Relational Algebra

- **Basis for SQL**
- **Operands**
  - Relations
- **Operators**
  - **Core**: Set operators, SPJ, rename
  - **Extended**: duplicate elimination, aggregation, grouping, sorting, outer-join

**Selection**

- \( \sigma_c(R) \) or \( \text{SELECT}_c(R) \)
  - Choose rows of \( R \) that satisfies condition \( C \)
  - \( C \) is a boolean expression of constants and \( R \)'s attributes

**Projection**

- \( \pi_L(R) \) or \( \text{PROJ}_L(R) \)
  - Choose columns of \( R \)
  - \( L \) is a list of \( R \)'s attributes
  - Eliminate duplicates in the results (set semantics)
**Projection Example**

\[ \pi_{\text{beer}, \text{price}}(\text{Sells}) \]

<table>
<thead>
<tr>
<th>beer</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bud</td>
<td>2.50</td>
</tr>
<tr>
<td>Miller</td>
<td>2.75</td>
</tr>
<tr>
<td>Miller</td>
<td>3.00</td>
</tr>
</tbody>
</table>

**Product**

- \( R_1 \times R_2 \)
  - Concatenate each tuple from \( R_1 \) with each tuple from \( R_2 \)
  - Also called Cross Product or Cartesian Product

**Product Example**

\[ R_1 \times R_2 \]

\[
\begin{array}{ccc}
A & B & R_1 \times R_2 \\
1 & 2 & 1 & 2 & 5 & 6 \\
3 & 4 & 3 & 4 & 5 & 6 \\
5 & 6 & 3 & 4 & 7 & 8 \\
7 & 8 & 3 & 4 & 9 & 10 \\
9 & 10 & & & & \\
\end{array}
\]

**Theta Join**

- \( R_1 \bowtie c_{k} R_2 \) or \( R_1 \JOIN c(R_1 \times R_2) \)

**Theta Join Example**

\[ \bowtie \text{name} = \text{bar} \]

<table>
<thead>
<tr>
<th>name</th>
<th>address</th>
<th>bar</th>
<th>beer</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe's</td>
<td>Maple St.</td>
<td>Joe's</td>
<td>Bud</td>
<td>2.50</td>
</tr>
<tr>
<td>Sue's</td>
<td>River Rd.</td>
<td>Sue's</td>
<td>Miller</td>
<td>2.75</td>
</tr>
<tr>
<td>Joe's</td>
<td>Maple St.</td>
<td>Joe's</td>
<td>Miller</td>
<td>3.00</td>
</tr>
<tr>
<td>Sue's</td>
<td>River Rd.</td>
<td>Sue's</td>
<td>Miller</td>
<td>3.00</td>
</tr>
</tbody>
</table>

**Natural Join**

- \( R_1 \bowtie \bowtie R_2 \) or \( R_1 \JOIN R_2 \)
  - Implies equality condition of the attributes with the same name
  - Only one column from each pair of equated attributes is kept in results
Natural Join Example

<table>
<thead>
<tr>
<th>Bars</th>
<th>Sells</th>
</tr>
</thead>
<tbody>
<tr>
<td>bar</td>
<td>address</td>
</tr>
<tr>
<td>Joe's</td>
<td>Maple St.</td>
</tr>
<tr>
<td>Sue's</td>
<td>River Rd.</td>
</tr>
<tr>
<td>Sue's</td>
<td>River Rd.</td>
</tr>
<tr>
<td>Sue's</td>
<td>River Rd.</td>
</tr>
</tbody>
</table>

Bars $\bowtie$ Sells

<table>
<thead>
<tr>
<th>bar</th>
<th>address</th>
<th>beer</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe's</td>
<td>Maple St.</td>
<td>Bud</td>
<td>2.50</td>
</tr>
<tr>
<td>Joe's</td>
<td>Maple St.</td>
<td>Miller</td>
<td>2.75</td>
</tr>
<tr>
<td>Sue's</td>
<td>River Rd.</td>
<td>Bud</td>
<td>2.50</td>
</tr>
<tr>
<td>Sue's</td>
<td>River Rd.</td>
<td>Miller</td>
<td>3.00</td>
</tr>
</tbody>
</table>

Rename

$\rho_{S(A_1,A_2,...,A_n)}(R)$ or $\text{RENAME}_{S(A_1,A_2,...,A_n)}(R)$
- Rename a relation and its attributes
- Rename a relation only: $\rho_s(R)$

Rename Example

<table>
<thead>
<tr>
<th>Bars</th>
<th>name</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>bar</td>
<td>address</td>
<td></td>
</tr>
<tr>
<td>Joe's</td>
<td>Maple St.</td>
<td></td>
</tr>
<tr>
<td>Sue's</td>
<td>River Rd.</td>
<td></td>
</tr>
</tbody>
</table>

$\rho_{\text{bar|bar, address}}(\text{Bars})$

<table>
<thead>
<tr>
<th>bar</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe's</td>
<td>Maple St.</td>
</tr>
<tr>
<td>Sue's</td>
<td>River Rd.</td>
</tr>
</tbody>
</table>

Set Operators

- Union: $\cup$
- Intersection: $\cap$
- Difference: $-$

- Two relations must have the same schemas
  - Same number of attributes
  - Same attribute names
  - Same attribute types

Set Operator Examples

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

$\cup$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

$\cap$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

$-$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Expressions (Queries)

- Find the name of the bars that are either on Maple St. or sells Bud for less than 3 dollars.

