Relational Database Operations in SQL - Part II
Ordering, Aggregation, and Grouping

Hui Chen [a]

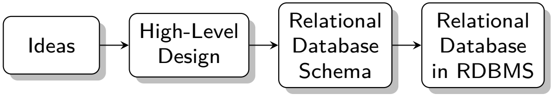
[a]CUNY Brooklyn College, Brooklyn, NY, USA

April 22, 2025

Outline

Outline

Overview


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

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.

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.

Relational Algebra on Sets and Bags

- Projection
- Selection
- Product
- Join
- Union, Intersection, and Difference
- 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

Selected Topics in SQL

- Ordering the Output
- Aggregate Processing
- Eliminating Duplicates
- Subquery
- Views
- Procedural SQL

Outline

Ordering the Output

To order the output of a resulting relation, use the `ORDER BY` clause

```
ORDER BY <list of attributes>
```

Ordering the Output: Example

Example 1:

```
SELECT *
FROM Movies
WHERE studioName = 'Disney' and year = 1990
ORDER BY length, title
```

Example 2:

```
SELECT title, idnum, sname
FROM Courses AS c INNER JOIN Enrollment AS e
WHERE c.idnum = e.cidnum
ORDER BY e.sname;
```

Example 3:

```
SELECT title, idnum, sname
FROM Courses AS c INNER JOIN Enrollment AS e
WHERE c.idnum = e.cidnum
ORDER BY EXTRACT(YEAR FROM e.since);
```

Outline

Eliminating Duplicates

To eliminating duplicates from the output, use the `DISTINCT` keyword after `SELECT`

```
SELECT DISTINCT <list of attributes>
```

Eliminating Duplicates: Examle

Example 1:

```
SELECT DISTINCT *
FROM Movies
WHERE studioName = 'Disney' and year = 1990
ORDER BY length, title
```

Example 2:

```
SELECT DISTINCT title, idnum, sname
FROM Courses AS c INNER JOIN Enrollment AS e
WHERE c.idnum = e.cidnum
ORDER BY e.sname;
```

Example 3:

```
SELECT DISTINCT title, idnum, sname
FROM Courses AS c INNER JOIN Enrollment AS e
WHERE c.idnum = e.cidnum
ORDER BY EXTRACT(YEAR FROM e.since);
```

Outline

Aggregation Operators

SQL defines 5 aggegation operators

`SUM`, `AVG`, `MIN`, `MAX`, and `COUNT`

Aggregation Operators: Examples

Example 1:

```
SELECT AVG(hours)
FROM Courses;
```

Example 2:

```
SELECT COUNT(name)
FROM Students;
```

Example 3:

```
SELECT COUNT(DISTINCT name)
FROM Students;
```

Outline

Grouping

To group tuples in the output, we use a `GROUP BY` clause, following the `WHERE` clause

```
SELECT ...
FROM ...
WHERE ...
GROUP BY <list of attributes>
```

Grouping:Example

Example 1:

```
SELECT e.sname, e.sphone
FROM Enrollment AS e INNER JOIN Courses AS c
WHERE e.cidnum = c.idnum;
GROUP by e.sname, e.sphone;
```

Example 2:

```
SELECT e.sname, e.sphone, SUM(c.hours) as totalhours
FROM Enrollment AS e INNER JOIN Courses AS c
WHERE e.cidnum = c.idnum
GROUP by e.sname, e.sphone;
```

Condition on Grouping

Use the HAVING clause to group only selected tutples.

```
SELECT ...
FROM ...
WHERE ...
GROUP BY <list of attributes>
HAVING <condition>
```

HAVING Clause: Example

Example 1:

```
SELECT e.sname, e.sphone, e.since, SUM(c.hours) as total
FROM Enrollment AS e INNER JOIN Courses AS c
WHERE e.cidnum = c.idnum
GROUP by e.sname, e.sphone;
HAVING e.since >= '2020-01-01';
```

Outline

Assignment

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