INFO2120 – INFO2820 – COMP5138 Database Systems

Week 2: Conceptual Database Design

(Kifer/Bernstein/Lewis – Chapter 4; Ramakrishnan/Gehrke – Chapter 2)

Dr. Uwe Röhm
School of Information Technologies

Outline

- Conceptual Database Design using the
  - Entity Relationship Model
  - Database Design with UML
- Case Study

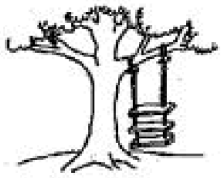
Based on slides from Kifer/Bernstein/Lewis (2006) “Database Systems” and from Ramakrishnan/Gehrke (2003) “Database Management Systems”, and including material from Ullman, Fekete and Röhm.

How to make an idea a reality?

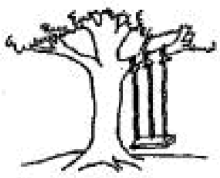
Source: the-self-build-guide.co.uk

Source: www.bg-propeties.info

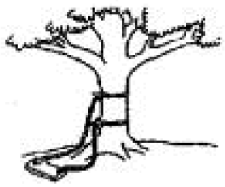
There are some pitfalls though…



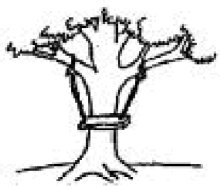
As proposed by the project sponsor.



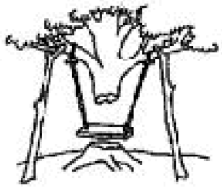
As specified in the project request.



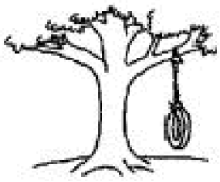
As designed by the senior architect.



As produced by the engineers.



As installed at the user's site.



What the customer really wanted.

Database Design Sequence

Topic of INFO2110

Requirements Analysis

- Understand…
  - what data is to be stored
  - what applications must be built
  - what operations are most frequent

Conceptual Design

- Develop…
  - high-level description of the data closely matching how users think of the data
  - Works as communication vehicle

Logical Design

- Convert…
  - conceptual design into a logical database schema

Physical Design

- Convert…
  - logical schema into a physical schema for a specific DBMS and tuned for app.

Conceptual Data Model

- Goal: Specification of database schema
- Methodology:
  - **Conceptual Design**: A technique for understanding and capturing business information requirements *graphically*
    - depicts the associations among different categories of data within a business or information system.
  - Convert conceptual database design to DDL
- Conceptual Database Design does *not* imply how data is implemented, created, modified, used, or deleted.
  - Works as communication vehicle
  - Facilitate planning, operation & maintenance of various data resources
- An conceptual data model is *model & database independent*

Conceptual Data Models

- Entity-Relationship Model (ERM)
- Object-oriented Data Models
  - Unified Modelling Language (UML)
  - OMT, Booch, …
- Etc.
  - Object Role Modelling (ORM)
  - Semantic Object Model (SOM)
  - Semantic Data Models (SDM)
  - KL-ONE etc.

This Week's Agenda

- Introduction
- **Entity Relationship Model**
- Unified Modelling Language

Entity Relationship Model

- Data model for conceptual database design:
  High-level *graphical representation* of what data needs to be contained in the system.
  - Used to interpret, specify, and document database systems.
  - Tools: ERwin, Sybase Power Designer, …
  - E-R diagram templates for drawing tools, e.g., Microsoft Visio
- First designed by Peter Chen in 1976
  - Several variations have since appeared
  - Here: **enhanced** or **extended E-R model**

Entity Relationship Model (cont'd)

- A data modeling approach that depicts the associations among different categories of data within a business or information system.
  - 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?
- A database `schema' in the ER Model is represented pictorially (***ER diagrams***).
  - We can convert an ER diagram into a relational schema.
- It is about what data needs to be stored
  - It does **not** imply how data is created, modified, used, or deleted.

Entities

- **Entity**: a person, place, object, event, or concept about which you want to gather and store data.
  - it must be distinguishable from other entities
  - Example: John Doe, unit COMP5138, account 4711
- **Entity Type** (also: **entity set**): a collection of entities that share common properties or characteristics
  - Example: students, courses, accounts
  - Rectangle represent entity type
  - Note: entity sets need not to be disjoint (e.g. person who is manager)
- **Attribute**: describes one aspect of an entity type
  - Example: people have *names* and *addresses*
  - depicted by an ellipses

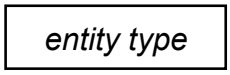
Entity Type

- An Entity Type is described by a set of attributes
  - Descriptive properties possessed by all members of an entity type
  - Example: Person has ID, Name, Address, Hobbies
- ***Domain***: possible values of an attribute
  - In contrast to relational model values can be complex / set-oriented!
    - **Simple** and **composite** attributes.
    - **Single-valued** and **multi-valued** attributes
  - Example see next slide
- ***Key***: minimal set of attributes that uniquely identifies an entity in the set (several such candidate keys possible)
  - One chosen as **Primary Key** (PK) => depicted by underlining attr.
- ***Entity Schema***: entity type name, attributes (+domains), PK

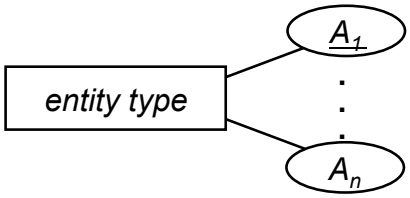
Graphical Representation in E-R Diagram

Symbols:

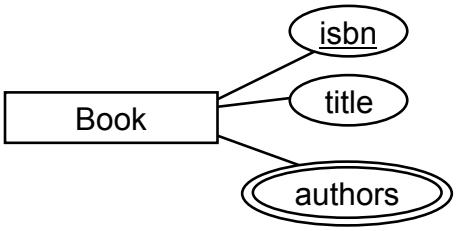
- Entity Types represented by a rectangle

