

UIC

CS480
Database Systems
5 - The ER Model & Database Design
Course webpage
Boris Glavic
bglavic@uic.edu

1

UIC

Database Design & The ER-Model

The Database Design

The ER-Model

Recap

2

UIC

The Database Design

The Database Design
The Design Process

The ER-Model

Recap

3

UIC **Textbook**

Database System Concepts

Textbook: Chapter 7

4

UIC **The Database Design**

The Database Design
The Design Process

UIC **Database Design**

```

graph TD
    World((World)) -- "???" --> Relational[Relational DB Schema]
  
```

5

UIC **Requirement Analysis**

```

graph TD
    World((World)) -- "Requirement Analysis" --> Mental[Mental Model]
    Mental -- "???" --> Relational[Relational DB Schema]
  
```

UIC **Example Requirement Analysis - Zoo**

- The zoo stores information about animals, cages, and zoo keepers.
- Animals are of a certain species and have a name. For each animal we want to record its weight and age.
- Each cage is located in a section of the zoo. Cages can house animals, but there may be cages that are currently empty. Cages have a size in square meter.
- Zoo keepers are identified by their social security number. We store a first name, last name, and for each zoo keeper. Zoo keepers are assigned to cages they have to take care of (clean, ...). Each cage that is not empty has a zoo keeper assigned to it. A zoo keeper can take care of several cages. Each zoo keeper takes care of at least one cage.

6

UIC **Example Requirement Analysis - Music Collection**

- Let's do it!

UIC **Conceptual Modeling**

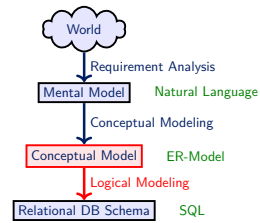
```

graph TD
    World((World)) -- "Requirement Analysis" --> Mental[Mental Model]
    Mental -- "Conceptual Modeling" --> Conceptual[Conceptual Model]
    Conceptual -- "???" --> Relational[Relational DB Schema]
  
```

7



Logical Modeling



8



The ER-Model

The Database Design

The ER-Model

Introduction
Entities & Attributes
Relationships
ER Diagrams
Translating ER-models into Relational Schemas
Design Considerations
ER-model Inheritance



The ER-Model

The ER-Model

Introduction
Entities & Attributes
Relationships
ER Diagrams
Translating ER-models into Relational Schemas
Design Considerations
ER-model Inheritance
Alternative Diagram Notation

9



The ER-model

- In the **Entity-Relationship Model**, **databases** are modeled as
 - One or more sets of entities of the same type
 - Relationships between the entities
- An **entity** is an object that exists and is distinguishable from other objects
- Entities have **attributes**
 - e.g., people have *names* and *addresses*
- An **entity set** is a set of entities of the same type that share the same attributes
 - e.g., *sets of people, set of classes, set of students, ...*

9



The ER-Model

The ER-Model

Introduction
Entities & Attributes
Relationships
ER Diagrams
Translating ER-models into Relational Schemas
Design Considerations
ER-model Inheritance
Alternative Diagram Notation

10



Instructor Entity Set

- entities in entity set **instructor** have attributes **instid** and **instname**

instid	instname
22222	Einstein
76354	Singh
98354	Kim
10101	Srinivasan
45565	Katz
76766	Crick

instructor

10



Attributes

- An entity is represented by a set of **attributes**, that are descriptive properties possessed by all members of an entity set

- instructor** = (ID, name, street, city, salary)
- course** = (course-id, title, credits)

Domains

- Domain** - the set of permitted values for each attribute

11



Attribute types

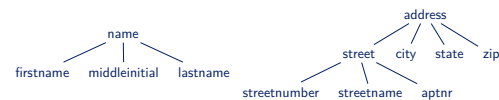
- Simple** and **composite** attributes
- Single-valued** and **multivalued** attributes
 - Example: multivalued attribute: *phone-numbers*
- Derived** attributes
 - Can be computed from other attributes
 - Example: *age*, given *dateofbirth*

12



Composite Attributes

- Composite attributes** consist of multiple **component** attributes
- Component attributes may be composite too



13



Keys

- Entities can have **keys**
- Just like in the relational model a key is a set of attributes whose values uniquely identify entities in an entity set

