Relational Algebra

CISC637, Lecture #8

Ben Carterette

Formal Query Languages

- SQL is a query language for relational databases
- The relational model admits *formal* query languages based on mathematical principles:
  - **Relational algebra**, based on operators over relations
  - **Relational calculus**, based on declarative statements about data
- The algebra more directly supports computation
  - A relational algebra query implies a sequence of steps that can be taken to execute it
  - It is a *procedural language*
- SQL is an implementation of relational algebra

Relational Algebra

- Basic operators:
  - **Selection:** $\sigma_P(R)$
    select a subset of records from relation instance R matching condition P
  - **Projection:** $\pi_X(R)$
    select a subset of fields X from relation instance R
  - **Cross-product:** $R1 \times R2$
    concatenate each record in R1 with each record in R2
  - **Set-difference:** $R1 - R2$
    return records in R1 that are not in R2
  - **Union:** $R1 \cup R2$
    return records in either R1 or R2

Relational Algebra

- Derived and advanced operators:
  - **Intersection:** $R1 \cap R2$
    return records in both R1 and R2
  - **Join:** $R1 \bowtie R2$
    combine information from relations R1 and R2
  - **Division:** $R1/R2$
    return records in R1 that "match" *every* record in R2 in a subset of fields
  - **Renaming:** $\rho(R2(M), R1)$
    relation instance R1 is named R2 and its fields are renamed according to mapping M
  - **Aggregate functions:** ${}_XG_f(R)$
    calculate aggregating function f on relation instance R grouped by fields X

Preliminaries

- Operations are applied to *relation instances*, and the result of an operation is a relation instance
  - Schemas of input relations are fixed
  - Schemas of output relations are also fixed, though may be different from input relation schemas
- Algebra is all about **composing** operators
  - Every operator takes relation instances as input and returns relation instances
  - Algebra is **closed** under these operators

Selection & Projection

- Selection returns a relation consisting of all records matching a logical *selection condition* P
  - We use Greek letter sigma (σ) for the selection function
  - $\sigma_P(R)$ consists only of records in R for which P is true
  - Schema of $\sigma_P(R)$ is the same as schema of R
- Projection returns a relation with only the fields indicated
  - We use Greek letter pi (π) for the projection function
  - $\pi_X(R)$ has only the fields of R specified in the list X
  - Schema of $\pi_X(R)$ is a subset of the schema of R
  - Records in $\pi_X(R)$ same as records in R, but duplicates are dropped

Projection & Selection

- *Generalized projection* - selecting arithmetic functions of fields
  - $\pi_F(R)$, where F is an arithmetic function of one or more fields and constant values
- Selection and projection operators can be *composed*
  - $\pi_X(\sigma_P(R))$ returns records in R for which P is true, and with only fields specified in X
  - Equivalent to SQL query

    `SELECT X FROM R WHERE P`

Set Operators

- Union, intersection, and set-difference
- These take two relation instances R1 and R2 and return a new instance
- R1 and R2 must be *union-compatible*
  - Same number of fields
  - Corresponding fields must have same domain
- The schema of the new relation is defined to be the schema of R1

Cross-Product

- Cross-product R1 × R2 pairs each record in R1 with each record in R2 to create a new relation of concatenated records
- Schema is defined to be the concatenation of R1 and R2's schemas
  - If both have a field with the same name, refer to that field by number
  - Or rename it using the ρ operator

 17

Joins

- Combine two relations R1 and R2 on records matching some logical condition p
  - $R1 \bowtie_p R2 \equiv \sigma_p(R1 \times R2)$
- Schema is the same as the cross-product schema (except in special cases)
  - Equijoin: P is an equality condition relating fields in R1 and R2
    - Most joins are equijoins
  - Natural join: P is an equality condition relating fields with the same name in R1 and R2
    - Most joins are natural joins
    - For natural joins, we don't have to write P
    - Just write $R1 \bowtie R2$

 18

Outer Joins

- Outer joins preserve records even if they do not match the condition p
  – Missing values filled in with *null*s
- **Left outer join** R1 ⟕$_P$ R2 preserves every record in R1 and uses *null*s for R2 fields when there is no record in R2 that satisfies P
- **Right outer join** R1 ⟖$_P$ R2 preserves every record in R2 and uses *null*s for R1 fields when there is no record in R1 that satisfies P
- **Full outer join** R1 ⟗$_P$ R2 combines left and right outer joins, preserving all records in both

 21

Aggregation Functions

- Calculate a function of a set of values, returning a single value
  – $G_f(R)$, where f is a function of a field in R, returns a relation consisting of the value returned by f
  – E.g. $G_{avg(salary)}$(instructors) calculates the average salary of all instructors
- Aggregate values by group:
  – $_XG_f(R)$ calculates f for distinct groups defined by values of field X
  – E.g. $_{dept\_name}G_{avg(salary)}$(instructors) calculates average salaries for instructors in each department

 23

Relational Algebra Summary

- Most important operators:
  - **Selection:** $\sigma_P(R)$
    select a subset of records from relation instance R matching condition P
  - **Projection:** $\pi_X(R)$
    select a subset of fields X from relation instance R
  - **Join:** R1 ⋈ R2
    combine information from relations R1 and R2
  - **Aggregate functions:** ${}_XG_f(R)$
    calculate aggregating function f on relation instance R grouped by fields X

