ADVANCED SQL I

CS 564- Fall 2018

WHAT IS THIS LECTURE ABOUT

- SQL: Set Operators
 - UNION/EXCEPT/INTERSECT
 - duplicates in SQL
- SQL: Nested Queries
 - IN/EXISTS/ALL
 - correlated queries

SET AND MULTISET OPERATORS

SET OPERATORS: REFRESHER

$$R = \{1, 2, 3\}$$

$$S = \{1, 2, 4, 5\}$$

- Intersection:
- Union:
- Difference:

$$R \cap S = \{1, 2\}$$

$$R \cup S = \{1, 2, 3, 4, 5\}$$

$$R - S = \{3\}$$

$$S - R = \{4, 5\}$$

SET OPERATORS IN SQL

SQL supports set operations between the outputs of subqueries:

- (subquery) INTERSECT (subquery)
- (subquery) UNION (subquery)
- (subquery) EXCEPT (subquery)

SET OPERATORS: INTERSECT

SELECT A FROM R
INTERSECT
SELECT A FROM S;

5

output A 1 2

Returns the tuples that belong in both subquery results

SET OPERATORS: UNION

SELECT A FROM R
UNION
SELECT A FROM S;

R

A	
1	
1	
1	
2	
3	

S

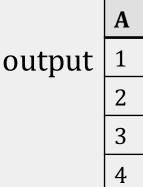
F	1	
1	L	

1

2

<u>___</u>

5



5

Returns the tuples that belong in either subquery results

SET OPERATORS: EXCEPT

SELECT A FROM R

EXCEPT

SELECT A FROM S;

R

1 1

1

3

S

1

1

4

5

output

A

Returns the tuples that belong in the first and not the second subquery result

SEMANTICS

- When using set operators, SQL eliminates all duplicate tuples
- We can modify the semantics by using the keyword ALL (e.g. UNION ALL)
- When using ALL, the operators are evaluated using multiset (or bag) semantics

SET OPERATORS: UNION ALL

SELECT A FROM R UNION ALL **SELECT** A **FROM** S;

output

3

5

3

S

Α

4

5

The number of copies of each tuple is the **sum** of the number of copies in the subqueries

SET OPERATORS: INTERSECT ALL

SELECT A FROM R
INTERSECT ALL
SELECT A FROM S;

R A 1

1

1

3

S

1

1

4

5

output

A1

1

2

The number of copies of each tuple is the minimum of the number of copies in the subqueries

SET OPERATORS: EXCEPT ALL

SELECT A FROM R
EXCEPT ALL
SELECT A FROM S;

R

1 1

1

3

S

<u>A</u> 1

_ 1

2

4

5

output

A1
3

The number of copies of each tuple is the difference (if positive) of the number of copies in the subqueries

DISCUSSION ON DUPLICATES

- When doing projection:
 - easier to avoid eliminating duplicates
 - tuple-at-a-time processing
- When doing intersection, union or difference:
 - more efficient to sort the relations first
 - at that point you may as well eliminate the duplicates anyway

NESTED QUERIES

NESTED QUERIES

A parenthesized SELECT-FROM-WHERE statement (*subquery*) can be used as a value in a number of places:

- in **FROM** clauses
- in WHERE clauses

```
FROM Country C
WHERE C.code =
    (SELECT C.CountryCode
    FROM City C
WHERE C.name = 'Berlin');
```

Can you rewrite this query without a subquery (unnesting)?

NESTING

- We can write nested queries because the SQL language is compositional
- Everything is represented as a multiset
- Hence the output of one query can be used as the input to another (nesting)

NESTED QUERIES

Find all countries in Europe with population more than 50 million

Can you unnest this query?

SET-COMPARISON OPERATOR: IN

Find all countries in Europe that have **some** city with population more than 5 million

SET-COMPARISON OPERATOR: EXISTS

Find all countries in Europe that have **some** city with population more than 5 million

```
FROM Country C correlated subquery

WHERE C.Continent = 'Europe'

AND EXISTS (SELECT *

FROM City T

WHERE T.Population > 5000000

AND T.CountryCode = C.Code);
```

SET-COMPARISON OPERATOR: ANY

Find all countries in Europe that have **some** city with population more than 5 million

SET-COMPARISON OPERATORS

Find all countries in Europe that have **all** cities with population less than 1 million

SET-COMPARISON OPERATORS: ALL

Find all countries in Europe that have **all** cities with population less than 1 million