Relational Database Design: Part I
Introduction to Databases
CompSci 316 Fall 2015

Announcements (Tue. Sep. 1)

• Homework #1 due in two weeks
  • Get started early on your homework!
• AWS credit codes emailed
  • Those who added after 9pm last night will hear from me tonight
• More details on the course project available next week

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)
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

<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>

- Is name a key of User?

  - Key declarations are part of the schema

---

More examples of keys

- Member (uid, gid)

- Address (street_address, city, state, zip)

  - A relation can have multiple keys!
    - We typically pick one as the "primary" key, and **underline** all its attributes
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
  - A few popular ones:
    - Entity/Relationship (E/R) model
    - Object Definition Language (ODL)
    - UML (Unified Modeling Language)
- Translate specification to the data model of DBMS
  - Relational, XML, object-oriented, etc.
- Create DBMS schema

But what about ORM?

- Automatic object-relational mappers are made popular by rapid Web development frameworks
  - For example, with Python SQLAlchemy:
    - You declare Python classes and their relationships
    - It automatically converts them into database tables
    - If you want, you can just work with Python objects, and never need to be aware of the database schema or write SQL
  - But you still need designer discretion in all but simple cases
  - Each language/library has its own syntax for creating schema and for querying/modifying data
  - Quirks and limitations cause portability problems
  - They are not necessarily easier to learn than SQL
Entity-relationship (E/R) model

- Historically and still very popular
- Concepts applicable to other design models as well
- Can think of as a “watered-down” object-oriented design model
- Primarily a design model—not directly implemented by DBMS
- Designs represented by E/R diagrams
  - We use the style of E/R diagram covered by the GMUW book; there are other styles/extensions
  - Very similar to UML diagrams

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

  ![Diagram showing relationship between Users and Groups]

• Where do the dates go?
  - With Users?
  - With Groups?

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? Can we capture historical membership information as well?

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

• 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

\[ \text{Users} \xrightarrow{\text{parent}} \text{Parent} \]
\[ \text{Users} \xrightarrow{\text{frnd}} \text{Friend} \]

\( n \)-ary relationships

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

\[ \text{Users} \xrightarrow{\text{Initiator}} \text{Groups} \]

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?

\[ \text{Users} \xrightarrow{\text{member}} \text{Groups} \]

\( n \)-ary versus binary relationships

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

\[ \text{Users} \xrightarrow{\text{member}} \text{Groups} \]

Wrong!
Next: two special relationships

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

... is a kind of ...

Weak entity sets

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

• The key of 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

• \( E \) is called a **weak entity set**, drawn as a double rectangle
  • The relationship sets through which \( E \) 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?
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

```
                      member
                         |
Users --IsMemberOf--> Membership
                      |
                         |
Membership --IsMemberOf--> Groups
```

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!)

```
          PaidUsers
          /    |
    gid    name
          |     |
   avatar
          |     |
    uid   name
```

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

Case study 1: first design

Case study 1: second design

- Technically, nothing in this design prevents a city in state $X$ from being the capital of another state $Y$, but oh well...
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

Case study 2: second design

Is the extra complexity worth it?