DATABASE SYSTEM PRINCIPLE

– ENTITY-RELATIONSHIP MODEL

李旭东
LEEXUDONG@NANKAI.EDU.CN
NANKAI UNIVERSITY

OBJECTIVES

- Design Process
- Modeling
- Constraints
- E-R Diagram
- Design Issues
- Weak Entity Sets
- Extended E-R Features
- Reduction to Relation Schemas

DESIGN PHASES

- (1)The initial phase of database design is to characterize fully the data needs of the prospective database users.
- (2)Next, the designer chooses a data model and, by applying the concepts of the chosen data model, translates these requirements into a conceptual schema of the database.
- (3)A fully developed conceptual schema also indicates the functional requirements of the enterprise.
  - In a "specification of functional requirements", users describe the kinds of operations (or transactions) that will be performed on the data.

DESIGN PHASES（CONT.,）

- (4)The process of moving from an abstract data model to the implementation of the database proceeds in two final design phases.
  - Logical Design – Deciding on the database schema. Database design requires that we find a "good" collection of relation schemas.
    - Business decision – What attributes should we record in the database?
    - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
  - Physical Design – Deciding on the physical layout of the database

DESIGN ALTERNATIVES选择

- In designing a database schema, we must ensure that we avoid two major pitfalls:
  - Redundancy
- A bad design may repeat information
  - Incompleteness
    - A bad design may make certain aspects of the enterprise difficult or impossible to model

DESIGN APPROACHES

- Entity Relationship Model (covered in this chapter)
  - Models an enterprise as a collection of *entities* and *relationships*
    - **Entity实体**: a "thing" or "object" that is distinguishable from other objects
      - Described by a set of *attributes*
    - **Relationship关系**: an association among several entities
  - Represented diagrammatically by an *entity-relationship diagram:*
- Normalization Theory (规范化理论)
  - Formalize what designs are bad, and test for them

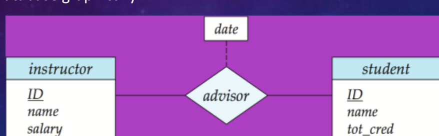
ENTITY-RELATIONSHIP MODEL

ER MODEL – DATABASE MODELING

- The ER data mode was developed to facilitate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database.
- The ER model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema. Because of this usefulness, many database-design tools draw on concepts from the ER model.
- The ER data model employs three basic concepts:
  - entity sets, relationship sets, attributes.

ER MODEL – DATABASE MODELING

- The ER model also has an associated diagrammatic representation, the ER diagram, which can express the overall logical structure of a database graphically.



BIBLIOGRAPHY : E-R MODEL

Dr. Peter Chen（陳品山）at Louisiana State University (LSU)

http://www.csc.lsu.edu/~chen/

- Chen P S. The Entity-Relationship Model—Toward a Unified View of Data[J]. Acm Transactions on Database Systems, 1976, 1(1):9-36.
- Prof. Peter Chen received his Ph.D. from Harvard University and has held regular and visiting faculty appointments at MIT, UCLA and Harvard.
- He is the originator of the **Entity-Relationship Model (ER Model)**, which serves as the foundation of many systems analysis and design methodologies, computer-aided software engineering (CASE) tools, and repository systems.
- Now "Entity-Relationship Model (ER Model)," "Entity-Relationship Diagram (ER Diagram)," and "Peter Chen" have become commonly used terms in "online" dictionaries, books, articles, web pages, course syllabi, and commercial product brochures.

ENTITY SETS

- An entity is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- An entity set is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays

ENTITY SETS

- An entity is represented by a set of attributes
  - i.e., descriptive properties possessed by all members of an entity set.
  - Example:

    instructor = (ID, name, street, city, salary)
    course= (course_id, title, credits)
- A subset of the attributes form a primary key of the entity set; i.e., uniquely identifiying each member of the set.

ENTITY SETS -- *INSTRUCTOR* AND *STUDENT*

instructor

instructor_ID	instructor_name
76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

student

student-ID	student_name
98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

RELATIONSHIP SETS

- A relationship is an association among several entities

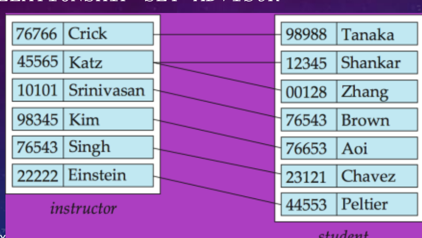
  Example:
  44553 (Peltier) advisor 22222 (Einstein)
  student entityrelationship set *instructor* entity

- A relationship set is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

  $$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

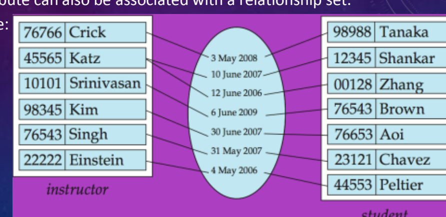
  where $(e_1, e_2, \ldots, e_n)$ is a relationship
  - Example: $(44553,22222) \in$ *advisor*

RELATIONSHIP SET *ADVISOR*



RELATIONSHIP SETS (CONT.)

