SQL Unit 7
Set Operations

Ryan Nixon

Another Way To Select

- Recall that Joins allow us to combine the rows from two tables into a single resultset
- WHERE can filter those results
- SELECT will display only the columns we choose
- It turns out there is one more way to specify how your data is handled

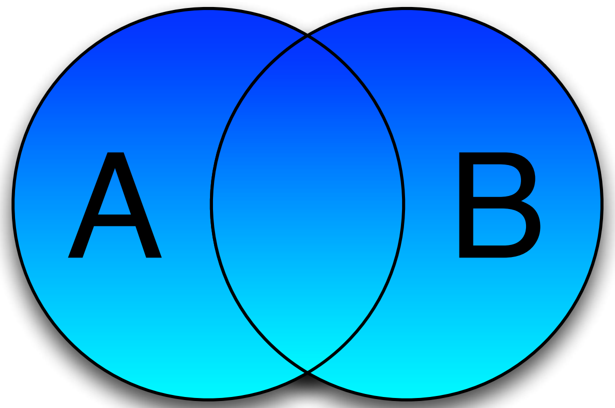
Set Operations

- Set operations -- Working with result**sets** as if they were collections of items, or tuples
- Using set theory, we can combine or intersect sets to make substantial changes to the resultset

Back to WHERE

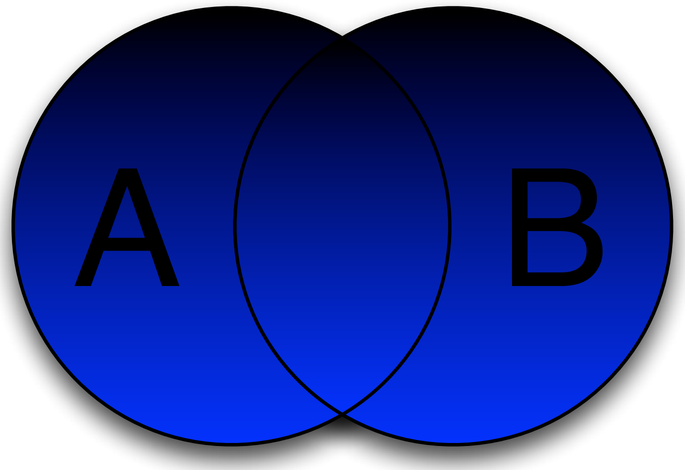
- Remember AND and OR in the WHERE clause?
  - AND will require both the condition on the left and the condition on the right to be true
  - OR will require one condition on the left or right to be true, or both
- Sets work similar to this but at a higher level; instead of working with columns, they work with whole rows

Set Notation



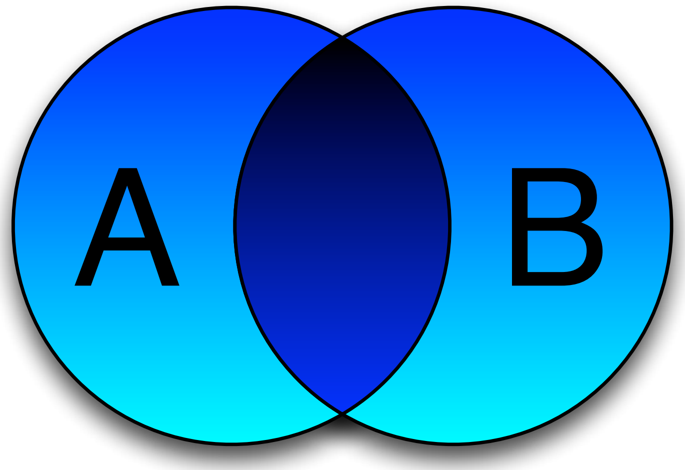
- You may already be familiar with set notation. The symbols used in those directly correlate to their SQL implementations:
  - A∪B - A UNION B (SQL OR)
  - A∩B - A INTERSECTION B (SQL AND)

