SFWRENG 3DB3 Tutorial

Week 4

Oct 4, 2021

Levin Noronha

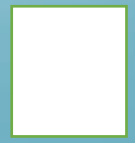
Select-From-Where

SELECT desired attributes

FROM one or more tables

WHERE condition about tuples of the tables

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



Complex Conditions in WHERE Clause

- Boolean operators AND, OR, NOT
- Comparisons =, <>, <, >, <=, >=
- Example: Using Sells(bar, beer, price), find the price Joe's Bar charges for Bud

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

- SQL includes a **between** comparison operator
- Example: Find the names of all instructors with salary between $90,000 and $100,000 (that is, *salary* >= 90000 AND *salary* <= 100000

```
select name
from instructor
where salary between 90000 and 100000
```

Credit: Zheng Zheng

LIKE operator

- Search for a pattern in a column of string values
  - WHERE column_name LIKE pattern
- Pattern can include wildcards like:
  - “%”: wildcard that represents 0 or more characters
  - “_”: wildcard that represents 1 character
  - and some other wildcards commonly found in regex
- For e.g., using Drinkers(name, address, phone), find drinkers whose names begins with “J” and ends with “n”
  - SELECT name
    FROM Drinkers
    WHERE name LIKE “J%n”

NULL values

- Columns use NULL in place of missing or inapplicable values
- Cannot use comparison operators to test for NULL
- Find drinkers who do not have an address
  - SELECT name
    FROM Drinkers
    WHERE address IS NULL
- Find drinkers who have an address
  - SELECT name
    FROM Drinkers
    WHERE address IS NOT NULL

Multi-relational Queries

- Combine data from multiple relations
- Distinguish attributes of the same name using “<relation>.<attribute>”

CROSS JOIN

- VERY EXPENSIVE. NOT COMMONLY USED.
- Produces cartesian product of two relations
- Number of rows in result = number of rows in first table * number of rows in second table
- For e.g.,

  SELECT * FROM beers, drinkers

  SELECT * FROM beers CROSS JOIN drinkers

name	manf
Bud	Budweiser
Bud Light	Coors

name	addr	phone
Barry	1 King St	123
John	2 Queen St	345

name	manf	name	addr	phone
Bud	Budweiser	Barry	1 King St	123
Bud Light	Coors	John	2 Queen St	345
Bud	Budweiser	Barry	1 King St	123
Bud Light	Coors	John	2 Queen St	345

EQUI JOIN

- JOIN tables Likes and Frequents to find each drinker's favourite drink and bar
- Add a WHERE clause

```
SELECT l.drinker, l.beer, f.bar
FROM likes l, frequents f
WHERE l.drinker = f.drinker
---------------------------------------
SELECT l.drinker, l.beer, f.bar
FROM likes l JOIN frequents f
ON l.drinker = f.drinker
```

