Designing Relational Database Models

Daniel Zappala

CS 360 Internet Programming
Brigham Young University
Relational Databases
Relational Databases

- each database contains a collection of tables
  - each row is a unique record
  - each column is an attribute of the record

<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
<th>totalRevenue</th>
<th>budgetedRevenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2002</td>
<td>1.07857e+07</td>
<td>6.08307e+06</td>
</tr>
<tr>
<td>2</td>
<td>2003</td>
<td>7.6935e+06</td>
<td>1.55783e+07</td>
</tr>
<tr>
<td>3</td>
<td>2004</td>
<td>7.20024e+06</td>
<td>6.72688e+06</td>
</tr>
<tr>
<td>4</td>
<td>2005</td>
<td>7.66746e+06</td>
<td>8.8527e+06</td>
</tr>
<tr>
<td>5</td>
<td>2006</td>
<td>9.39535e+06</td>
<td>1.17699e+07</td>
</tr>
<tr>
<td>6</td>
<td>2007</td>
<td>7.97184e+06</td>
<td>1.50126e+07</td>
</tr>
</tbody>
</table>
Attributes

- attributes have data types
- primary key: one or more keys that together uniquely identify each row in a table

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Integer</td>
<td>primary key</td>
</tr>
<tr>
<td>date</td>
<td>Integer</td>
<td>unique</td>
</tr>
<tr>
<td>totalRevenue</td>
<td>Float</td>
<td></td>
</tr>
<tr>
<td>budgetedRevenue</td>
<td>Float</td>
<td></td>
</tr>
<tr>
<td>totalExpenditures</td>
<td>Float</td>
<td></td>
</tr>
<tr>
<td>budgetedExpenditures</td>
<td>Float</td>
<td></td>
</tr>
</tbody>
</table>
## Relationships

- form relationships between tables using identifiers
- **one-to-many, one-to-one** and **many-to-many** relationships

### Fund Table (Citizen Budget)

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>totalRevenue</th>
<th>budgetedRevenue</th>
<th>yearID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GENERAL FUND</td>
<td>2.06581e+06</td>
<td>2.64328e+06</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>CAPITAL PROJECT FUND</td>
<td>7.63334e+06</td>
<td>2.02127e+06</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>WATER AND SEWER FUND</td>
<td>1.08651e+06</td>
<td>1.41852e+06</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>GENERAL FUND</td>
<td>1.76936e+06</td>
<td>1.681e+06</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>CAPITAL PROJECT FUND</td>
<td>4.32644e+06</td>
<td>3.60852e+06</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>WATER AND SEWER FUND</td>
<td>1.53088e+06</td>
<td>1.01848e+07</td>
<td>2</td>
</tr>
</tbody>
</table>
Types of Relationships

• one-to-one
  • exactly one instance of the first entity for each instance of the second entity
  • example: customer has exactly one set of login information

• one-to-many
  • one or more instances of the second entity for each instance of the first entity
  • example: each customer can place more than one order, but each order is made by only one customer

• many-to-many
  • each entity is related to more than one instance of the other entity
  • example: a bowtie can come have more than one category (red, paisley), and a category can have more than one bowtie
Entity Relationship Modeling
Entity-Relationship Model

- visualizes database and its relationships
  - tables (entities): rectangles
  - attributes: ellipses
  - relationship: diamond

- primary key is underlined

- relationship is annotated with an M, showing a one-to-many relationship
Entities and Relationships

- Citizen Budget
  - a year has many funds
  - a fund has many categories
  - a category has many items
- rules: if you delete a fund, you must delete all of its categories
- store example
  - bowties
  - customer
  - order
- purchasing action creates a relationship between customer, order, and bowtie
  - must associate one customer with each order
  - customers can make more than one order
  - each order has one or more bowties
- You should wear a bowtie!
Resolving Many-to-Many Relationships

- A bowtie can come have more than one category (red, paisley), and a category can have more than one bowtie.
Normalization
Designing Tables

- customer table
  - id
  - name
  - address

- order table
  - id
  - customerID
  - bowtieID
  - quantity

- can only order one type of bowtie in a single order
  - solution: add “bowtieID2”, “bowtieID3”, “quantity2”, “quantity3” to the order table
  - must decide on a maximum number of bowties per order (horrors!)
  - must decide on empty values if an order has fewer than this
Normalization

- better solution: store the items that make up an order
- **items** table
  - id
  - price
  - quantity
  - bowtieID
  - orderId
- when do you add a table versus or more attributes?
  - normalize the database according to a set of rules
  - [MySQL article](#)
Example
Example

- CitizenBudget models
- uses SQLAlchemy