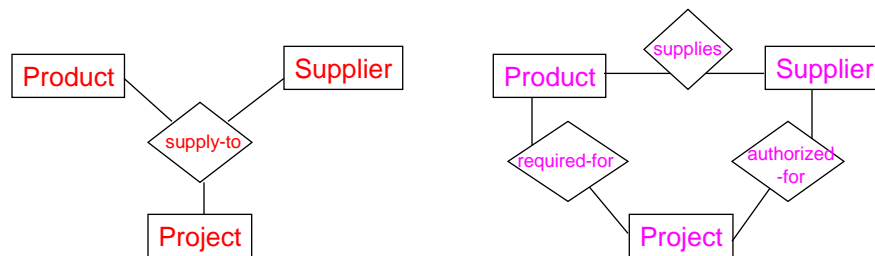
Weak Entity Sets and Identifying Relationship sets: Alternative Notation



Weak Entity Sets and Identifying Relationship sets: Alternative Notation (cont.)



Ternary vs. Binary Relationship Sets (Ternary = 3-way, Binary = 2-way)



These two schemas are not equivalent!
When would we use a ternary relationship set?
When would we use three binary relationship sets?

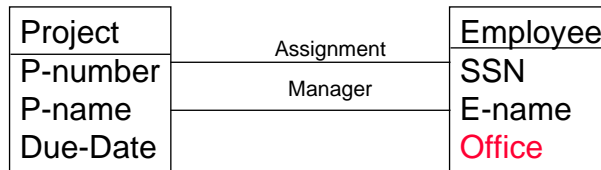
Ternary vs. Binary Relationship Sets (Cont.)

The ternary relationship set means that a Supplier must be authorized to supply a particular part to a particular project. e.g., **Office-Depot can supply laser printer paper to project 112.** **Office-Max can supply paper clips to Project 112.** **Office-Max can supply pencils to project 115.** (But based on that much information, **Office-Max can't supply pencils to 112.**)

Ternary vs. Binary Relationship Sets (Cont.)

The three binary relationship sets each represent something distinct. A Supplier can be authorized to supply certain products (**Office-Max can supply pencils**). A Project can require certain products (**112 needs pencils**). And a Supplier can be authorized to supply a certain project. (**Office-Max supplies 112**)
Therefore: **Office-Max can supply pencils to 112.**

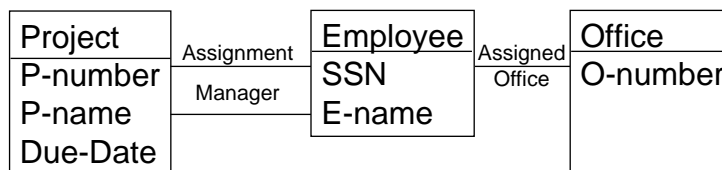
Duality: **entity** ↔ **value**
 and **attribute** ↔ **relationship**



Should **Office** be an **attribute of Employee**? or a **separate entity set**? Most attributes can be “promoted” to an entity set and some entities can be “demoted” to an attribute value.

This explains why there are so many different ways to design a schema.

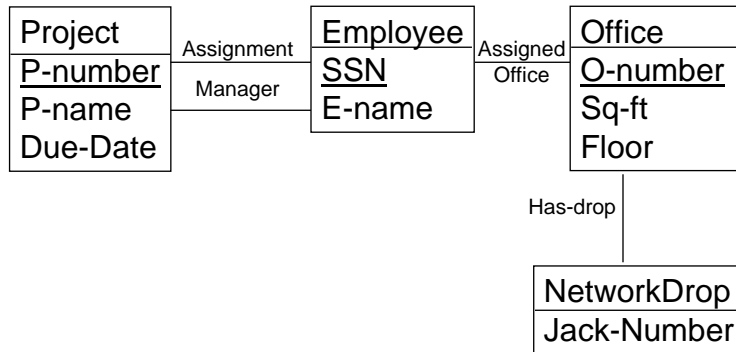
Entity vs. Value of an Attribute



What are some reasons to model Office as an entity set?

- An employee can have more than one office
- There are other attributes of Office
- Office needs to participate in other relationship sets such as a relationship set connecting to furniture or telephones or network drops (located in the office)

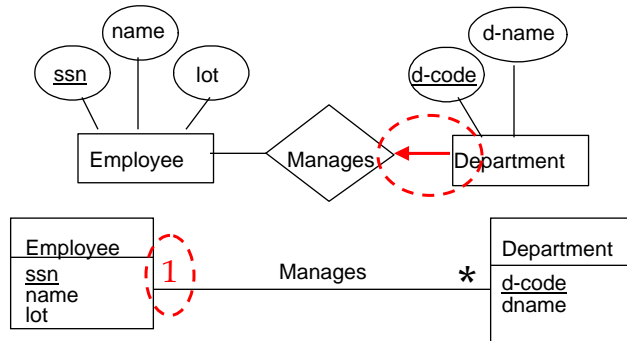
Entity vs. Value of an Attribute



Key Constraints - as described in the text

(limiting participation in relationship set to at most 1 entity)

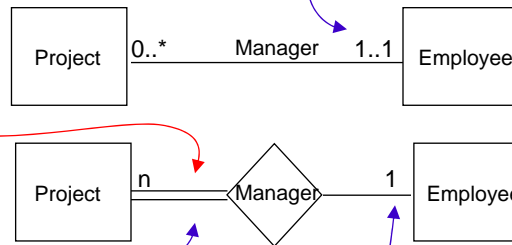
same as maximum multiplicity of 1 in UML



Each dept has at most one manager, according to the *key constraint* on Manages.

Participation Constraint - as in text: when every entity **MUST** participate in a relationship set

a Project has exactly one manager



**a Project MUST have a manager
and there is at most 1 employee who is manager**