Relational Database Design

CISC637, Lecture #10

Ben Carterette

Database Design Process

- Start with enterprise requirements
  - → Translate to a conceptual model
  - → Translate to a logical model
  - → Translate to a physical model
- **Iterate** each step to ensure that every requirement that *needs to be* captured and that *can be* captured *actually is* captured

Database Design Process

- Start with enterprise requirements
  - →Translate to a conceptual model
    - Entity-Relationship model (E-R model)
  - →Translate to a logical model
    - Relational model
  - →Translate to a physical model
    - Implement in a database

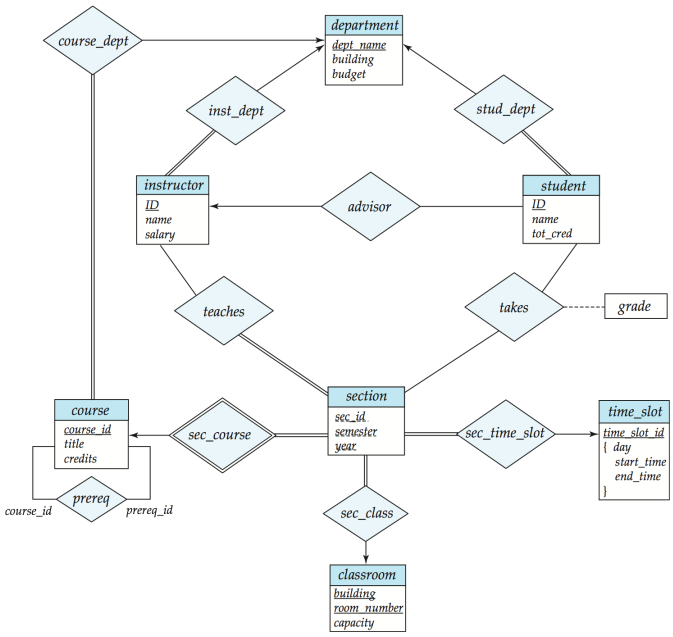
Qualities of Good RDB Design

- Correctly captures requirements of enterprise
- Maintains data integrity and accuracy
  - Referential integrity
- Minimal redundancy
  - Doesn't store the same data in more than one place
  - Except for foreign keys
- Efficient querying, data updates

- Fundamental design trade-off:
  - Fewer large tables versus more small tables

University Requirements

- Requirements regarding people in the university:
  - instructors are identified by an ID num, and we need to store their name and salary
    - every instructor must also be associated with exactly one department
  - students are identified by an ID num, and we need to store their name and total credits
    - every student must also be associated with exactly one department
  - a student can have at most one instructor as advisor
    - instructors can advise any number of students
- Requirements regarding courses and course scheduling:
  - courses have IDs, titles, and number of credits
    - every course must be associated with exactly one department
  - courses are scheduled into sections
    - a section of a course is identified by a section number, semester, and year
    - a section must be associated with exactly one course, though a course can have multiple sections
    - each section takes place in exactly one classroom (which has a certain capacity), and at exactly one of a pre-determined set of time slots
  - students take a section of a course for a grade
  - each section is taught by one or more instructors
  - some courses have one or more prerequisite courses
- Requirements regarding departments:
  - departments are identified by a name, and we need to store their home building and budget

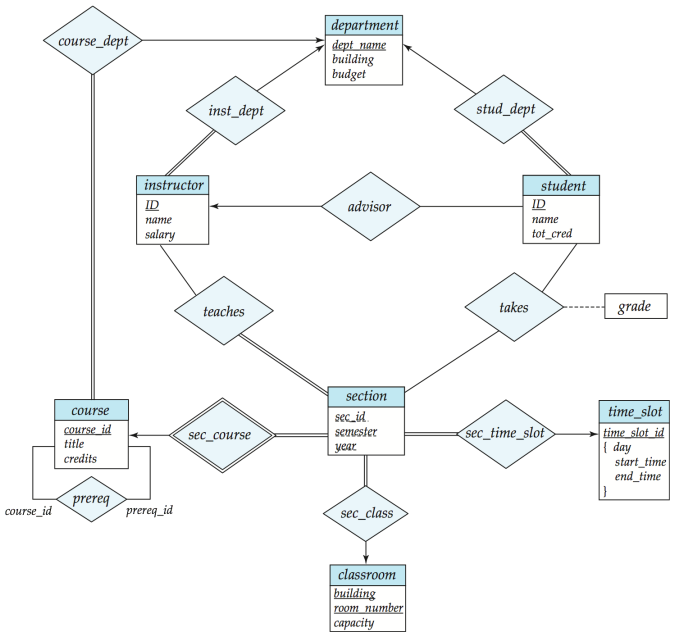


 8

Translating an E-R Diagram to Relational Schema

- Basic steps:
  1. Define a relation for each entity set
     - Primary key of relation = unique identifier of entity
  2. Define a relation for each relationship set
     - Primary key of relation = tuple of unique identifiers of entity sets involved
     - Define foreign keys to relations representing entity sets
  3. Modify keys of relations defined in step 2 to capture mapping constraints as necessary
  4. Eliminate identical relations
     - Relations that have exactly the same fields and keys
  5. Merge relations with the same keys to capture participation constraints as necessary
  6. Check normal forms and normalize relations as needed

