Relational Algebra
Relational Algebra

- Procedural language
- Six basic operators
  - select: $\sigma$
  - project: $\Pi$
  - union: $\cup$
  - set difference: $-$
  - Cartesian product: $\times$
  - rename: $\rho$
- The operators take one or two relations as inputs and produce a new relation as the result
Composition of Operations

- Building expressions using multiple operations
- Example: \( \sigma_{A=C}(r \times s) \)
Rename Operation

- Name, and therefore to refer to, the results of relational-algebra expressions
  - Refer to a relation by more than one name
- Example: $\rho_x(E)$ returns the expression $E$ under the name $X$
- If a relational-algebra expression $E$ has arity $n$, then $\rho_{x(A_1, A_2, \ldots, A_n)}(E)$ returns the result of expression $E$ under the name $X$, and with the attributes renamed to $A_1$, $A_2$, $\ldots$, $A_n$
Banking Example

- branch (branch_name, branch_city, assets)
- customer (customer_name, customer_street, customer_city)
- account (account_number, branch_name, balance)
- loan (loan_number, branch_name, amount)
- depositor (customer_name, account_number)
- borrower (customer_name, loan_number)
Example Queries

• Find all loans of over $1200

\[ \sigma_{\text{amount} > 1200} \ (\text{loan}) \]

• Find the loan number for each loan of an amount greater than $1200

\[ \Pi_{\text{loan\_number}} \ (\sigma_{\text{amount} > 1200} \ (\text{loan})) \]

\text{loan} \ (\text{loan\_number}, \ \text{branch\_name}, \ \text{amount})
Example Queries

• Find the names of all customers who have a loan, an account, or both, from the bank
  \[ \Pi_{\text{customer\_name}}(\text{borrower}) \cup \Pi_{\text{customer\_name}}(\text{depositor}) \]

• Find the names of all customers who have a loan and an account at the bank
  \[ \Pi_{\text{customer\_name}}(\text{borrower}) \cap \Pi_{\text{customer\_name}}(\text{depositor}) \]

depositor (customer\_name, account\_number)
borrower (customer\_name, loan\_number)
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
  – Answer 1
    \[ \Pi_{\text{customer\_name}}(\sigma_{\text{branch\_name} = \text{"Perryridge"}} (\sigma_{\text{borrower\_loan\_number} = \text{loan\_loan\_number}} (\text{borrower} \times \text{loan}))) \]
  – Answer 2
    \[ \Pi_{\text{customer\_name}}(\sigma_{\text{loan\_loan\_number} = \text{borrower\_loan\_number}} (\sigma_{\text{branch\_name} = \text{"Perryridge"}} (\text{loan}) \times \text{borrower})) \]
Example Queries

• Find the largest account balance
  – Aggregate max is not directly supported in relational algebra
  – Find those balances that are not the largest
    • Rename account relation as d so that we can compare each account balance with all the others
    – Use set difference to find the max balance accounts

\[ \pi_{\text{balance}}(\text{account}) - \pi_{\text{account.balance}} \left( \sigma_{\text{account.balance} < \text{d.balance}} (\text{account} \times \rho_{\text{d}} (\text{account})) \right) \]

account (account_number, branch_name, balance)
Formal Definition

• A basic expression in the relational algebra consists of either one of the following:
  – A relation in the database
  – A constant relation

• Let $E_1$ and $E_2$ be relational-algebra expressions; the following are all relational-algebra expressions:
  – $E_1 \cup E_2$
  – $E_1 - E_2$
  – $E_1 \times E_2$
  – $\sigma_p(E_1)$, $P$ is a predicate on attributes in $E_1$
  – $\Pi_S(E_1)$, $S$ is a list consisting of some of the attributes in $E_1$
  – $\rho_x(E_1)$, $x$ is the new name for the result of $E_1$
Additional Operations

• The additional operations do not add any power to the relational algebra, but can simplify writing common queries
  – Set intersection
  – Natural join
  – Division
  – Assignment
### Set-Intersection Operation – Example

