Grouping, Aggregation, Subqueries

PDBM 7.3–7.3.1.4

Dr. Chris Mayfield

Department of Computer Science
James Madison University

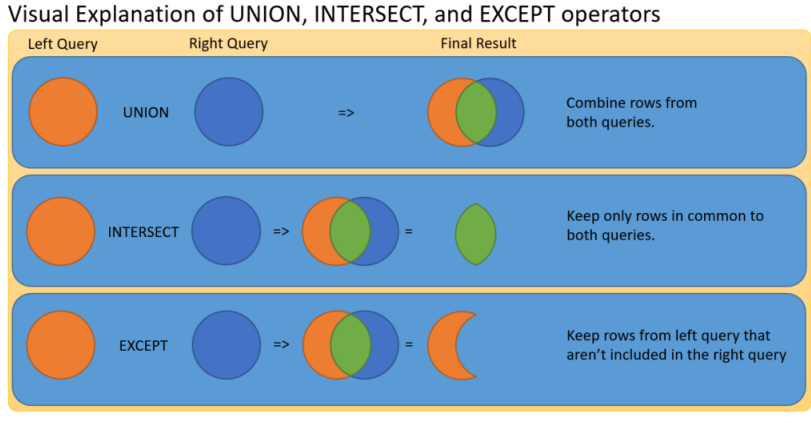
Feb 22, 2022

state	city	revenue
Maharashtra	Achalpur	10000
Uttar Pradesh	Achhnera	20000
Gujarat	Adalaj	10000
Uttar Pradesh	Agra	30000
Gujarat	Ahmedabad	40000
Maharashtra	Ahmednagar	10000
Maharashtra	Akola	40000

⇩

state	sum(revenue)
Maharashtra	60000
Uttar Pradesh	50000
Gujarat	50000

©graspsql.com



Aggregation

Common functions

- COUNT, SUM, AVG, MIN, MAX
- https://www.postgresql.org/docs/11/static/functions-aggregate.html

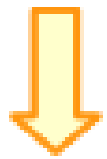
```
SELECT min(year), max(year), count(year)
FROM movie
WHERE title = 'Frozen';
```

More commonly used in groups:

```
SELECT year, count(*)
FROM movie
WHERE title = 'Frozen'
GROUP BY year
ORDER BY year;
```

GROUP BY

state	city	revenue
Maharashtra	Achalpur	10000
Uttar Pradesh	Achhnera	20000
Gujarat	Adalaj	10000
Uttar Pradesh	Agra	30000
Gujarat	Ahmedabad	40000
Maharashtra	Ahmednagar	10000
Maharashtra	Akola	40000



state	sum(revenue)
Maharashtra	60000
Uttar Pradesh	50000
Gujarat	50000

HAVING vs WHERE

- `WHERE` = before grouping
- `HAVING` = after grouping

```
-- titles used more than 50 times
SELECT m.title, k.kind, count(*)
FROM movie AS m
  JOIN kind_type AS k ON m.kind_id = k.id
WHERE year > 2010
  AND m.title NOT LIKE '(%'
GROUP BY m.title, k.kind
HAVING count(*) > 50
ORDER BY count DESC
```

Aggregation with NULLs

Three ways of counting:

- `SELECT count(*) FROM cast_info; -- count rows`
- `SELECT count(person_id) FROM cast_info; -- count values`
- `SELECT count(DISTINCT person_id) FROM cast_info;`

NULL values are ignored!

```
SELECT
  m.title, m.year, m.kind_id, -- see GROUP BY
  max(r.info) AS max_runtime, count(*)
FROM movie AS m
  LEFT JOIN movie_info AS r
    ON m.id = r.movie_id AND r.info_id = 1
GROUP BY m.title, m.year, m.kind_id
```

NULLs can be confusing!

R	
a	*b*
NULL	NULL

```
SELECT a, count(b)
FROM R
GROUP BY a;
```

- Returns (NULL, 0)

```
SELECT a, sum(b)
FROM R
GROUP BY a;
```

- Returns (NULL, NULL)

SQL Exercises: `imdb`

How many of each kind of movie?

