Relational Algebra
Relational Query Languages

Recall: Query = “Retrieval Program”

Language Examples:

Theoretical:
1. Relational Algebra
2. Relational Calculus
   a. Tuple Relational Calculus (TRC)
   b. Domain Relational Calculus (DRC)

Practical:
1. SQL (originally: SEQUEL from System R)
2. Quel (used in Ingres)
3. Datalog (Prolog-like – used in research lab systems)

Theoretical QLs give semantics to Practical QLs
Relational Algebra

• Basic Operators
  1. select (σ)
  2. project (π)
  3. union (∪)
  4. set difference (–)
  5. cartesian product (×)
  6. rename (ρ)

• Closure Property
Select ($\sigma$)

Notation: $\sigma_{\text{predicate}}\ (\text{Relation})$

Relation: Can be name of table or result of another query

Predicate:

1. Simple
   - attribute$_1 = $ attribute$_2$
   - attribute = constant value (also: $\neq$, $<$, $>$, $\leq$, $\geq$)

2. Complex
   - predicate AND predicate
   - predicate OR predicate
   - NOT predicate

Idea:

Select rows from a relation based on a predicate
## Bank Database

### 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>
<tr>
<td>Brighton</td>
<td>A-201</td>
<td>900</td>
</tr>
<tr>
<td>Redwood</td>
<td>A-222</td>
<td>700</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-217</td>
<td>750</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>
<tr>
<td>Johnson</td>
<td>A-201</td>
</tr>
<tr>
<td>Jones</td>
<td>A-217</td>
</tr>
<tr>
<td>Lindsay</td>
<td>A-222</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>
<tr>
<td>Williams</td>
<td>Nassau</td>
<td>Princeton</td>
</tr>
<tr>
<td>Adams</td>
<td>Spring</td>
<td>Pittsfield</td>
</tr>
<tr>
<td>Johnson</td>
<td>Alma</td>
<td>Palo Alto</td>
</tr>
<tr>
<td>Glenn</td>
<td>Sand Hill</td>
<td>Woodside</td>
</tr>
<tr>
<td>Brooks</td>
<td>Senator</td>
<td>Brooklyn</td>
</tr>
<tr>
<td>Green</td>
<td>Walnut</td>
<td>Stanford</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>Horseneck</td>
<td>0.4M</td>
</tr>
<tr>
<td>R.H.</td>
<td>Horseneck</td>
<td>8M</td>
</tr>
<tr>
<td>Pownel</td>
<td>Bennington</td>
<td>0.3M</td>
</tr>
<tr>
<td>N. Town</td>
<td>Rye</td>
<td>3.7M</td>
</tr>
<tr>
<td>Brighton</td>
<td>Brooklyn</td>
<td>7.1M</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>
<tr>
<td>Curry</td>
<td>L-93</td>
</tr>
<tr>
<td>Smith</td>
<td>L-11</td>
</tr>
<tr>
<td>Williams</td>
<td>L-17</td>
</tr>
<tr>
<td>Adams</td>
<td>L-16</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>
Select ( $\sigma$ )

Notation: $\sigma_{\text{predicate}}(\text{Relation})$

$$\sigma \text{ bcity = “Brooklyn” } (\text{branch}) =$$

<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>Brighton</td>
<td>Brooklyn</td>
<td>7.1M</td>
</tr>
</tbody>
</table>

$$\sigma \text{ assets > $8M } (\sigma \text{ bcity = “Brooklyn” } (\text{branch})) =$$

<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>
</tbody>
</table>
Project ( $\pi$ )

Notation: $\pi_{A1, \ldots, An}$ (Relation)

- Relation: name of a table or result of another query
- Each $A_i$ is an attribute
- Idea: $\pi$ selects columns (vs. $\sigma$ which selects rows)

$$\pi_{\text{cstreet, ccity}} (\text{customer}) =$$

<table>
<thead>
<tr>
<th>cstreet</th>
<th>ccity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>Park</td>
<td>Pittsfield</td>
</tr>
<tr>
<td>Putnam</td>
<td>Stanford</td>
</tr>
<tr>
<td>Nassau</td>
<td>Princeton</td>
</tr>
<tr>
<td>Spring</td>
<td>Pittsfield</td>
</tr>
<tr>
<td>Alma</td>
<td>Palo Alto</td>
</tr>
<tr>
<td>Sand Hill</td>
<td>Woodside</td>
</tr>
<tr>
<td>Senator</td>
<td>Brooklyn</td>
</tr>
<tr>
<td>Walnut</td>
<td>Stanford</td>
</tr>
</tbody>
</table>
Project ( $\pi$ )

$$\pi_{\text{bcity}} (\sigma_{\text{assets} > 5M} (\text{branch})) = \begin{array}{l}
\text{bcity} \\
\text{Brooklyn} \\
\text{Horseneck}
\end{array}$$