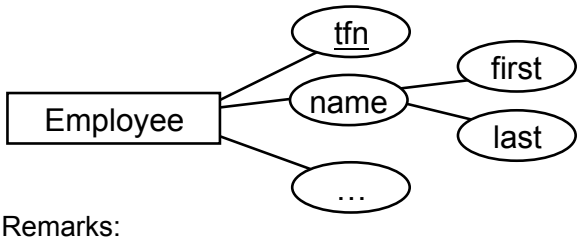


- Attributes depicted by ellipses
  - Double ellipses for multi-valued attributes
  - Keys are underlined



Examples:





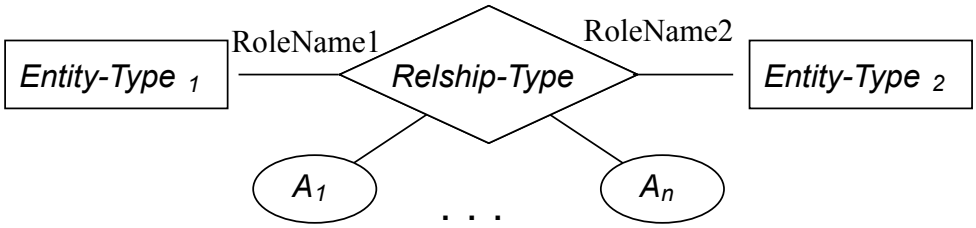
Remarks:
Book.authors is a *multi-valued attribute*;
Employee.name is a *composite attribute*.

Relationships

- **Relationship**: relates two or more entities
  - number of entities is also known as the **degree** of the relationship
  - Example: John *is enrolled in* INFO2120
- **Relationship Type** (**R.ship Set**): set of similar relationships
  - Formally: a relation among $n \geq 2$ entities, each from entity sets:
    $\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$
  - Example: **`Student`** (entity type) related to **`UnitOfStudy`** (entity type) by **`EnrolledIn`** (relationship type).
- Distinction:
  - relation (relational model) - set of tuples
  - relationship (E-R Model) – describes relationship between entities
  - Both entity sets and relationship sets (E-R model) may be represented as relations (in the relational model)

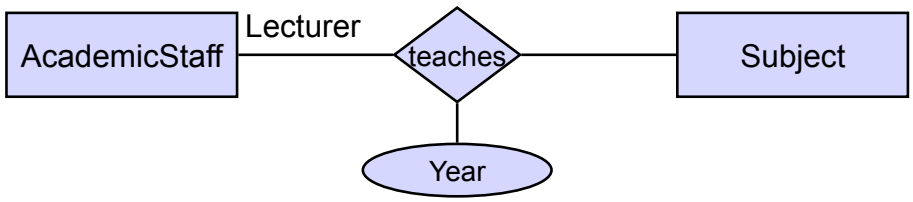
Graphical Representation of Relationships in E-R Diagrams

Symbol:



- Diamonds represent relationship types
- Lines link attributes to entity types and entity types to relationship types.
- Roles are edges labeled with role names

Example



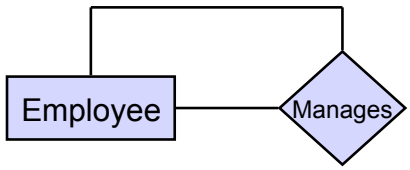
Relationship Attributes & Roles

- **Relationship-Attribute**
  Relationships can also have additional properties
  - E.g., John enrolles in INFO2120 *in* the first semester 2010
  - John and INFO2120 are related
  - 2010sem1 describes the relationship - value of the *Semester* attribute of the **`EnrolledIn`** relationship set

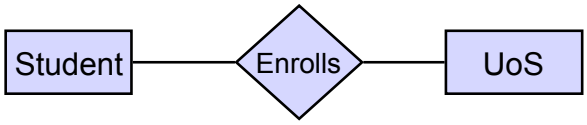
- **Relationship-Role**
  Each participating entity can be named with an explicit role.
  - E.g. John is value of *Student* role, INFO2120 value of *Subject* role
  - useful for relationship that relate elements of same entity type
  - Example: **`Supervises( Employee:Manager, Employee )`**

Relationship Degree

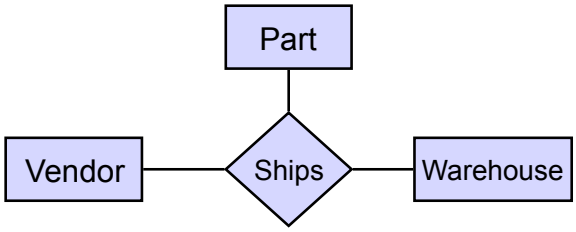
- Degree of a Relationship:
  # of entity types involved
  - Unary Relationship (Recursive)



  - Binary Relationship (80%)



  - Ternary Relationship

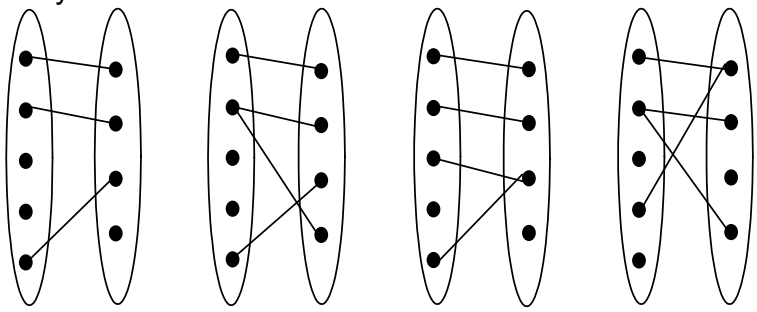


Schema of Relationship Types

- The combination of the primary keys of the participating entity types forms a super key of a relationship.
  - Example:
    (student_Id, UoS_number) is the super key of *Enrolls*
  - One must consider the mapping cardinality of the relationship when deciding what are the candidate keys
- Relationship Set Schema:
  - Relationship name
  - Role names (or: names of participating entity sets)
  - Relationship attributes and their types
  - key