How many of each type of info?

What else would you like to know?

Hello TPC-H

The Transaction Processing Performance Council is a non-profit corporation that defines performance benchmarks for comparing database systems.

- All the major database companies from industry
- Founded in 1988; new benchmarks every few years

TPC-H is a business-oriented "decision support" database consisting of customers, orders, parts, suppliers, etc.

- The data is synthetically generated
- A driver simulates many transactions

Set Operations

PDBM 7.3.1.11 and PG 7.4, 10.5

Set operations

These queries eliminate duplicates (unless you use `ALL` keyword)

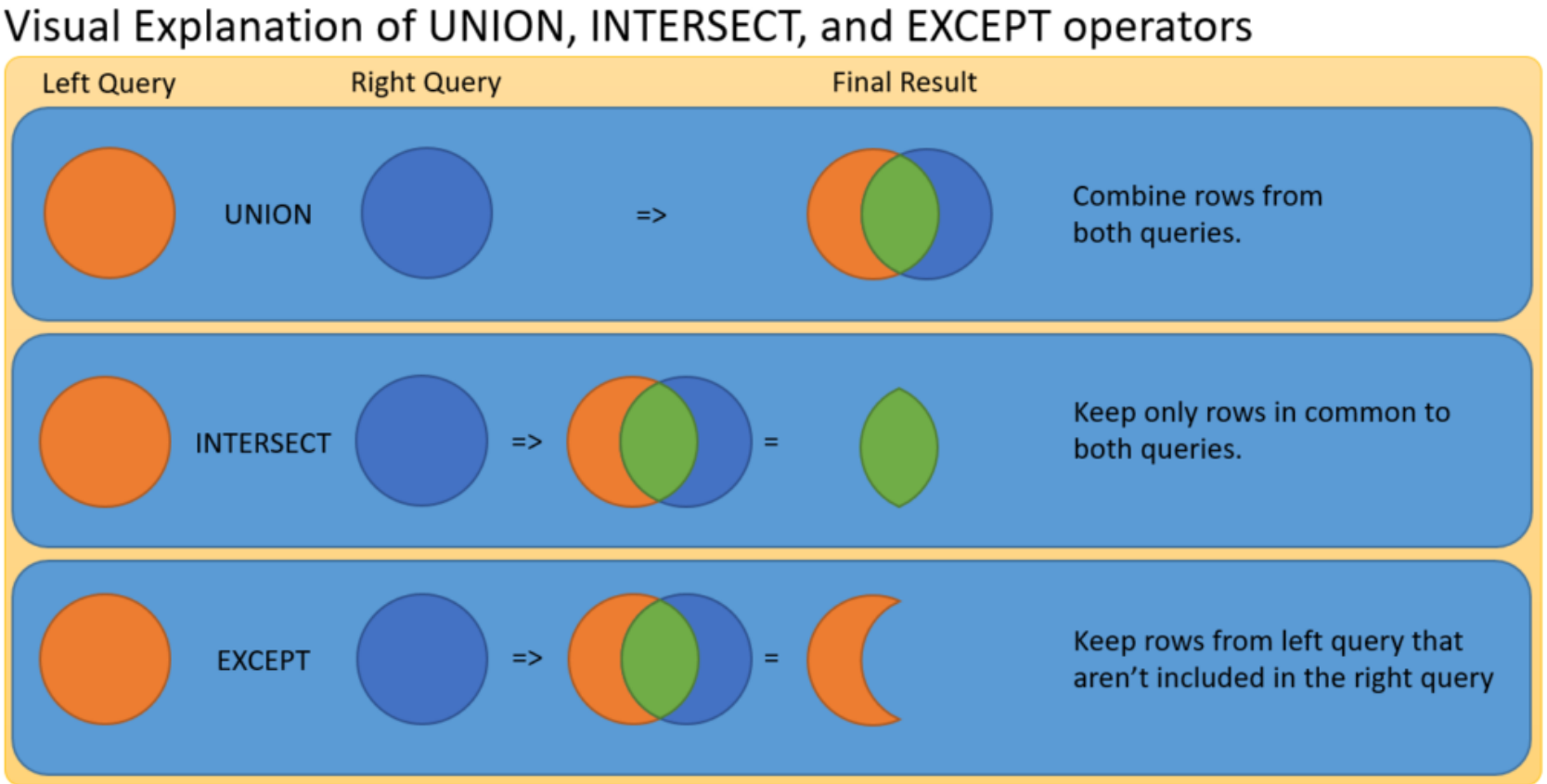
```
SELECT c_name, c_address FROM customer
UNION -- or UNION ALL
SELECT s_name, s_address FROM supplier;

SELECT c_name, c_address FROM customer
INTERSECT -- or INTERSECT ALL
SELECT s_name, s_address FROM supplier;

SELECT c_name, c_address FROM customer
EXCEPT -- or EXCEPT ALL
SELECT s_name, s_address FROM supplier;
```

