Lecture Topics

I. Background
   - Foundation for database query language
   - SQL is based on relational algebra with some extensions

II. Operators
   - Projection
   - Selection
   - Cartesian Product
   - Union
   - Difference
   - Intersection
   - Complement

II.1 Sets and Operations
   - If A, B, and C are sets
     \[ U : \text{union, } A \cup B = \{ x | x \in A \lor x \in B \} \]
     \[ \cap : \text{intersection, } A \cap B = \{ x | x \in A \land x \in B \} \]
     \[ - : \text{difference, } A - B = \{ x | x \in A \land x \notin B \} \]
   - Cartesian Product

The above form an algebra, i.e.,
   you can perform operations on results of operations, such as
   \[(A \cap B) \times (C \cup A)\]

II.2 Relation
   - Relations are sets of tuples
   - Tuples have a domain
   - Assume domain is linearly ordered
     i.e., \( x \leq y, x = y, \text{ or } y \geq x \)

II.3 Projection
   - Any operation on relations produces a relation.

II.3.1 Projection
   \[ R \begin{array}{cccc}
   A & B & C & D \\
   \hline
   1 & 2 & 3 & 4 \\
   \end{array} \]
   \[ \pi_{\text{AB}}(R) \]
   Select A, B
   From R
   \[ \begin{array}{cc}
   A & B \\
   \hline
   1 & 2 \\
   \end{array} \]
II. 2 Selection

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

\[ \sigma_{A \leq 2} (R) \]

\text{SELECT } * \text{ FROM } R \text{ WHERE } A \leq 2

\[ \begin{array}{cccc}
A & B & C & D \\
1 & 2 & 4 & 8
\end{array} \]

The condition (predicate) can be specified by a boolean formula

\[ T, \land, \lor \text{ on atomic conditions} \]

atomic conditions are:

- a comparison between columns
- a comparison between column and constant

II. 3 Cartesian Product

\[ R \begin{array}{cc}
A & B \\
1 & 2
\end{array}, S \begin{array}{cc}
C & B & D \\
10 & 10 & 10 \\
20 & 20 & 20
\end{array} \]

\[ R \times S \]

\text{SELECT } R.A, R.B, C, S.B, D \text{ FROM } R, S

\[ \begin{array}{ccccccc}
A & B & C & S.B & D \\
1 & 10 & 10 & 10 & 10 \]

II. 4 Union

\[ \begin{array}{cccc}
R & A & B & S & A & B \\
1 & 10 & & 1 & 10 \\
2 & 20 & & 3 & 20
\end{array} \]

\[ R \cup S \]

\text{SELECT } * \text{ FROM } R \text{ UNION SELECT } * \text{ FROM } S

\[ \begin{array}{cccc}
A & B \\
1 & 10 \]

Must be union compatible

- require same set of same-arity relations
- columns should "probably" be from the same domains

II. 5 Difference

\[ R \begin{array}{cc}
A & B \\
1 & 2
\end{array}, S \begin{array}{cc}
A & B \\
1 & 10 \\
2 & 20
\end{array} \]

\[ R - S \]

\text{SELECT } * \text{ FROM } R \text{ MINUS SELECT } * \text{ FROM } S

\[ \begin{array}{cccc}
A & B \\
2 & 20
\end{array} \]

- \text{EXCEPT sometimes instead of MINUS}
- must be union compatible

-useful to correlate information from two tables
Intersection

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

**R \( \cap \) S**

**Select**

FROM **R**

INTERSECT

**Select**

FROM **S**

- Union compatibility required
- Not fundamental

**R \( \cap \) S = R - (R - S)**

**III. Complementarity**

- Given:
  - Person (Name, Sex, Age)
  - Birth (Parent, Child)

**Produce:**

- Answer (Father, Daughter)

**Strategy:**

- Need two distinct copies of 'Person'

**Select P as Father, C as Daughter**

FROM PERSON, BIRTH, PERSON AS PERSON1
WHERE P-PERSON.N AND C-PERSON1.N AND PERSON.P = 'M' AND PERSON1.S = 'F'

- Produce: Answer (GRANDPARENT, GRANDCHILD)

- (GRAND-GRANDPARENT, GRAND-GRANDCHILD)
  - Cartesian product of Grandparent and with (Parent, Child), equally on the immediate person.

- (GRAND \( \times \) - parent, GRAND \( \times \) - child)?

- (Ancestor, Descendant)?

Not possible in relational algebra.

**Proof Sketch:** (by induction)

- Any R.A. query is limited in how many relations, or copy of relations, it can refer to

- Computing arbitrary (Ancestor, Descendant) pairs can't be done if the query is limited by the number of relations or copies of relations it can refer to

**Another Example:**

Can't compute transitive closure with R.A.

- You can with SQL 1999 standard
Example Schema:

Marriage
PK, FK, PK
Wife
PK
Husband
Age

Person
PK
Name
Set
Age

Birth
PK, PK, Parent
PK, PK, Child