The set intersection operation finds the common elements between two relations. Here are the examples:

#### Relation $r$:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>1</td>
</tr>
<tr>
<td>α</td>
<td>2</td>
</tr>
<tr>
<td>β</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Relation $s$:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>2</td>
</tr>
<tr>
<td>β</td>
<td>3</td>
</tr>
</tbody>
</table>

The set intersection of $r$ and $s$, denoted as $r \cap s$, is:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>2</td>
</tr>
</tbody>
</table>
Set-Intersection Operation

- \( r \cap s = \{ t \mid t \in r \text{ and } t \in s \} \)
  - In basic operators, we only have set difference but no intersection

- Assume:
  - \( r, s \) have the same arity
  - attributes of \( r \) and \( s \) are compatible

- \( r \cap s = r - (r - s) \)
Natural Join Operation – Example

$$r \times s$$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>1</td>
<td>α</td>
<td>a</td>
</tr>
<tr>
<td>β</td>
<td>2</td>
<td>γ</td>
<td>a</td>
</tr>
<tr>
<td>γ</td>
<td>4</td>
<td>β</td>
<td>b</td>
</tr>
<tr>
<td>α</td>
<td>1</td>
<td>γ</td>
<td>a</td>
</tr>
<tr>
<td>δ</td>
<td>2</td>
<td>β</td>
<td>b</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>α</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>β</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>γ</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>δ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>1</td>
<td>α</td>
<td>a</td>
<td>α</td>
</tr>
<tr>
<td>α</td>
<td>1</td>
<td>α</td>
<td>a</td>
<td>γ</td>
</tr>
<tr>
<td>α</td>
<td>1</td>
<td>α</td>
<td>a</td>
<td>α</td>
</tr>
<tr>
<td>δ</td>
<td>2</td>
<td>β</td>
<td>b</td>
<td>δ</td>
</tr>
</tbody>
</table>
Natural-Join Operation

• Let r and s be relations on schemas R and S respectively. \( r \bowtie s \) is a relation on schema \( R \cup S \) obtained as follows:
  – Consider each pair of tuples \( t_r \) from r and \( t_s \) from s
  – If \( t_r \) and \( t_s \) have the same value on each of the attributes in \( R \cap S \), add a tuple \( t \) to the result, where
    • \( t \) has the same value as \( t_r \) on r
    • \( t \) has the same value as \( t_s \) on s
Example

• \( R = (A, B, C, D) \)
• \( S = (E, B, D) \)
• Result schema = \( (A, B, C, D, E) \)
• \( r \bowtie s \) is defined as

\[
\Pi_{r.A, r.B, r.C, r.D, s.E} (\sigma_{r.B = s.B \land r.D = s.D} (r \times s))
\]
### Division Operation – Example

#### Relations

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>1</td>
</tr>
<tr>
<td>α</td>
<td>2</td>
</tr>
<tr>
<td>α</td>
<td>3</td>
</tr>
<tr>
<td>β</td>
<td>1</td>
</tr>
<tr>
<td>γ</td>
<td>1</td>
</tr>
<tr>
<td>δ</td>
<td>1</td>
</tr>
<tr>
<td>δ</td>
<td>3</td>
</tr>
<tr>
<td>δ</td>
<td>4</td>
</tr>
<tr>
<td>ε</td>
<td>6</td>
</tr>
<tr>
<td>ε</td>
<td>1</td>
</tr>
<tr>
<td>β</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
</tr>
<tr>
<td>β</td>
</tr>
</tbody>
</table>

#### Query

\[ r \div s \]
Division Operation

• Let $r$ and $s$ be relations on schemas $R$ and $S$ respectively where $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$ and $S = (B_1, \ldots, B_n)$
  – The result of $r \div s$ is a relation on schema $R - S = (A_1, \ldots, A_m)$
  – $r \div s = \{ t \mid t \in \Pi_{R-S}(r) \land \forall u \in s ( tu \in r ) \}$, where $tu$ means the concatenation of tuples $t$ and $u$ to produce a single tuple
• Suited to queries that include the phrase “for all”
Another Division Example

