BİL 354 – Veritabanı Sistemleri

Relational Algebra
(İlişkisel Cebir)
Relational Queries

- **Query languages**: Allow manipulation and retrieval of data from a database.

- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic.
  - Allows for much optimization.

- **Query Languages ≠ programming languages**!
  - QLs not intended to be used for complex calculations.
  - QLs support easy, efficient access to large data sets.
Formal Relational Query Languages

Two mathematical Query Languages form the basis for “real” languages (e.g. SQL), and for implementation:

- **Relational Algebra**: More operational, very useful for representing execution plans.
- **Relational Calculus**: Lets users describe what they want, rather than how to compute it. (Nonoperational, *declarative*.)
What is Relational Algebra?

- An algebra whose operands are relations or variables that represent relations.
- Operators are designed to do the most common things that we need to do with relations in a database.
  - The result is an algebra that can be used as a *query language* for relations.
Relational Algebra

- Procedural language
- Six basic operators
  - select
  - project
  - union
  - set difference
  - Cartesian product
  - rename
- The operators take one or more relations as inputs and give a new relation as a result.
### Select Operation – Example

**Relation r**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>α</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>β</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>β</td>
<td>12</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>β</td>
<td>23</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

**σ_{A=B \land D > 5}(r)**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>α</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>β</td>
<td>23</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
Select Operation

- Notation: \( \sigma_p(r) \)
- \( p \) is called the selection predicate
- Defined as:

\[
\sigma_p(r) = \{ t \mid t \in r \text{ and } p(t) \}
\]

Where \( p \) is a formula in propositional calculus consisting of terms connected by: \( \land \) (and), \( \lor \) (or), \( \lnot \) (not)

Each term is one of:

- \(<\text{attribute}> \ op \ <\text{attribute}> \) or \(<\text{constant}>\)
  where \( op \) is one of: \(=, \neq, >, \geq, <, \leq\)

- Example of selection:
  \[ \sigma_{\text{branch-name="Perryridge"}}(\text{account}) \]
Project Operation – Example

- Relation \( r \):

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>10</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>20</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>30</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>40</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

- \( \Pi_{A,C}(r) \)

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

\[\Pi_{A,C}(r) = \begin{array}{c|c}
\alpha & 1 \\
\alpha & 1 \\
\beta & 1 \\
\beta & 2 \\
\end{array}\]
Project Operation

- Notation:

\[ \Pi_{A_1, A_2, \ldots, A_k} (r) \]

where \( A_1, A_2 \) are attribute names and \( r \) is a relation name.

- The result is defined as the relation of \( k \) columns obtained by erasing the columns that are not listed.

- Duplicate rows removed from result, since relations are sets.

- E.g. To eliminate the *branch-name* attribute of *account*

\[ \Pi_{account-number, balance} (account) \]
Union Operation – Example

- Relations $r$, $s$:

  $r$:  
  \[
  \begin{array}{c|c}
  A & B \\
  \hline
  \alpha & 1 \\
  \alpha & 2 \\
  \beta & 1 \\
  \end{array}
  \]

  $s$:  
  \[
  \begin{array}{c|c}
  A & B \\
  \hline
  \alpha & 2 \\
  \beta & 3 \\
  \end{array}
  \]

  $r \cup s$:  
  \[
  \begin{array}{c|c}
  A & B \\
  \hline
  \alpha & 1 \\
  \alpha & 2 \\
  \beta & 1 \\
  \beta & 3 \\
  \end{array}
  \]
Union Operation

- Notation: \( r \cup s \)
- Defined as:
  \[
  r \cup s = \{ t \mid t \in r \text{ or } t \in s \}
  \]
- For \( r \cup s \) to be valid:
  1. \( r, s \) must have the same \textit{arity} (same number of attributes)
  2. The attribute domains must be \textit{compatible} (e.g., 2nd column of \( r \) deals with the same type of values as does the 2nd column of \( s \))
- E.g. to find all customers with either an account or a loan
  \[
  \Pi_{\text{customer-name}} (\text{depositor}) \cup \Pi_{\text{customer-name}} (\text{borrower})
  \]
### Set Difference Operation – Example

- **Relations** $r$, $s$:

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>2</td>
</tr>
<tr>
<td>$\beta$</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>2</td>
</tr>
<tr>
<td>$\beta$</td>
<td>3</td>
</tr>
</tbody>
</table>

- **$r - s$**:

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>$\beta$</td>
<td>1</td>
</tr>
</tbody>
</table>
Set Difference Operation

- Notation $r - s$
- Defined as:
  \[ r - s = \{ t \mid t \in r \text{ and } t \not\in s \} \]
- Set differences must be taken between compatible relations.
  - $r$ and $s$ must have the same arity
  - Attribute domains of $r$ and $s$ must be compatible
### Cartesian-Product Operation - Example