- An attribute can also be associated with a relationship set.
- Example:



DEGREE度 OF A RELATIONSHIP SET

- binary relationship二元关系
  - involve two entity sets (or degree two).
  - most relationship sets in a database system are binary.
- Relationships between more than two entity sets are rare
  - Example: *students* work on research *projects* under the guidance of an *instructor*.
  - relationship *proj_guide* is a ternary relationship between *instructor*, *student*, and *project*

ATTRIBUTES属性

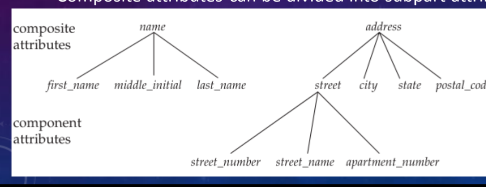
- Domain
  - For each attribute, there is a set of permitted values, called the domain域, or value set值集
- Example
  - the domain of attribute semester might be strings from the set {Fall, Winter, Spring, Summer}

ATTRIBUTE OF AN ENTITY SET

- an attribute of an entity set is a function that maps from the entity set into a domain.
- Since an entity set may have several attributes, each entity can be described by a set of (attribute, data value) pairs, one pair for each attribute of the entity set.
- Example: instructor
  - {(ID , 76766), (name, Crick), (dept_name, Biology), (salary, 72000)}

ATTRIBUTE TYPES 1/3

- Simple简单 and composite复合 attributes
  - Composite attributes can be divided into subpart attributes



ATTRIBUTE TYPES 2/3

- Single-valued and multivalued attributes
- example
  - The student ID attribute for a specific student entity refers to only one student ID
  - An instructor may have zero, one, or several phone numbers, and different instructors may have different numbers of phones.
    - This type of attribute is said to be multivalued

ATTRIBUTE TYPES 3/3

- Derived attribute派生属性
  - The value for this type of attribute can be derived from the values of other related attributes or entities.
- Suppose that the instructor entity set has an attribute age that indicates the instructor’s age. If the instructor entity set also has an attribute date of birth, we can calculate age from date of birth and the current date
  - Thus, age is a derived attribute

VALUE OF ATTRIBUTE

- null value
  - An attribute takes a null value when an entity does not have a value for it.
  - The null value may indicate “not applicable“不适用的
  - that is, that the value does not exist for the entity

CONSTRAINTS约束

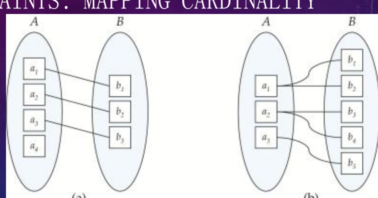
CONSTRAINTS约束

- An E-R enterprise schema may define certain constraints to which the contents of a database must conform
  - mapping cardinalities映射基数
  - participation constraints参与约束
  - Key码

CONSTRAINTS: MAPPING CARDINALITY(映射基数)

- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one, One to many, Many to one, Many to many

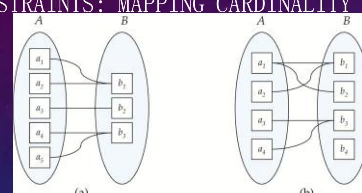
CONSTRAINTS: MAPPING CARDINALITY



One to one One to many

Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

CONSTRAINTS: MAPPING CARDINALITY



Many to one Many to many

Note: Some elements in A and B may not be mapped to any elements in the other set

CONSTRAINTS: PARTICIPATION CONSTRAINTS参与约束

- The participation of an entity set E in a relationship set R is said to be total全部 if every entity in E participates in at least one relationship in R.
- If only some entities in E participate in relationships in R, the participation of entity set E in relationship R is said to be partial部分.

CONSTRAINTS: KEY码

- We must have a way to specify how entities within a given entity set are distinguished.
- The values of the attribute values of an entity must be such that they can uniquely identify the entity.
- Superkey超码, candidate key候选码, primary key主码

CONSTRAINTS: KEY OF RELATIONSHIP SET 1/4

- Let R be a relationship set involving entity sets E1 , E2 ,..., En
- Let primary-key(Ei) denote the set of attributes that forms the primary key for entity set Ei
- Assume for now that the attribute names of all primary keys are unique.
- The composition of the primary key for a relationship set depends on the set of attributes associated with the relationship set R.

©LXD

CONSTRAINTS: KEY OF RELATIONSHIP SET 2/4

Case1:

- If the relationship set R has no attributes associated with it, then the set of attributes:

primary-key(E1) ∪ primary-key(E2) ∪ ⋯ ∪ primary-key(En)

describes an individual relationship in set R.

©LXD

CONSTRAINTS: KEY OF RELATIONSHIP SET 3/4

Case 2

- If the relationship set R has attributes a1 ,a2 ,...,a m associated with it, then the set of attributes:

primary-key(E1) ∪ primary-key(E2) ∪ ⋯ ∪ primary-key(En) ∪ { a1 ,a2 ,..., am } describes an individual relationship in set R.

©LXD

CONSTRAINTS: KEY OF RELATIONSHIP SET 4/4

- In both of the above cases, the set of attributes

primary-key(E1) ∪ primary-key(E2) ∪ ⋯ ∪ primary-key(En)

forms a superkey for the relationship set.

- For nonbinary relationships, if no cardinality constraints are present then the superkey formed is the only candidatekey, and it is chosen as the primarykey.

©LXD

REMOVING REDUNDANT ATTRIBUTES IN ENTITY SETS

©LXD

REMOVING REDUNDANT ATTRIBUTES IN ENTITY SETS

- Suppose we have entity sets:
  - *instructor*, with attributes: *ID*, *name*, *dept_name, salary*
  - *department,* with attributes: *dept_name, building, budget*
- We model the fact that each instructor has an associated department using ***a relationship set*** *inst_dept*
- The attribute *dept_name* appears in both entity sets. Since it is the primary key for the entity set *department*, it replicates information present in the relationship and is therefore redundant in the entity set *instructor* and needs to be removed.