Multiplicity of Relationships

- Consider **`Works_In`**: An employee can work in many departments; a department can have many employees.
- In contrast, each department has at most one manager
- We find examples of each style of relationship
  - Think carefully about both directions:
  - To how many instances of B can a given instance of A be related?
  - To how many instances of A can a given instance of B be related?
- Warning: natural language can be confusing
  - Many-to-1 means each A is related to *at most* 1 of B

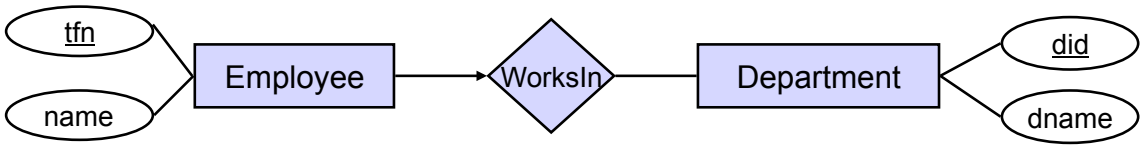


1-to-1 | 1-to Many | Many-to-1 | Many-to-Many

Multiplicities are depicted in E-R diagrams as constraints…
(see next slides)

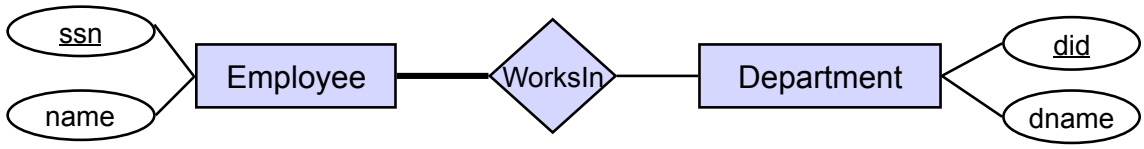
Key Constraints

- If, for a particular participant entity type, each entity participates in *at most one* relationship, the corresponding role is a key of relationship type
  - E.g., *Employee* role is unique in *WorksIn*
  - but there may be employees who are working in no department at all
  - also called: **one-to-many** or **1:N relationship**
- Representation in E-R diagram: arrow
- Example: An employee works in at most one department.



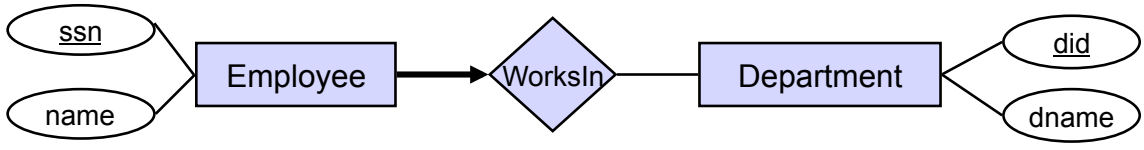
Participation Constraint

- If every entity participates in *at least one* relationship, a *participation constraint* holds:
  - A participation constraint of entity type *E* having role ρ in relationship type *R* states that for *e* in *E* there is an *r* in *R* such that $\rho(r) = e$.
  - also called a **total participation** of *E* in *R*
    - A participation that is not total is said to be **partial**
- Representation in E-R diagram: thick line
- Example: every employee works in at least one department



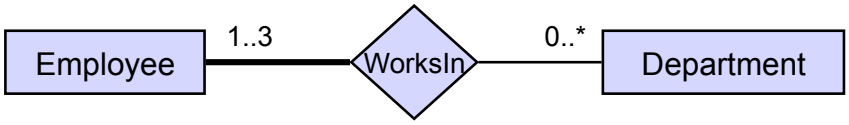
Participation and Key Constraint

- If every entity participates in *exactly one* relationship, both a participation and a key constraint hold.
  - Again: 1:N relationship
- Representation in E-R diagrams: thick arrow
- Example:
  Every employee works in exactly one department



Cardinality Constraints

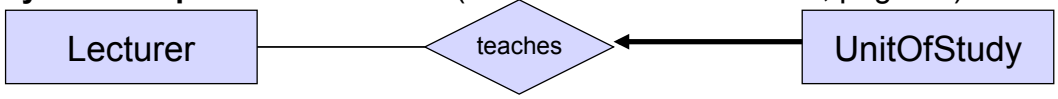
- Generalisation of key and participation constraints
- A **cardinality constraint** for the participation of an entity set E in a relationship R specifies how often an entity of set E participates in R at least (minimum cardinality) and at most (maximum cardinality).
  - In an ER-diagram we annotate the edge between an entity type E and relationship R with min..max, where min is the minimum cardinality and max the maximum cardinality. If no maximal cardinality is specified, we set '*' as max number ("don´t care").
- Example: Every employee works in 1 to 3 departments.



Comparison of Notations

"A lecturer can teach several subjects." "Every UoS is taught by exactly one lecturer."

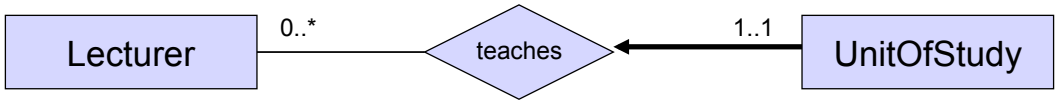
With Key & Participation Constraint: (cf. Ramakrishnan/Gehrke, page 33)



- partial participation
- multiple side

- total participation
- key constraint on department side

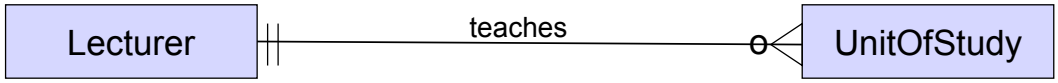
With Cardinality Constraints: (cf. Kifer/Bernstein/Lewis, pages 76/77 and 82/83)



- partial participation
- multiple side

- total participation
- key constraint on department side

"Crow's-foot" notation: (e.g. Hoffer, ed7, pages 95,116-122 – as used in INFO1003)



- mandatory one

- optional many

Be aware that the Crow's foot notation is inverse to our notation!

