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)
Reduction to Relation Schemas

- Entity sets and relationship sets can be expressed uniformly as *relation schemas* that represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of schemas.
- For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding entity set or relationship set.
- Each schema has a number of columns (generally corresponding to attributes), which have unique names.

Representing Entity Sets With Simple Attributes

- A strong entity set reduces to a schema with the same attributes `student(ID, name, tot_cred)`
- A weak entity set becomes a table that includes a foreign key for the primary key of the identifying strong entity set `section (course_id, sec_id, sem, year)`
Representing 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:

  \( \text{advisor}(s\_id, i\_id) \)

Redundancy 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(s) to the “many” side, containing the primary key of the “one” side

• Example:
  • get rid of inst_dept by: \( \text{instructor}(ID, \text{name}, \text{salary}) \rightarrow \text{instructor}(ID, \text{dept}\_\text{name}, \text{name}, \text{salary}) \)
Redundancy of Schemas (Cont.)

• For one-to-one relationship sets, either side can be chosen to act as the “many” side
  • That is, extra attribute can be added to either of the tables corresponding to the two entity sets

\[
\begin{align*}
\text{Instructor} & \quad | \quad \text{department} \\
\text{ID} & \rightarrow \text{inst_dept} & \text{dept_name} & \rightarrow \\
\text{name} & & \text{building} & \\
\text{salary} & & \text{budget} & \\
\end{align*}
\]

\[
\text{instructor}(\text{ID, dept_name, name, salary})
\]

or

\[
\text{department}(\text{dept_name, ID, building, budget})
\]

Redundancy of Schemas (Cont.)

• If participation is partial on the “many” side, replacing a relationship schema by an extra attribute in the schema corresponding to the “many” side could result in null values (generally avoided)
  • i.e. the approach in the previous slides does not work
  • need to represent relationship as a separate table

• Relationship set linking a weak entity set to its identifying strong entity set is redundant.
  • Example: The section schema already contains the attributes that would appear in the sec_course schema

• Unless otherwise instructed, assume we wish to avoid NULLs when converting to relations, i.e. remove redundant relationship schemas only when total participation on side where adding attribute.
**Composite Attributes**

• Composite attributes flattened out
  • Example: given entity set *instructor*
    • with composite attribute *name*
    • with component attributes *first_name* and *last_name*
    • replace with *name_first_name* and *name_last_name*
  • Prefix omitted if there is no ambiguity

• Ignoring multivalued attributes, extended instructor schema is
  • *instructor*(ID, first_name, middle_initial, last_name, street, street_number, street_name, apt_number, city, state, zip, { phone_number }, date_of_birth, age ( )

**Multivalued Attributes**

• Multivalued attribute *M* of entity *E* represented by a separate schema *EM*
  • Schema *EM* includes *E*’s primary key and attribute corresponding to *M*
  • Example: Multivalued attribute *phone_number* of *instructor*:
    • *inst_phone*= ( ID, phone_number)

  • Each value of the multivalued attribute maps to separate tuple of *EM*:
    • *instructor* entity with primary key 22222 and numbers 456-7890 and 123-4567 maps to:
      • (22222, 456-7890)
      • (22222, 123-4567)
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)
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

```
• department(dept_name, building, budget)
• instructor(iD, name, salary)
• course(course_id, title, credits)
• section
• student(sID, 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(_id, 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..)

department(dept_name, building, budget)
instructor(ID, name, salary)
course(course_id, title, credits)
section(sec_id, course_id, semester, year)
student(ID, 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(_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

```
department(dept_name, building, budget)
instructor(ID, dept_name, name, salary)
course(course_id, title, credits)
section(sec_id, course_id, 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

```
department(dept_name, building, budget)
instructor(ID, dept_name, name, salary)
course(course_id, title, credits)
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)
```

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

```plaintext
> 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)
```

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

```plaintext
> 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(_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

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

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 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_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.
  
  • Replace $R$ between entity sets $A$, $B$ and $C$ by an entity set $E$, and $R_A$, $R_B$, $R_C$, relating $E$ with $A$, $B$, and $C$
  
  • Create a special identifying attribute for $E$
  
  • Add any attributes of $R$ to $E$
  
  • For each relationship $(a_i, b_i, c_i)$ in $R$
    - create a new entity $e_i$ in the entity set $E$
    - add $(e_i, a_i)$ to $R_A$, etc.
A Movie Industry Schema

Relational Database Design

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, 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)
- produced_by(movie_name, movie_year, cert#)
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, star_name)
- stars(name, addr, gender, birthdate)
- execs(name, cert#)
- studios(stud_name, address, pres#)

Is this a good idea???

Relational Database Design

or

“Troubles With 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)

---

Movie(title, year, length, inColor, studioName, producerC#, starName)

<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
<th>Length</th>
<th>inColor</th>
<th>StudioName</th>
<th>prodC#</th>
<th>StarName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star wars</td>
<td>1977</td>
<td>121</td>
<td>Yes</td>
<td>Fox</td>
<td>128</td>
<td>Hamill</td>
</tr>
<tr>
<td>Star wars</td>
<td>1977</td>
<td>121</td>
<td>Yes</td>
<td>Fox</td>
<td>128</td>
<td>Fisher</td>
</tr>
<tr>
<td>Star wars</td>
<td>1977</td>
<td>121</td>
<td>Yes</td>
<td>Fox</td>
<td>128</td>
<td>H. Ford</td>
</tr>
<tr>
<td>King Kong</td>
<td>2005</td>
<td>187</td>
<td>Yes</td>
<td>Universal</td>
<td>150</td>
<td>Watts</td>
</tr>
<tr>
<td>King Kong</td>
<td>1933</td>
<td>100</td>
<td>no</td>
<td>RKO</td>
<td>20</td>
<td>Fay</td>
</tr>
</tbody>
</table>

**Issues:**

1. Redundancy ➔ higher storage, inconsistencies (“anomalies”)
   
   *update anomalies, insertion anomalies*

2. Need nulls
   
   Unable to represent some information without using nulls

   *How to store movies w/o actors (pre-productions etc)*?