©LXD

REMOVING REDUNDANT ATTRIBUTES IN ENTITY SETS

A good entity-relationship design does not contain redundant attributes:

- **classroom**: with attributes (*building*, *room_number*, *capacity*).
- **department**: with attributes (*dept_name*, *building*, *budget*).
- **course**: with attributes (*course_id*, *title*, *credits*).
- **instructor**: with attributes (*ID*, *name*, *salary*).
- **section:** with attributes (*course_id*, *sec_id*, *semester*, *year*).
- **student**: with attributes (*ID*, *name*, *tot_cred*).
- **time_slot**: with attributes (*time_slot_id*, {(*day*, *start_time*, *end_time*) }).

REMOVING REDUNDANT ATTRIBUTES IN ENTITY SETS

- **inst_dept**: relating instructors with departments.
- **stud_dept**: relating students with departments.
- **teaches**: relating instructors with sections.
- **takes**: relating students with sections, with a descriptive attribute *grade*.
- **course_dept**: relating courses with departments.
- **sec_course**: relating sections with courses.
- **sec_class**: relating sections with classrooms.
- **sec_time_slot**: relating sections with time slots.
- **advisor**: relating students with instructors.
- **prereq**: relating courses with prerequisite courses.

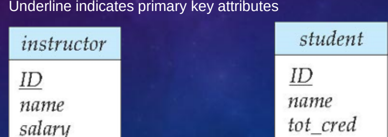
ENTITY-RELATIONSHIP DIAGRAMS

ENTITY-RELATIONSHIP DIAGRAMS

- E-R diagram can express the overall logical structure of a database graphically
- Basic structure:
  - **Rectangles divided into two parts** represent entity sets. The first part, which in this textbook is shaded blue, contains the name of the entity set. The second part contains the names of all the attributes of the entity set.
  - **Diamonds** represent relationship sets.
  - **Undivided rectangles** represent the attributes of a relationship set. Attributes that are part of the primary key are underlined.
  - **Lines** link entity sets to relationship sets.
  - **Dashed lines** link attributes of a relationship set to the relationship set.
  - **Double lines** indicate total participation of an entity in a relationship set.
  - **Double diamonds** represent identifying relationship sets linked to weak entity sets (we discuss identifying relationship sets and weak entity sets later,

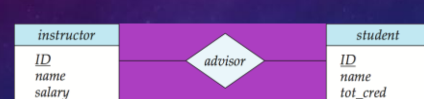
ENTITY SETS

- Entities can be represented graphically as follows:
  - Rectangles represent entity sets.
  - Attributes listed inside entity rectangle
  - Underline indicates primary key attributes

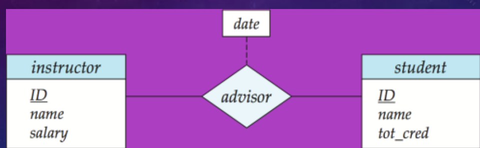


RELATIONSHIP SETS

- Diamonds represent relationship sets.



RELATIONSHIP SETS WITH ATTRIBUTES



©LXD

CARDINALITY CONSTRAINTS映射基数

- We express cardinality constraints by drawing the following line between the relationship set and the entity set
  - either a directed line (→), signifying "one,"
  - or an undirected line (—), signifying "many,"

©LXD

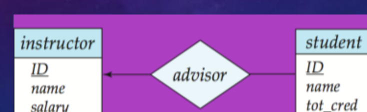
CARDINALITY CONSTRAINTS映射基数

- One-to-one relationship between an *instructor* and a *student* :
  - A student is associated with at most one *instructor* via the relationship *advisor*
  - A *student* is associated with at most one *department* via *stud_dept*

©LXD

CARDINALITY CONSTRAINTS映射基数

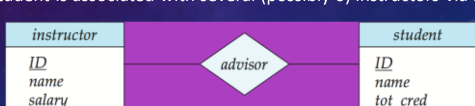
- One-to-many relationship between an *instructor* and a *student*
  - an instructor is associated with several (including 0) students via *advisor*
  - a student is associated with at most one instructor via advisor,



©LXD

CARDINALITY CONSTRAINTS映射基数: MANY-TO-MANY RELATIONSHIP

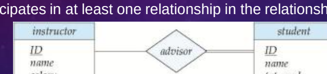
- An instructor is associated with several (possibly 0) students via *advisor*
- A student is associated with several (possibly 0) instructors via *advisor*



©LXD

CARDINALITY CONSTRAINTS映射基数: TOTAL AND PARTIAL PARTICIPATION

- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set



participation of *student* in *advisor* relation is total

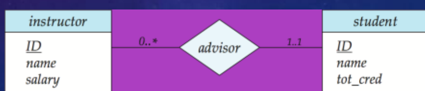
- every *student* must have an associated instructor

- Partial participation: some entities may not participate in any relationship in the relationship set
- Example: participation of *instructor* in *advisor* is partial

©LXD

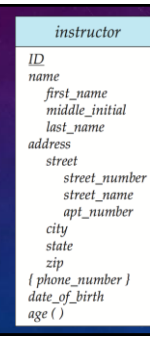
NOTATION FOR EXPRESSING MORE COMPLEX CONSTRAINTS

- A line may have an associated minimum and maximum cardinality, shown in the form *l..h*, where *l* is the minimum and *h* the maximum cardinality
  - A minimum value of 1 indicates total participation.
  - A maximum value of 1 indicates that the entity participates in at most one relationship
  - A maximum value of * indicates no limit.