14



The ER-Model

Introduction
 Entities & Attributes
Relationships
 ER Diagrams
 Translating ER-models into Relational Schemas
 Design Considerations
 ER-model Inheritance
 Alternative Diagram Notation



- A **relationship** is an association between two or more entities

- relationship advisor between entities
 - 44553 (Peltier) (a student)
 - 22222 (Einstein) (an instructor)

15

15



Relationship Sets

- A **relationship set** is a relation between entities from $n \geq 2$ entities from a given selection of entity sets $\{E_i\}$, i.e., a subset of:

$$\{(e_1, \dots, e_n) \mid \forall i \in [1, n] : e_i \in E_i\}$$

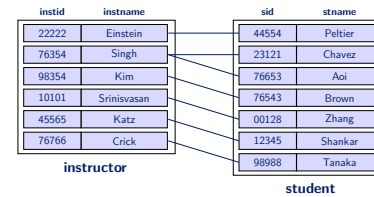
- n is called the **degree** of a relationship set (most relationships are binary: $n = 2$)

$(44553, 22222) \in \text{advisor}$

16



Example Advisor Relationship Set

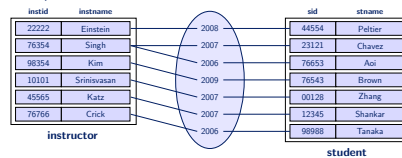


17



Attributes on Relationships

- Relationships can also have **attributes** to store additional information about the relationship

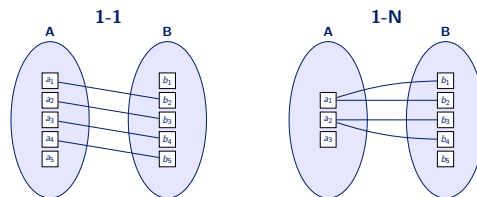


18

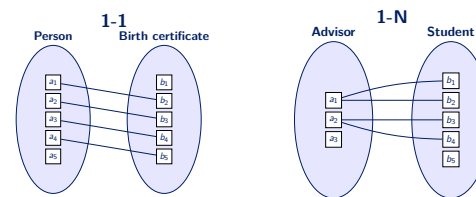
19



Cardinality Constraints



Cardinality Constraints Example

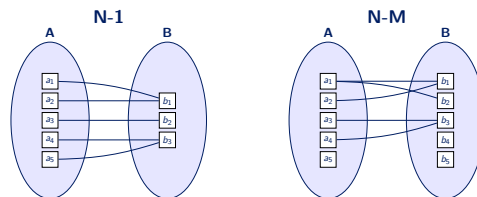


20

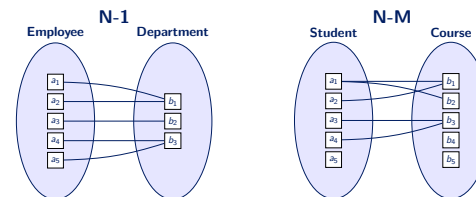
21



Cardinality Constraints



Cardinality Constraints Example



22

23



Partial Participation

- Allow specification of whether participation is **total** (1) or **partial** (0)
- For binary relationships we get 16 cases
- 1-1**
 - 0:1-0:1
 - 0:1-1:1
 - 1:1-0:1
 - 1:1-1:1
- 1-N**
 - 0:1-0:N
 - 0:1-1:N
 - 1:1-0:N
 - 1:1-1:N
- N-1**
 - 0:N-0:1
 - 0:N-1:1
 - 1:N-0:1
 - 1:N-1:1
- N-M**
 - 0:N-0:M
 - 0:N-1:M
 - 1:N-0:M
 - 1:N-1:M

24



Upper & Lower Bound Participation

- Alternative approach: specify **lower** and **upper bounds** for the number of relationships an entity can participate

Remark

- This is the exact opposite of the 1-N notation!

25



The ER-Model

The ER-Model

Introduction
Entities & Attributes
Relationships
ER Diagrams
Translating ER-models into Relational Schemas
Design Considerations
ER-model Inheritance
Alternative Diagram Notation

26



Purpose of ER-Diagrams

- An **ER-diagram** is a visual representation of an ER-model
- Useful to communicate concepts to an expert

Alternative Notations

- There are several common visual representations that have been used in ER-diagrams
- We will use one particular one here, but showcase others at the end so you can understand them when you see them in the wild!

