Database Lecture 6
Logical Design

Addendum

Dr. Jefferson Fong

Reference: 2.1-2.4, 6.8.6

Midterm Exam Schedule:

Nov. 15 Sat. 09:30 - 11:30

Venue: TBD

Lecture 1 - 7

Lab 1- 7

Project Groups

- Is in iSpace under Section 1002 / 1005
- Every group should have at least 1 or 2 people taken System and Web Development workshop.
  - Let us know if that's not so.
- Project description due Sun Nov 9.

Database Design Process

Physical World → Conceptual Design produces **ER diagrams**

(Lectures 2 to 5)

→ Logical Design produces **schema for tables**

(this lecture)

→ SQL codes

Relational Model

Slide 3

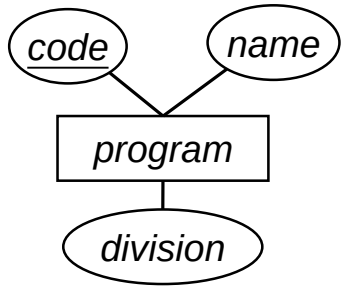
- Relational model uses tables to represent data and the relationship among the data.

Slide 4

- In relational model, **relation** = table, **tuple** = row, **attribute** = column.
  - A point in 3D Cartesian plane is represented as a **tuple** (x, y, z).
  - Examples for database table in Slide 5.

Slide 6

- An ER diagram's attribute is not always same as an attribute (column) in a table.
  - For some simple cases, they are the same.
  - For more general cases, they are not; more later in Relationship Sets..



Slide 7

- Candidate key - minimal set means smallest set among super keys.
- In MySQL or Access, we are usually interested in the primary key.
  - Candidate key is a candidate for the primary key
- At bottom,
  - {year, gpa} **cannot** be a **super key** because the **values** in the table **can change** or rows can be added; it's possible that in **another instance**, more than one person can have the **same year** and **gpa**.

Slide 9

- With primary key, you can find all other info in that table.

  (1) From table **student**, can find David's **student_ID**.

  (2) From table ***borrow*** (using David's **student_ID**), can find the **ISBN** of the book David borrowed.

  (3) From table ***book*** (using the **ISBN**), can find the name of the book.

Slide 10

- Exercise 1 – B and D.
- Exercise 2
  - Unlike Slide 7's example, the meanings of the attributes are not specified.
  - So we can only check the table's values for possible keys.
  - {A, B} is a **super key** because there are **no duplication** in the table when we consider columns A and B together
    - i.e. no duplication in {A1, B1}, {A1, B2}, {A2, B1}
    - So {A, B} uniquely determine every row.
  - {C, D} is **not** a super key because duplication in {C1, D1}, **{C2, D1}, {C2, D1}.**
    - **{C2, D1}** does **not** uniquely determine a row.
  - Candidate keys – smallest set among super keys.
    - Has 2 elements in this example.

Slide 12 Entity Sets

- Consider **strong entity** first.
  - Treat weak entity, composite attribute and multi-value attribute in Slide 18-20.

Slide 13-17. Relationship sets for **strong entities**, 10 cases, depending on **constraints**.

- **General guideline**: use the **simplest** way to **uniquely** determine the relationships between the two strong entities.
- **Case 1 to 3** (**Slide 13**), **many-to-many** relationship
  - Simple cases.
  - Need the **key** from each entity because of the "**many**" constraint.
  - So the **two keys** from the two **entities** form the **relationship's key**.
  - E.g. Case 3 (**Slide 14)**
  - Entities: *program* = (*code*, …), *course* = (*course_name*, …)
    - From *code*, we can determines all attributes in the entity *program*.
    - From c*ourse_name*, we can determine all attributes in the entity *course*.
  - Relationship: *offer* = (code, course_name)
    - The key for relationship *offer* is the combination of the keys for the entities.

Slide 15

Case 4 to 6, **one to** XXX relationship

- In the diagram, the entity at left has constraint "**one**".

- The **entities schemas** are similar to Cases 1 to 3.
  - Create tables for the strong entities in usual way.
  - Slight change on the relationship's key.
- Case 4, **relationship's key** can be **either entity's key.**
  - 1-to-1 relationship; from the key of one entity, we can determine the key of the other entity.
  - E.g. suppose the program-offer-course in Slide 14 is a one-to-one relationship.
    - Not true in the real world, but suppose it is for this example.
    - Then the schema for relationship is offer(code, course_name)
    - From the key "code", we can determine course_name in this one-to-one relationship.
- Case 5 and 6, **relationship's key** is the **entity key** of the "**many**" side.
  - Don't need the key from the "one" entity; the "many" all choose that "one".

Logical Design – Relationship Sets

Slide 16-17

- Cases 7 to 10 **don't need** a **relationship schema**!
  - Only need to change the **entity** schema(s) slightly.
  - All these cases have (at least) "one" entity, with total participation on the other entity.
  - Just add the key from the "one" entity as **foreign key** in the table of the other (total participation) entity.
  - Example Case 10: **many instructors** in **one program**, with total participation in both sides.
  - In the *instructor* table, add the foreign key "code" (key of the "one" entity *program*); that totally determines the relationship.
  - Note: we still need the schema for the entity *program.*

Logical Design – Relationship Sets

Slide 16-17 (continue)

- Suppose we try to use this method on Case 6, with "*instructor*" not totally participating. What happens?
  - Try to use instructor = (id, …, code)
  - For a **non-participating instructor** (with id 222) not in any program, the value of the foreign key "code" would be NULL
  - A key cannot have a NULL value; if try to query using that key, the query would fail.

id	Code
111	1
222	NULL

Slide 18 Weak entity sets

- Similar to Case 9, one course to many sections (full participation).
- "Section" (with double rectangle) is a weak entity.
  - Just add the key ***course_name*** from the identifying strong entity to the schema for the weak entity.
  - Then **course_name** and **section_num** together form a **key** to the weak entity *section*.
  - Don't need a schema for the "belong".
  - Note: still need the strong entity course = (course_name).