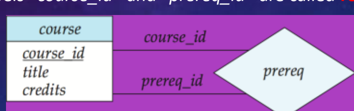
NOTATION FOR EXPRESSING MORE COMPLEX CONSTRAINTS

- composite attribute name
- a multivalued attribute phone number
- a derived attribute age



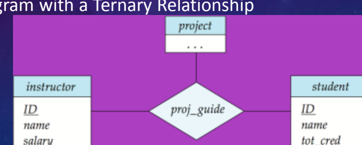
ROLES角色

- Entity sets of a relationship need not be distinct
  - Each occurrence of an entity set plays a “role” in the relationship
- The labels “*course_id*” and “*prereq_id*” are called roles.



NON-BINARY RELATIONSHIP SETS

- Most relationship sets are binary
- There are occasions when it is more convenient to represent relationships as non-binary.
- E-R Diagram with a Ternary Relationship



CARDINALITY CONSTRAINTS ON TERNARY RELATIONSHIP三元联系

- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
  - For exampe, an arrow from *proj_guide* to *instructor* indicates each student has at most one guide for a project
- If there is more than one arrow, there are two ways of defining the meaning.
  - Example (cont.,)

CARDINALITY CONSTRAINTS ON TERNARY RELATIONSHIP

- If there is more than one arrow, there are two ways of defining the meaning.
  - For example, a ternary relationship *R* between *A*, *B* and *C* with arrows to *B* and *C* could mean
    1. Each *A* entity is associated with a unique entity from *B* and *C* or
    2. Each pair of entities from (*A, B*) is associated with a unique *C* entity, and each pair (*A, C*) is associated with a unique *B*
  - Each alternative has been used in different formalisms
  - To avoid confusion we outlaw more than one arrow

WEAK ENTITY SETS弱实体集

- Consider a *section* entity, which is uniquely identified by a *course_id*, *semester*, *year*, and *sec_id*.
- Clearly, section entities are related to course entities. Suppose we create a relationship set *sec_course* between entity sets *section* and *course*.
- Note that the information in *sec_course* is redundant, since *section* already has an attribute *course_id*, which identifies the course with which the section is related.
- **One option** to deal with this redundancy is to get rid of the relationship *sec_course*; however, by doing so the relationship between *section* and *course* becomes implicit in an attribute, which is not desirable.

WEAK ENTITY SETS (CONT.)

- An alternative way to deal with this redundancy is to not store the attribute *course_id* in the *section* entity and to only store the remaining attributes section_id, *year*, and *semester.*
- However, the entity set *section* then does not have enough attributes to identify a particular *section* entity uniquely; although each *section* entity is distinct, sections for different courses may share the same *section_id*, *year*, and *semester*.
- To deal with this problem, we treat the relationship ***sec_course*** as a special relationship that provides extra information, in this case, the *course_id*, required to identify *section* entities uniquely.

WEAK ENTITY SETS (CONT.)

- The notion of **weak entity set** formalizes the above intuition.
- A weak entity set is one whose existence is dependent on another entity, called its **identifying entity**;
- Instead of associating a primary key with a weak entity, we use the identifying entity, along with extra attributes called **discriminator分辨符** to uniquely identify a weak entity.
- An entity set that is not a weak entity set is termed a **strong entity set强实体集**

WEAK ENTITY SETS (CONT.)

- Every weak entity must be associated with an identifying entity标识实体
- that is, the weak entity set is said to be **existence dependent存在依赖** on the identifying entity set.
- The identifying entity set标识实体集(属主实体集) is said to **own拥有** the weak entity set that it identifies. The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship标识性联系**.
- Note that the relational schema we eventually create from the entity set *section* does have the attribute *course_id*, for reasons that will become clear later, even though we have dropped the attribute *course_id* from the entity set *section.*

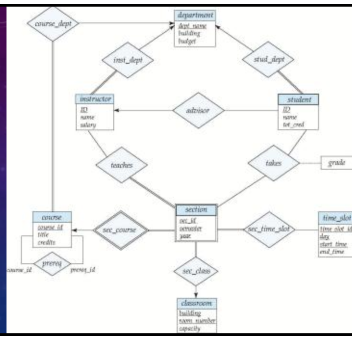
WEAK ENTITY SETS (CONT.)

- In E-R diagrams, a weak entity set is depicted via a double rectangle.
- We underline the discriminator分辨符 of a weak entity set with a dashed line.
- The relationship set connecting the weak entity set to the identifying strong entity set is depicted by a double diamond.
- The primary key of a weak entity set is formed by the primary key of the identifying entity set, plus the weak entity set's discriminator

WEAK ENTITY SETS (CONT.)

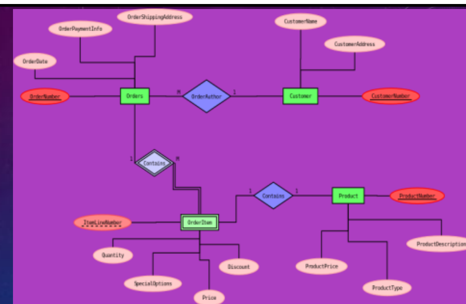
- Primary key for *section* – (*course_id, sec_id, semester, year*)