Set operations are not joins!

Venn diagrams



https://www.essentialsql.com/sql-set-operations-sql-server/

Example query

```
-- customers with negative balance
SELECT c_custkey, c_name, c_acctbal
FROM customer
  JOIN orders ON c_custkey = o_custkey
  JOIN lineitem ON o_orderkey = l_orderkey
WHERE c_acctbal < 0
EXCEPT
-- customers who live in Africa
SELECT c_custkey, c_name, c_acctbal
FROM customer
  JOIN nation ON c_nationkey = n_nationkey
  JOIN region ON n_regionkey = r_regionkey
WHERE r_name = 'AFRICA'
-- ORDER BY and LIMIT apply to entire query
ORDER BY c_acctbal
LIMIT 50
```

Note: HW4 doesn't require set operations

Subqueries

PDBM 7.3.1.7–10 and PG 7.2, 9.22

Subqueries with scalar values

"Find the name of the professor who teaches CS 474."

```
SELECT name
FROM professor
  JOIN teach ON pid = prof_pid
WHERE dept = 'CS' AND number = 474;
```

Another way:

```
SELECT name
FROM professor
WHERE pid = (  -- subquery in WHERE should return a single row
  SELECT prof_pid
  FROM teach
  WHERE dept = 'CS' AND number = 474);
```

More common: FROM subqueries

```
SELECT name
FROM MovieExec, (
  SELECT producerC#
  FROM Movies, StarsIn
  WHERE title = movieTitle
    AND year = movieYear
    AND starName = 'Harrison Ford'
  ) AS Ford
WHERE cert# = Ford.producerC#;
```

Subquery relations must be given a name

- (i.e., `AS Ford`)

Subquery expressions

Let R be a relation and t be a tuple from R

- `EXISTS R`
- `t IN R`
- `t > ALL R`
- `t > ANY R`

"Remove from the `Ships` relation all ships that sunk in battle."

```
DELETE FROM Ships
WHERE name IN (
  SELECT ship FROM Outcomes
  WHERE result = 'sunk');
```

Use `NOT` to negate `EXISTS`, `IN`, `ALL`, `ANY`
https://www.postgresql.org/docs/11/static/functions-subquery.html

Correlated subqueries

"Find course names that have been used for multiple courses."

```
SELECT name
FROM courses AS c1
WHERE name IN (
  SELECT name
  FROM courses AS c2
  WHERE c2.dept <> c1.dept
     OR c2.number <> c1.number);
```

Read this query from the inside out!

Another exercise

```
--
\echo QUERY #8
\echo
-- Find the minimum cost supplier for each part.
--
-- You must use a subquery to receive full credit.
-- Hint: Find the minimum cost of each part first.
--
-- Schema: ps_partkey, ps_suppkey, min_supplycost
--  Order: ps_partkey
```