**Question:** Does the result of Project always have the same number of tuples as its input?
Union ($\cup$)

**Notation:** $Relation_1 \cup Relation_2$

$R \cup S$ valid only if:

1. $R$, $S$ have same number of columns (arity)
2. $R$, $S$ corresponding columns have same name and domain (compatibility)

**Example:**

$$(\pi_{cname\ (depositor)}) \cup (\pi_{cname\ (borrower)}) =$$

**Schema:**

<table>
<thead>
<tr>
<th>Depositor</th>
<th>Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td>$cname$</td>
<td>$cname$</td>
</tr>
<tr>
<td>$acct_no$</td>
<td>$lno$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$cname$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnson</td>
</tr>
<tr>
<td>Smith</td>
</tr>
<tr>
<td>Hayes</td>
</tr>
<tr>
<td>Turner</td>
</tr>
<tr>
<td>Jones</td>
</tr>
<tr>
<td>Lindsay</td>
</tr>
<tr>
<td>Jackson</td>
</tr>
<tr>
<td>Curry</td>
</tr>
<tr>
<td>Williams</td>
</tr>
<tr>
<td>Adams</td>
</tr>
</tbody>
</table>
Set Difference (−)

Notation: $Relation_1 - Relation_2$

R - S valid only if:

1. $R, S$ have same number of columns (arity)
2. $R, S$ corresponding columns have same domain (compatibility)

Example:

$$(\pi_{\text{bname}}(\sigma_{\text{amount} \geq 1000}(\text{loan}))) - (\pi_{\text{bname}}(\sigma_{\text{balance} < 800}(\text{account}))) =$$
What About Intersection?

Remember:

\[ R \cap S = R - (R - S) \]
Cartesian Product ( \( \times \) )

**Notation:** \( \text{Relation}_1 \times \text{Relation}_2 \)

\( R \times S \) like cross product for mathematical relations:

- *every tuple of R appended to every tuple of S*
- *flattened!!!*

**Example:**

\[
depositor \times borrower =
\]

<table>
<thead>
<tr>
<th>depositor. cname</th>
<th>acct_no</th>
<th>borrower. cname</th>
<th>lno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Jones</td>
<td>L-17</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Smith</td>
<td>L-23</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Hayes</td>
<td>L-15</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Jackson</td>
<td>L-14</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Curry</td>
<td>L-93</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Smith</td>
<td>L-11</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Williams</td>
<td>L-17</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Adams</td>
<td>L-16</td>
</tr>
<tr>
<td>Smith</td>
<td>A-215</td>
<td>Jones</td>
<td>L-17</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**How many tuples in the result?**

**A:** 56
Rename (ρ)

Notation:  $\rho_{\text{identifier}} (\text{Relation})$

renames a relation, or

Notation:  $\rho_{\text{identifier}_0 (\text{identifier}_1, \ldots, \text{identifier}_n)} (\text{Relation})$

renames relation and columns of n-column relation

Use:

massage relations to make $\cup$, $-$ valid, or $\times$ more readable
Rename \( \rho \)

Notation: \( \rho_{\text{identifier}_0}(\text{identifier}_1, \ldots, \text{identifier}_n) \) (Relation)

Example:

\[ \rho_{\text{result}}(\text{dcname}, \text{acctno}, \text{bcname}, \text{lno}) \) (\text{depositor} \times \text{borrower}) = \]

<table>
<thead>
<tr>
<th>dcnname</th>
<th>acctno</th>
<th>bcname</th>
<th>lno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Jones</td>
<td>L-17</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Smith</td>
<td>L-23</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Hayes</td>
<td>L-15</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Jackson</td>
<td>L-14</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Curry</td>
<td>L-93</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Smith</td>
<td>L-11</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Williams</td>
<td>L-17</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-101</td>
<td>Adams</td>
<td>L-16</td>
</tr>
<tr>
<td>Smith</td>
<td>A-215</td>
<td>Jones</td>
<td>L-17</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Example Query in RA

