SQL:
Data Manipulation Language

Author: Diane Horton

UNIVERSITY OF
TORONTO

[Slides 3-24 are covered by Prep4]

Introduction

- So far, we have defined database schemas and queries mathematically.
- SQL is a formal language for doing so with a DBMS.
- "Structured Query Language", but it's for more than writing queries.
- Two sub-parts:
    - DDL (Data Definition Language), for defining schemas.
    - DML (Data Manipulation Language), for writing queries and modifying the database.

PostgreSQL

- We'll be working in PostgreSQL, an open-source relational DBMS.
- Learn your way around the documentation; it will be very helpful.
- Standards?
  - There are several, the most recent being SQL:2008.
  - The standards are not freely available. Must purchase from the International Standards Organization (ISO).
  - PostgreSQL supports most of it SQL:2008.
  - DBMSs vary in the details around the edges, making portability difficult.

Heads up: SELECT vs σ

- In SQL,
  - "SELECT" is for choosing columns, *i.e.*, π.
  - Example:

```
SELECT surName
FROM Student
WHERE campus = 'StG';
```

- In relational algebra,
  - "select" means choosing rows, *i.e.*, σ.

A high-level language

- SQL is a very high-level language.
  - Say "what" rather than "how."
- You write queries without manipulating data. Contrast languages like Java or C++.
- Provides physical "data independence:"
  - Details of how the data is stored can change with no impact on your queries.
- You can focus on readability.
  - But because the DBMS optimizes your query, you get efficiency.

Basic queries

Meaning of a query with one relation

```
SELECT name
FROM Course
WHERE dept = 'CSC';
```

$\pi_{name}(\sigma_{dept="csc"}(Course))$

… and with multiple relations

```
SELECT name
FROM Offering, Took
WHERE Offering.id = Took.oid and
      dept = 'CSC';
```

$\pi_{name}(\sigma_{Offering.id=Took.id \wedge dept='csc'}(Offering \times Took))$

Temporarily renaming a table

- You can rename tables (just for the duration of the statement):

```
SELECT e.name, d.name
FROM employee e, department d
WHERE d.name = 'marketing'
AND e.name = 'Horton';
```

- Can be convenient vs the longer full names:

```
SELECT employee.name, department.name
FROM employee, department
WHERE department.name = 'marketing'
AND employee.name = 'Horton';
```

- This is like ρ in relational algebra.

Self-joins

- As we know, renaming is *required* for self-joins.
- Example:

```
SELECT e1.name, e2.name
FROM employee e1, employee e2
WHERE e1.salary < e2.salary;
```

* In SELECT clauses

- A * in the SELECT clause means "all attributes of this relation."
- Example:

```
SELECT *
FROM Course
WHERE dept = 'CSC';
```

Renaming attributes

- Use `AS «new name»` to rename an attribute in the result.
- Example:

```
SELECT name AS title, dept
FROM Course
WHERE breadth;
```

Complex Conditions in a WHERE

- We can build boolean expressions with operators that produce boolean results.
  - comparison operators: `=, <>, <, >, <=, >=`
  - and many other operators:
    see section 6.1.2 of the text and chapter 9 of the postgreSQL documentation.
- Note that "not equals" is unusual: `<>`
- We can combine boolean expressions with:
  - Boolean operators: `AND, OR, NOT`.

Example: Compound condition

- Find 3rd- and 4th-year CSC courses:

```
SELECT *
FROM Offering
WHERE dept = 'CSC' AND cnum >= 300;
```

ORDER BY

- To put the tuples in order, add this as the final clause:
  `ORDER BY «attribute list» [DESC]`
- The default is ascending order; DESC overrides it to force descending order.
- The attribute list can include expressions: e.g., `ORDER BY sales+rentals`
- The ordering is the last thing done before the SELECT, so all attributes are still available.

Case-sensitivity and whitespace

- Example query:

```
SELECT surName
FROM Student
WHERE campus = 'StG';
```

- Keywords, like `SELECT`, are not case-sensitive.
  - One convention is to use uppercase for keywords.
- Identifiers, like `Student` are not case-sensitive either.
  - One convention is to use lowercase for attributes, and a leading capital letter followed by lowercase for relations.
- Literal strings, like `'StG'`, are case-sensitive, and require single quotes.
- Whitespace (other than inside quotes) is ignored.

Expressions in SELECT clauses

- Instead of a simple attribute name, you can use an expression in a SELECT clause.
- Operands: attributes, constants
  Operators: arithmetic ops, string ops
- Examples:

```
SELECT sid, grade+10 AS adjusted
FROM Took;

SELECT dept||cnum
FROM course;
```

Expressions that are a constant

- Sometimes it makes sense for the whole expression to be a constant (something that doesn't involve any attributes!).
- Example:

```
SELECT dept, cNum,
   'satisfies' AS breadthRequirement
FROM Course
WHERE breadth;
```

Pattern operators

- Two ways to compare a string to a pattern by:
  - *«attribute»* `LIKE` *«pattern»*
  - *«attribute»* `NOT LIKE` *«pattern»*
- Pattern is a quoted string
  - `%` means: any string
  - `_` means: any single character
- Example:

```
SELECT *
FROM Course
WHERE name LIKE '%Comp%';
```

Aggregation and Grouping

Computing on a column

- We often want to compute something across the values in a column.
- `SUM`, `AVG`, `COUNT`, `MIN`, and `MAX` can be applied to a column in a SELECT clause.
- Also, `COUNT(*)` counts the number of tuples.
- We call this aggregation.
- Note: To stop duplicates from contributing to the aggregation, use `DISTINCT` inside the brackets. (Does not affect `MIN` or `MAX`.)
- Example: aggregation.txt

Grouping

- Example: group-by.txt
- If we follow a SELECT-FROM-WHERE expression with GROUP BY <attributes>
  - The tuples are grouped according to the values of those attributes, and
  - any aggregation gives us a single value per group.

Restrictions on aggregation

- If any aggregation is used, then each element of the SELECT list must be either:
  - aggregated, or
  - an attribute on the GROUP BY list.
- Otherwise, it doesn't even make sense to include the attribute.

Continuing on with SQL

HAVING Clauses

- Example: having.txt
- WHERE let's you decide which tuples to keep.
- Similarly, you can decide which *groups* to keep.
- Syntax:

```
. . .
GROUP BY «attributes»
HAVING «condition»
```

- Semantics:
  Only groups satisfying the condition are kept.

Restrictions on HAVING clauses

- Outside subqueries, HAVING may refer to attributes only if they are either:
  - aggregated, or
  - an attribute on the GROUP BY list.
- (Same requirement as for SELECT clauses with aggregation)