Weak Entities

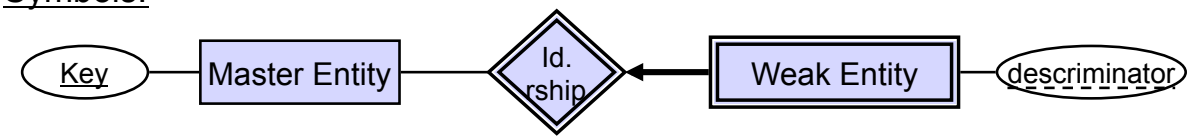
- ***Weak entity type***: An entity type that does not have a primary key.
  - Can be seen as an **exclusive 'part-of' relationship**
  - Its existence depends on the existence of one or more *identifying entity types*
  - it must relate to the identifying entity set via a total, one-to-many *identifying relationship type* from the identifying to the weak entity set
  - Examples:
    child from parents, *payment* of a loan

- The ***discriminator*** (or ***partial key***) of a weak entity type is the set of attributes that distinguishes among all the entities of a weak entity type related to the same owning entity.

- The primary key of a weak entity type is formed by the primary key of the strong entity type(s) on which the weak entity type is existence dependent, plus the weak entity type's discriminator.

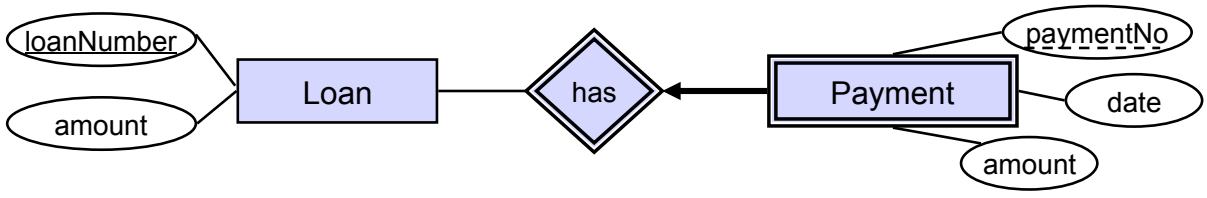
Representation of Weak Entity Types

Symbols:



- We depict a weak entity type by double rectangles.
- Identifying relationship depicted using a double diamond
- underline the discriminator of a weak entity type with a dashed line

Example:



- paymentNumber: discriminator of the payment entity type
- Primary key for payment: (loanNumber, paymentNumber)

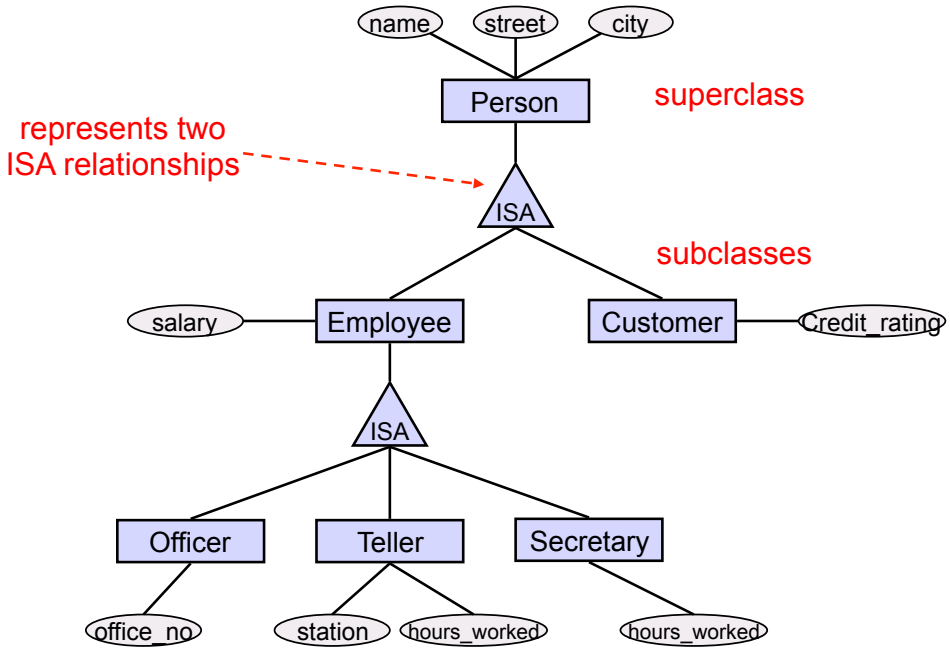
Enhanced E-R Model

- ER model in its original form did not support
  - SPECIALIZATION/ GENERALIZATION
  - ABSTRACTIONS ('aggregation')
- This let to development of 'Enhanced' ER model
  - Includes all modeling concepts of basic ER
  - Additional some object-oriented concepts: subclasses/superclasses, specialization/generalization, categories, attribute inheritance
  - The resulting model is sometimes called the enhanced-ER or Extended ER (E2R or EER) model
    - used to model applications more completely and accurately if needed
- If we will talk about E-R model, we always mean EER model

Generalisation / Specialisation

- Arranging of entity types in a type hierarchy.
  - Determine entity types whose set of properties are actual a subset of another entity type.
- Definition **Generalisation** / **Specialisation** / **Inheritance**:
  Two entity types *E* and *F* are in an ISA-relationship ("*F* is a *E*"), if
  (1) the set of attributes of *F* is a superset of the set of attributes of *E*, and
  (2) the entity set *F* is a subset of the entity set of *E* ("each *f* is an *e*")
- One says that *F* is a *specialisation* of *E* (*F* is **subclass**) and *E* is a *generalisation* of *F* (*E* is **superclass**).
  - Example: `Student` is a subclass of `Person`
- **Attribute inheritance** – a lower-level entity type inherits all the attributes and relationship participations of its supertype.
- Depicted by a triangle component labeled IsA