26



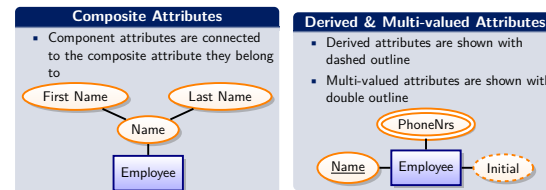
Visual Elements - Entities



27



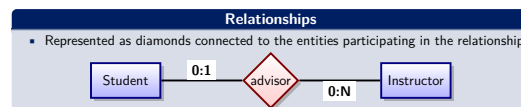
Visual Elements - Composite Attributes



28



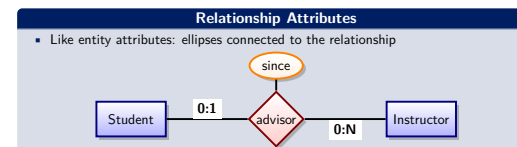
Visual Elements - Relationships



29



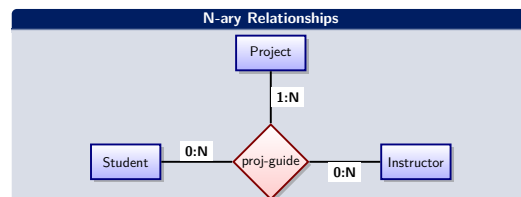
Visual Elements - Relationship Attributes



30



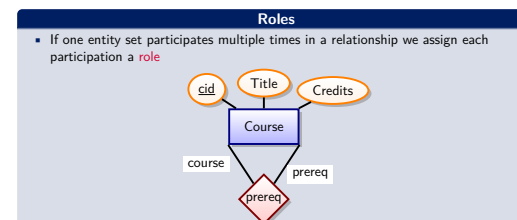
Visual Elements - N-ary Relationships



31



Visual Elements - Roles



32

Let's design an ER-model for parts of our university database example

1. Identify entities
2. Identify Relationships
3. Determine Attributes
4. Determine Cardinality Constraints
5. Iteratively Refine Model

33

34

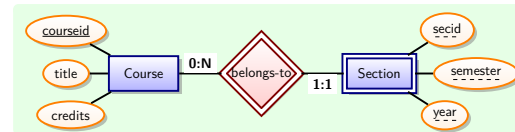
Weak Entities

- The types of entities we have seen so far are **strong entities**
- A **weak entity** is an entity that can only be identified through its relationship with another entity
- The **partial key** or **discriminator** is a set of attributes from the **weak entity** that together with the strong entity identifies the weak entity
- The **identifying relationship** connects the weak entity to the **identifying entity**
 - The relationship has to be 1:N and **total** to ensure that each weak entity from the set is associated with exactly one entity from the identifying entity set

Weak Entity Example

Visual Notation

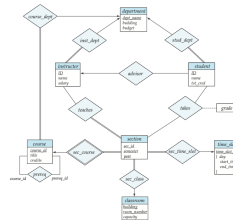
- The weak entity and identifying relationships are shown with double outlines
- Discriminator attributes are shown with dashed underline



35

36

Full University ER-model



The ER-Model

The ER-Model

Introduction

Entities & Attributes

Relationships

ER Diagrams

Translating ER-models into Relational Schemas

Design Considerations

ER-model Inheritance

Alternative Diagram Notation

37

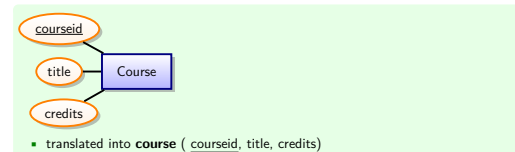
38

Overview

- The translation needs to **preserve the semantics** of entities and relationships
- Cardinality constraints** should be enforced in the relational schema where possible
- Entity and relationship sets are translated into relations
 - sometimes it is not necessary to create a separate relation for a relationship

Strong Entities

- Every **strong entity** (entities that are not weak) is translated into a new relation
- Simple attributes** are represented as attributes of this relation
- Derived attributes** are ignored
- The **key** of the entity is translated as the **primary key** of the relation

