Reduction to Relational Schemas

Outline

- Relational Algebra (6.1)
- E/R Model (7.2 - 7.4)
- E/R Diagrams (7.5)
- Reduction to Schema (7.6)
- Relational Database Design (7.7)
- Functional Dependencies (8.1 – 8.4)
- Normalization (8.5 – 8.7)
- Relational Query Languages
- SQL Basics
- Formal Semantics of SQL
ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now

- Schema per relationship set

lots of foreign key dependences (weak, relationships...)

department(dept_name, building, budget)
instructor(ID, name, salary)
course(course_id, title, credits)
section(sec_id, course_id, semester, semester, year)
student(ID, name, tot_cred)
classroom(building, room_number, capacity)
time_slot(time_slot_id, {time_slot, day, start_time, end_time})
inst_dept
stud_dept
teaches
takes
advisor(s_id)
course_dept(course_id, dept_name)
sec_time_slot(course_id, sec_id, semester, year, time_slot_id)
sec_course(mess)
prereq(course_id, prereq_id)
sec_class(course_id, sec_id, semester, year, building, room_number)

lots of foreign key dependences (weak, relationships...)

department(dept_name, building, budget)
instructor(ID, name, salary)
course(course_id, title, credits)
section(sec_id, course_id, semester, semester, year)
student(ID, name, tot_cred)
classroom(building, room_number, capacity)
time_slot(time_slot_id, {day, start_time, end_time})
inst_dept
stud_dept
teaches
takes
advisor(s_id)
course_dept(course_id, dept_name)
sec_time_slot(course_id, sec_id, semester, year, time_slot_id)
sec_course(mess)
prereq(course_id, prereq_id)
sec_class(course_id, sec_id, semester, year, building, room_number)

lots of foreign key dependences (weak, relationships...)
ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now

- Schema per relationship set
  - lots of foreign key dependences (weak, relationships...)

```sql
CREATE TABLE department (dept_name, building, budget);
CREATE TABLE instructor (ID, name, salary);
CREATE TABLE course (course_id, title, credits);
CREATE TABLE section (sec_id, course_id, semester, year);
CREATE TABLE student (ID, name, tot_cred);
CREATE TABLE classroom (building, room_number, capacity);
CREATE TABLE time_slot (time_slot_id, day, start_time, end_time);
CREATE TABLE inst_dept (ID, dept_name);
CREATE TABLE study_dept (ID, dept_name);
CREATE TABLE teaches (ID, sec_id, semester, course_id, year);
CREATE TABLE takes (ID, course_id, sec_id, semester, year, grade);
CREATE TABLE advisor (i_id, s_id);
CREATE TABLE course_dept (course_id, dept_name);
CREATE TABLE sec_time_slot (course_id, sec_id, semester, year, time_slot_id);
CREATE TABLE sec_course (course_id, sec_id, semester, year);
CREATE TABLE prereq (course_id, prereq_id);
CREATE TABLE sec_class (course_id, sec_id, semester, building, room_number);
```

lots of foreign key dependences (weak, relationships...)
ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set

lots of foreign key dependences (weak, relationships..)

department(dept_name, building, budget)
instructor(ID, dept_name, name, salary)
course(course_id, title, credits)
section(sec_id, course_id, semester, semester, year)
student(ID, name, tot_cred)
classroom(building, room_number, capacity)
time_slot(time_slot_id, day, start_time, end_time)

lots of foreign key dependences (weak, relationships..)
ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now

- Schema per relationship set

lots of foreign key dependences (weak, relationships...)

---

ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now

- Schema per relationship set

lots of foreign key dependences (weak, relationships...)

---

department(dept_name, building, budget)

instructor(id, dept_name, name, salary)

course(course_id, title, credits, dept_name)

section(sec_id, course_id, semester, year)

student(id, dept_name, name, tot_cred)

classroom(building, room_number, capacity)

time_slot(time_slot_id, day, start_time, end_time)

inst_dept(id, dept_name)

stud_dept(id, dept_name)

teaches(id, sec_id, semester, course_id, year)

takes(id, course_id, sec_id, semester, year, grade)

advisor(i_id, s_id)

course_dept(course_id, dept_name)

sec_time_slot(course_id, sec_id, semester, year, time_slot_id)

sec_course(course_id, sec_id, semester, year)

prereq(course_id, prereq_id)

sec_class(course_id, sec_id, semester, year, building, room_number)

lots of foreign key dependences (weak, relationships...)
ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set

Design Issues

- **Binary versus n-ary relationship sets**
  Although it is possible to replace any nonbinary (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.

- **Placement of relationship attributes**
  e.g., attribute *date* as attribute of *advisor* or as attribute of *student*
Binary Vs. Non-Binary Relationships

- Some relationships that appear to be non-binary may be better represented using binary relationships
  - E.g., A ternary relationship \textit{parents}, relating a child to his/her father and mother, is best replaced by two binary relationships, \textit{father} and \textit{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: \textit{proj\_group}, with several project members

Converting Non-Binary Relationships to Binary Form

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Create a special identifying attribute for \textit{E}
  - Add any attributes of \textit{R} to \textit{E}
  - For each relationship \((a_i, b_i, c_i)\) in \textit{R}
    - create a new entity \(e_i\) in the entity set \textit{E}
    - add \((e_i, a_i)\) to \textit{R}_A, etc.
Where did we come up with the schema that we used?

- E.g. why not store the actor names with movies?

If from an E-R diagram, then:

- Did we make the right decisions with the E-R diagram?

Goals:

- Formal definition of what it means to be a “good” schema.
- How to achieve it.
Relational Schemas and Redundancy

- movies(name, year, len)
- stars(name, addr, gender, birthdate)
- execs(name, cert#)
- studios(stud_name, address)

- in(star_name, movie_name, movie_year)
- made_by(movie_name, movie_year)
- produced_by(movie_name, movie_year, cert#)
- helmed_by(cert#, stud_name)
Relational Schemas and Redundancy

- movies(name, year, len)
- stars(name, addr, gender, birthdate)
- execs(name, cert#)
- studios(stud_name, address, pres#)

- in(star_name, movie_name, movie_year)
- made_by(movie_name, movie_year)
- produced_by(movie_name, movie_year, cert#)
Relational Schemas and Redundancy

- movies(name, year, len, prod#)
- stars(name, addr, gender, birthdate)
- execs(name, cert#)
- studios(stud_name, address, pres#)

- in(star_name, movie_name, movie_year)
- made_by(movie_name, movie_year)

Relational Schemas and Redundancy

- movies(name, year, len, prod#, studio_name)
- stars(name, addr, gender, birthdate)
- execs(name, cert#)
- studios(stud_name, address, pres#)

- in(star_name, movie_name, movie_year)
- made_by(movie_name, movie_year)
Relational Schemas and Redundancy

- movies(name, year, len, prod#, studio_name)
- stars(name, addr, gender, birthdate)
- execs(name, cert#)
- studios(stud_name, address, pres#)

- in(star_name, movie_name, movie_year)