E-R DIAGRAM FOR A UNIVERSITY ENTERPRISE



©LXD

E-R DIAGRAM FOR A CUSTOMER ORDERS



From https://en.wikipedia.org/wiki/Weak_entity

©LXD

EXAMPLE

- 给某商场会员卡业务画出相关实体关系图
  - 某商场可以为顾客办理会员卡;
  - 每个顾客只能办理一张会员卡;
  - 顾客信息包括顾客姓名、地址、电话、身份证号;
  - 会员卡信息包括号码、等级和积分。
  - 顾客具有多个地址和多个电话号码，地址包括省、市、区、街道;
  - 电话号码包括区号、号码。

©LXD

REDUCTION TO RELATIONAL SCHEMAS

©LXD

REDUCTION TO RELATIONAL SCHEMAS

- We can represent a database that conforms to an E-R database schema by a collection of relation schemas.
- For each entity set and for each relationship set in the database design, there is a unique relation schema to which we assign the name of the corresponding entity set or relationship set.
- Both the E-R model and the relational database model are abstract, logical representations of real-world enterprises.
  - Because the two models employ similar design principles, we can convert an E-R design into a relational design.

©LXD

REDUCTION TO RELATIONAL SCHEMAS: REPRESENTATION OF STRONG ENTITY SETS WITH SIMPLE ATTRIBUTES

- A strong entity set reduces to a schema with the same attributes

student(<u>ID</u>, name, tot_cred)

classroom (building, <u>room_number</u>, capacity)
department (<u>dept_name</u>, building, budget)
course (<u>course_id</u>, title, credits)
instructor (<u>ID</u>, name, salary)
student (<u>ID</u>, name, tot_cred)

©LXD

REDUCTION TO RELATIONAL SCHEMAS: REPRESENTATION OF STRONG ENTITY SETS WITH COMPLEX ATTRIBUTES 1/2

- (1) composite attribute

instructor (*ID*, *first_name*, *middle_name*, *last_name*, *street_number*, *street_name*, *apt_number*, *city*, *state*, *zip_code*, *date_of_birth*)

©LXD

REDUCTION TO RELATIONAL SCHEMAS: REPRESENTATION OF STRONG ENTITY SETS WITH COMPLEX ATTRIBUTES 2/2

- (2) Multivalued attribute
  - Example: the entity set instructor, which includes the multivalued attribute phone number
    - For this multivalued attribute, we create a relation schema

instructor_phone (<u>*ID*</u>, <u>*phone_number*</u>)

©LXD

REDUCTION TO RELATIONAL SCHEMAS: REPRESENTATION OF WEAK ENTITY

- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set
- Let A be a weak entity set with attributes a1, a2 ,...,am. Let B be the strongentity set on which A depends. Let the primary key of B consist of attributes b1 , b2 ,...,bn.
- We represent the entity set A by a relation schema called A with one attribute for each member of the set:{a1 ,a2 ,...,am } ∪ {b1 ,b2 ,...,bn }
- Example: *Supporting on delete cascade*

section (course_id, sec_id, sem, year)

©LXD

REDUCTION TO RELATIONAL SCHEMAS: REPRESENTATION OF RELATIONSHIP SETS

- Let R be a relationship set, let a1 ,a2 ,...,am be the set of attributes formed by the union of the primary keys of each of the entity sets participating in R
- Let the descriptive attributes (if any) of R be b1 ,b2 ,...,bn.
- We represent this relationship set by a relation schema called R with one attribute for each member of the set:

{a1 ,a2 ,...,am } ∪ {b1 ,b2 ,...,bn }

©LXD

REDUCTION TO RELATIONAL SCHEMAS: REPRESENTATION OF RELATIONSHIP SETS

- For a binary many-to-many relationship, the union of the primary-key attributes from the participating entity sets becomes the primary key.
- For a binary one-to-one relationship set, the primary key of either entity set can be chosen as the primary key. The choice can be made arbitrarily.
- For a binary many-to-one or one-to-many relationship set, the primary key of the entity set on the "many" side of the relationship set serves as the primary key.
- Cont.,

©LXD

REDUCTION TO RELATIONAL SCHEMAS: REPRESENTATION OF RELATIONSHIP SETS

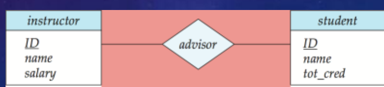
- For an n-ary relationship set with out any arrows on its edges, the union of the primary key-attributes from the participating entity sets becomes the primary key.
- For an n-ary relationship set with an arrow on one of its edges, the primary keys of the entity sets not on the "arrow" side of the relationship set serve as the primary key for the schema.

©LXD

REDUCTION TO RELATIONAL SCHEMAS: REPRESENTATION OF RELATIONSHIP SETS

- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set *advisor*

advisor = (*s_id, i_id*)



REDUCTION TO RELATIONAL SCHEMAS: REPRESENTATION OF RELATIONSHIP SETS

teaches (*ID*, *course_id*, *sec_id*, *semester*, *year*)
takes (*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)
prereq (*course_id*, *prereq_id*)
advisor (*s_ID*, *i_ID*)
sec_course (*course_id*, *sec_id*, *semester*, *year*)
sec_time_slot (*course_id*, *sec_id*, *semester*, *year*, *time_slot_id*)
sec_class (*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*)
inst_dept (*ID*, *dept_name*)
stud_dept (*ID*, *dept_name*)
course_dept (*course_id*, *dept_name*)

REDUCTION TO RELATIONAL SCHEMAS: REDUNDANCY OF SCHEMAS 1/2

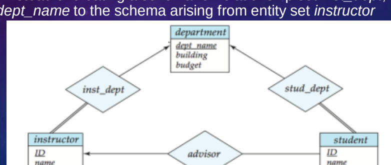
- A relationship set linking a weak entity set to the corresponding strong entity set is treated specially
- the weak entity set section is dependent on the strong entity set course via the relationship set sec_course.
- The primary key of section is {course id, sec_id, semester, year} and the primary key of course is course id.
- Since sec_course has no descriptive attributes, the sec_course schema has attributes course id, sec_id, semester, and year.
- Thus, the sec_course schema is redundant

REDUCTION TO RELATIONAL SCHEMAS: REDUNDANCY OF SCHEMAS 2/2

- In general, the schema for the relationship set linking a weak entity set to its corresponding strong entity set is redundant
- and does not need to be present in a relational database design based upon an E-R diagram.