Superclass/Subclass Example

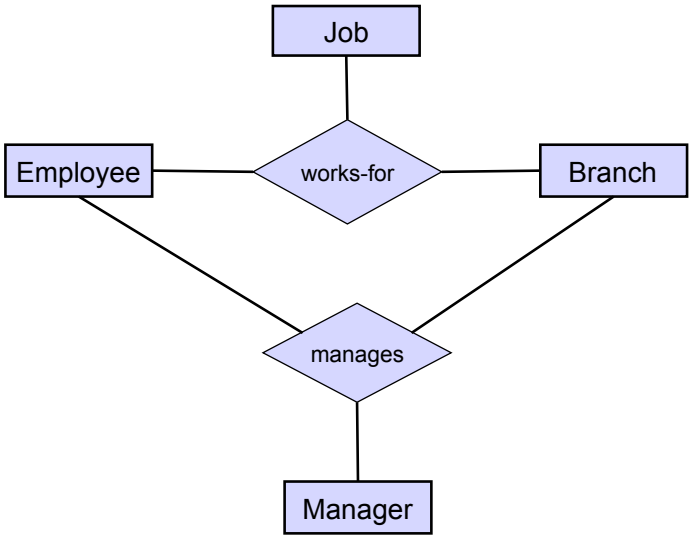


Constraints on ISA Hierarchies

- We can specify *overlap* and *covering* constraints for ISA hierarchies:
- Overlap Constraints
  - **Disjoint**
    - an entity can belong to only one lower-level entity set
    - Noted in E-R diagram by writing *disjoint* next to the ISA triangle
  - **Overlapping** (the default - *opposite to Ramakrishnan/Gehrke book*)
    - an entity can belong to more than one lower-level entity set
- Covering Constraints
  - **Total**
    - an entity must belong to one of the lower-level entity sets
    - Denoted with a thick line between the ISA-triangle and the superclass
  - **Partial** (the default)
    - an entity need not belong to one of the lower-level entity sets

Aggregation

- Consider a ternary relationship *works-on*
- Suppose we want to record managers for *tasks* performed by an employee at a branch.
- Like this?



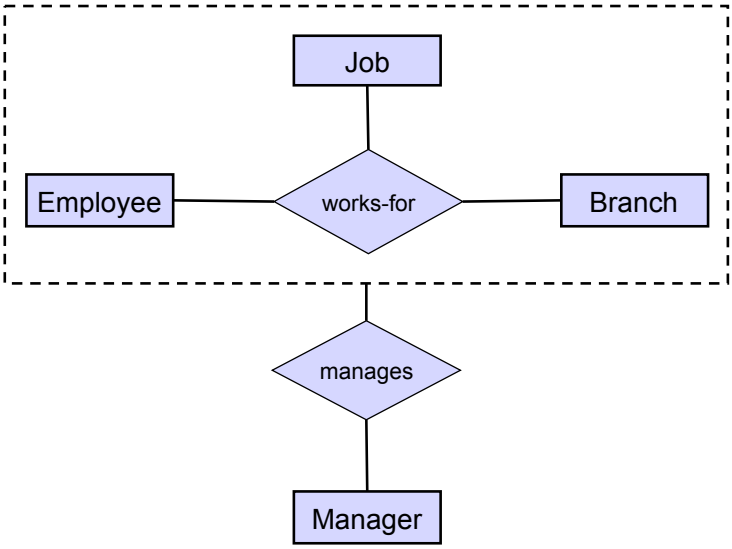
Source:
Silberschatz/Korth/Sudarshan:
Database System Concepts, 2002.

Aggregation (Cont.)

- Relationship sets *works-on* and *manages* represent overlapping information
  - Every *manages* relationship corresponds to a *works-on* relationship
  - However, some *works-on* relationships may not correspond to any *manages* relationships
    - So we can't discard the *works-on* relationship
- Eliminate this redundancy via ***aggregation***
  - Aggregation allows an relationship set to be treated as an (abstract) entity set for the purpose of participating in another relationships
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

E-R Diagram With Aggregation

- Without introducing redundancy, the following diagram represents:
  - An employee works on a particular job at a particular branch
  - An employee, branch, job combination may have an associated manager



Source:
Silberschatz/Korth/Sudarshan:
Database System Concepts, 2002.

Case Study

- Assignment 1: A Treasure Hunt Game

This Week's Agenda

- Introduction
- Entity Relationship Model
- **Unified Modeling Language**

Based on slides from Ullman (Database Systems - A first Course) and Silberschatz/Korth/Sudarshan: *Database System Concepts*, 2002.

UML

- UML: Unified Modeling Language
  - UML is designed to model object-oriented software, but has been adapted as a database modeling language.
  - --- a more "object-oriented" version of E/R
- UML has many components to graphically model different aspects of an entire software system
  - Use Case
  - Activity
  - Statechart
  - Class
- UML Class Diagrams correspond to E-R Diagram, but there are several differences.