- Note: JOIN is interchangeable with INNER JOIN

Likes

drinker	beer
Barry	Bud
John	Bud Light

Frequents

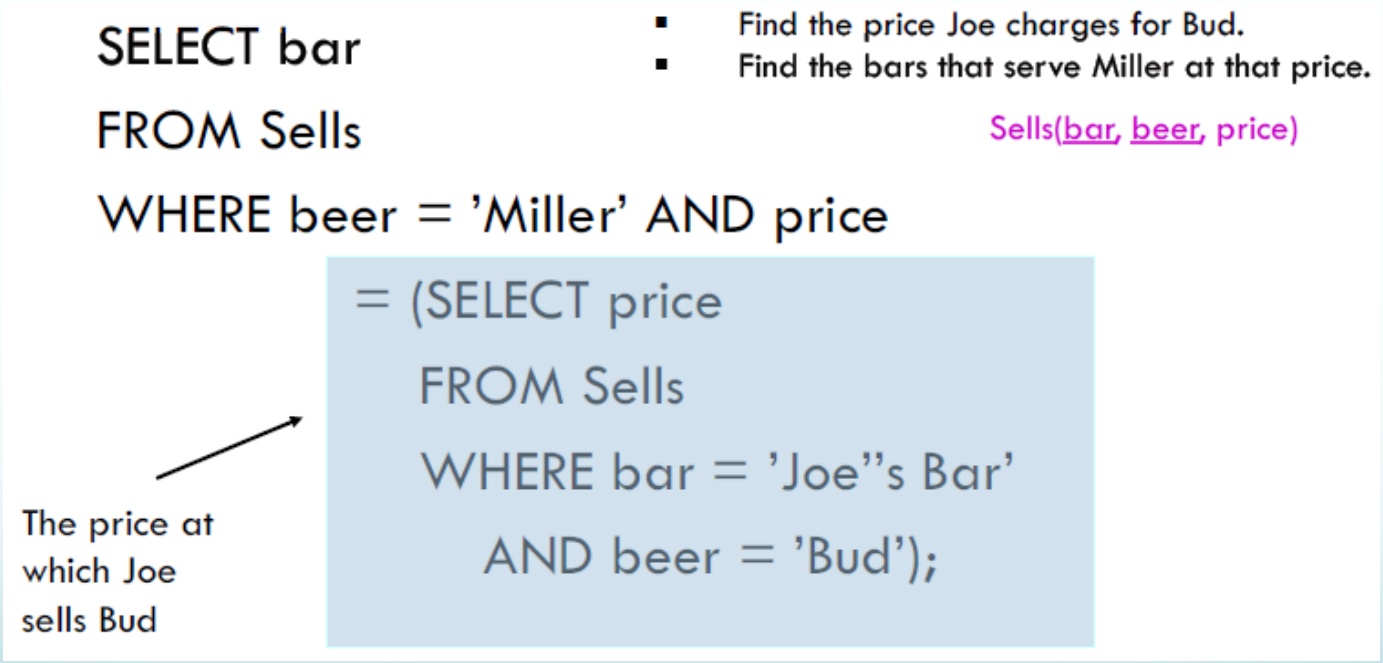
drinker	bar
Barry	Joe's
John	Callaghan's
Mary	Joe's

l.drinker	l.beer	f.drinker	f.bar
Barry	Bud	Barry	Joe's
John	Bud Light	John	Callaghan's

l.drinker	l.beer	f.bar
Barry	Bud	Joe's
John	Bud Light	Callaghan's

Subqueries

- Using Sells(bar, beer, price), find the bars that serve Miller for the same price Joe charges for Bud



ANY and ALL operators

- Perform comparison between a single value and a list of values
- Can be used with WHERE, and HAVING
- Syntax:
  - SELECT …
    FROM table
    WHERE column_name COMPARE_OP ANY/ALL
      (SELECT column_name
      FROM …
      WHERE …)
- ANY returns TRUE if any of subquery values satisfy the condition
- ALL returns TRUE if all of subquery values satisfy the condition

ANY operator

- Using Sells(bar, beer, price) and Likes(drinker, beer), find drinkers that like beers sold at price greater than 10

```
SELECT DISTINCT drinker
FROM Likes
WHERE beer = ANY
        (SELECT DISTINCT beer
        FROM Sells
        WHERE price > 10)
```

ALL operator

- Using Sells(bar, beer, price) and Beers(name, manf), find the beer manufacturer(s) whose beer(s) sell at the highest price