REDUCTION TO RELATIONAL SCHEMAS: COMBINATION合并 OF SCHEMAS

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side
- Example: Instead of creating a schema for relationship set *inst_dept*, add an attribute *dept_name* to the schema arising from entity set *instructor*



REDUCTION TO RELATIONAL SCHEMAS: COMBINATION合并 OF SCHEMAS

- For one-to-one relationship sets, either side can be chosen to act as the "many" side
  - That is, an extra attribute can be added to ***either*** of the tables corresponding to the two entity sets
- If participation is *partial* on the "many" side, replacing a schema by an extra attribute in the schema corresponding to the "many" side could result in null values

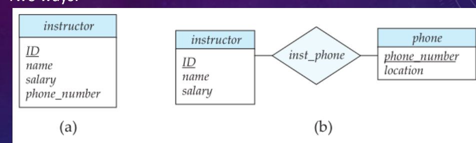
ENTITY-RELATIONSHIP DESIGN ISSUES

- Use of Entity Sets versus Attributes
- Use of Entity Sets versus Relationship Sets
- Binary versus n-ary Relationship Sets
- Placement of Relationship Attributes

©LXD

USE OF ENTITY SETS VERSUS ATTRIBUTES

Two ways:

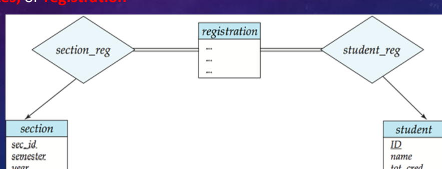


The main difference then is that treating a phone as an entity better models a situation where one may want to keep extra information about a phone, such as its location, or its type (mobile, IP phone, or plain old phone), or all who share the phone.

©LXD

USE OF ENTITY SETS VERSUS RELATIONSHIP SETS

- Possible guideline is to designate a relationship set to describe an action that occurs between entities
  - takes, or registration



©LXD

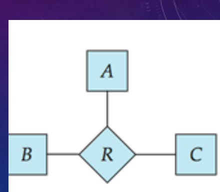
BINARY VERSUS N-ARY(N元) RELATIONSHIP SETS

- Although it is possible to replace any non-binary (*n*-ary, for *n* > 2) relationship set by a number of distinct binary relationship sets, a *n*-ary relationship set shows more clearly that several entities participate in a single relationship.
- Some relationships that appear to be non-binary may be better represented using binary relationships
  - For example, a ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
    - Using two binary relationships allows partial information (e.g., only mother being known)
  - But there are some relationships that are naturally non-binary
    - Example: *proj_guide*

©LXD

CONVERTING NON-BINARY RELATIONSHIPS TO BINARY FORM 1/3

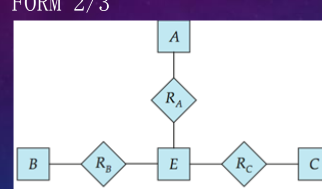
- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.



(a)

©LXD

CONVERTING NON-BINARY RELATIONSHIPS TO BINARY FORM 2/3



(b)

©LXD

CONVERTING NON-BINARY RELATIONSHIPS TO BINARY FORM 3/3

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Replace *R* between entity sets A, B and C by an entity set *E*, and three relationship sets:

  1. R_A, relating *E* and *A* 2. R_B, relating *E* and *B* 3. R_C, relating *E* and *C*

  - Create an identifying attribute for *E and* add any attributes of *R* to *E*
  - For each relationship (a_i, b_i, c_i) in *R,* create

    1. a new entity e_i in the entity set *E* 2. add (e_i, a_i) to R_A

    3. add (e_i, b_i) to R_B 4. add (e_i, c_i) to R_C

CONVERTING NON-BINARY RELATIONSHIPS TO BINARY FORM

- Also need to translate constraints
  - Translating all constraints may not be possible
  - There may be instances in the translated schema that cannot correspond to any instance of *R*
    - Exercise: *add constraints to the relationships* R_A, R_B *and* R_C to ensure that a newly created entity corresponds to exactly one entity in each of entity sets *A, B* and *C*
  - We can avoid creating an identifying attribute by making E a weak entity set (described shortly) identified by the three relationship sets

EXTENDED E-R FEATURES

EXTENDED E-R FEATURES

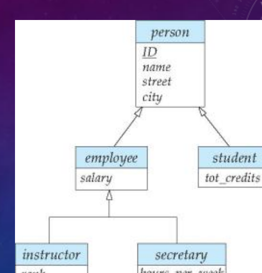
- Specialization
- Generalization
- Attribute Inheritance
- Constraints on Generalizations
- Aggregation

SPECIALIZATION特化

- Top-down design process
- we designate sub-groupings within an entity set that are distinctive from other entities in the set.
- These sub-groupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle* component labeled ISA (e.g., *instructor* "is a" *person*).