UML Class Diagrams

- Entity sets are shown as boxes, and attributes are shown within the box, rather than as separate ellipses in E-R diagrams.
- Binary relationship sets are represented in UML by just drawing a line connecting the entity sets. The relationship set name is written adjacent to the line.
- The role played by an entity set in a relationship set may also be specified by writing the role name on the line, adjacent to the entity set.
- The relationship set name may alternatively be written in a box, along with attributes of the relationship set, and the box is connected, using a dotted line, to the line depicting the relationship set.
- Non-binary relationships drawn using diamonds, just as in ER diagrams

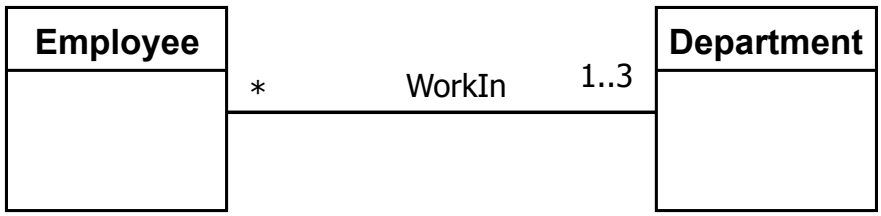
UML Classes

- Sets of objects, with attributes (*state*) and methods (*behavior*).
- Attributes have types.
- **PK** indicates an attribute in the primary key (optional) of the object.
- Methods have declarations = arguments (if any) and return type.
  - correspond for us to derived attributes in the E/R model

Student
PK sid : integer name : string birthdate : date speaks : set(string)
languages() : integer

Relationship Type = UML Association

- Binary relationships between classes.
- Represented by named lines (no diamonds as in E/R).
- Multiplicity at each end.
  - *m* ..*n* means between *m* and *n* of these associate with one on the other end.
  - * = “infinity”; e.g. 1..* means “at least one.”

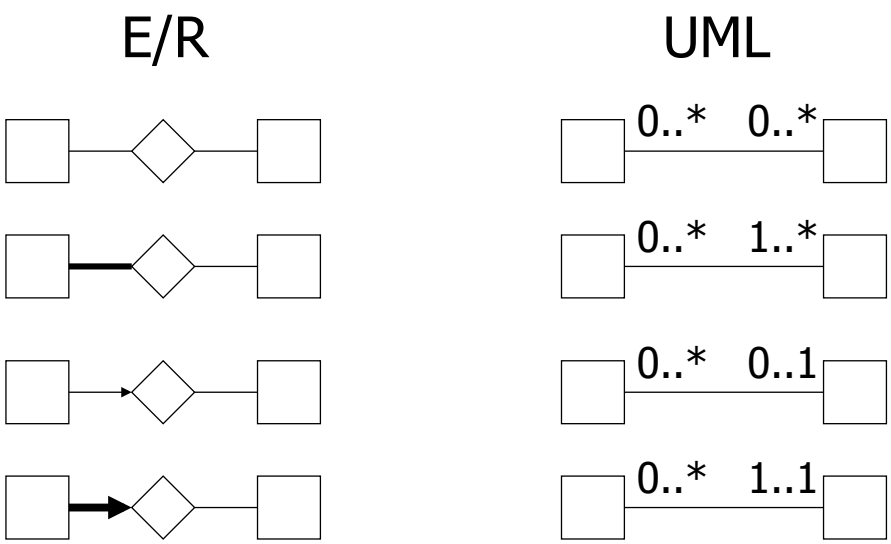


UML Class Diagrams (cont’d)

- Cardinality constraints are specified in the form *l..h*, where *l* denotes the minimum and *h* the maximum number of relationships an entity can participate in.
  - the positioning of the constraints is OPPOSITE to the positioning of cardinality constraints in our E-R notation
    - it is the same as the positioning of constraints in crow’s foot notation.
  - The constraint *0..** on the E_2 side and *0..1* on the E_1 side means that each E_2 entity can participate in at most one relationship, whereas each E_1 entity can participate in many relationships; in other words, the relationship is many to one from E_2 to E_1.
  - Single values, such as 1 or * may be written on edges; The single value 1 on an edge is treated as equivalent to 1..1, while * is equivalent to 0..*.

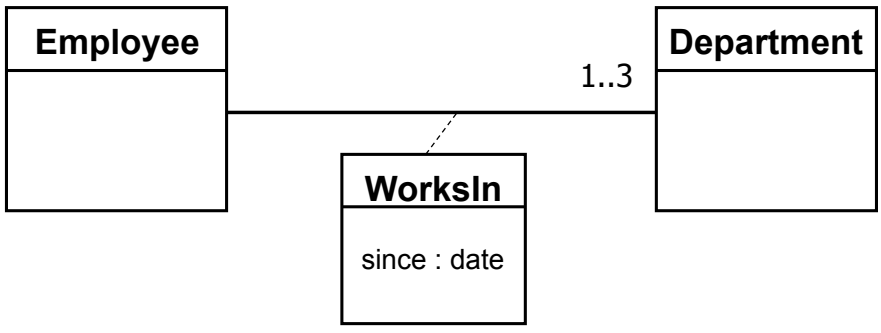
Source: Silberschatz/Korth/Sudarshan: *Database System Concepts*, 2002.

Comparison with E/R Multiplicities



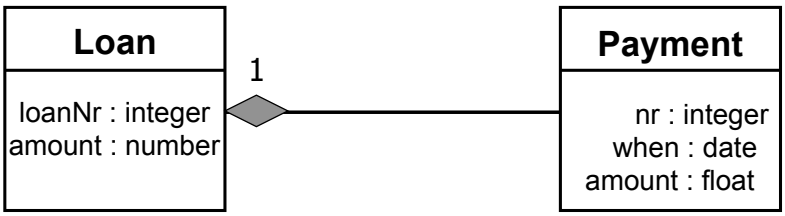
Association Class

- Attributes on associations are permitted.
  - Called an *association class*.
  - Analogous to attributes on relationships in E/R.



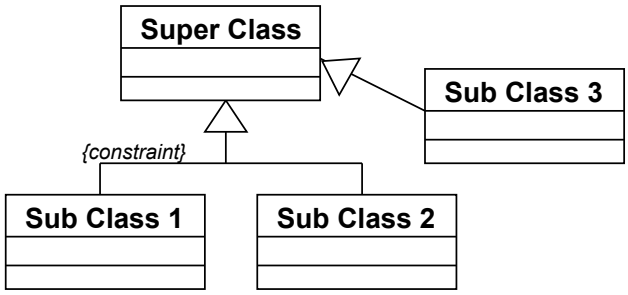
UML Composition

- A special form of an UML association
- An association with the implication that every object is definitely *owned* by one object on the other side.
  - existance dependency(!)
- Represented by solid diamond at owner.
- Corresponds to the concept of *Weak Entities* in E/R.

