SQL: Structured Query Language

Abdu Alawini

University of Illinois at Urbana-Champaign

CS411: Database Systems

Some slides are adopted from D. Maier, S. Davidson and A. Parameswaran

Learning Objectives

After this lecture, you should be able to:

- Write **single-relation SQL** queries
- Write SQL queries with **complex WHERE conditions**

SQL: Structured Query Language

The standard language for relational data

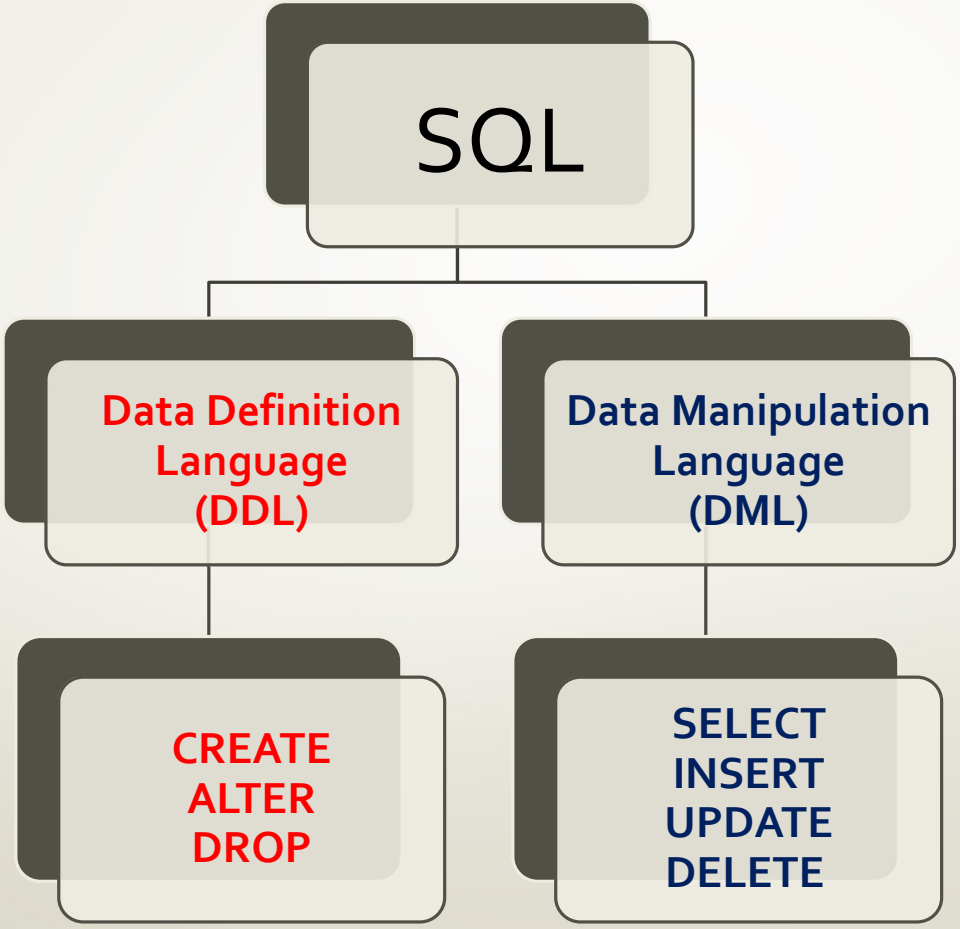
- Invented by folks at IBM, esp. Don Chamberlin

Separated into

- DDL (data definition language)
- DML (data manipulation language)

DML based on relational calculus, which we discuss later

SQL – the language we use to talk to the Database Management System



SQL (cont.)

SQL is a standard…
and there have been a series of SQL standards:
1986, 1989, 1992 (SQL2), 1999 (SQL3), …,
SQL:2011

But DBMS products differ in how much of the standard they support … and how many extra features they have.

Single-Relation Query Form

- The principal form of a single-relation query is:

SELECT desired attributes
FROM one table (relation)
WHERE condition about tuples of the table

Example Query

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

```
SELECT      Number, Owner
FROM        Account
WHERE       Type = "savings";
```

Number	Owner
103	J. Smith

How a Single-Relation SQL query is evaluated

Third, the SELECT clause tells us
which columns to keep in the query answer.

```
SELECT    Number, Owner
FROM      Account
WHERE     Type = "savings"
```

First, the FROM clause
tells us the input tables.

Second, the WHERE clause
is evaluated for all rows from the
input table.

Renaming Attributes

- If you want the result to have different attribute names, use "AS <new name>" to rename an attribute.
- Example based on Account(number, owner, balance, type):

```
SELECT Number AS Acc_Num, Owner
FROM Accounts
WHERE Type = "savings";
```

Acc_Num	Owner
103	J. Smith

Another Database Schema

- We will be using the following database schema as a running example.
  - Underline indicates key attributes.

    Drinks(<u>name</u>, manf)

    Cafe(<u>name</u>, addr, license)

    Customers(<u>name</u>, addr, phone)

    Likes(<u>customer</u>, <u>drink</u>)

    Sells(<u>cafe</u>, <u>drink</u>, price)

    Frequents(<u>customer</u>, <u>cafe</u>)

Expressions in SELECT Clauses

- Any expression that makes sense can appear as an element of a SELECT clause.

- Example: from Sells (cafe, drink, price):

```
SELECT cafe, drink, price * 120 AS priceInYen
FROM Sells;
```

Creating temporary tables using INTO

We can create a name for the query answer:

```
SELECT   Owner INTO temp3
FROM     Account
WHERE    Type = "checking";
```

temp3 can be used as a table in subsequent queries!

REMEMBER TO DELETE YOUR TEMPORARY TABLES!

Complex Conditions in WHERE Clause

- From Sells (cafe, drink, price), find the price Caffe bene charges for Mocha:

```
SELECT price
FROM Sells
WHERE cafe = 'Caffe bene' AND
      drink = 'Mocha';
```

WHERE Clause Syntax

What you can use in WHERE:

- attribute names of the relation(s) used in the FROM.
- comparison operators: =, <>, <, >, <=, >=
- apply arithmetic operations: stockprice*2
- operations, comparisons on strings
- pattern matching: s LIKE p
- special stuff for comparing dates and times.

See the textbook for more…

Then, create bigger expressions using Boolean connectives

Syntax: Patterns and "LIKE"

- WHERE clauses can have conditions in which a string is compared with a pattern, to see if it matches.

- General form: <Attribute> LIKE <pattern>
  or <Attribute> NOT LIKE <pattern>

- Pattern is a quoted string with
  - % "any string"
  - _ "any character."

Example

- From Customers(name, addr, phone) find the customers that have phone numbers with middle three numbers 555:

```
SELECT name
FROM Customers
WHERE phone LIKE '%555-_ _ _ _';
```

Ordering the results

```
SELECT *
FROM  Account
WHERE Name LIKE 'J%'
ORDER BY Balance
```

- Ordering is ascending, unless you specify the DESC keyword.
- Ties are broken by the second attribute on the ORDER BY list, etc.