SPECIALIZATION

- **Overlapping重叠特化**
  - *employee* and *student*
- **Disjoint不相交特化**
  - *instructor* and *secretary*
- Total and partial



REPRESENTING SPECIALIZATION VIA SCHEMAS

- Method 1:
  - Form a schema for the higher-level entity
  - Form a schema for each lower-level entity set, include primary key of higher-level entity set and local attributes

schema	attributes
person	ID, name, street, city
student	ID, tot_cred
employee	ID, salary

  - Drawback: getting information about, an *employee* requires accessing two relations, the one corresponding to the low-level schema and the one corresponding to the high-level schema

©LXD

REPRESENTING SPECIALIZATION VIA SCHEMAS

- Method 2:
  - Form a schema for each entity set with all local and inherited attributes

schema	attributes
person	ID, name, street, city
student	ID, name, street, city, tot_cred
employee	ID, name, street, city, salary

  - Drawback: *name, street* and *city* may be stored redundantly for people who are both students and employees

©LXD

GENERALIZATION概化

- A bottom-up design process – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization are used interchangeably.

©LXD

ATTRIBUTE INHERITANCE属性继承

- a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked
  - single inheritance
  - multiple inheritance
    - the resulting structure is said to be a lattice(格)

©LXD

CONSTRAINTS ON GENERALIZATIONS

- To model an enterprise more accurately, the database designer may choose to place certain constraints on a particular generalization
  - Condition-defined
  - User-defined
  - Disjoint不相交/Overlapping重叠
  - Completeness constraint

©LXD

CONSTRAINTS ON GENERALIZATIONS: CONDITION-DEFINED条件定义的

- In condition-defined lower-level entity sets, membership is evaluated on the basis of whether or not an entity satisfies an explicit condition or predicate
- For example, assume that the higher-level entity set student has the attribute studenttype. All student entities are evaluated on the defining studenttype attribute. Only those entities that satisfy the condition studenttype="graduate" are allowed to belong to the lower-level entity set graduate student.

©LXD

CONSTRAINTS ON GENERALIZATIONS: USER-DEFINED用户自定义的

- User-defined lower-level entity sets are not constrained by a membership condition;
- rather, the database user assigns entities to a given entity set
- For instance, let us assume that, after 3 months of employment, university employees are assigned to one of four work teams

CONSTRAINTS ON GENERALIZATIONS: DISJOINT/OVERLAPPING

- Disjoint不相交
  - A disjointness constraint requires that an entity belong to no more than one lower-level entity set
- Overlapping重叠
  - In overlapping generalizations, the same entity may belong to more than one lower-level entity set within a single generalization

DESIGN CONSTRAINTS ON A SPECIALIZATION/GENERALIZATION

- **Completeness constraint完全性约束**
  - specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.
  - total: an entity must belong to one of the lower-level entity sets
  - partial: an entity need not belong to one of the lower-level entity sets

DESIGN CONSTRAINTS ON A SPECIALIZATION/GENERALIZATION

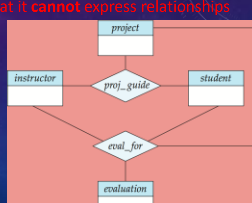
- Partial generalization is the default
  - We can specify total generalization in an ER diagram by adding the keyword **total** in the diagram and drawing a dashed line from the keyword to the corresponding hollow arrow-head to which it applies (for a total generalization), or to the set of hollow arrow-heads to which it applies (for an overlapping generalization).
- The *student* generalization is total
  - All student entities must be either graduate or undergraduate. Because the higher-level entity set arrived at through generalization is generally composed of only those entities in the lower-level entity sets, the completeness constraint for a generalized higher-level entity set is usually total

CONSTRAINTS ON GENERALIZATIONS: AGGREGATION聚集

- One limitation of the E-R model is that it **cannot** express relationships among relationships

■Consider the ternary relationship *proj_guide*, which we saw earlier

■Suppose we want to record evaluations of a student by a guide on a project



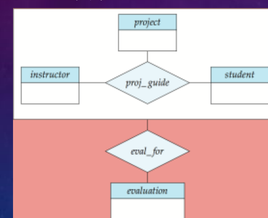
CONSTRAINTS ON GENERALIZATIONS: AGGREGATION聚集

- Relationship sets *eval_for* and *proj_guide* represent overlapping information
  - Every *eval_for* relationship corresponds to a *proj_guide* relationship
  - However, some *proj_guide* relationships may not correspond to any *eval_for* relationships
    - So we can't discard the *proj_guide* relationship

CONSTRAINTS ON GENERALIZATIONS: AGGREGATION聚集

- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity
- Eliminate this redundancy via *aggregation* without introducing redundancy, the following diagram represents:
  - A student is guided by a particular instructor on a particular project
  - A student, instructor, project combination may have an associated evaluation

CONSTRAINTS ON GENERALIZATIONS: AGGREGATION聚集



CONSTRAINTS ON GENERALIZATIONS: AGGREGATION聚集

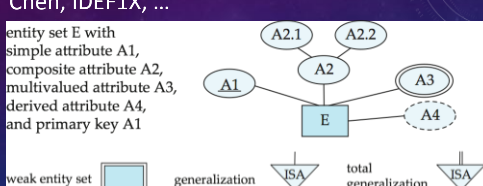
- To represent aggregation, create a schema containing
  - Primary key of the aggregated relationship,
  - The primary key of the associated entity set
  - Any descriptive attributes
- example:
  - The schema *eval_for* is:

    eval_for (*s_ID, project_id, i_ID, evaluation_id*)
  - The schema *proj_guide* is redundant.

