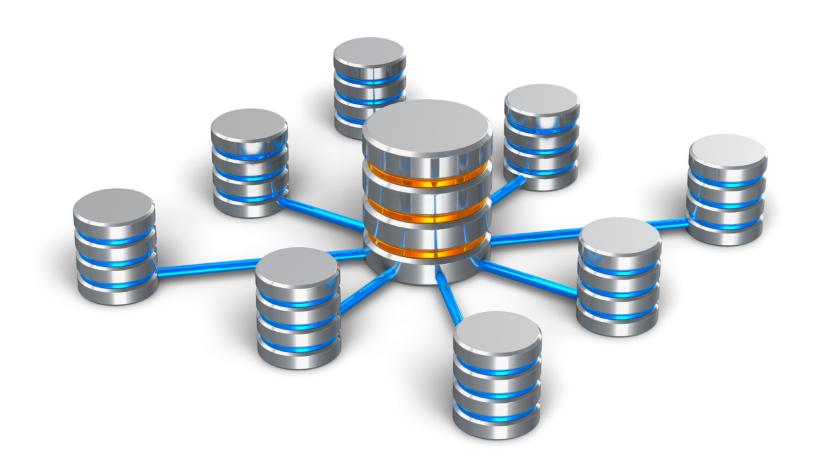
CSE 180

Constraints, and Queries



Dev October 12 2023, Section 3

NOT NULL constraint

- The specified column cannot have a NULL value
- Always a column constraint

```
CREATE TABLE Bars (
  bar VARCHAR(30),
  addr VARCHAR(50),
  license VARCHAR(50) NOT NULL,
  PRIMARY KEY (bar)
);
```

UNIQUE constraint

- Data in the column, or groups of columns, is unique in the table
- Can be a column constraint or a table constraint

```
CREATE TABLE Bars (
  bar VARCHAR(30),
  addr VARCHAR(50),
  license VARCHAR(50) UNIQUE,
  PRIMARY KEY (bar)
);
```

```
CREATE TABLE Bars (
  bar VARCHAR(30),
  addr VARCHAR(50),
  license VARCHAR(50),
  PRIMARY KEY (bar),
  UNIQUE(license)
);
```

SELECT statement

- Retrieves rows from zero or more tables
- All tables in FROM are computed upon
- Conditions under WHERE are looked for
- Other operators such as WITH, HAVING, GROUPBY will be explored later in the course

SELECT Statement examples

```
SELECT * FROM Bars;

SELECT * FROM Sells WHERE price < 8;

SELECT beer FROM Sells WHERE price < 8 AND bar='The Red Room';</pre>
```

SELECT WHERE

- Where clause is used to filter records
- Examples:
- SELECT * FROM table WHERE firstName='Elle';
- SELECT * FROM table WHERE id=1;
- SELECT * FROM beers WHERE price ≤ 2.5;

Operators in WHERE

Operator	Meaning	Use
=	Equal	Used with most types to check equality
<, >, <=, >=	Comparison	Works on Numeric and Date types
<>	Not Equal to	Works on most types
BETWEEN	Value in range	eg.WHERE Price BETWEEN 20 AND 30;
LIKE	Pattern match	WHERE beer LIKE 'I%';: Looks for all beers starting with I
IN	Checks values in a column	WHERE courseNo IN (180,130,111); or WHERE courseNo in (SELECT)
IS NULL/ IS NOT NULL	Checks if a value is/ isn't null	Note that equality and comparison operators cannot be used with NULL. NULL is not equal to NULL

AND, OR, NOT

- AND: Combine multiple WHERE clauses, all must be TRUE
- OR: Combine multiple WHERE clauses, at least one must be TRUE
- NOT: Pick Complement

```
SELECT * FROM BEERS WHERE beer='IPA' AND manf='Russian River Brewing Co.';

SELECT * FROM BEERS WHERE beer='White Claw' OR beer='IPA';

SELECT * FROM BEERS WHERE NOT beer='White Claw';
```

ORDER BY

 Orders results into ascending or descending orders on a particular column or a group of columns

```
SELECT Beer, Bar, Price FROM Sells
WHERE Price<5
ORDER BY Price ASC;
```