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>a</td>
<td>α</td>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>α</td>
<td>a</td>
<td>γ</td>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>α</td>
<td>a</td>
<td>γ</td>
<td>b</td>
<td>1</td>
</tr>
<tr>
<td>β</td>
<td>a</td>
<td>γ</td>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>β</td>
<td>a</td>
<td>γ</td>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>γ</td>
<td>a</td>
<td>γ</td>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>γ</td>
<td>a</td>
<td>γ</td>
<td>b</td>
<td>1</td>
</tr>
<tr>
<td>γ</td>
<td>a</td>
<td>β</td>
<td>b</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ r \]

\[ s \]

\[ r \div s \]
Properties of Division Operation

- Let \( q = r \div s \), \( q \) is the largest relation satisfying \( q \times s \subseteq r \)
- Let \( r(R) \) and \( s(S) \) be relations, and let \( S \subseteq R \), \( r \div s = \prod_{R \setminus S}(r) - \prod_{R \setminus S}((\prod_{R \setminus S}(r) \times s) - \prod_{R \setminus S,S}(r)) \)
  - \( \prod_{R \setminus S,S}(r) \) simply reorders attributes of \( r \)
  - \( \prod_{R \setminus S}\left(\prod_{R \setminus S}(r) \times s\right) - \prod_{R \setminus S,S}(r) \) gives those tuples \( t \) in \( \prod_{R \setminus S}(r) \) such that for some tuple \( u \in s \), \( tu \not\in r \)
Assignment Operation

- The assignment operation (←) provides a convenient way to express complex queries
  - Write query as a sequential program consisting of a series of assignments followed by an expression whose value is displayed as a result of the query
  - Assignment must always be made to a temporary relation variable
- Example: compute \( r \div s \)
  - \( \text{temp}_1 \leftarrow \Pi_{R-S}(r) \), \( \text{temp}_2 \leftarrow \Pi_{R-S}((\text{temp}_1 \times s) - \Pi_{R-S,S}(r)) \)
  - result = \( \text{temp}_1 - \text{temp}_2 \)
- The result to the right of the ← is assigned to the relation variable on the left of the ←
  - May use variable in subsequent expressions
Bank Example Queries

• Find the names of all customers who have a loan and an account at bank
  \[
  \Pi_{\text{customer\_name}}(\text{borrower}) \cap \Pi_{\text{customer\_name}}(\text{depositor})
  \]

• Find the name of all customers who have a loan at the bank and the loan amount
  \[
  \Pi_{\text{customer\_name, loan\_number, amount}}(\text{borrower} \bowtie \text{loan})
  \]
Bank Example Queries

• Find all customers who have an account from at least the “Downtown” and the “Uptown” branches
  – Answer 1

\[
\Pi_{\text{customer\_name}} (\sigma_{\text{branch\_name} = \text{“Downtown”}} (\text{depositor} \bowtie \text{account})) \cap \\
\Pi_{\text{customer\_name}} (\sigma_{\text{branch\_name} = \text{“Uptown”}} (\text{depositor} \bowtie \text{account}))
\]

– Answer 2: using a constant relation

\[
\Pi_{\text{customer\_name}, \text{branch\_name}} (\text{depositor} \bowtie \text{account}) \\
\div \rho_{\text{temp}(\text{branch\_name})} (\{\{\text{“Downtown”}, \{\text{“Uptown”}\}\})
\]
Example Queries

• Find all customers who have an account at all branches located in Brooklyn city

\[ \Pi_{\text{customer\_name, branch\_name}}(\text{depositor} \bowtie \text{account}) \div \Pi_{\text{branch\_name}}(\sigma_{\text{branch\_city} = "Brooklyn"}(\text{branch})) \]
Summary

• Examples of relational algebra expressions
• Additional operators
  – Do not add any power to the relational algebra, but can simplify writing common queries
  – Set intersection
  – Natural join
  – Division
  – Assignment
To-Do List

• Translate the relational algebra expression examples into SQL
• What can you observe?