Slide 19 Composite Attribute

- In the schema, we don't need the composite attribute *course_code*.
- Put the component attributes *domain* and *course_number* in the schema.

Slide 20 Multivalued Attributes

- The multi-value *phone* (in double ellipse) is stored in a separate schema with the entity's key *id*; i.e. person_phone = (*id*, *phone*).
  - So name and phone numbers only need to be listed once.

Slide 21 ISA, **overlapping** or **partial participation**

(i) **Overlapping** means a person can be both a student and instructor,

(ii) **Partial participation** means a person can be neither student nor instructor (such as a staff person).

- For these two cases,
  - The 3 entities are really separate; a person can be student, instructor, neither or both.
  - So have id as the key in all 3 entities.
  - Under various situations, can join the appropriate entities using the key *id*.
  - E.g. if a person is a student but not instructor, join person and student using *id*.
  - E.g. if a person is both student and instructor, join person, student and instructor using *id*;

Slide 22 ISA, (iii) **disjoint** and **total**

- A person **cannot** be both a student and instructor, and a person must be either a student or instructor.
  - No need for a schema for *person*; put those info (id and name) in *student* or *instructor*.
  - Since *student* and *instructor* are **disjoint**, the info for each person cannot be in both *student* and *instructor*; no redundancy.
  - **Total participation** means student and instructor account for all person.
  - If we try to use this method in Slide 21, we get redundancy (*person* is both *student* and *teacher*) or missing info for *person* (missing *name* for a non-teaching staff).

Slide 23 Multi-ary Relationship Set

- The relationship schema contains the keys of all entity sets involved.
  - From these keys, can get the info from all entities.

Slide 24 Aggregation

- Add *project_ID* to the schema enroll (in last slide) to get the schema of doing.
  - See Slide 27 for complete ER diagram
  - Also is solution to last week's exercises.

Slide 25

- Both *student* and *instructor* has attribute *id*; so rename to *student_ID* and *instructor_ID* respectively in the *doing* relationship.

Slide 26 exercises

Q1a. In addendum for Slide 17 exercise for Case 10, we tried apply Case 10 method (adding a foreign key to the "many" entity) to Case 6.

- Get NULL for *code* when *instructor* is non-participating.

Q1b. Now consider the relationship for "student" and "course"

- A student can take many courses, and a course can be taken by many students.
  - Some students don't take any course, and some course don't have any student.
- So this should be Case ___?
- For simplicity, let the entities to be *student*(*id*, *name*) and *course*(*ccode*, *cname*).
- What happens if we try to apply the method used in Case 7-10?
- Try to combine the two tables at right.
  - Don't know what values to put in for *code*.

id	name
111	Jefferson
222	Steven
333	Goliath

ccode	cname
COMP1	DBMS
COMP2	JwD
COMP3	NNDL

Exercises

Q1b. We get redundancies or create tuples that don't exists.

sid	sname	ccode	cname
222	Jefferson	COMP1	DBMS
222	Jefferson	COMP2	JwD
222	Jefferson	COMP3	NNDL

Q2. We get redundancies

id	name	phone
111	Jefferson	111-22222
111	Jefferson	999-33333
222	Steven	111-33333
222	Steven	999-44444
333	Goliath	

- If we use the wrong method, we get
  - (a) NULL value in the key of a table (Q1a)
  - (b) Unclear what value to enter; create tuples that don't exist (Q1b).
  - (c) Redundancy in the table (Q2)

Q3:

“Program PD instructor”, 1 to 1 with total participation for program.

Case 7 – add the foreign key (director_id) to the schema of the total participation.

$$programs = (\underline{p_{code}}, p\_name, division, director\_id)$$

“Program offer course”, many to many with total participation for program and course.

Case ???, see slide 14.

“Course” and “Section” – weak entity.

Team Meeting

- Get together with your group.
- Decide on project description.

Summary of ER Diagram Design

entity set

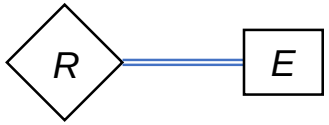
relationship set

attribute

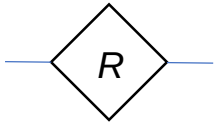
weak entity set

identifying relationship set

multivalued attribute

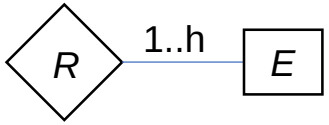


total participation

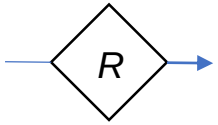


many-to-many relationship

derived attribute

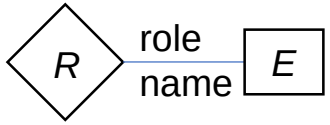


cardinality limits

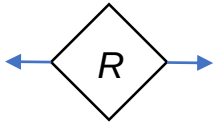


many-to-one relationship

primary key

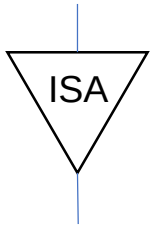


role indicator

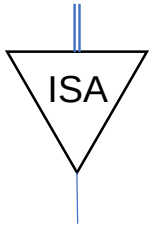


one-to-one relationship

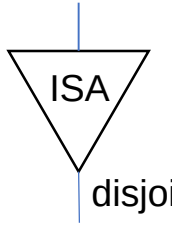
discriminator



ISA



total generalization



disjoint generalization