CMSC 424 – Database design
Lecture 5:
Relational Model
Queries
SQL...maybe

Book: Chap. 2
Chap. 3...maybe

Mihai Pop
Logistics

• Homework...NOW
• No office hours tomorrow!
• Oracle accounts... have you tried them?
• SQL assignment will be issued Thursday
  – part 1 due Tuesday
  – part 2 due a week from Tuesday (February 26)
  – NO EXTENSIONS
Method 1:

\[ E = (a_1, \ldots, a_n) \]
\[ E_1 = (a_1, b_1, \ldots, b_m) \]
\[ E_2 = (a_1, c_1, \ldots, c_k) \]
Subclasses

Method 1:
\[ E = (a_1, \ldots, a_n) \]
\[ E_1 = (a_1, b_1, \ldots, b_m) \]
\[ E_2 = (a_1, c_1, \ldots, c_k) \]

Method 2:
\[ E_1 = (a_1, \ldots, a_n, b_1, \ldots, b_m) \]
\[ E_2 = (a_1, \ldots, a_n, c_1, \ldots, c_k) \]
Subclasses example:

Method 1:

Account = (acct_no, balance)
SAccount = (acct_no, interest)
CAccount = (acct_no, overdraft)

Method 2:

SAccount = (acct_no, balance, interest)
CAccount = (acct_no, balance, overdraft)

Q: When is method 2 not possible?

A: When subclassing is partial
Keys and Relations

• Recall:
  – Keys: Sets of attributes that allow us to identify entities
  – Very loosely speaking, tuples === entities

• Just as in E/R Model:
  – Superkeys, candidate keys, and primary keys
Keys

- **Superkey**
  - set of attributes of table for which every row has distinct set of values

- **Candidate key**
  - Minimal such set of attributes

- **Primary key**
  - DB Chosen Candidate key
  - Plays a very important role
    - E.g. relations typically sorted by this
Keys

• Also act as integrity constraints
  – i.e., guard against illegal/invalid instance of given schema

e.g., Branch = (bname, bcity, assets)

<table>
<thead>
<tr>
<th>bname</th>
<th>bcity</th>
<th>assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brighton</td>
<td>Brooklyn</td>
<td>5M</td>
</tr>
<tr>
<td>Brighton</td>
<td>Boston</td>
<td>3M</td>
</tr>
</tbody>
</table>

Invalid
Keys

• In fact, keys are one of the primary ways to enforce constraints/structure
• Consider a one-to-many relationship e.g.
  – Between customers and accounts
  – The relational model will be:
    • Customers(\texttt{custid}, custname,…)
    • Accounts(\texttt{accountid}, custid, balance,…)
  – Allows for multiple accounts per customer, but not multiple customers per account
    • Not possible to store such information
• In other words, constraints will lead to less representation power
  – Contrast with:
    • Customers(\texttt{custid}, custname,…)
    • Accounts(\texttt{accountid}, balance,…)
    • CustomerHasAccounts(\texttt{custid}, \texttt{accountid})
More on Keys

• Determining Primary Keys
  – If relation schema derived from E-R diagrams, we can determine the primary keys using the original entity and relationship sets
  – Otherwise, same way we do it for E-R diagrams
    • Find candidate keys (minimal sets of attributes that can uniquely identify a tuple)
    • Designate one of them to be primary key

• Foreign Keys
  – If a relation schema includes the primary key of another relation schema, that attribute is called the foreign key
Schema Diagram for the Banking Enterprise
Next

- Query language for operating on the relations
- Theoretical:
  - Relational Algebra
  - Tuple Relational Calculus
  - Domain Relational
- Practical:
  - SQL (loosely based on TRC)
  - Datalog
Next…

• Query language for operating on the relations
• Theoretical:
  – Relational Algebra
  – Tuple Relational Calculus
  – Domain Relational
• Practical:
  – SQL (loosely based on TRC)
  – Datalog
### Account

<table>
<thead>
<tr>
<th>bname</th>
<th>acct_no</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>Mianus</td>
<td>A-215</td>
<td>700</td>
</tr>
<tr>
<td>Perry</td>
<td>A-102</td>
<td>400</td>
</tr>
<tr>
<td>R.H.</td>
<td>A-305</td>
<td>350</td>
</tr>
</tbody>
</table>