JOIN

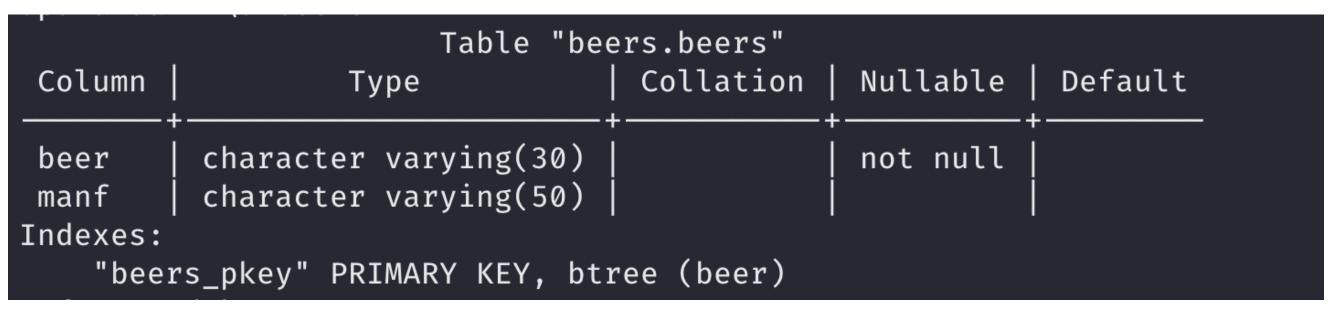
- Joins combine two or more tables based on a related column
- They can be explicitly specified with FROM table LEFT/RIGHT/INNER/FULL JOIN table ON column
- Or can be specified with
 FROM tablea a, tableb b
 WHERE a related column = b related column
- There are different types of JOIN including self Join

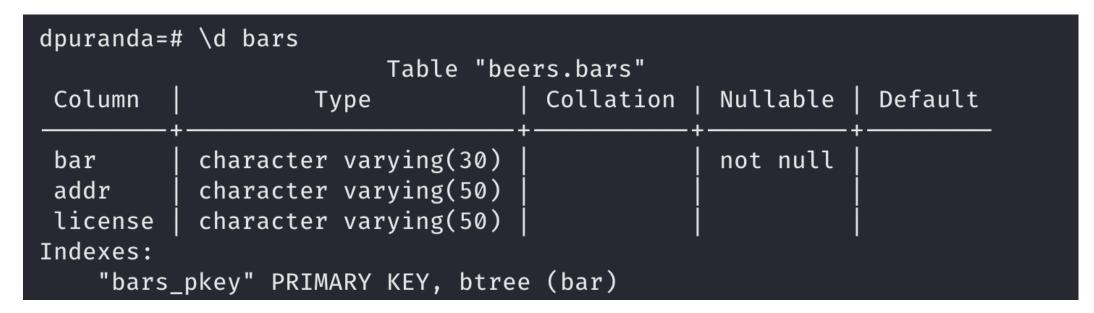
Breaking Down Complex Queries

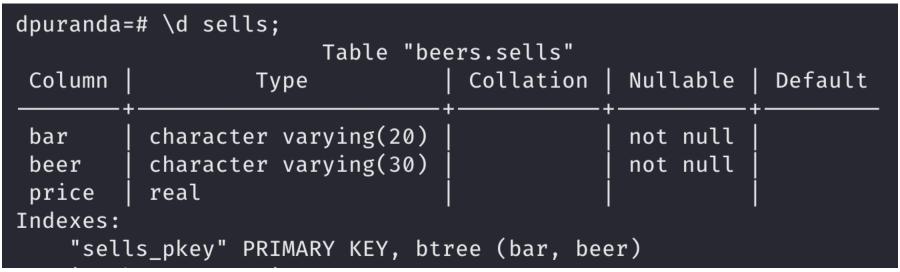
- 1. Get Data: Figure out the required Tables: These will inform your FROM
- 2. Figure out the necessary JOIN: which tables do you need to join? On which columns?
- 3. Filter: Use your WHERE clause to filter the cartesian product into required results
- 4. Return: SELECT the fields necessary and Name them
- 5. ORDER BY: Select Ascending or Descending order

Breaking Down Complex Queries

Find bars whose license is not NULL, and display all the beers sold in each bar which cost less than or equal to \$5 in ascending order of the barName and the price. For each bar, your results should appear as barName, beer, manufacturer, price.