• Determine \textit{Ino} for loans that are for an amount that is larger than the amount of some other loan. (i.e. \textit{Ino} for all non-minimal loans)

Can do in steps:

\begin{align*}
\text{Temp}_1 & \leftarrow \ldots \\
\text{Temp}_2 & \leftarrow \ldots \text{ Temp}_1 \ldots \\
\ldots
\end{align*}
Example Query in RA

1. Find the base data we need

   \[ \text{Temp}_1 \leftarrow \pi_{\text{ino,amt}} (\text{loan}) \]

2. Make a copy of (1)

   \[ \text{Temp}_2 \leftarrow \rho_{\text{Temp}_2 (\text{ino2,amt2})} (\text{Temp}_1) \]
3. Take the cartesian product of 1 and 2

\[\text{Temp}_3 \leftarrow \text{Temp}_1 \times \text{Temp}_2\]
Example Query in RA

4. Select non-minimal loans

\[ \text{Temp}_4 \leftarrow \sigma_{\text{amt} > \text{amt2}} (\text{Temp}_3) \]

5. Project on lno

\[ \text{Result} \leftarrow \pi_{\text{lno}} (\text{Temp}_4) \]

… or, if you prefer…

\[ \pi_{\text{lno}} (\sigma_{\text{amt} > \text{amt2}} (\pi_{\text{lno,amt}} (\text{loan}) \times (\rho_{\text{Temp2 (lno2,amt2)}} (\pi_{\text{lno,amt}} (\text{loan})))))) \]
Relational Algebra

1. SELECT (\(\sigma\))
2. PROJECT (\(\pi\))
3. UNION (\(\cup\))
4. SET DIFFERENCE (\(\setminus\))
5. CARTESIAN PRODUCT (\(\times\))
6. RENAME (\(\rho\))

- Relational algebra gives semantics to practical query languages
- Above set: minimal relational algebra
  \(\Rightarrow\) will now look at some redundant (but useful!) operators
Review

Express the following query in the RA:

*Find the names of customers who have both accounts and loans*

\[
T_1 \leftarrow \rho_{T_1} (\text{cname2, lno}) (\text{borrower})
\]

\[
T_2 \leftarrow \text{depositor} \times T_1
\]

\[
T_3 \leftarrow \sigma_{\text{cname} = \text{cname2}} (T_2)
\]

\[
\text{Result} \leftarrow \pi_{\text{cname}} (T_3)
\]

*Above sequence of operators (\(\rho\), \(\times\), \(\sigma\)) very common.*

*Motivates additional (redundant) RA operators.*
Relational Algebra

Additional Operators

1. Natural Join (⋈)
2. Division (÷)
3. Generalized Projection (π)
4. Aggregation
5. Outer Joins (⋈⋈⋈)
6. Update (←) (we’ve already been using this)

• 1&2 Redundant: Can be expressed in terms of minimal RA
  e.g. depositor ⋈.borrower =
  \[ \pi \ldots(\sigma\ldots(depositor \times \rho\ldots(borrower))) \]

• 3 – 6 Added for extra power
Natural Join

Notation: \( Relation_1 \bowtie Relation_2 \)

Idea: combines \( \rho, \times, \sigma \)

\[
\begin{array}{|c|c|c|c|}
\hline
A & B & C & D \\
\hline
1 & \alpha & + & 10 \\
2 & \alpha & - & 10 \\
2 & \alpha & - & 20 \\
3 & \beta & + & 10 \\
\hline
\end{array}
\quad \bowtie \quad
\begin{array}{|c|c|}
\hline
E & B & D \\
\hline
\text{‘a’} & \alpha & 10 \\
\text{‘a’} & \alpha & 20 \\
\text{‘b’} & \beta & 10 \\
\text{‘c’} & \beta & 10 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|c|}
\hline
A & B & C & D & E \\
\hline
1 & \alpha & + & 10 & \text{‘a’} \\
2 & \alpha & - & 10 & \text{‘a’} \\
2 & \alpha & - & 20 & \text{‘a’} \\
3 & \text{B} & + & 10 & \text{‘b’} \\
3 & \beta & + & 10 & \text{‘c’} \\
\hline
\end{array}
\]

depositor \bowtie borrower

\equiv

\pi_{\text{cname,acct_no,lno}} (\sigma_{\text{cname}=\text{cname2}} (\text{ depositor } \times \rho_{t(\text{cname2,lno})} (\text{ borrower})))
Division