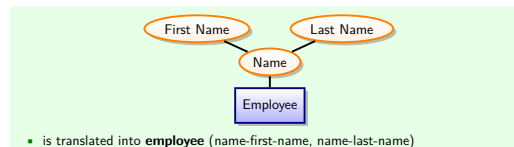


38

39

Composite Attributes

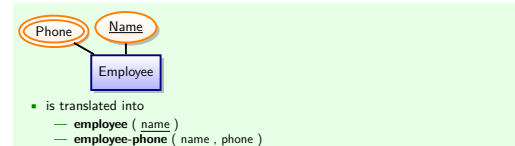
- Composite attributes** are translated by **flattening** the hierarchies



- is translated into **employee** (name-first-name, name-last-name)

Multivalued Attributes

- Each **multi-valued attribute** is translated into a separate relation
- The attributes of this relation is the key attributes of the entity and the multi-valued attribute
- The primary key of this relation are all attributes



40

41

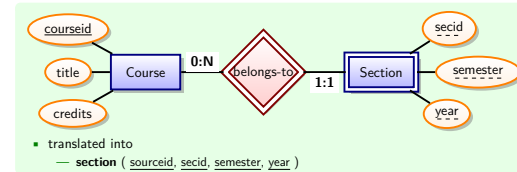


Weak Entities

- Every **weak** is translated into a new relation
- Similar to strong entities attributes of the entity are added to this relation
- Additionally, the key of the identifying strong entity is added as a foreign key to the relation for the weak entity



Weak Entities - Example



42

43

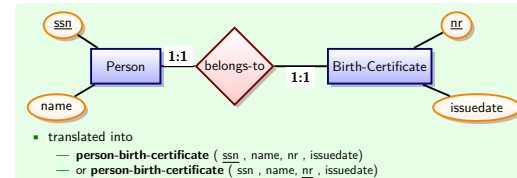


Relationships - 1-1 - total

- As each entity from either side is connected to **exactly one** entity from the other side, we can merge the two relations for the two entities
 - all of the attributes from both entities
 - the attributes of the relationship
- The key of the merged relation is the one of the keys of the two entities (we can select either one)



Relationships - 1-1 - total - Example



44

45

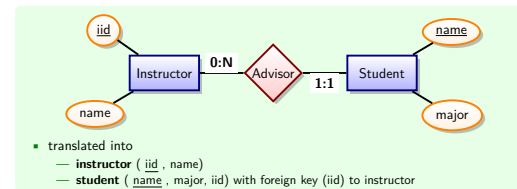


Relationships - 1-1 partial, 1-N and N-1

- For a **1-N** relationship (and symmetrically **N-1**):
 - each entity from the **1** side may be connected to many entities from the **N**
 - each entity from the **N** side is only connect to at most one entity from the **1** side
- We can add the key of the **1** side to the relations for the **N** as a foreign key to record which entity from the **1** each entity is connected to
- If a **1-1** relationship is partial, then creating a single relation may lead to a lot of null values
 - Deal with it the same way as 1-N



Relationships - 1-1 partial, 1-N and N-1 - Example



46

47

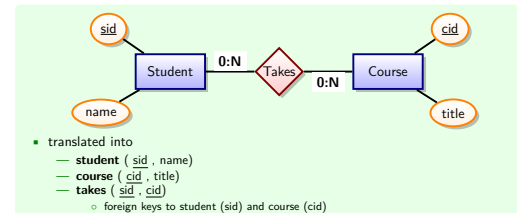


Relationships - N-M and n-ary

- An **N-M** or **n-ary** relationship (for $n \geq 3$) for both sides entities may participate more than once in the relationship
 - Adding a foreign key would require repeating tuples from the referencing table (**redundancy**)
- Such relationships are translated into **separate relations**
- Consisting of
 - the key attributes of all participating entities
 - all attributes of the relationship
- The key of the new relation is the combination of all primary keys
- The relation has foreign keys to the relations for all participating entities



Relationships - N-M - Example



48

49



ER-model to Relational Translation Summary

Rule 1: Strong entities

- Strong entity E**
 - Create relation with all simple and flattened composite attributes of E
 - The primary key is the key of E