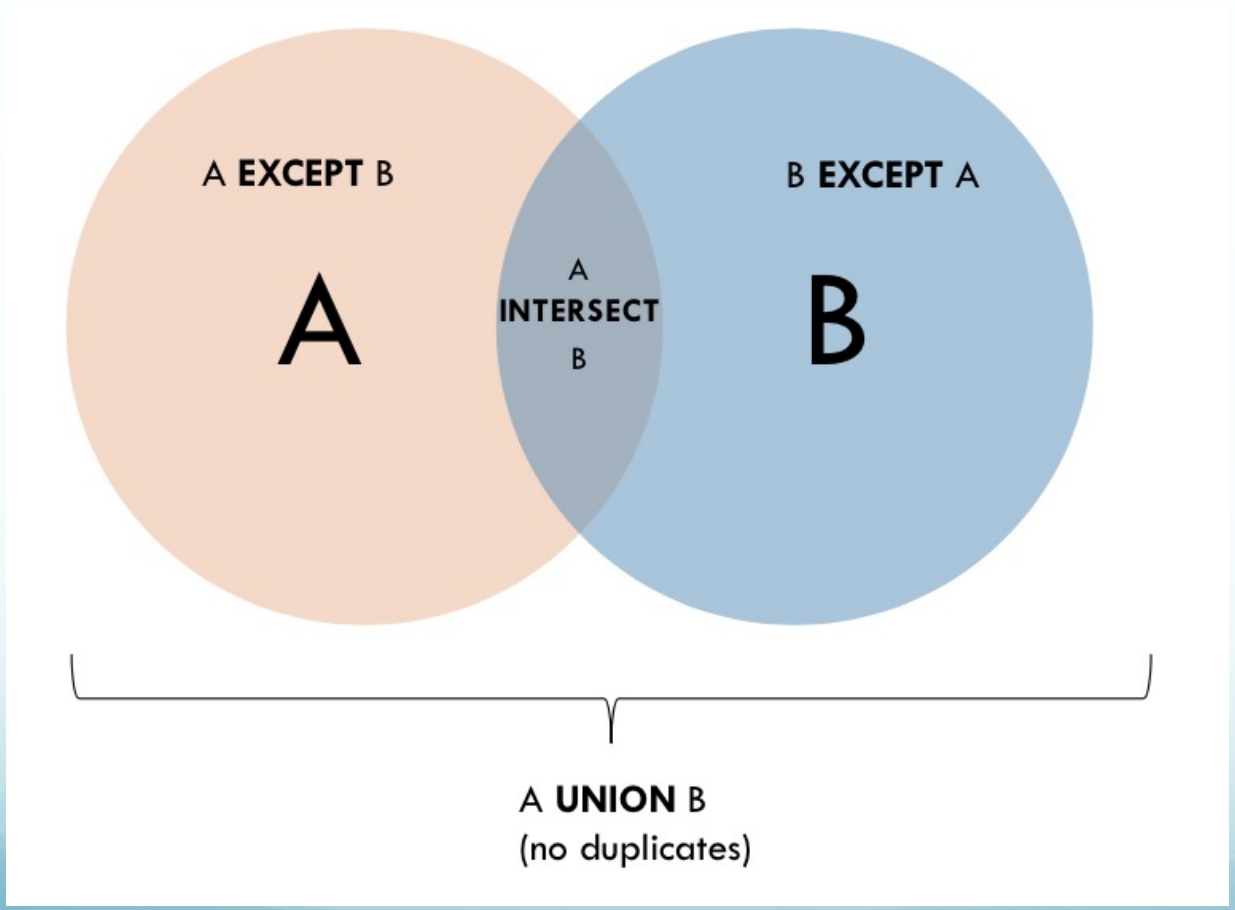
```
SELECT DISTINCT manf
FROM Sells s INNER JOIN Beers b ON s.beer = b.name
WHERE price >= ALL
        (SELECT price
        FROM Sells)
```

IN operator

- Allows multiple values to be specified in WHERE clause
- WHERE col IN (value1, value2, …)
- WHERE col IN (SELECT col FROM …)
- Using Drinkers(name, addr, phone) and Frequents(drinker, bar), find drinkers who do not frequent bars
  - SELECT name
    FROM Drinkers
    WHERE name NOT IN (Select drinker FROM Frequents)

Union, Intersection, Difference

- (<subquery>) UNION/INTERSECT/EXCEPT (<subquery)
-



Note: requires both relations to have the same number of columns with compatible types

name	manf
Bud	Budweiser
Bud Light	Budweiser
Coors	Coors
Heineken	Heineken

INTERSECT

name	manf
Bud	Budweiser
Bud Light	Budweiser

=

name	manf
Bud	Budweiser
Bud Light	Budweiser

name	manf
Bud	Budweiser
Bud Light	Budweiser
Coors	Coors
Heineken	Heineken

EXCEPT

name	manf
Bud	Budweiser
Bud Light	Budweiser

=

name	manf
Coors	Coors
Heineken	Heineken

name	manf
Bud	Budweiser
Bud Light	Budweiser
Coors	Coors
Heineken	Heineken

UNION

name	manf
Bud	Budweiser
Boxer	Boxer

=

name	manf
Bud	Budweiser
Bud Light	Budweiser
Coors	Coors
Heineken	Heineken
Boxer	Boxer

Note: to preserve duplicates, use UNION ALL instead.

EXCEPT example

Note: EXCEPT requires both relations to have the same number of columns with compatible types

- Find beer manufacturers that only sell one beer

(SELECT * FROM Beers)

EXCEPT

(SELECT b1.name, b2.manf FROM Beers b1, Beers b2

WHERE b1.manf = b2.manf AND b1.name <> b2.name)

Beers

name	manf
Bud	Budweiser
Bud Light	Budweiser
Coors	Coors
Heineken	Heineken

EXCEPT

name	manf
Bud	Budweiser
Bud Light	Budweiser

=

name	manf
Coors	Coors
Heineken	Heineken