Relational Algebra on Bags

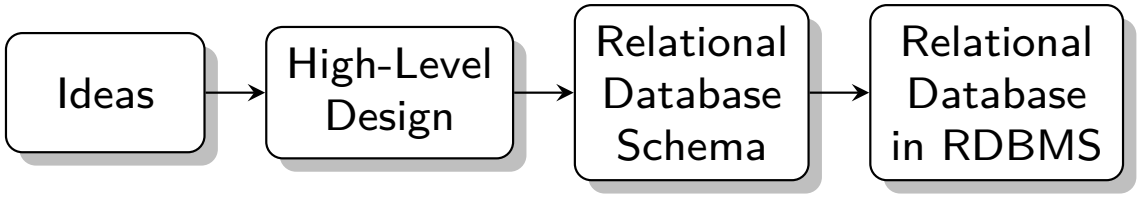
Hui Chen [a]

[a]CUNY Brooklyn College, Brooklyn, NY, USA

March 3, 2022

Outline

Overview


Ideas
High-Level Design
Relational Database Schema
Relational Database in RDBMS

Outline

Introduction to SQL

SQL (pronounced as "sequel") is the principal language used to describe and manipulate relational database, and has several aspects:

- Data definition language (DDL).
  - SQL includes commands to create database objects such as tables, indexes, and views, as well as commands to define access rights to those database objects.
  - Topics of this lecture: SQL commands to create database tables (relations)
- Data manipulation language (DML).
  - SQL includes commands to insert, update, delete, and retrieve data within the database tables.
- Transaction control language (TCL).
  - The DML commands in SQL are executed within the context of a transaction.
- Data control language (DCL).
  - Data control commands are used to control access to data objects.

To Discuss Subset of DML

- Relational Algebra
- Query Databases

Outline

Why Relational Algebra

- SQL, incorporates relational algebra at its center, and many SQL programs are really "syntactically sugared" relational algebra expressions
- When a DBMS processes queries, it first the first thing translate queries into relational algebra or a very similar internal representation.
- Help us understand queries.

What's Algebra?

A mathematical system consisting of

- Operands – variables or values from which new values can be constructed.
- Operators – symbols denoting procedures that construct new values from given values.

Relational Algebra

An algebra whose operands are relations or variables that represent relations.

- designed to do the most common tasks with relations in a relational database
- can be used as a query language for relations

Relational Algebra on Bags

Examined relational algebra on sets, now examine relational algebra on bags

- A bag (or multiset) is like a set, but an element may appear more than once.
- Examples, $\{1, 2, 1, 3\}$, $\{'a', 'a', 'a', 'b'\}$, $\{1, 2, 3\}$, and $'a', 'b'$

Why Bags

- SQL is actually a bag language.
- Some operations, like projection, are more efficient on bags than sets.

Bag Union

An element appears in the union of two bags the sum of the number of times it appears in each bag.

- Example: $\{1, 2, 1\} \cup \{1, 1, 2, 3, 1\} = \{1, 1, 1, 1, 1, 2, 2, 3\}$

Bag Intersection

An element appears in the intersection of two bags the minimum of the number of times it appears in either.

- Example: $\{1, 2, 1, 1\} \cap \{1, 2, 1, 3\} = \{1, 1, 2\}$

Bag Difference

An element appears in the difference $A - B$ of bags as many times as it appears in A, minus the number of times it appears in B; however, never less than 0 times.

- Example: $\{1, 2, 1, 1\} - \{1, 2, 3\} = \{1, 1\}$
- Example: $\{1, 2, 1, 1\} - \{1, 1, 1, 1, 3\} = \{2\}$

Operations on Bags

- Selection applies to each tuple, so its effect on bags is like its effect on sets.
- Projection also applies to each tuple, but as a bag operator, we *do not eliminate duplicates.*
- Products and joins are done on each pair of tuples, so duplicates in bags have no effect on how we operate.

Extended Operators

- Duplicate-elimination operator δ
- Aggregation operators, e.g., sum, average, min, max
- Grouping operator γ combines grouping and aggregation (see the aggregation operators above)
- Extended projection π – extending π with computation
- Sorting operator τ
- Outer-join operator $⟗$, $⟕$, and $⟖$

Outline

Assignment

Let's work on an assignment using paper and pencil/pen ...