Rule 2: Weak Entities

- Weak entity W** identified by entity E through relationship R
 - Create relation with attributes of W and R and PK(E)
 - The PK is PK(E) and the discriminator attributes of R
 - PK(E) is a foreign key to E



ER-model to Relational Translation Summary

Rule 3: Binary Relationships 1-1

- Binary 1-1 Relationship R** between entities A and B
 - If **no side is total** do one of:
 - Add PK(A) to B as a foreign key. Add attributes of R to A
 - Add PK(B) to A as a foreign key. Add attributes of R to B
 - If **one side is total** (say A):
 - Add PK(A) to B as a foreign key. Add attributes of R to A
 - If **both sides are total**:
 - Merge the relations for A and B
 - Choose either PK(A) or PK(B) as the primary key for the merged relation
 - Add attributes of R to the merged relation

50

51



Rule 4: Binary Relationships 1-N / N-1

Binary 1-1 Relationship R between entities A and B where A is the 1 side

- Add primary key $PK(A)$ of the 1 side to the N side as a foreign key
- Add attributes of R to the N side (B)

52



Rule 5: n-ary and N-M Relationships

- n-ary relationship or N-M binary relationship R between entities $E_1, \dots, E_k$
- Create a new relation for R
- Attributes of this relation are:
 - $PK(E_i)$ for each E_i
 - Attributes of R
- The primary key of this relation is $PK(E_1), \dots, PK(E_k)$
- Each $PK(E_i)$ in this relation is a foreign key to E_i

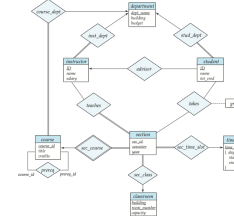
53



Rule 6: Multi-valued Attributes

- Multi-valued attribute A of entity E
 - Create a new relation with attributes A and $PK(E)$
 - The PK is all attributes of the new relation
 - $PK(E)$ is a foreign key to E

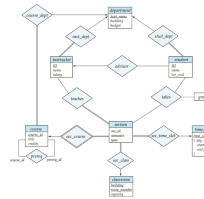
54



55



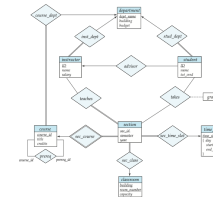
- Rule 1: strong entities
 - department (dept_name, building, budget)
 - instructor (ID, name, salary)
 - student (ID, name, tot_cred)
 - course (course_id, title, credits)
 - time_slot (time_slot_id)
 - classroom (building, room_number, capacity)
- Rule 2: weak entities
 - section (course_id, sec_id, semester, year)



56



- Rule 3: Relationships 1-1
 - None exists
- Rule 4: Relationships 1-N / N-1
 - department (dept_name, building, budget)
 - instructor (ID, name, salary, dept_name)
 - student (ID, name, tot_cred, dept_name)
 - course (course_id, title, credits, dept_name)
 - time_slot (time_slot_id)
 - classroom (building, room_number, capacity)
 - section (course_id, sec_id, semester, year, room_building)



57



- Rule 5: Relationships N-M and n-ary
 - department (dept_name, building, budget)
 - instructor (ID, name, salary, dept_name)
 - student (ID, name, tot_cred, dept_name, instr_ID)
 - course (course_id, title, credits, dept_name)
 - time_slot (time_slot_id)
 - classroom (building, room_number, capacity)
 - section (course_id, sec_id, semester, year, room_building, room_number, time_slot_id)
 - prereq (course_id, prereq_id)
 - teaches (ID, course_id, sec_id, semester, year)
 - takes (course_id, sec_id, semester, year, grade)



58



- Rule 6: Multi-valued attributes
 - department (dept_name, building, budget)
 - instructor (ID, name, salary, dept_name)
 - student (ID, name, tot_cred, dept_name, instr_ID)
 - course (course_id, title, credits, dept_name)
 - time_slot (time_slot_id)
 - time_slot_day (time_slot_id, start_time, end_time)
 - classroom (building, room_number, capacity)
 - section (course_id, sec_id, semester, year, room_building, room_number, time_slot_id)
 - prereq (course_id, prereq_id)
 - teaches (ID, course_id, sec_id, semester, year)
 - takes (course_id, sec_id, semester, year, grade)



