Relational Database Design using E/R

Introduction to Databases
CompSci 316 Spring 2020
Announcements (Thu. Jan. 16)

All on course website schedule

• Reminder: HW1 due next Tuesday 01/21

• HW2 on RA to be posted next Tuesday 01/21, due on 01/28
  • HW2-Q1 on gradiance already open if you want to start early
  • Check gradiance code on Sakai announcements

• In-class lab on RA next Tuesday 01/21
  • Part of HW2 (~ 2 questions) in class to get the set up ready with TAs help
  • Last 30-40 mins of Tuesday’s lecture
  • You can work in groups of size 2 or 3, but would submit your own solution
  • You can submit by the next day -- 10% extra credit for finishing all questions correctly in class (last timestamp <= 4:20 pm)!

• In-class quiz next Thursday 01/23
  • You can work in groups of size 2 or 3, but would submit your own solution
  • 50% for attempt, 50% for correct answer
  • What if you miss a class? We would drop 25% (ceiling) of the lowest grades while calculating your final score for quiz, i.e. if we have 4 quizzes 1 dropped, 5-8 quizzes 2 dropped, ...

• Quiz or Lab -- you can submit while not being in the class too, but you would miss the fun of discussing with others (+ help from TAs for Labs)!
Announcements (Tue. Jan. 21)

• Reminder: HW1 due tonight 01/21, Tuesday, 11:59 pm
  • Later 5% penalty per hour
  • Do not forget collaboration.txt
  • Still have two OH: 5-7 LSRC D301 (this week only), 7-9 Soc Psych 128

• HW2 on RA to be posted today Tuesday 01/21, due on 01/28
  • HW2-Q1 on gradiance already open if you want to start early
  • Check gradiance code on Sakai announcements

• In-class lab on RA today
  • would start around 3:30 pm

• In-class quiz Thursday 01/23
  • See announcement in last lecture or on Sakai for details

• Next Tuesday 01/28: Project mixer
  • Project details to be published soon
  • Presentation by your UTA Jane Li on their project from last semester
  • Guest lecture by Danai Adkisson from Duke OIT colab on Web/app development, Flask, and what help you can receive
  • If we have more time, we would have group discussions in small groups (5-10) in 1-2 rounds on project ideas among yourselves

• 316 textbooks available through Duke Libraries
  • see Sakai announcement
Relational model: review

- A database is a collection of relations (or tables)
- Each relation has a set of attributes (or columns)
- Each attribute has a name and a domain (or type)
- Each relation contains a set of tuples (or rows)

How do we know which relations and attributes to have?
Example: Users, Groups, Members

**Users**
Each has `uid` (unique id), name, age, pop (popularity)

**Groups**
Each has `gid` (unique id), name

**Member**
Records `fromDate` (when a user joined a group)
Keys

• A set of attributes $K$ is a **key** for a relation $R$ if
  • In no instance of $R$ will two different tuples agree on all attributes of $K$
    • That is, $K$ can serve as a “tuple identifier”
  • No proper subset of $K$ satisfies the above condition
    • That is, $K$ is **minimal**

• Example: **User** ($uid$, $name$, $age$, $pop$)
  • $uid$ is a key of **User**
  • $age$ is not a key (not an identifier)
  • {$uid$, $name$} is not a key (not minimal)
Schema vs. instance

- Is *name* a key of *User*?
  - Yes? Seems reasonable for this instance
  - No! User names are not unique in general

- Key declarations are part of the schema

<table>
<thead>
<tr>
<th>uid</th>
<th>name</th>
<th>age</th>
<th>pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>142</td>
<td>Bart</td>
<td>10</td>
<td>0.9</td>
</tr>
<tr>
<td>123</td>
<td>Milhouse</td>
<td>10</td>
<td>0.2</td>
</tr>
<tr>
<td>857</td>
<td>Lisa</td>
<td>8</td>
<td>0.7</td>
</tr>
<tr>
<td>456</td>
<td>Ralph</td>
<td>8</td>
<td>0.3</td>
</tr>
</tbody>
</table>
More examples of keys

• Member (uid, gid)
  • {uid, gid}
    ➔ A key can contain multiple attributes

• Address (street_address, city, state, zip)
  • {street_address, city, state}
  • {street_address, zip}
    ➔ A relation can have multiple keys!
    • We typically pick one as the “primary” key, and underline all its attributes, e.g., Address (street_address, city, state, zip)
Use of keys