Notation: \( \text{Relation}_1 \div \text{Relation}_2 \)

Idea: expresses “for all” queries

Find values for A in r which have corresponding B values for all B values in s
Another way to look at it: $\div$ and $\times$

The largest value of $i$ such that: $i \times 3 \leq 17$

Relational Division

<table>
<thead>
<tr>
<th>$r$</th>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$\delta$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\delta$</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

$s \div \begin{array}{c} B \\ 1 \\ 2 \end{array} = t \begin{array}{c} A \\ \alpha \\ \delta \end{array}$

The largest value of $t$ such that: $(t \times s \subseteq r)$
# Division

## A More Complex Example

<table>
<thead>
<tr>
<th>$r$</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>$a$</td>
<td>$\alpha$</td>
<td>$a$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$a$</td>
<td>$\gamma$</td>
<td>$a$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$a$</td>
<td>$\gamma$</td>
<td>$b$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>$a$</td>
<td>$\gamma$</td>
<td>$a$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>$a$</td>
<td>$\gamma$</td>
<td>$b$</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>$a$</td>
<td>$\gamma$</td>
<td>$a$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>$a$</td>
<td>$\gamma$</td>
<td>$b$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>$a$</td>
<td>$\beta$</td>
<td>$b$</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

$\div \quad s$  

<table>
<thead>
<tr>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
</tr>
</tbody>
</table>

$t$  

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>$a$</td>
<td>$\gamma$</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>$a$</td>
<td>$\gamma$</td>
</tr>
</tbody>
</table>

$?$
Division Adds No Power

Definition in terms of the basic algebra operation
Let \( r(R) \) and \( s(S) \) be relations, and let \( S \subseteq R \)

\[
r \div s = \prod_{R-S}(r) - \prod_{R-S}((\prod_{R-S}(r) \times s) - \prod_{R-S,S}(r))
\]

To see why

- \( \prod_{R-S,S}(r) \) simply reorders attributes of \( r \)

- \( \prod_{R-S}((\prod_{R-S}(r) \times s) - \prod_{R-S,S}(r)) \) gives those tuples \( t \) in
  \( \prod_{R-S}(r) \) such that for some tuple \( u \in s \), \( tu \not\in r \).
Generalized Projection

Notation: $\pi_{e_1, \ldots, e_n} (Relation)$

$e_1, \ldots, e_n$ can include arithmetic expressions – not just attributes

Example

<table>
<thead>
<tr>
<th>cname</th>
<th>limit</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jones</td>
<td>5000</td>
<td>2000</td>
</tr>
<tr>
<td>Turner</td>
<td>3000</td>
<td>2500</td>
</tr>
</tbody>
</table>

credit = 

Then...

$\pi_{\text{cname, limit - balance}} (\text{credit}) =$

<table>
<thead>
<tr>
<th>cname</th>
<th>limit - balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jones</td>
<td>3000</td>
</tr>
<tr>
<td>Turner</td>
<td>500</td>
</tr>
</tbody>
</table>
 Aggregate Functions and Operations

- **An aggregate function** takes a collection of values and returns a single value as a result.
  
  - **avg**: average value
  - **min**: minimum value
  - **max**: maximum value
  - **sum**: sum of values
  - **count**: number of values

- **Aggregate operation** in relational algebra

  \[ E \quad g \quad F_1(A_1), F_2(A_2), \ldots, F_n(A_n) \quad (E) \]

  - \( E \) is any relational-algebra expression
  - \( G_1, G_2, \ldots, G_n \) is a list of attributes on which to group
    (can be empty)
  - Each \( F_i \) is an aggregate function
  - Each \( A_i \) is an attribute name

CSCI1270, Lecture 2
Aggregate Operation – Example

- Relation $r$:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>7</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$\beta$</td>
<td>7</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\beta$</td>
<td>3</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\beta$</td>
<td>10</td>
</tr>
</tbody>
</table>

$g_{\text{sum}(c)}(r)$

- No grouping

$\text{sum-C}$

27
Aggregate Operation – Example

Relation *account* grouped by *branch-name*:

<table>
<thead>
<tr>
<th>branch-name</th>
<th>account-number</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perryridge</td>
<td>A-102</td>
<td>400</td>
</tr>
<tr>
<td>Perryridge</td>
<td>A-201</td>
<td>900</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-217</td>
<td>750</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-215</td>
<td>750</td>
</tr>
<tr>
<td>Redwood</td>
<td>A-222</td>
<td>700</td>
</tr>
</tbody>
</table>

\[ \text{branch-name} \ g \ \text{sum(balance)} \ (\text{account}) \]

<table>
<thead>
<tr>
<th>branch-name</th>
<th>sum(balance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perryridge</td>
<td>1300</td>
</tr>
<tr>
<td>Brighton</td>
<td>1500</td>
</tr>
<tr>
<td>Redwood</td>
<td>700</td>
</tr>
</tbody>
</table>
Aggregate Functions (Cont.)