Set Notation



- A∪B - A UNION B (SQL OR)
- Return the Union of query A and query B. This will include all rows that occur in either resultset or both of them

Set Notation

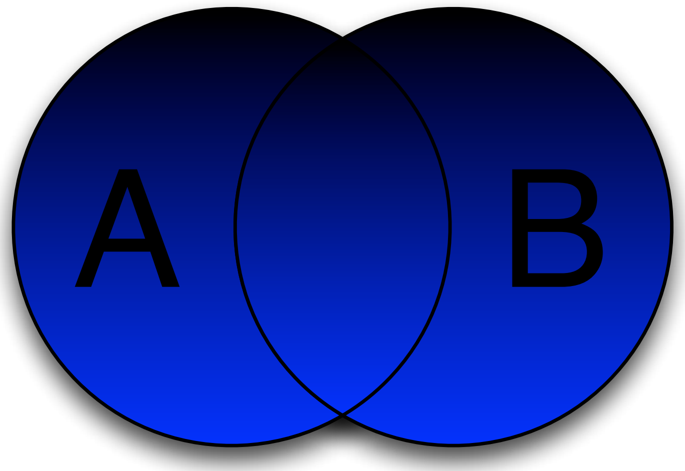


- A∩B - A INTERSECTION B (SQL AND)
- Return the intersection of query A and query B. This will include only rows that occur in *both* resultsets

Set Notation

- In practice, the set operations will also remove duplicate rows that are encountered. This is similar to applying DISTINCT to the query

Unions



-
  ```
  SELECT *
  FROM person
  WHERE last = 'Smith'
  UNION
  SELECT *
  FROM person
  WHERE last = 'Jones'
  ```

Unions

A

person				
id	first	last	gender_id	address_id
30663453	Lisa	Smith	F	738
30729595	Allen	Smith	M	185
30782165	Norbert	Smith	M	735

$\cup$

person				
id	first	last	gender_id	address_id
30600610	Joyce	Jones	F	520
30717694	Steve	Jones	M	260
30799275	Ken	Jones	M	559
30840089	Benjamin	Jones	M	100

B

$\rightarrow$

person				
id	first	last	gender_id	address_id
30600610	Joyce	Jones	F	520
30663453	Lisa	Smith	F	738
30717694	Steve	Jones	M	260
30729595	Allen	Smith	M	185
30782165	Norbert	Smith	M	735
30799275	Ken	Jones	M	559
30840089	Benjamin	Jones	M	100

Unions

- Simply putting UNION between the two queries will trigger the set operation
- Take a look at the query again. Doesn't this behavior look familiar? Is there another way to do it?

Unions

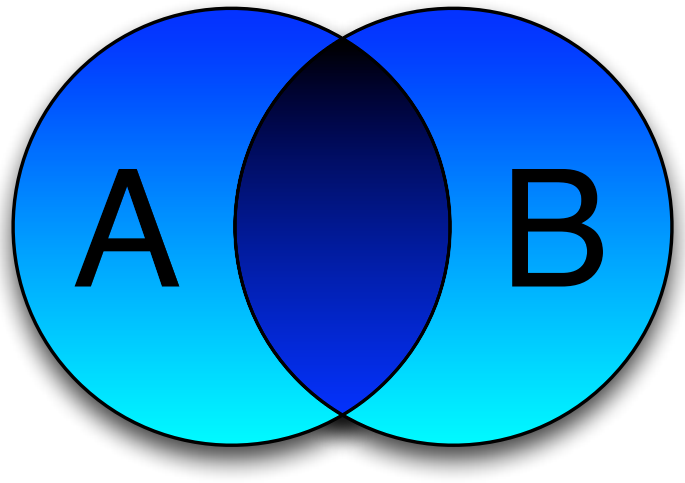
-
  ```
  SELECT *
  FROM person
  WHERE last = ‘Smith’ OR last = ‘Jones’
  ```
- This would produce the same resultset. In this way UNION acts exactly like the OR.
- UNION however is very beneficial when more complex queries are in use