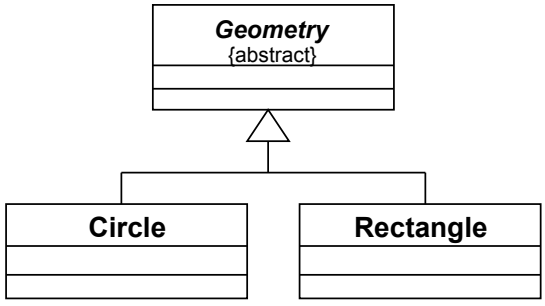


UML Inheritance Hierarchies

- General ISA Relationship
  - optional constraint:
    - **complete**
      $\cup$ ext(SubClass$_i$) = ext(SuperC)
    - **disjoint**
      $\cap$ ext(SubClass$_i$) = {}
    - **incomplete**
    - **overlapping**

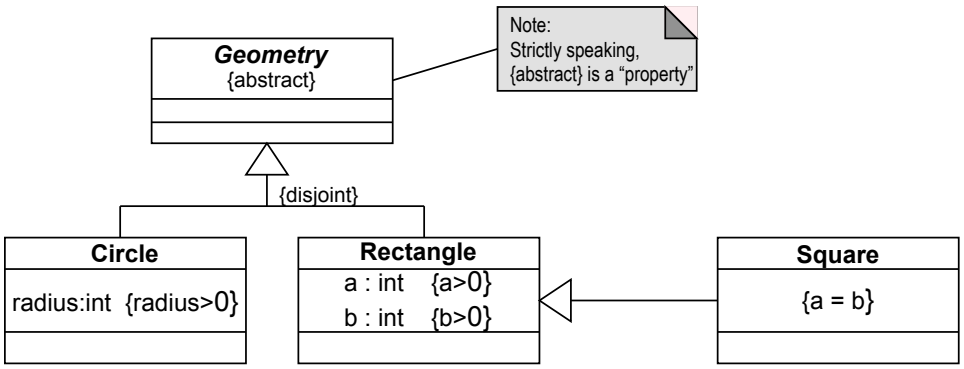


- Abstract classes:
  - Classes which cannot have any instances.
  - in particular will have no extend

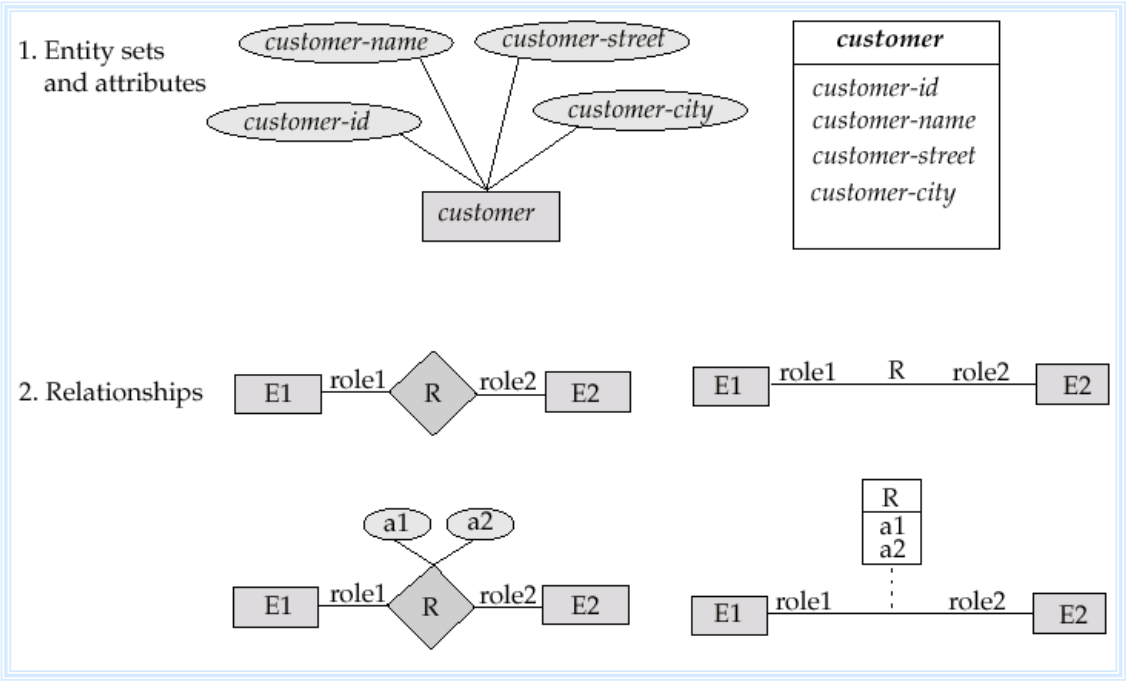


Constraints in UML

- Predicates in curly braces can be added to *every* UML element
- Some pre-defined constraints (cf. previous slides)
- Semantic can be free-form, or using the UML-own OCR ("object constraint language")