### Branch

<table>
<thead>
<tr>
<th>bname</th>
<th>bcity</th>
<th>assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>Brooklyn</td>
<td>9M</td>
</tr>
<tr>
<td>Redwood</td>
<td>Palo Alto</td>
<td>2.1M</td>
</tr>
<tr>
<td>Perry</td>
<td>Horseneck</td>
<td>1.7M</td>
</tr>
<tr>
<td>Mianus</td>
<td>R.H.</td>
<td>0.4M</td>
</tr>
</tbody>
</table>

### Depositor

<table>
<thead>
<tr>
<th>cname</th>
<th>acct_no</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnson</td>
<td>A-101</td>
</tr>
<tr>
<td>Smith</td>
<td>A-215</td>
</tr>
<tr>
<td>Hayes</td>
<td>A-102</td>
</tr>
<tr>
<td>Turner</td>
<td>A-305</td>
</tr>
</tbody>
</table>

### Borrower

<table>
<thead>
<tr>
<th>cname</th>
<th>lno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jones</td>
<td>L-17</td>
</tr>
<tr>
<td>Smith</td>
<td>L-23</td>
</tr>
<tr>
<td>Hayes</td>
<td>L-15</td>
</tr>
<tr>
<td>Jackson</td>
<td>L-14</td>
</tr>
</tbody>
</table>

### Customer

<table>
<thead>
<tr>
<th>cname</th>
<th>cstreet</th>
<th>ccity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>Hayes</td>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>Curry</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>Lindsay</td>
<td>Park</td>
<td>Pittsfield</td>
</tr>
<tr>
<td>Turner</td>
<td>Putnam</td>
<td>Stanford</td>
</tr>
</tbody>
</table>

### Loan

<table>
<thead>
<tr>
<th>bname</th>
<th>lno</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>L-17</td>
<td>1000</td>
</tr>
<tr>
<td>Redwood</td>
<td>L-23</td>
<td>2000</td>
</tr>
<tr>
<td>Perry</td>
<td>L-15</td>
<td>1500</td>
</tr>
<tr>
<td>Downtown</td>
<td>L-14</td>
<td>1500</td>
</tr>
<tr>
<td>Mianus</td>
<td>L-93</td>
<td>500</td>
</tr>
<tr>
<td>R.H.</td>
<td>L-11</td>
<td>900</td>
</tr>
<tr>
<td>Perry</td>
<td>L-16</td>
<td>1300</td>
</tr>
</tbody>
</table>
Example Queries

Find the names of all customers who have a loan at the Perryridge branch.

\[ \Pi_{\text{customer\_name}} (\sigma_{\text{branch\_name}=\text{“Perryridge”}} (\sigma_{\text{borrower.loan\_number} = \text{loan.loan\_number} (\text{borrower x loan}))) ) \]

Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank.

\[ \Pi_{\text{customer\_name}} (\sigma_{\text{branch\_name}=\text{“Perryridge”}} (\sigma_{\text{borrower.loan\_number} = \text{loan.loan\_number} (\text{borrower x loan}))) - \Pi_{\text{customer\_name}}(\text{depositor}) ) \]
Example Queries

Find the names of all customers who have a loan at the Perryridge branch.

- Query 1
  \[\Pi_{\text{customer\_name}} (\sigma_{\text{branch\_name} = "Perryridge"} (\sigma_{\text{borrower\_loan\_number} = \text{loan\_loan\_number}} (\text{borrower} \times \text{loan})))\]

- Query 2
  \[\Pi_{\text{customer\_name}} (\sigma_{\text{loan\_loan\_number} = \text{borrower\_loan\_number}} (\sigma_{\text{branch\_name} = "Perryridge"} (\text{loan}) \times \text{borrower})))\]
Example Queries

Find the largest account balance

**Strategy:**

- Find those balances that are *not* the largest
  - Rename `account` relation as `d` so that we can compare each account balance with all others
- Use set difference to find those account balances that were *not* found in the earlier step.

*The query is:*

\[
\Pi_{\text{balance}}(\text{account}) - \Pi_{\text{account.balance}} \\
(\sigma_{\text{account.balance} < d.\text{balance}} (\text{account} \times \rho_d (\text{account})))
\]