• More constraints on data, fewer mistakes
• Look up a row by its key value
  • Many selection conditions are “key = value”
• “Pointers” to other rows (often across tables)
  • Example: Member (uid, gid)
    • uid is a key of User
    • gid is a key of Group
    • A Member row “links” a User row with a Group row
  • Many join conditions are “key = key value stored in another table”
Database design

• Understand the real-world domain being modeled
• Specify it using a database design model
  • More intuitive and convenient for schema design
  • But not necessarily implemented by DBMS
• We will cover
  • Entity/Relationship (E/R) model

• Then
  1. Translate specification to the data model of DBMS
     • Relational, XML, object-oriented, etc.
  2. Create DBMS schema
Entity-relationship (E/R) model

• Historically and still very popular

• Designs represented by E/R diagrams
  • We use the style of E/R diagram covered by the GMUW book; there are other styles/extensions
Start of Lecture-5
Announcements (Tue. Jan. 21)

• HW2 and Lab1 deadline extended by two days Thursday 1/30, 11:59 pm
  • Deadlines will be updated
• Questions on RA syntax?
  • Short RA tutorial in all office hours starting tomorrow
  • Extra office hour by Yuchao 1:30-2:30 pm tomorrow (Fri) – see Piazza announcement later for the room info
• Suggestion:
  • You may want to solve the queries on paper first before trying on autograder/RATest with correct syntax
• Next Tuesday 01/28: Project mixer
  • Presentation by your UTA Jane Li on their project from last semester
  • Guest lecture by Danai Adkisson from Duke Colab on Web/app development, Flask, and what help you can receive from them
  • If we have more time, we would have group discussions in small groups (5-10) in 1-2 rounds on project ideas among yourselves
Example: Users, Groups, Members

**Users**
Each has **uid** (unique id), name, age, pop (popularity)

**Groups**
Each has **gid** (unique id), name

**Member**
Records **fromDate**
(when a user joined a group)
E/R basics

- **Entity**: a “thing,” like an object
- **Entity set**: a collection of things of the same type, like a relation of tuples or a class of objects
  - Represented as a rectangle
- **Relationship**: an association among entities
- **Relationship set**: a set of relationships of the same type (among same entity sets)
  - Represented as a diamond
- **Attributes**: properties of entities or relationships, like attributes of tuples or objects
  - Represented as ovals
An example E/R diagram

• Users are members of groups

• A **key** of an entity set is represented by underlining all attributes in the key
  • A key is a set of attributes whose values can belong to at most one entity in an entity set—like a key of a relation
Attributes of relationships

• Example: a user belongs to a group since a particular date

• Where do the dates go?
  • With Users?
    • But a user can join multiple groups on different dates
  • With Groups?
    • But different users can join the same group on different dates
  • With IsMemberOf!
E/R diagram for Beers Database?

- **Drinkers** are frequent bars “X” times a week.
- **Bars** each have an address.
- **Bars** serve beers at price “Y”.
- **Drinkers** like beers.
- **Beers** each have a brewer.

Keys?
More on relationships

• There could be multiple relationship sets between the same entity sets
  • Example: *Users IsMemberOf Groups; Users Likes Groups*

• In a relationship set, each relationship is uniquely identified by the entities it connects
  • Example: *Between Bart and “Dead Putting Society”, there can be at most one IsMemberOf relationship and at most one Likes relationship*

☞ What if Bart joins DPS, leaves, and rejoins? How can we modify the design to capture historical membership information?

☞ Make an entity set of *MembershipRecords*
Multiplicity of relationships

- \( E \) and \( F \): entity sets
- **Many-many**: Each entity in \( E \) is related to 0 or more entities in \( F \) and vice versa
  - Example:

- **Many-one**: Each entity in \( E \) is related to 0 or 1 entity in \( F \), but each entity in \( F \) is related to 0 or more in \( E \)
  - Example:

- **One-one**: Each entity in \( E \) is related to 0 or 1 entity in \( F \) and vice versa
  - Example:

- “One” (0 or 1) is represented by an arrow
- “Exactly one” is represented by a rounded arrow
Roles in relationships

• How do we model “Friendship” among Users?
• An entity set may participate more than once in a relationship set
  ❯ May need to label edges to distinguish roles
• Examples
  • Users may be parents of others; label needed
  • Users may be friends of each other; label not needed
\( n \)-ary relationships

- Example: a user must have an initiator in order to join a group

Rule for interpreting an arrow into entity set \( E \) in an \( n \)-ary relationship:

- Pick one entity from each of the other entity sets; together they can be related to at most one entity in \( E \)

- Exercise: hypothetically, what do these arrows imply?
n-ary versus binary relationships

• Can we model n-ary relationships using just binary relationships?