**Relations r, s:**

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

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>β</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>β</td>
<td>20</td>
<td>b</td>
</tr>
<tr>
<td>γ</td>
<td>10</td>
<td>b</td>
</tr>
</tbody>
</table>

**r x s:**

<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>10</td>
<td>a</td>
</tr>
<tr>
<td>α</td>
<td>1</td>
<td>β</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>α</td>
<td>1</td>
<td>β</td>
<td>20</td>
<td>b</td>
</tr>
<tr>
<td>α</td>
<td>1</td>
<td>γ</td>
<td>10</td>
<td>b</td>
</tr>
<tr>
<td>β</td>
<td>2</td>
<td>α</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>β</td>
<td>2</td>
<td>β</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>β</td>
<td>2</td>
<td>β</td>
<td>20</td>
<td>b</td>
</tr>
<tr>
<td>β</td>
<td>2</td>
<td>γ</td>
<td>10</td>
<td>b</td>
</tr>
</tbody>
</table>
Cartesian-Product Operation

- Notation \( r \times s \)
- Defined as:
  \[
  r \times s = \{ t, q \mid t \in r \text{ and } q \in s \}
  \]
- Assume that attributes of \( r(R) \) and \( s(S) \) are disjoint. (That is, \( R \cap S = \emptyset \)).
- If attributes of \( r(R) \) and \( s(S) \) are not disjoint, then renaming must be used.
Composition of Operations

- Can build expressions using multiple operations

- Example: $\sigma_{A=C}(r \times s)$

- $r \times s$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td>$\alpha$</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td>$\beta$</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td>$\beta$</td>
<td>20</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td>$\gamma$</td>
<td>10</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\alpha$</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\beta$</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\beta$</td>
<td>20</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\gamma$</td>
<td>10</td>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>

- $\sigma_{A=C}(r \times s)$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td>$\alpha$</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\beta$</td>
<td>20</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\beta$</td>
<td>20</td>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>
Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to 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

$$\rho_X(A_1, A_2, ..., A_n)(E)$$

returns the result of expression $E$ under the name $X$, and with the attributes renamed to $A_1, A_2, ..., A_n$. 
Banking Example

branch (branch-name, branch-city, assets)

customer (customer-name, customer-street, customer-only)

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

\[ \Pi_{\text{loan-number}} (\sigma_{\text{amount} > 1200} (\text{loan})) \]
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 bank.

$$\Pi_{\text{customer-name}}(\text{borrower}) \cap \Pi_{\text{customer-name}}(\text{depositor})$$
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
    \[
    \Pi_{\text{customer-name}}(\sigma_{\text{loan.loan-number} = \text{borrower.loan-number}}(\sigma_{\text{branch-name} = “Perryridge”}(\text{loan}) \times \text{borrower}))
    \]
Find the largest account balance

- Rename `account` relation as `d`
- The query is:

\[ \Pi_{\text{balance}}(\text{account}) - \Pi_{\text{account.balance}}(\sigma_{\text{account.balance} < d.\text{balance}}(\text{account} \bowtie d (\text{account}))) \]
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 $E_1$
We define additional operations that do not add any power to the relational algebra, but that simplify common queries.

- Set intersection
- Natural join
- Division
- Assignment
Set-Intersection Operation

- Notation: \( r \cap s \)
- Defined as:
  \[ r \cap s = \{ t \mid t \in r \text{ and } t \in s \} \]
- Assume:
  - \( r, s \) have the same arity
  - attributes of \( r \) and \( s \) are compatible
- Note: \( r \cap s = r - (r - s) \)
Set-Intersection Operation - Example

- Relation r, s:

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

r

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

s

- $r \cap s$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Natural-Join Operation

- Notation: $r \natural s$

- Let $r$ and $s$ be relations on schemas $R$ and $S$ respectively. Then, $r \natural s$ 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
    
    1. $t$ has the same value as $t_r$ on $r$
    2. $t$ has the same value as $t_s$ on $s$

- Example:

  $R = (A, B, C, D)$
  $S = (E, B, D)$
  
  Result schema = $(A, B, C, D, E)$
  
  $r \natural s$ is defined as:

  $$
  \Pi_{r.A, r.B, r.C, r.D, s.E} (\sigma_{r.B = s.B \land r.D = s.D} (r \times s))
  $$

  ©Silberschatz, Korth and Sudarshan
Natural Join Operation – Example

- Relations r, s:

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

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

\[ r \bowtie s \]

