INTRODUCTION TO SQL

SQL

SFWR ENG 3DB3 – FALL 2016

MICHAEL LIUT (LIUTM@MCMASTER.CA)
DEPARTMENT OF COMPUTING AND SOFTWARE
MCMASTER UNIVERSITY

WHY SQL?

- SQL is a very-high-level language.
  - Structured Query Language
  - Say "what to do" rather than "how to do it."
  - Avoid a lot of data-manipulation details needed in procedural languages like C++ or Java.
- DBMSs determine the "best" method of executing a query.
  - This is called "query optimization".

2

DATABASE SCHEMAS IN SQL

- SQL is primarily a query language, used for retrieving data from a database.
  - Data Manipulation Language (DML)
- But SQL also includes a data-definition component for describing database schemas.
  - Data Definition Language (DDL)

3

SELECT-FROM-WHERE STATEMENTS

SELECT → desired attribute(s)

FROM → one or more tables (i.e. entity-sets)

WHERE → condition about tuples of the tables

4

RECALL OUR EXAMPLE

- Our SQL queries will be based on the following database schema.
  - Underlines indactes key attributes.

Beers (name, manf)
Bars (name, addr, license)
Drinkers (name, addr, phone)
Likes (drinker, beer)
Sells (bar, beer, price)
Frequents (drinker, bar)

5

EXAMPLE

- Using Beers(name, manf), what beers are made by Anheuser-Busch?

```
SELECT   name
FROM     Beers
WHERE    manf = 'Anheuser-Busch';
```

6

RESULT OF QUERY

name
Bud
Bud Light
Michelob
…

- The answer is a relation with a single attribute, name, and tuples with the name of each beer by Anheuser-Busch, such as Bud.

7

MEANING OF SINGLE-RELATION QUERY

- Begin with the relation in the FROM clause.
- Apply the selection indicated by the WHERE clause.
- Apply the extended projection indicated by the SELECT clause.

8

OPERATIONAL SEMANTICS - GENERAL

- Think of a *tuple variable* visiting each tuple of the relation mentioned in FROM.
- Check if the tuple assigned to the tuple variable satisfies the WHERE clause.
- If so, compute the attributes or expressions of the SELECT clause using the components of this tuple.

9

OPERATIONAL SEMANTICS

Tuple-variable *t* loops over all tuples

name	manf
Bud	Anheauser-Busch

If so, include t.name in the result

Check if Anheuser-Busch

10

EXAMPLE

- What beers are made by Anheuser-Busch?

```
SELECT name
FROM   Beers
WHERE  manf = 'Anheuser-Busch';
```

OR

```
SELECT t.name
FROM   Beers t
WHERE  t.manf = 'Anheuser-Busch';
```

NOTE: these two queries are identical.

11

ASTERISKS (*) IN SELECT CLAUSES

- When there is one relation in the **FROM** clause, * in the **SELECT** clause stands for "all attributes of this relation".
- **Example**: Using Beers(name, manf):

```
SELECT  *
FROM    Beers
WHERE   manf = 'Anheuser-Busch';
```

12

RESULT OF QUERY

name	manf
Bud	Anheuser-Busch
Bud Light	Anheuser-Busch
Michelob	Anheuser-Busch
…	…

- The result now has each of the attributes of Beers.

13

RENAMING ATTRIBUTES

- If you want the result to have different attribute names, use "AS <new name>" to rename an attribute.
- **Example**: Using Beers(name, manf):

```
SELECT  name AS beer, manf
FROM    Beers
WHERE   manf = 'Anheuser-Busch';
```

14

RESULT OF QUERY

beer	manf
Bud	Anheuser-Busch
Bud Light	Anheuser-Busch
Michelob	Anheuser-Busch
…	…

- The result now has each of the attributes of Beers.

15

EXPRESSIONS IN SELECT CLAUSES

- Any valid expression can appear as an element of a SELECT clause.
- **EXAMPLE**: Using Sells(bar, beer, price):

```
SELECT  bar, beer, price*80 AS priceInJPY
FROM    Sells;
```

16

RESULT OF QUERY

bar	beer	priceInJPY
Joe's	Bud	285
Sue's	Miller	342
John's	Michelob	532
…	…	…

- The result now has each of the attributes of Beers.

17

EXAMPLE: CONSTANTS AS EXPRESSIONS

- Using Likes(drinker, beer):

```
SELECT   drinker, 'likes Bud' AS whoLikesBud
FROM     Likes
WHERE    beer = 'Bud';
```

18

RESULT OF QUERY

drinker	whoLikesBud
George	likes Bud
Frank	likes Bud
Jenny	likes Bud
…	…

19

COMPLEX CONDITIONS IN WHERE CLAUSE

- Boolean operators: **AND, OR, NOT**.
- Comparison operators: =, <>, <, >, <=, >=.

20

EXAMPLE: COMPLEX CONDITION

- Using Sells(bar, beer, price), find the price that Joe's Bar charges for Bud.

```
SELECT price
FROM Sells
WHERE bar = 'Joe``s Bar' AND beer = 'Bud';
```

21

PATTERNS

- A condition can compare a string to a pattern by:
  - <Attribute> **LIKE** <pattern> **or** <Attribute> **NOT LIKE** <pattern>
- *Pattern* is a quotes string:
  - % = "any string";
  - __ = "any character".

22

EXAMPLE: LIKE

- Using Drinkers(name, addr, phone) find the drinkers with exchange *555*:

```
SELECT name
FROM DRINKERS
WHERE phone LIKE '%555-____';
```

23

NULL VALUES

- Tuples in SQL relations can have NULL as a value for one or more components.
- The meaning is dependent on the context. In general, there are two common cases:
  1. *Missing Values*: e.g., we know Joe's Bar has some address, but we don't know what it is.
  2. *Inapplicable*: e.g., the value of attribute spouse for an unmarried person.

24

COMPARING NULL'S TO VALUES

- The logic of a conditions in SQL are 3 pronged:
  1. **TRUE**
  2. **FALSE**
  3. **UNKNOWN**
     - Comparing any value (including NULL) with NULL yields UNKNOWN.

A tuple is in a query answer iff the **WHERE** clause is **TRUE** (not **FALSE** or **UNKNOWN**).

25