UNION ALL

- There is another modifier that can be used, UNION ALL, which will keep any duplicates around.
- This may be useful if the results are going to be aggregated in some way

Intersections

-
  ```
  SELECT *
  FROM class
  WHERE credits > 1
  LIMIT 5
  INTERSECT
  SELECT *
  FROM class
  WHERE credits < 3
  LIMIT 5
  ```



Intersections

A

class			
id	name	course	credits
70004	Principles of Financial Accounting I	A101	3
70006	Principles of Financial Accounting II	A102	3
70007	Bookkeeping for Business I	A120	3
70014	Principles of Financial Accounting	A201	3
70017	Principles of Managerial Accounting	A202	3
70020	Introduction to Computerized Accounting	A222	3

$\cap$

class			
id	name	course	credits
70116	Foundations of the United States Air Force I	A101	1
70117	US Air Force Leadership Laboratory	A150	1
70118	Evolution of Air and Space Power I	A201	2
70128	Aircraft Ground Operations and Safety	A170	1
70132	Fundamentals of Aircraft Electronics Lab	A174L	2
70133	Drawing and Precision Measurement	A175	2

B

$\rightarrow$

class			
id	name	course	credits
70118	Evolution of Air and Space Power I	A201	2
70132	Fundamentals of Aircraft Electronics Lab	A174L	2
70133	Drawing and Precision Measurement	A175	2

Intersections

- Intersections have a very similar syntax to Unions, but a completely different behavior
- Again, take a look at the query. What does that look familiar to?

Intersections

-
  ```
  SELECT *
  FROM class
  WHERE credits > 1 AND credits < 3
  ```
- Intersections are exactly like the AND operator when they work with resultsets
- Just like Unions, this only works well in complex queries

A Compatibility Note

- For the sake of illustration, I just lied a bit
- The INTERSECT keyword does not work in many SQL implementations (although it does work in SQLite)
- But that's okay! After this short tangent you'll see something *BETTER!*

Dealing With Columns

- Note that the previous examples all use * and are from the same table. This is not a requirement
- As long as the types are similar and the number of columns are the same, you may interchange tables
- This is known as union compatibility

Dealing With Columns

- Note that I said ‘similar’ when dealing with types. There is an implicit cast that occurs when dealing with columns of similar type
- CHAR(10) + CHAR(50) = CHAR(50)
- INT + FLOAT = FLOAT
- The cast will take the largest or most flexible field and use that in the resultset

Dealing With Columns

- With columns, the counts must be the same. As this is unlikely to occur when tables are being compared, you can "pad" columns with default values until they match
  - `person_student: person_id, NULL, out_of_state, NULL`
  - `person_faculty: person_id, department_id, 0, bio`
- Note that this value doesn't have to be NULL

Dealing With Columns

-
  ```
  SELECT person_id, NULL, out_of_state, NULL
  FROM person_student
  LIMIT 5
  UNION
  SELECT person_id, department_id, 0, bio
  FROM person_faculty
  LIMIT 5
  ```

Dealing With Columns

A

person_student			
person_id	NULL	out_of_state	NULL
30676986		0	
30677070		0	
30679836		0	
30680792		1	
30681904		0	

U

person_faculty			
person_id	department_id	0	bio
30745137	7	0	
30745820	1	0	
30748325	30	0	
30752649	24	0	I like cats.
30756264	9	0	

B

$\longrightarrow$

person_id	NULL	out_of_state	NULL
30676986		0	
30677070		0	
30679836		0	
30680792		1	
30681904		0	
30745137	7	0	
30745820	1	0	
30748325	30	0	
30752649	24	0	I like cats.
30756264	9	0	