$\pi_{\text{name | address}}(\text{Bars}) \bowtie \sigma_{\text{beer = \text{'Bud'}} \text{ AND price < 3.0}}(\rho_{\text{bar|bar, address}}(\text{Bars}) \bowtie \text{Sells})$

- Relational Operator Precedence
  - Unary operators
  - Binary operators
  - Set operators
    - Intersection
    - Union, Difference
**Linear Notation for Expressions**

\[ R(\text{bar, address}) := \text{Bars} \]
\[ S := R \bowtie \text{Sells} \]
\[ T := \sigma_{\text{address}=\text{"Maple St."} \text{ OR (beer=\text{"Bud"} AND price < 3.0)}}(S) \]
\[ \text{Ans} := \pi_{\text{bar}} \]

**Extended Algebra**

- Eliminate duplicates
- Sort tuples
- Extended projection
- Grouping and aggregation
- Outerjoin

**Duplicate Elimination**

\[ \delta(R) \text{ or } \text{DELTA}(R) \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

**Sorting**

\[ \pi_{L}(R) \text{ or } \text{TAU}_{L}(R) \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

**Extended Projection**

\[ \pi_{L}(R) \text{ or } \text{PROJ}_{L}(R) \]

- Allow arithmetic expressions of R's attributes in L

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>A+B</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

**Aggregation Operators**

- Aggregation operators apply to column(s) of a relation and produces a single result
- SUM, AVG, COUNT, MIN, MAX

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th>SUM(A) = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>AVG(A) = 2</td>
</tr>
<tr>
<td>B</td>
<td>COUNT(A) = 2</td>
</tr>
<tr>
<td>C</td>
<td>MIN(A) = 1</td>
</tr>
<tr>
<td>D</td>
<td>MAX(A) = 3</td>
</tr>
</tbody>
</table>
**Grouping**

- \( \gamma_l(R) \) or \( \text{GAMMA}_l(R) \)
  - \( L \) is a list of elements that are
  - Individual attributes (grouping attributes)
  - \( \text{AGG}(A) \) where \( \text{AGG} \) is one of the aggregation operators and \( A \) is an attribute

**Grouping Example**

\[
\begin{array}{ccc}
A & B & C \\
1 & 4 & 2 \\
1 & 3 & 3 \\
1 & 2 & 2 \\
2 & 3 & 3 \\
3 & 2 & 4 \\
2 & 2 & 3 \\
\end{array}
\]  

\[
\begin{array}{ccc}
A & B & C \\
1 & 4 & 2 \\
1 & 2 & 2 \\
2 & 3 & 3 \\
2 & 2 & 3 \\
3 & 2 & 4 \\
\end{array}
\]

**Outer Joins**

- \( \bowtie_l \) or \( \text{OUTERJOIN} \) (full outer join)
- \( \bowtie_l \) or \( \text{LEFTJOIN} \) (left outer join)
- \( \bowtie_r \) or \( \text{RIGHTJOIN} \) (right outer join)

**Dangling Tuples**

- When two relations join, a tuple is said to be **dangling** if it does not match any tuple in the other relation.

**Outer Join Examples**

\[
\begin{array}{ccc}
R \bowtie_l S & R \bowtie_l S & R \bowtie_r S \\
1 & 2 & 3 & 1 & 2 & 3 & 1 & 2 & 3 \\
3 & 4 & NULL & 3 & 4 & NULL & NULL & 5 & 6 \\
\end{array}
\]

**Summary of Operators**

- UNION
- INTERSECTION
- DIFFERENCE
- SELECT
- PROJECT
- PROJECTION
- PRODUCT
- JOIN
- RENAME
- DELTA
- TAU
- GAMMA
- PROJECTION
- OUTERJOIN
- LEFTJOIN
- RIGHTJOIN
- SUM, AVG, COUNT, MIN, MAX
Set vs. Bag

- Bag (or multiset) allows duplicates while set does not
- SQL is a bag language
  - A relation may contain duplicate tuples
  - Only eliminate duplicate tuples when the query explicitly asks for it
  - Certain operations like projection are much more efficient on bags than sets.

Relational Operators on Bags

- Most relational operators work the same for both sets and bags — just keep the duplicates
- Union, Intersection, Difference

```
R = {1, 1, 2, 3, 3, 4 }, S = {1, 2, 3, 5}
R \cup S = ??
R \cap S = ??
R - S = ??
```

Bag Laws != Set Laws

- Some laws hold for both sets and bags
  - E.g. R \cup S = S \cup R
- Some laws do not
  - E.g. S \cup S = S

Exercises - Schema

- Senators( senator_id, senator_name, state, party )
- Bills( bill_id, bill_name )
- Votes( senator_id, bill_id, vote )
  - Vote can be Yes, No, or Abstain

Exercises - Queries

- Find the state which senator Hillary Clinton represents.
- Find the names of the bills on which Senator Hillary Clinton has voted Yes.
- Find the number of Republican senators in the Senate.
- Find the names of the bills which have been passed, i.e. the bills that have more Yes votes than No votes.

Constraints on Relations

- Constraints
  - not null, unique, primary key, references (foreign key) ...
- Express constraints using relational algebra
  - R = 0
  - R \subseteq S
Functional Dependency

- In Senators, \{senator_id\}, \{senator_name\}
- In Bills, \{bill_id\}, \{bill_name\}
- In Votes, \{senator_id,bill_id\}, \{vote\}

Other Constraints

- Referential Integrity
- NOT NULL
- Unique
- Enumeration
  - E.g. vote can only be Yes, No, or Abstain