Unit 2: Relational Algebra as a Query Language

• General Problem: How to query the information that is contained relational tables?

• SQL is the most important commercial query language for relational databases

• The essential principles of SQL are based upon relational algebra
  – Instead of doing algebra with numbers, we work with tables
  – Example: Form the union \( R \cup S \) of tables \( R \) and \( S \)
• Relational algebra is the essence of SQL, but it is far simpler than the full SQL language

• These ideas can be applied to a wide variety of practical problems
Relations

• Relational algebra deals with relations, which are sets of tuples (rows) drawn from suitable domains

• The order of the rows, and the number of times a row appears, do not matter

• Assuming the columns are labeled, their order does not matter

• The following two relations are equal:
Operations on relations

There are several fundamental operations on relations. We will describe them in turn:

- Union
- Difference
- Projection
- Selection
- Cartesian product
**Union**

- For relations R and S, the union of R and S, designated $R \cup S$, is the relation that contains all of the rows of R and all of the rows of S.

- The operation should make sense: R and S should have the same attributes.

- We refer to this as “union compatibility” of relations.

- Example:
DIFFERENCE

• For relations R and S, the difference of R and S, designated $R - S$, is the relation that contains all of the rows of R that do not appear in S

• The two relations should be union compatible

• Example:
**Projection**

• If R is a relation, this operation produces a relation by discarding some of the attributes

• Let A be a subset of the attributes of R

• Then the *projection* of R onto A, denoted by $\pi_A(R)$, is the relation obtained by restricting R to just the attributes in A

• Example:
**Selection**

- If R is a relation, this operation produces a relation by specifying a filtering condition on the set of rows.
- Let F be a boolean formula whose variables are attributes of R.
  - Example: Suppose that the attributes of R are \{A,B,C,D,E\}. Then F could be \(A \leq C \& D = 4\).
- Then the *selection* of R by F, written \(\sigma_F(R)\), is the relation containing all the rows of R that satisfy F.
- Example:
**Cartesian product**

- For relations $R$ and $S$, this operation produces the *product* relation $R \times S$ which contains all of the concatenations of a row from the first relation followed by a row from the second relation.

- Formally, $R \times S = \{ \text{concat}(t_1, t_2) \mid t_1 \in R, \; t_2 \in S \}$

- We may think of the rows of $R \times S$ as “long tuples”

- Example:
From relational algebra to queries

• These algebraic operations allow us to define a large number of interesting queries for relational databases

• In order to express our examples, we will use the following procedural operations:
  – Assignment of an expression to a new variable. In our case it will be assignment of a relational expression to a relational variable. For instance, $R := \sigma_{A>0}(S)$
  – Renaming of an attribute in a relation
A Small Example

Consider the following three relations:

- **INFO(PERSON, SEX, AGE)**. The key is PERSON.
- **BIRTH(PARENT, CHILD)**. This relation specifies who is a parent of whom. For each child, there will be one record for the mother and one for the father.
- **MARRIAGE(HUSBAND, WIFE, DATE)**. The key is \{HUSBAND, WIFE\}. The DATE specifies when the marriage took place.
Query Number 1

• Query: Produce a relation R(PERSON) consisting of all women who are less than 10 years old
• Note that all the information required can be obtained from looking at a single relation, INFO
• \( R := \pi_{PERSON}(\sigma_{AGE \leq 10 \land SEX = F}(INFO)) \)
Query Number 2

• Produce the relation D-BIRTH(PARENT,DAUGHTER), which relates parents to their daughters.

• Here, even though the answer comes only from the single relation BIRTH, we still have to check in the relation INFO what the SEX of the CHILD is.

• To do that, we create the cartesian product of the two relations: INFO and BIRTH. This gives us “long tuples” consisting of a tuple in INFO and a tuple in BIRTH.
• For this query, two tuples should match if:
  – BIRTH.CHILD = INFO.PERSON
  – The SEX of INFO.PERSON is F

• We therefore obtain the following solution:
  1. TMP := \( \sigma_{CHILD=PERSON \, \& \, \text{SEX}=F}(INFO \times BIRTH) \)
  2. D-BIRTH := \( \pi_{PARENT,CHILD\rightarrow DAUGHTER}(TMP) \)
Query Number 3

• Produce a relation FD-BIRTH(FATHER, DAUGHTER), which relates fathers to their daughters

• Here we have to simultaneously look at two copies of the relation INFO, as we have to determine both the SEX of the PARENT and the SEX of the CHILD

• We need to have two different copies of INFO
Solution to Query Number 3

1. INFO2 := INFO

2. TMP := $\sigma_{\phi}(INFO \times BIRTH \times INFO2)$

3. FD-BIRTH :=

$$\pi_{PARENT\rightarrow FATHER, CHILD\rightarrow DAUGHTER}(TMP)$$

where $\phi$ is the formula:

- PARENT=INFO.PERSON AND
- CHILD=INFO2.PERSON AND
- INFO.SEX=M AND
- INFO2.SEX=F
Two More Queries

Produce the following relations:

- `FS-INLAW(FATHER_IN_LAW,SON_IN_LAW)`
- `GRAND(GRANDPARENT,GRANDCHILD)`
Sidenote: Relational Algebra is not a Universal Query Language

- It is impossible in relational algebra to compute the relation $R(\text{ANCESTOR}, \text{DESCENDANT})$
- The proof is by induction. It is reasonably simple, but cumbersome.
- General idea: Any relational algebra query involves a fixed number of copies of relations
- But this limits the distance between ancestors and descendants that can be searched by the query
Natural Join

- For reasons, which we will see later, it is useful to introduce the concept of natural join
- Suppose we have relations $R(A,B,C)$ and $S(A,C,D)$
- Then their natural join is:
  \[
  \pi_{R.A,R.B,R.C,S.D}(\sigma_{R.A=S.A \land R.C=S.C}(R \times S'))
  \]
- In effect, we glue the tuples together by equating columns with the same name, and then remove duplicate columns
- Natural join appears naturally when information about an object is combined from several tables
Optimization Notes

- Projections and all kinds of selections are cheaper than cartesian products or joins
- In general, equality selections are cheaper than nonequality selections
- Equality selections on an index are cheaper than projections
Exercises

Let’s try to answer the typical queries of the introductory section.

1. Give names of all employees born before 1960

2. Give names and addresses of all employees who took out a book before 1985.8.1

3. Find names and addresses of employees who had flu
Topics Covered in this Unit

- Five basic relational algebra operators
- Use of these operators to solve queries