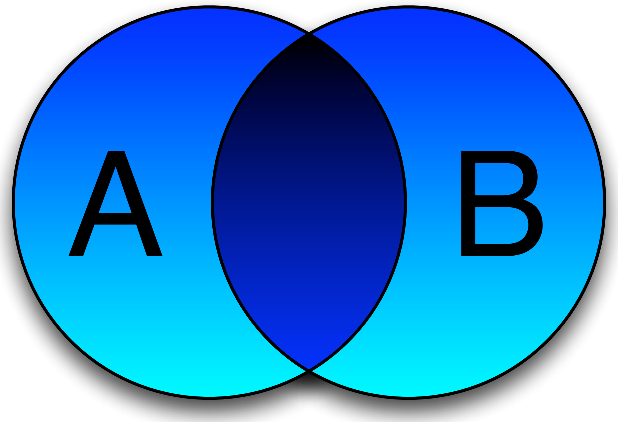
Dealing With Columns

- You might note that the header for the resultset had NULLs in it.
- It takes the headers from query A, so if you would like better headers you can add aliases to the NULL columns

Additional Operations

- Union and Intersect directly relate to WHERE operators, but there are 2 more that indirectly relate:
  - $A \subset B$ - A IN B
  - A - B - A NOT IN B

IN



- IN...Intersect...oh, I see what you did there.
- It's true, they perform the same actions, but in much different ways
- The IN operator, of the 4 in this unit, is by far the most common

IN

-
  ```
  SELECT *
  FROM person_student
  WHERE person_id IN (
    SELECT person_id
    FROM person_faculty)
  LIMIT 5
  ```

IN

person_student	
person_id	out_of_state
30770668	0
30707731	0
30752649	0
30799694	0
30700976	0

$\subset$

person_faculty
person_id
30748325
30752649
30756264
30759542
30759864

$\longrightarrow$

person_id	out_of_state
30752649	

IN

- Notice how the operation behaves the same, selecting all rows that appear in both, but the syntax is vastly different
- With IN you can intersect two resultsets based off of a common column. Almost like a join, but in this case no additional rows or columns are added

A Familiar Example

-
  ```
  SELECT *
  FROM person
  INNER JOIN class_registration reg
     ON (reg.person_id = person.id
       AND reg.role_id = ‘S’)
  INNER JOIN class_registration reg2
     ON (reg2.person_id = person.id
       AND reg2.role_id = ‘F’)
  ```

A Familiar Example

-
  ```
  SELECT *
  FROM person
  INNER JOIN class_registration reg
     ON (reg.person_id = person.id)
  WHERE role_id = ‘S’
     AND person_id IN (
       SELECT person_id
       FROM class_registration
       WHERE role_id = ‘F’)
  ```

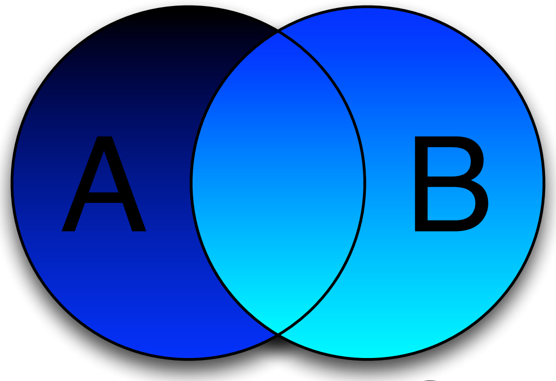
IN

- Also notice that there is an entire query in those parentheses. This (surprise) is what an inline view or subquery looks like
- It defines another query whose resultset is used as a parameter for IN
- Even better: since the resultset is only one column, you can substitute the query for actual values

IN

-
  ```
  SELECT *
  FROM person
  WHERE last IN (‘Smith’, ‘Jones’)
  ```
- This query is exactly like the Union example. It filters all person rows down to those with the last name ‘Smith’ OR ‘Jones’

NOT IN



- You can apply the NOT operator to IN as well:

-
  ```
  SELECT *
  FROM person
  WHERE last NOT IN ('Smith',
  'Jones')
  ```

- This predictably will return all persons whose last name is NOT Smith or Jones

Reminders

- Assignment 7 up by Friday. Due 11/19
- Independent lab time Monday in SSB 172
- Lecture next Wednesday in classroom