Result of aggregation does not have a name
  – Can use rename operation to give it a name
  – For convenience, we permit renaming as part of aggregate operation

\[ \text{branch-name} \ g \ \text{sum(balance)} \ as \ \text{sum-balance} \ (\text{account}) \]
Outer Joins

Motivation:

\[
\begin{array}{|c|c|c|}
\hline
\text{bname} & \text{lno} & \text{amt} \\
\hline
\text{Downtown} & \text{L-170} & \text{3000} \\
\text{Redwood} & \text{L-230} & \text{4000} \\
\text{Perry} & \text{L-260} & \text{1700} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{cname} & \text{lno} \\
\hline
\text{Jones} & \text{L-170} \\
\text{Smith} & \text{L-230} \\
\text{Hayes} & \text{L-155} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|}
\hline
\text{bname} & \text{lno} & \text{amt} & \text{cname} \\
\hline
\text{Downtown} & \text{L-170} & \text{3000} & \text{Jones} \\
\text{Redwood} & \text{L-230} & \text{4000} & \text{Smith} \\
\hline
\end{array}
\]

Join result loses…

\[\rightarrow \text{any record of Perry}\]
\[\rightarrow \text{any record of Hayes}\]
Outer Joins

1. Left Outer Join (\(\bowtie\))
   - *preserves all tuples in left relation*

\[
\begin{array}{|c|c|c|}
\hline
\text{bname} & \text{lno} & \text{amt} \\
\hline
\text{Downtown} & \text{L-170} & 3000 \\
\text{Redwood} & \text{L-230} & 4000 \\
\text{Perry} & \text{L-260} & 1700 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{cname} & \text{lno} \\
\hline
\text{Jones} & \text{L-170} \\
\text{Smith} & \text{L-230} \\
\text{Hayes} & \text{L-155} \\
\hline
\end{array}
\]

\[
\text{loan} \bowtie \text{borrower} =
\]

\[
\begin{array}{|c|c|c|c|}
\hline
\text{bname} & \text{lno} & \text{amt} & \text{cname} \\
\hline
\text{Downtown} & \text{L-170} & 3000 & \text{Jones} \\
\text{Redwood} & \text{L-230} & 4000 & \text{Smith} \\
\text{Perry} & \text{L-260} & 1700 & \bot \\
\hline
\end{array}
\]

\(\bot = \text{NULL}\)
Outer Joins

2. Right Outer Join (▷◁)

- *preserves all tuples in right relation*

\[
\text{loan} \triangleright◁ \text{borrower} =
\]

<table>
<thead>
<tr>
<th>bname</th>
<th>lno</th>
<th>amt</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>L-170</td>
<td>3000</td>
<td></td>
</tr>
<tr>
<td>Redwood</td>
<td>L-230</td>
<td>4000</td>
<td></td>
</tr>
<tr>
<td>Perry</td>
<td>L-260</td>
<td>1700</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\downarrow = \text{NULL}
\]

CSCI1270, Fall 2008, Lecture 2
3. **Full Outer Join (⟂⟂)**

- *preserves all tuples in both relations*

\[
\text{loan} \, \Box \Box \, \text{borrower} = \quad \\
\begin{array}{ccc}
\text{bname} & \text{lno} & \text{amt} \\
\hline
\text{Downtown} & \text{L-170} & 3000 \\
\text{Redwood} & \text{L-230} & 4000 \\
\text{Perry} & \text{L-260} & 1700 \\
\hline
\end{array} \\
\begin{array}{cc}
\text{cname} & \text{lno} \\
\hline
\text{Jones} & \text{L-170} \\
\text{Smith} & \text{L-230} \\
\text{Hayes} & \text{L-155} \\
\hline
\end{array}
\]

\[\bot = \text{NULL}\]
Update

Notation: Identifier $\leftarrow$ Query

Common Uses:

1. **Deletion**: $r \leftarrow r \ominus s$
   - e.g., $\text{account} \leftarrow \text{account} \ominus \sigma_{\text{bname}=\text{Perry}}(\text{account})$
   - (deletes all Perry accounts)

2. **Insertion**: $r \leftarrow r \cup s$
   - e.g., $\text{branch} \leftarrow \text{branch} \cup \{(\text{Waltham}, \text{Boston}, 7\text{M})\}$
   - (inserts new branch with $\text{bname} = \text{Waltham}, \text{bcity} = \text{Boston}, \text{assets} = 7\text{M}$)
   - e.g., $\text{depositor} \leftarrow \text{depositor} \cup (\rho_{\text{temp}}(\text{cname,acct_no})(\text{borrower}))$
   - (adds all borrowers to depositors, treating lno’s as acct_no’s)

3. **Update**: $r \leftarrow \pi_{e_1,\ldots,e_n}(r)$
   - e.g., $\text{account} \leftarrow \pi_{\text{bname,acct_no,bal*1.05}}(\text{account})$
   - (adds 5% interest to account balances)
Views

• Limited access to DB.
• Tailored schema

• Consider a person who needs to know a customer’s loan number but has no need to see the loan amount. This person should see a relation described as:

\[ \Pi_{\text{customer-name, loan-number}} (\text{borrower} \bowtie \text{loan}) \]

• A relation that is made visible to a user as a “virtual relation” is called a view.
View Definition

- A view is defined using the create view statement which has the form

  ```
  create view v as <query expression>
  ```

  where <query expression> is any legal relational algebra query expression. The view name given as v.

- Once a view is defined, the view name can be used to refer to the virtual relation that the view generates.

- View definition is not the same as creating a new relation by evaluating the query expression. Rather, a view definition causes the saving of an expression to be substituted into queries using the view.
View Examples

• Consider the view (named \textit{all-customer}) consisting of branches and their customers.

\textbf{create view} \textit{all-customer} as
\[
\Pi_{\text{branch-name, customer-name}} (\text{depositor} \Join \text{account}) \\
\cup \Pi_{\text{branch-name, customer-name}} (\text{borrower} \Join \text{loan})
\]

• We can find all customers of the Perryridge branch by writing:

\[
\Pi_{\text{customer-name}} \\
(\sigma_{\text{branch-name} = \text{“Perryridge”}} (\text{all-customer}))
\]
Updates Through View

- Database modifications expressed as views must be translated to modifications of the actual relations in the database.
- Consider the person who needs to see all loan data in the loan relation except amount. The view given to the person, branch-loan, is defined as:

\[
\text{create view branch-loan as } \\
\quad \Pi_{\text{branch-name, loan-number}} (\text{loan})
\]

- Since we allow a view name to appear wherever a relation name is allowed, the person may write:

\[
\text{branch-loan} \leftarrow \text{branch-loan} \cup \{("Perryridge", L-37)\}
\]
Updates Through Views (Cont.)

• The previous insertion must be represented by an insertion into the actual relation $loan$ from which the view $branch-loan$ is constructed.

• An insertion into $loan$ requires a value for $amount$. The insertion can be dealt with by either.
  
  – rejecting the insertion and returning an error message to the user.
  
  – inserting a tuple (“L-37”, “Perryridge”, $null$) into the $loan$ relation
Updates Through Views (Cont.)

• Some updates through views are impossible to translate.

create view v as $\sigma_{\text{branch-name} = \text{“Perryridge”}}(\text{account})$

$v \leftarrow v \cup (\text{L-99, Downtown, 23})$

• Others cannot be translated uniquely

\textit{all-customer} \leftarrow \textit{all-customer} \cup (\text{Perryridge, John})

• Have to choose loan or account, and create a new loan/account number!
Views Defined Using Other Views

• One view may be used in the expression defining another view

• A view relation $v_1$ is said to depend directly on a view relation $v_2$ if $v_2$ is used in the expression defining $v_1$

• A view relation $v_1$ is said to depend on view relation $v_2$ if either $v_1$ depends directly on $v_2$ or there is a path of dependencies from $v_1$ to $v_2$
View Expansion

• Let view $v_1$ be defined by an expression $e_1$ that may itself contain uses of view relations.

• View expansion of an expression repeats the following replacement step:

  repeat
  Find any view relation $v_i$ in $e_1$
  Replace the view relation $v_i$ by the expression defining $v_i$
  until no more view relations are present in $e_1$

• As long as the view definitions are not recursive, this loop will terminate