Picking Tables

- Find bars whose license is not NULL, and display all the beers sold in each bar which cost less than or equal to \$5 in ascending order of the barName and the price. For each bar, your results should appear as barName, beer, manufacturer, price.
 - Identify all the tables you need to operate on
 - Which attributes do we need to utilize?

Picking Tables

- Find bars whose license is not NULL, and display all the beers sold in each bar which cost less than or equal to \$5 in ascending order of the barName and the price. For each bar, your results should appear as barName, beer, manufacturer, price.
 - We need to use bars table (for license), and sells table (for cost) and beers table (for manufacturer)
 - We do not need the frequents or drinkers table
 - This gives us an idea for our FROM clause

JOINS

```
SELECT ba.bar AS barName, s.beer, be.manf AS manufacturer, price
FROM beers be, bars ba, sells s
WHERE ba.bar = s.bar
AND be.beer = s.beer;
```

WHERE conditions

- Find bars whose license is not NULL, and display all the beers sold in each bar which cost less than or equal to \$5 in ascending order of the barName and the price. For each bar, your results should appear as barName, beer, manufacturer, price.
 - Need to check license and price
 - WHERE condition is where JOINS will be performed as well
 - What is the common factor between Table Sells and Table Bars?

Picking SELECT

- Find bars whose license is not NULL, and display all the beers sold in each bar which cost less than or equal to \$5 in ascending order of the barName and the price. For each bar, your results should appear as barName, beer, manufacturer, price.
 - The attributes that we need to display will be the attributes we SELECT
 - Appear as tells us to us AS modifier to set column name

ORDER BY

Find bars whose license is not NULL, and display all the beers sold in each bar which cost less than or equal to \$5 in ascending order of the barName and the price. For each bar, your results should appear as barName, beer, manufacturer, price.

Putting it all Together

```
SELECT ba.bar AS barName, s.beer, be.manf AS manufacturer, price FROM beers be, bars ba, sells s
WHERE ba.bar = s.bar
AND be.beer = s.beer
AND ba.license IS NOT NULL
AND s.price \leq 5
ORDER BY barName, price ASC;
```

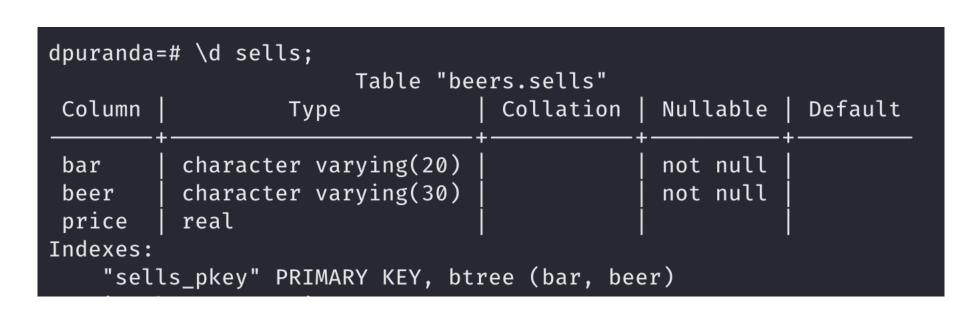
Self JOIN

 List all the Bars that sell beer that is sold at multiple bars, your output should be Bar, Beer, Price, ordered by beer name and price

FROM

- List all the Bars that sell beer that is sold at multiple bars, your output should be Bar, Beer, Price
- Sells, but we need two instances:
- Sells S1, and Sells S2





JOIN

- List all the Bars that sell beer that is sold at multiple bars, your output should be Bar, Beer, Price
- Here we need a self join to make sure that while the barname is not the same, the beer is

```
WHERE S1.bar < > S2.bar

AND S1.beer = S2. Beer
```

SELECT

 List all the Bars that sell beer that is sold at multiple bars, your output should be Bar, Beer, Price

SELECT S1.bar, S1.beer, S1.price

Putting it all together

```
AND S1.beer = S2.beer
ORDER BY S1.beer, S1.price;

SELECT DISTINCT S1.bar, S1.beer, S1.price
FROM Sells S1, Sells S2
WHERE S1.bar 	$ S2.bar
AND S1.beer = S2.beer
ORDER BY S1.beer, S1.price;
```

SELECT S1.bar, S1.beer, S1.price

FROM Sells S1, Sells S2

WHERE S1.bar ♦ S2.bar