10



11

1. Define Relations for Each Entity Set

student(ID, name, tot_cred)
instructor(ID, name, salary)
course(course_id, title, credits)
department(dept_name, building, budget)
classroom(building, room_no, capacity)

time_slot(time_slot_id, day, start_time, end_time)

section(course_id, sec_id, semester, year)

2. Define Relations for Each Relationship Set

stud_dept(ID, dept_name)
inst_dept(ID, dept_name)
course_dept(course_id, dept_name)
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_no)

takes(ID, course_id, sec_id, semester, year, grade)
teaches(ID, course_id, sec_id, semester, year)

prereq(course_id, prereq_id)

3. Modify Keys to Capture Mapping Constraints

- Simple rules:
  - Many-to-many relationships: no change to keys
    - takes(ID, course_id, sec_id, semester, year, grade)
    - teaches(ID, course_id, sec_id, semester, year)
    - prereq(course_id, prereq_id)
  - Many-to-one relationships: primary key is the key of the entity set on the "many" side
    - stud_dept(ID, dept_name)
    - inst_dept(ID, dept_name)
    - course_dept(course_id, dept_name)
    - advisor(s_ID, i_ID)
    - sec_time_slot(course_id, sec_id, semester, year, time_slot_id)
    - sec_class(course_id, sec_id, semester, year, building, room_no)
  - One-to-one relationships: pick one of the entity sets to give the primary key; define the other fields UNIQUE

4. Eliminate Identical Relations

- At this point we have two different relations representing sections
  - section(course_id, sec_id, semester, year)
  - sec_course(course_id, sec_id, semester, year)
- These are identical—we only need one
  - Keep section(course_id, sec_id, semester, year)

5. Merge Relations to Capture Participation Constraints

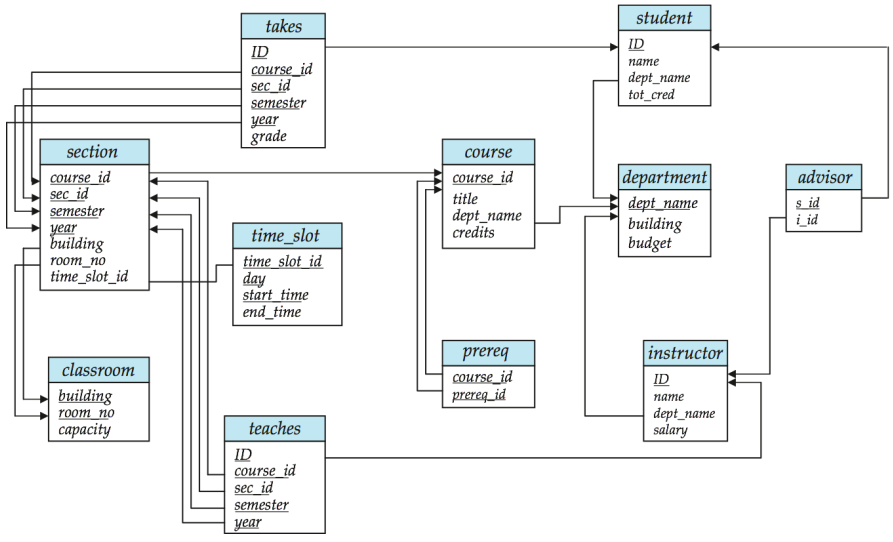
- First look for relations with the same primary key
  - student(ID, name, tot_cred) & advisor(s_ID, i_ID)
  - student(ID, name, tot_cred) & stud_dept(ID, dept_name)
  - takes(ID, course_id, sec_id, semester, year, grade) & teaches(ID, course_id, sec_id, semester, year)
  - section(course_id, sec_id, semester, year) & sec_time_slot(course_id, sec_id, semester, year, time_slot_id) & sec_class(course_id, sec_id, semester, year, building, room_no)

5. Merge Relations to Capture Participation Constraints

- Then look at participation constraints on E-R diagram:
  - No relationship: do nothing
    - No relationship between takes(ID, course_id, sec_id, semester, year, grade) & teaches(ID, course_id, sec_id, semester, year)
  - Partial participation: no change
    - student(ID, name, tot_cred) & advisor(s_ID, i_ID)
  - Total participation: merge relations and define foreign key NOT NULL
    - student(ID, name, tot_cred); stud_dept(ID, dept_name)
      → student(ID, name, tot_cred, dept_name)
    - instructor(ID, name, salary); inst_dept(ID, dept_name)
      → instructor(ID, name, salary, dept_name)
    - course(course_id, title, credits); course_dept(ID, dept_name)
      → course(course_id, title, credits, dept_name)
    - section(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_no)
      → section(course_id, sec_id, semester, year, building, room_no, time_slot_id)

6. Check Normal Forms

- Topic of next week's lectures



Other Things

- Hierarchies/inheritance
  - Define a relation for each entity set in the hierarchy
    - Each relation has primary key = unique identifier of top-level entity set
    - Each relation's primary key is also a foreign key to the relation representing the entity set it inherits from
  - When inheritance is disjoint, you can merge relations for parent and child entity sets

Other Things

- Aggregation
  - Relation representing eval_for relationship set needs to have as primary key the concatenation of primary keys on all four entity sets involved