ALTERNATIVE ER NOTATIONS

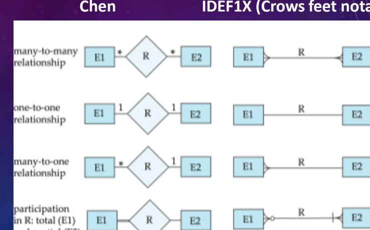
ALTERNATIVE ER NOTATIONS

- Chen, IDEF1X, …



ALTERNATIVE ER NOTATIONS

Chen IDEF1X (Crows feet notation)



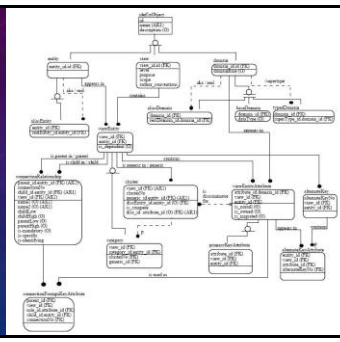
ALTERNATIVE ER NOTATIONS – IDEF

- In the mid-1970s, the Integrated Computer Aided Manufacturing (ICAM) Program
- IDEF (ICAM Definition) Methods
- IDEF0 used to produce a "function model" which is a structured representation of the activities or processes within the environment or system
- IDEF1 used to produce an "information model" which represents the structure and semantics of information within the environment or system
- IDEF2 used to produce a "dynamics model"

ALTERNATIVE ER NOTATIONS – IDEF1X

- Integration DEFinition for information modeling (IDEF1X) is a data modeling language for the development of semantic data models. IDEF1X is used to produce a graphical information model which represents the structure and semantics of information within an environment or system.
- IDEF1X permits the construction of semantic data models which may serve to support the management of data as a resource, the integration of information systems, and the building of computer databases. This standard is part of the IDEF family of modeling languages in the field of software engineering.

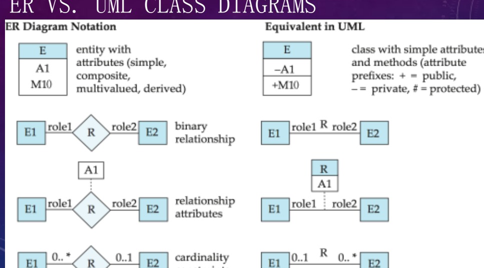
IDEF1X



UML

- UML: Unified Modeling Language
- UML has many components to graphically model different aspects of an entire software system
- UML Class Diagrams correspond to E-R Diagram, but several differences.

ER VS. UML CLASS DIAGRAMS

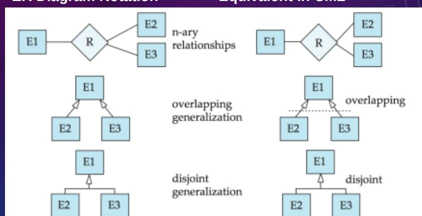


UML CLASS DIAGRAMS

- 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.

ER VS. UML CLASS DIAGRAMS

ER Diagram Notation **Equivalent in UML**



*Generalization can use merged or separate arrows independent of disjoint/overlapping

©LXD

OTHER ASPECTS OF DATABASE DESIGN

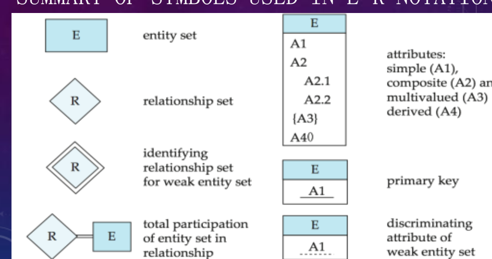
©LXD

OTHER ASPECTS OF DATABASE DESIGN

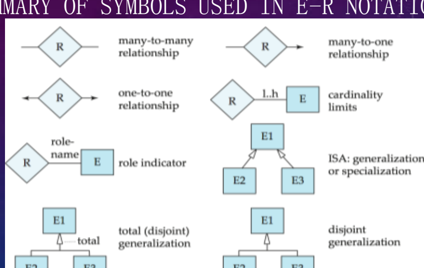
- Data Constraints and Relational Database Design
- Usage Requirements: Queries, Performance
  - Throughput吞吐量, Response time响应时间
- Authorization Requirements
- Data Flow, Workflow
- ...Database design is usually not a one-time activity. The needs of an organization evolve continually, and the data that it needs to store also evolve correspondingly

©LXD

SUMMARY OF SYMBOLS USED IN E-R NOTATION



SUMMARY OF SYMBOLS USED IN E-R NOTATION



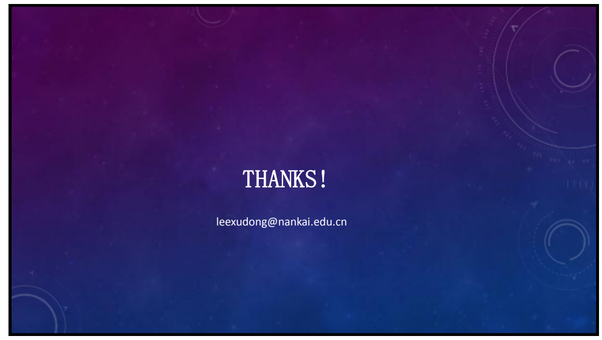
©LXD

SUMMARY

- Design Process
- Modeling
- Constraints
- E-R Diagram
- Design Issues
- Weak Entity Sets
- Extended E-R Features
- Reduction to Relation Schemas

©LXD


Q&A?


THANKS!
leexudong@nankai.edu.cn