59



The ER-Model

- Introduction
- Entities & Attributes
- Relationships
- ER Diagrams
- Translating ER-models into Relational Schemas
- Design Considerations
- ER-model Inheritance
- Alternative Diagram Notation

60



- Should phone be an entity or an attribute?
 - Reuse: designing phone number as an entity allows this entity to have relationships with several entities (e.g., student phone numbers)
 - Complexity and disassociation:
 - Adding extra entities increases the complexity of the schema
 - Using an attribute more closely associates phone numbers with instructors

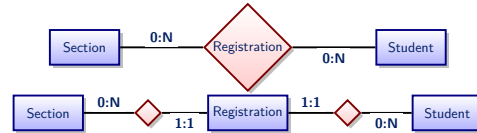


60



Entities vs. Relationships

- Should registration be a relationship or an entity?
 - Reuse:** if registration is an entity it can have relationships with additional entities, e.g., we could record the instructor that did put in an override to let the student register without prerequisites
 - Complexity:** creating a separate registration entity increases the number of elements in the diagram

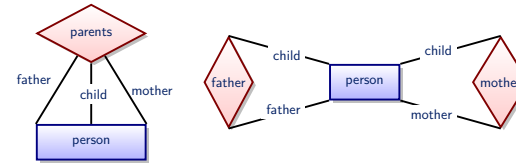


61



Multiple Binary vs. N-ary Relationships

- Parents ternary relationship vs. mother + father binary relationship

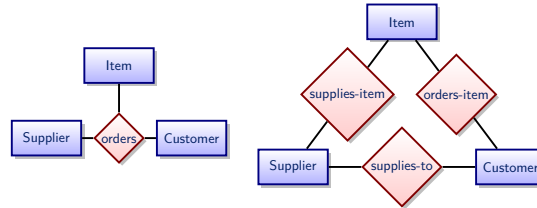


62



Multiple Binary vs. N-ary Relationships

- Orders vs. Supplies-to + Supplies-item + Orders-item



63



Multiple Binary vs. N-ary Relationships

Expressiveness of n-ary relationships

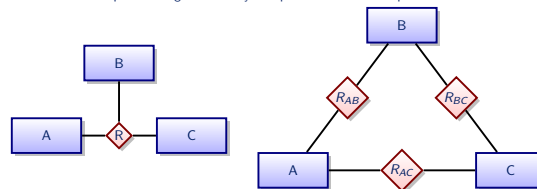
- A ternary relationship is more expressive than any number of binary relationships
 - With ternary relationships we can record which triples of entities are connected. However, binary relationships only encode pairwise connections.
- (Gunnar, chainsaw, Bob) records that
 - "Bob ordered a chainsaw from Gunnar"
- {(Gunnar, chainsaw), (chainsaw, Bob), (Gunnar, Bob)} records that
 - "Gunnar supplies chainsaws"
 - "Gunnar supplies to Bob"
 - "Bob has ordered a chainsaw"

64



Converting N-ary to Multiple Binary Relationships

- It is possible to convert n-ary relationships into equivalent binary relationships
 - This requires adding a new entity to represent the relationship



65



The ER-Model

The ER-Model

Introduction
 Entities & Attributes
 Relationships
 ER Diagrams
 Translating ER-models into Relational Schemas
 Design Considerations
 ER-model Inheritance
 Alternative Diagram Notation

66



Inheritance

Motivation

- Similar to motivation of inheritance in object-oriented programming
- Sharing commonalities among super entity and sub entity

Inheritance in the ER-model

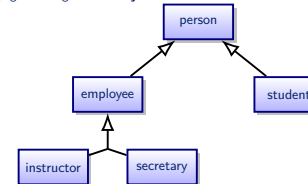
- The sub entity automatically inherits all relationships and attributes from the super entity
- Consider an super entity S and multiple sub entities U_1 to U_k inheriting from S
 - partial vs. total: if total then every S entity has to belong to at least on U_i
 - disjoint vs. overlapping: if overlapping then an entity can be part of more than one sub entity set

66



Inheritance Example

- Arrows indicate inheritance (from sub to super entity)
 - Multiple incoming arrows: overlapping
 - Merging incoming arrows: disjoint