No; for example:
• Ralph is in both abc and gov
• Lisa has served as initiator in both abc and gov
• Ralph was initiated by Lisa in abc, but not by her in gov
Next: two special relationships

... is part of/belongs to ...

... is a kind of ...

http://blogs.library.duke.edu/renovation/files/2012/08/Rubenstein-Library-First-Floor-Floorplan.jpg
http://www.sharky-jones.com/Sharkyjones/Artwork/taxonomy%20artwork/Class1.jpg
Weak entity sets

Sometimes, an entity’s identity depends on some others’

Can you come to my OH in 325?

D wing

LSRC

Sorry 325 in..?

D-wing of...?

Got it

LSRC
Weak entity sets

Sometimes, an entity’s identity depends on some others’

- The key of a weak entity set \( E \) comes not completely from its own attributes, but from the keys of one or more other entity sets
  - \( E \) must link to them via many-one or one-one relationship sets

- Example: *Rooms inside Buildings are partly identified by Buildings’ name*

- A weak entity set is drawn as a double rectangle

- The relationship sets through which it obtains its key are called *supporting relationship sets*, drawn as double diamonds
Weak entity set examples

- Seats in rooms in building

• Why must double diamonds be many-one/one-one?
  • With many-many, we would not know which entity provides the key value!
Remodeling $n$-ary relationships

- An $n$-ary relationship set can be replaced by a weak entity set (called a connecting entity set) and $n$ binary relationship sets.

Are they equivalent now?

Note that the multiplicity constraint for $IsMemberOf$ is lost.
ISA relationships

• Similar to the idea of subclasses in object-oriented programming: subclass = special case, fewer entities, and possibly more properties
  • Represented as a triangle (direction is important)
• Example: paid users are users, but they also get avatars (yay!)

```
<table>
<thead>
<tr>
<th>Users</th>
<th>Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>uid</td>
<td>gid</td>
</tr>
<tr>
<td>name</td>
<td>name</td>
</tr>
<tr>
<td>IsMemberOf</td>
<td>IsMemberOf</td>
</tr>
<tr>
<td>fromDate</td>
<td>fromDate</td>
</tr>
<tr>
<td>ISA</td>
<td>ISA</td>
</tr>
<tr>
<td>PaidUsers</td>
<td>PaidUsers</td>
</tr>
<tr>
<td>avatar</td>
<td>avatar</td>
</tr>
</tbody>
</table>
```

Automatically “inherits” key, attributes, relationships
Summary of E/R concepts

• Entity sets
  • Keys
  • Weak entity sets

• Relationship sets
  • Attributes of relationships
  • Multiplicity
  • Roles
  • Binary versus n-ary relationships
    • Modeling n-ary relationships with weak entity sets and binary relationships
  • ISA relationships
Case study 1

• Design a database representing cities, counties, and states
  • For states, record name and capital (city)
  • For counties, record name, area, and location (state)
  • For cities, record name, population, and location (county and state)

• Assume the following:
  • Names of states are unique
  • Names of counties are only unique within a state
  • Names of cities are only unique within a county
  • A city is always located in a single county
  • A county is always located in a single state
Start of Lecture-6
(after project mixer)
Announcements (Tue. Jan. 28)

• Reminder: HW2 and Lab1 due Thursday, 1/30, 11:59 pm

• Project team formation
  • See the email sent on sakai and piazza for shared google spreadsheet
  • Each standard project team should have 5 members
  • Open project teams may be more flexible in size based on the work
Case study 1: first design

- County area information is repeated for every city in the county
  - Redundancy is bad (why?)

- State capital should really be a city
  - Should “reference” entities through explicit relationships
Case study 1: second design

- Technically, nothing in this design prevents a city in state $X$ from being the capital of another state $Y$ ...
Case study 2

• Design a database consistent with the following:
  • A station has a unique name and an address, and is either an express station or a local station
  • A train has a unique number and an engineer, and is either an express train or a local train
  • A local train can stop at any station
  • An express train only stops at express stations
  • A train can stop at a station for any number of times during a day
  • Train schedules are the same everyday
Case study 2: first design

• Nothing in this design prevents express trains from stopping at local stations
  ✫ We should capture as many constraints as possible

• A train can stop at a station only once during a day
  ✫ We should not introduce unintended constraints
Case study 2: second design

Is the extra complexity worth it?