Formal Query Languages: Relational Algebra Review

DSC 208R - Data Management for Analytics

Introduction to Formal Query Languages

Formal query languages provide a theoretical foundation for database query processing. They are precisely defined, making it possible to analyze their expressive power and properties. Relational Algebra is a procedural query language, meaning it specifies *how* to obtain the result.

Relational Algebra

Relational Algebra is a collection of operators that take one or two relations (tables) as input and produce a new relation as output. All operators are "closed" which means their input and output are always relations.

Schemas Used in Examples

- **Movie** (<u>name</u>, year, genre)
- **ActedIN** (<u>actorname</u>, <u>moviename</u>)
- **Actor** (<u>name</u>, age)

Core Relational Algebra Operations

1. Selection (σ)

The selection operation filters rows (tuples) from a relation that satisfy a given predicate (condition).

- **Notation**: $\sigma_P(R)$
- **P**: A predicate (condition) involving attributes of R.
- **R**: The input relation.
- **Result**: A relation containing only the tuples from R for which P is true.

Example: Select movies produced after 2000.

$$\sigma_{\text{year}>2000}(\text{Movie}) \tag{1}$$

This expression returns a relation with all columns of 'Movie', but only for rows where 'year' is greater than 2000.

2. Projection (π)

The projection operation selects specific columns (attributes) from a relation and removes duplicate rows from the result.

- **Notation**: $\pi_{A_1,A_2,...,A_n}(R)$
- $A_1, A_2, ..., A_n$: A list of attributes to be retained.
- **R**: The input relation.
- **Result**: A relation containing only the specified attributes. Duplicates are eliminated.

Example: Return movie names and their genres.

$$\pi_{\text{name, genre}}(\text{Movie}) \tag{2}$$

3. Cartesian Product ($\times$)

The Cartesian product combines every tuple from one relation with every tuple from another relation.

- **Notation**: $R \times S$
- **R, S**: Input relations.
- **Result**: A new relation with all possible combinations of tuples from R and S. The schema of the result is a concatenation of the schemas of R and S.

Example: Combine Movie and ActedIN relations.

$$\text{Movie} \times \text{ActedIN} \tag{3}$$

This operation by itself is rarely useful in practice because it generates many meaningless combinations. It is typically followed by a selection operation to form a join.

4. Set Operations

Relational algebra includes standard set operations: Union ($\cup$), Intersection ($\cap$), and Set Difference ($\setminus$). For these operations to be valid, the relations must be **union-compatible**, meaning they have the same number of attributes and corresponding attributes have the same data types.

Union ($\cup$) Returns all tuples that are in R, or in S, or in both. Duplicates are eliminated.

- **Notation**: $R \cup S$

Example: Return all names from 'Actor' and 'Movie'.

$$\pi_{\text{name}}(\text{Actor}) \cup \pi_{\text{name}}(\text{Movie}) \tag{4}$$

Intersection ($\cap$) Returns all tuples that are common to both R and S.

- **Notation**: $R \cap S$

Set Difference ($\setminus$) Returns all tuples that are in R but not in S.

- **Notation**: $R \setminus S$

5. Rename (ρ)

The rename operation changes the name of a relation or its attributes.

- **Notation**: $\rho_{B_1,\ldots,B_n(R)}(R')$ or $\rho_S(R)$
- R': New name for the relation.
- $B_1, ..., B_n$: New names for the attributes of R.
- **R**: The input relation.

This operation is useful for ensuring unique attribute names in complex expressions, especially after a Cartesian product, and for making results more readable.

Example: Rename 'Movie' to 'Film'.

$$\rho_{\text{Film}}(\text{Movie}) \tag{5}$$

Derived Relational Algebra Operations (Joins)

Joins are combinations of Cartesian product and selection, offering a more convenient way to combine relations.

Theta Join ($\bowtie_\theta$)

A theta join combines the Cartesian product with a selection condition (θ).

- **Notation**: $R \bowtie_\theta S \equiv \sigma_\theta(R \times S)$
- θ: Any valid comparison predicate.

Example: Find movie names and actor names where movie name in 'Movie' equals movie name in 'ActedIN'.

$$\text{Movie} \bowtie_{\text{Movie.name=ActedIN.moviename}} \text{ActedIN} \tag{6}$$

Equi-join

An equi-join is a special case of theta join where the predicate θ consists only of equality comparisons.

Natural Join ($\bowtie$)

The natural join combines relations based on common attributes, performing an equi-join on all identically named attributes and then projecting out duplicate common attributes.

- **Notation**: $R \bowtie S$
- It automatically identifies common columns and joins on them.

Example: Join 'Movie' and 'ActedIN' naturally.

$$\text{Movie} \bowtie \text{ActedIN} \tag{7}$$

This specific example might not work as intended with the given schemas because 'Movie' has 'name' and 'ActedIN' has 'moviename', which are conceptually the same but have different names. To perform a natural join on these, one of the attributes would need to be renamed first.