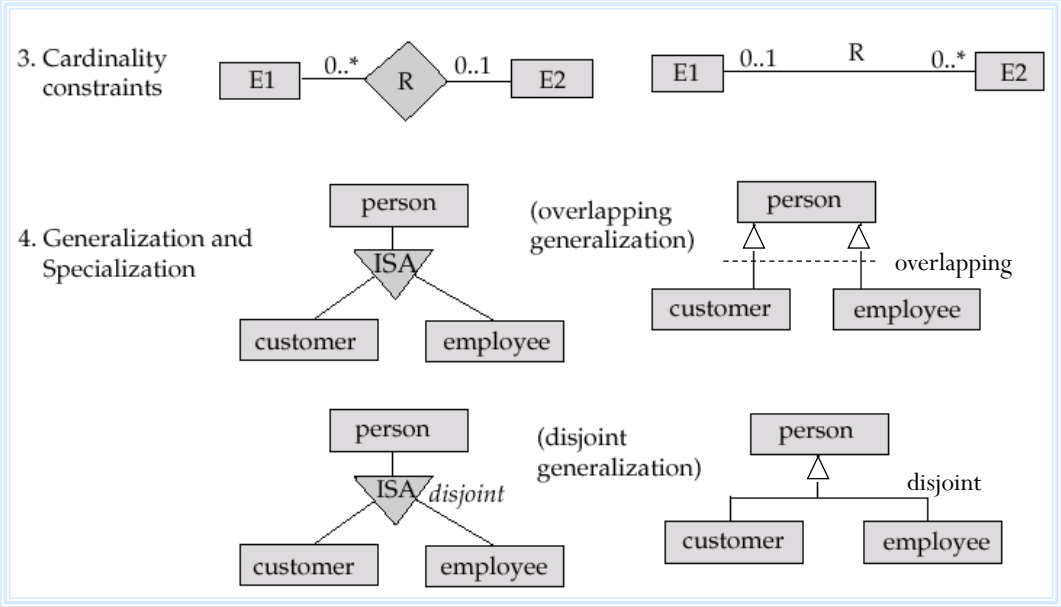


UML Class Diagram Notation



Source: Silberschatz/Korth/Sudarshan: *Database System Concepts*, 2002.

UML Class Diagrams (cont'd)



[Source: Korth/Silberschatz/Sudarshan]

*Note reversal of position in cardinality constraint depiction
*Generalization can use merged or separate arrows independent of disjoint/overlapping

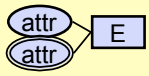
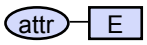
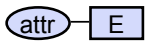
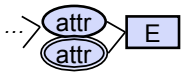
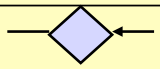
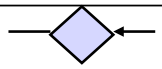
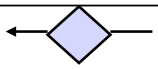
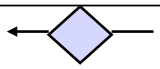
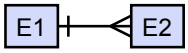
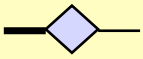
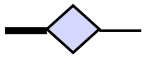
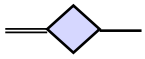
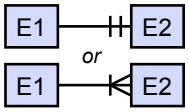
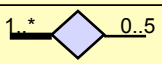
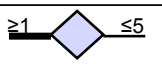
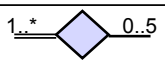
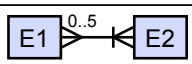
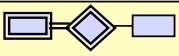
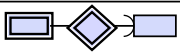
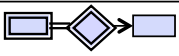
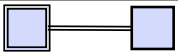
This week, you have learned…

- **The Database Design Process**
  - An understanding of the general database design process and the roles of conceptual and logical data modelling
- **Conceptual Data Modelling using the E-R Model**
  - Understanding and experience with conceptual data modelling using the entity-relationship model:
    - Basic Constructs: Entity, Attributes, Relationships, Cardinality Constraints
    - Advanced Concepts: Weak Entities, Inheritance
- **Introduction to UML**
  - Basic understanding on how to use UML class diagrams for data modelling

References

- Kifer/Bernstein/Lewis (2nd edition)
  - Chapter 4
  - *Lecture uses this ER-D notation and naming scheme; good section on UML*
- Ramakrishnan/Gehrke (3rd edition - the 'Cow' book)
  - Chapter 2
  - *similar ER-D notation with few differences (cf. appendix); incl. case study Internet Shop; unfortunately not much on UML (about 2 pages)*
- Ullman/Widom (3rd edition)
  - Chapter 4
  - *only basic ER-D with a few differences in its notation; small section on UML*
- Silberschatz/Korth/Sudarshan (5th edition - 'sailing boat')
  - Chapter 6
  - *standard ER-D notation with most details; just 2 pages UML overview*
- Elmasri/Navathe (5th edition)
  - Chapters 3 and 4
  - *uses a different notation for Enhanced E-R diagrams, especially inheritance*

Appendix: Notation Comparison

	Kifer / Bernstein / Lewis	Ramakrishnan / Gehrke	Ullman / Widom	Korth/ Silberschatz / Sudarshan	Hoffer / Prescott "Crows-Foot"
Entity Name	Entity Type	Entity Set *(plural names)*	Entity Set *(plural names)*	Entity Set	Entity Type
Attributes	 *only atomic; single-|set-valued*	 *only atomic & single valued*	 *only single valued (but mention variants with structs & sets)*	 *single-|set-valued composite attr. derived attributes*	 *single-|set-valued composite attr. derived attributes*
Key Constraints (1-many relationship)	 *(arrow from N-side to diamond)*	 *(arrow from N-side to diamond)*	 *(arrow from diamond to 1-side)*	 *(arrow from diamond to 1-side)*	 *(no diamond;tick on 1-side, crow's foot on many side)*
Participation Constraints	 *(thick line on total participation side)*	 *(thick line on total participation side)*	*n/a*	 *(double line - total participation side)*	
Cardinality Constraints	 *min..max notation*	*n/a*	 *limit constraint*	 *min..max notation*	 *on opposite side!*
Roles	yes	yes	yes	yes	yes
Weak Entity (& identifying rel.ship)					
ISA					

Administrativa

- Tutorials start this week
- Also: Homework starting this week
  - Deadline for homework: Monday next week, midnight
  - Electronic submission in
    - eLearning (menu item ‘Tutorials’, then ‘Homework Submission Week 2’)
- Next week: SQL Challenge starts
  and further introduction to Practical Assignment

Next Week

- Levels of Abstraction, Data Model, Schema
- The Relational Data Model
  - Relations
  - SQL Data Definitions
  - Keys
  - Views
- Readings:
  - Kifer/Bernstein/Lewis book, Chapter 3
  - Ramakrishnan/Gehrke book, Chapter 3