67



Translation to Relational Schema

- Several alternative methods
 - each of these methods has some advantages & disadvantages
 - the semantics of inheritance cannot be fully enforced in the relational model

68



Method 1: Separate Relations / Sharing

Method 1: Separate Relations / Sharing

- Create a relation for each super and sub entity
- Only repeat the key attributes of the super entity in the relations for the sub entities
- The primary key in each sub entity relation is a foreign key to the relation for the super entity

person (ID, name, street, city)
 student (ID, totcred)
 employee (ID, salary)

69



Method 1: Trade-offs

Advantages

- No repetition of common attributes
- Supports partial and overlapping inheritance without redundancy

Disadvantages

- Access all information about a sub entity requires joining multiple tables

70



Method 2: Separate Relations / Independent

Method 2: Separate Relations / Independent

- Create a relation for the super entity and every sub entity
- These relations have all (including inherited attributes)

- person** (ID, name, street, city)
- student** (ID, name, street, city, totcred)
- employee** (ID, name, street, city, salary)

71



Method 2: Trade-offs

Advantages

- No joins required to gather all information for an entity

Disadvantages

- Querying multiple entity types from the inheritance hierarchy requires union operations

72



Method 3: Single Relation

Method 3: Single Relation

- Create a single relation with all attributes from the super and all sub entities
- For **disjoint** inheritance add an attribute storing the type of the entity
- For **overlapping** inheritance add one boolean attribute per sub entity

- disjoint inheritance**
 - person** (ID, **type**, name, street, city, totcred, salary)
- overlapping inheritance**
 - person** (ID, **is_employee**, **is_student**, name, street, city, totcred, salary)

73



Method 3: Trade-offs

Advantages

- No join or union required to gather information about super / sub entities

Disadvantages

- Can lead to many NULL values for attributes that are not applicable

74



The ER-Model

The ER-Model

Introduction

Entities & Attributes

Relationships

ER Diagrams

Translating ER-models into Relational Schemas

Design Considerations

ER-model Inheritance

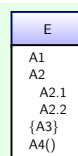
Alternative Diagram Notation

75



Attributes Inside Entity Boxes

- Include attributes inside the rectangle for an entity
 - composite attributes (indentation)
 - multi-valued attribute A written as {A}
 - derived attribute A written as A()

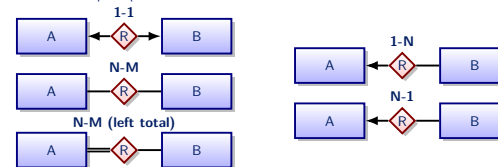


75



Edge Types for Cardinality Constraints

- Use different edge types to indicate cardinality constraints:
 - Arrow on 1 side of a relationship
 - Total participation indicated with a double line

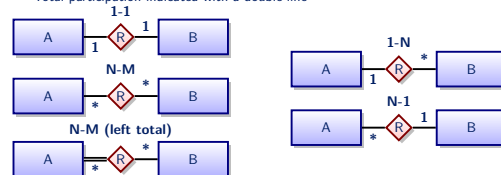


76



Edge Types - Chen Notation

- Use 1 for one side
- Use * for many side
- Total participation indicated with a double line

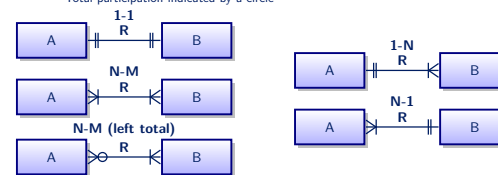


77



Edge Types - Crows Feet Notation

- Use different edge types to indicate cardinality constraints:
 - "Crow's feet" on the N side of a relationship
 - Total participation indicated by a circle



78



Recap

The Database Design

The ER-Model

Recap



Recap

- Database Design Process
 - Requirement Analysis, Conceptual Modeling, Logical Modeling
- ER-model
 - Entities: strong vs weak
 - Attributes: simple, composite, multi-valued, derived
 - Relationships and cardinality constraints
 - Translation to relational schema (rules 1 to 6)
 - Design considerations
 - Inheritance
 - Total vs. partial
 - Disjoint vs. overlapping