Today …

- Relational Algebra (wrap up)
- Normalization (intro)

Homework

- HW 2 (out)
### Example Tables

#### Branch

<table>
<thead>
<tr>
<th>branch_name</th>
<th>address</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>906 Main</td>
<td>444-5300</td>
</tr>
<tr>
<td>South Hill</td>
<td>3324 Perry</td>
<td>444-5301</td>
</tr>
</tbody>
</table>

#### Account

<table>
<thead>
<tr>
<th>acct_id</th>
<th>acct_name</th>
<th>main_branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Alice</td>
<td>Downtown</td>
</tr>
<tr>
<td>102</td>
<td>Bob</td>
<td>Downtown</td>
</tr>
<tr>
<td>103</td>
<td>Alice</td>
<td>South Hill</td>
</tr>
<tr>
<td>104</td>
<td>Chuck</td>
<td>Downtown</td>
</tr>
</tbody>
</table>

#### Loan

<table>
<thead>
<tr>
<th>acct_id</th>
<th>barcode</th>
<th>checkout_date</th>
<th>due_date</th>
<th>return_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>4242</td>
<td>8/12</td>
<td>8/26</td>
<td>8/24</td>
</tr>
<tr>
<td>101</td>
<td>4243</td>
<td>8/12</td>
<td>8/19</td>
<td>NULL</td>
</tr>
<tr>
<td>102</td>
<td>4242</td>
<td>8/25</td>
<td>9/7</td>
<td>8/29</td>
</tr>
<tr>
<td>101</td>
<td>4243</td>
<td>7/10</td>
<td>7/17</td>
<td>7/18</td>
</tr>
</tbody>
</table>
Query trees

The query:

\[ \pi_{\text{main\_branch}}(\text{Account} \bowtie_{\text{acct\_id}=\text{acct\_id}} (\sigma_{\text{return\_date} \geq \text{due\_date}}(\text{Loan}))) \]

Can also be draw as a “query tree” (i.e., a dataflow pipeline)

where ...

- relations are leaf nodes
- operators are internal, non-leaf nodes
- children outputs are inputs to parents (relations are their own outputs)
Relationships among operations

Some examples of relational algebra equivalences (not comprehensive)

1. \( R \bowtie_C S = S \bowtie_C R \)  
   ... commutative, also: \( \cup, \cap, \times \)

2. \( R \cap S = R - (R - S) \)

3. \( R \bowtie_C S = \sigma_C(R \times S) \)

4. \( \sigma_{C_1 \land C_2}(R) = \sigma_{C_2}(\sigma_{C_1}(R)) = \sigma_{C_1}(\sigma_{C_2}(R)) \)

5. \( \sigma_{C_1 \lor C_2}(R) = \sigma_{C_1}(R) \cup \sigma_{C_2}(R) \)

6. \( \sigma_{C_1}(R) \bowtie_{C_2} S = \sigma_{C_1}(R \bowtie_{C_2} S) \)  
   ... if \( C_1 \) mentions only attributes of \( R \)

7. \( \pi_A(\sigma_C(R)) = \sigma_C(\pi_A(R)) \)  
   ... if \( C \) only mentions attributes in \( A \)

(*) the 6th equivalence is called “pushing a select”
Terminology (review / cont)

Superkey
- non-empty set of relation attributes that uniquely identify a row
- no two rows can have the same combination of superkey attribute values

Minimal key
- A superkey with a minimal set of attributes (can’t remove one and be a key)

Candidate key
- Each minimal superkey for a relation is a candidate key
- A relation may have multiple minimal candidate keys

Primary key
- A candidate key a designer picks to be enforced by the DBMS
- Implies a certain optimized storage scheme for the relation (more later)
- Note a DBMS doesn’t enforce minimality of candidate keys

As you (hopefully) see with hw-1:
- it can be tricky to determine keys
- it can be tricky to pick a primary key (eg, because of foreign keys)
Issues with Redundancy

<table>
<thead>
<tr>
<th>eid</th>
<th>name</th>
<th>dept</th>
<th>dept_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A01</td>
<td>Alice</td>
<td>12</td>
<td>CS</td>
</tr>
<tr>
<td>A12</td>
<td>Eric</td>
<td>10</td>
<td>HR</td>
</tr>
<tr>
<td>A13</td>
<td>Eric</td>
<td>12</td>
<td>CS</td>
</tr>
<tr>
<td>A03</td>
<td>Anne</td>
<td>12</td>
<td>CS</td>
</tr>
</tbody>
</table>

“Update Anomaly”
- If the CS dept. changes its name, we must change multiple rows

“Insertion Anomaly”
- If a department has no employees, where do we store its id and name?

“Deletion Anomaly”
- If A12 quits, the information about the HR department will be lost

Anomalies are in addition to wasted space
- e.g., the dept. name is stored multiple times
Normal Forms and Decomposition

Database Normal Forms

Define “levels” of allowed redundancies based on attribute “dependencies”

- First Normal Form (atomic values)
- Second & Third Normal Forms (functional deps / allow some redundancy)
- Boyce–Codd Normal Form (BCNF) (no redundancy from functional deps)
- Fourth & Fifth Normal Forms (for multivalued dependencies)

⇒ We’ll talk about 2NF, 3NF, and BCNF

Normalization involves “decomposing” a table into smaller tables

- Decomposition based on the dependencies that exist
- Follow a decomposition algorithm
- We’ll give a sense for how this works (but not dive into algorithms)