<table>
<thead>
<tr>
<th></th>
<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>
<td></td>
</tr>
<tr>
<td>α</td>
<td>1</td>
<td>α</td>
<td>a</td>
<td>γ</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>1</td>
<td>γ</td>
<td>a</td>
<td>α</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>1</td>
<td>γ</td>
<td>a</td>
<td>γ</td>
<td></td>
</tr>
<tr>
<td>δ</td>
<td>2</td>
<td>β</td>
<td>b</td>
<td>δ</td>
<td></td>
</tr>
</tbody>
</table>
Division Operation

\[ r \div s \]

- Suited to queries that include the phrase “for all”.
- Let \( r \) and \( s \) be relations on schemas \( R \) and \( S \) respectively where
  \[ R = (A_1, \ldots, A_m, B_1, \ldots, B_n) \]
  \[ S = (B_1, \ldots, B_n) \]

The result of \( r \div s \) is a relation on schema \( R - S = (A_1, \ldots, A_m) \)

\[ r \div s = \{ t \mid t \in \Pi_{R-S}(r) \land \forall u \in s \ (tu \in r) \} \]
Division Operation – Example

Relations $r$, $s$:

\[
\begin{array}{|c|c|}
\hline
A & B \\
\hline
\alpha & 1 \\
\alpha & 2 \\
\alpha & 3 \\
\beta & 1 \\
\gamma & 1 \\
\delta & 1 \\
\delta & 3 \\
\delta & 4 \\
\epsilon & 6 \\
\epsilon & 1 \\
\beta & 2 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
B \\
\hline
1 \\
2 \\
\hline
\end{array}
\]

$r \div s$:

\[
\begin{array}{|c|}
\hline
A \\
\hline
\alpha \\
\beta \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
r \\
\hline
\end{array}
\]
Another Division Example

Relations $r, s$:

<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>$\alpha$</td>
<td>$a$</td>
<td>$\gamma$</td>
<td>$a$</td>
<td>$1$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$a$</td>
<td>$\gamma$</td>
<td>$b$</td>
<td>$1$</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$a$</td>
<td>$\gamma$</td>
<td>$a$</td>
<td>$1$</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$a$</td>
<td>$\gamma$</td>
<td>$b$</td>
<td>$3$</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>$a$</td>
<td>$\gamma$</td>
<td>$a$</td>
<td>$1$</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>$a$</td>
<td>$\gamma$</td>
<td>$b$</td>
<td>$1$</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>$a$</td>
<td>$\beta$</td>
<td>$b$</td>
<td>$1$</td>
</tr>
</tbody>
</table>

$s$

\[
\begin{array}{cc}
D & E \\
1 & 1 \\
1 & 1 \\
\end{array}
\]

$r \div S$:

<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 Operation (Cont.)

- **Property**
  - Let $q - r \div s$
  - Then $q$ is the largest relation satisfying $q \times s \subseteq r$

- **Definition in terms of the basic algebra operation**
  For relations $r(R)$ and $s(S)$, and let $S \subseteq R$

  $$r \div s = \Pi_{R \setminus S}(r) - \Pi_{R \setminus S}(\Pi_{R \setminus S}(r) \times s) - \Pi_{R \setminus S,S}(r)$$

  To see why
  - $\Pi_{R \setminus S,S}(r)$ simply reorders attributes of $r$
  - $\Pi_{R \setminus S}(\Pi_{R \setminus S}(r) \times s) - \Pi_{R \setminus S,S}(r))$ gives those tuples $t$ in
    $$\Pi_{R \setminus S}(r)$$ such that for some tuple $u \in s$, $tu \notin r.$
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: Write \( r \div s \) as

\[
\begin{align*}
\text{temp1} & \leftarrow \Pi_{R-S} (r) \\
\text{temp2} & \leftarrow \Pi_{R-S} ((\text{temp1} \times s) - \Pi_{R-S,S} (r)) \\
x \text{result} & = \text{temp1} - \text{temp2}
\end{align*}
\]

- The result to the right of the ← is assigned to the relation variable on the left of the ←.
- May use variable in subsequent expressions.
Example Queries

Find all customers who have an account from at least the “Downtown” and the Uptown” branches.

Query 1

$$\Pi_{CN}(\sigma_{BN=“Downtown”}(depositor \Join account)) \cap \Pi_{CN}(\sigma_{BN=“Uptown”}(depositor \Join account))$$

where $CN$ denotes customer-name and $BN$ denotes branch-name.

Query 2

$$\Pi_{customer-name, \ branch-name}(depositor \Join account) \div \rho_{temp(branch-name)}(\{(“Downtown”), (“Uptown”)\})$$
Example Queries

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

\[ \Pi_{customer-name, branch-name} (depositor \times account) \]
\[ \div \Pi_{branch-name} (\sigma_{branch-city = \text{"Brooklyn"}} (branch)) \]