Lecture 3: More SQL Basics

CS1106/CS6503– Introduction to Relational Databases

Dr Kieran T. Herley

2019-2020

School of Computer Science & Information Technology
University College Cork

Summary

Composite conditions using AND, OR and NOT. The IN operator. Ordering results. Renaming result columns.

Complex Conditions

SQL Conditions

```
SELECT last_name, first_name
FROM students
WHERE hometown = 'Cork';
```

```
SELECT last_name, first_name
FROM students
WHERE points >= 500;
```

Note that SQL *conditions*

```
hometown = 'Cork'
points >= 500
```

are either *true* or *false* in respect of each row in the table `students`

We think of these as espressions with *true*/*false* value

Single-Criterion WHERE Conditions

-

```
SELECT last_name, first_name
FROM students
WHERE hometown = 'Cork';
```

```
SELECT last_name, first_name
FROM students
WHERE points >= 500;
```

  Conditions filter rows based on single criterion

- What is we want all students that have Cork as their hometown *as well as* having at least 500 points?

Single-Criterion WHERE Conditions

-

```
SELECT last_name, first_name
FROM students
WHERE hometown = 'Cork';
```

```
SELECT last_name, first_name
FROM students
WHERE points >= 500;
```

  Conditions filter rows based on single criterion

- What is we want all students that have Cork as their hometown *as well as* having at least 500 points?
- Can express this using keyword **AND** to combine two conditions

```
SELECT last_name, first_name
FROM students
WHERE hometown = 'Cork' AND points >= 500;
```

Two-Criterion WHERE Conditions

```
SELECT last_name, first_name
FROM students
WHERE hometown = 'Cork' AND points >= 500;
```

- Result contains rows for which
  - hometown value is 'Cork' **and**
  - points value is at least 500

  NB both conditions must be satisfied

- **students**

. . .	*hometown*	. . .	*points*	
· · ·	Tralee	· · ·	350	
· · ·	Cork	· · ·	350	
· · ·	Limerick	· · ·	550	
· · ·	Cork	· · ·	500	✓

  Only the checked row satisfies *both* criteria

The Logical Operator AND

- Suppose $\boxed{\alpha}$, $\boxed{\beta}$ are two SQL conditions (with true/false answers).
- Then the combined condition

$$\boxed{\alpha} \ \textbf{AND} \ \boxed{\beta}$$

is true if and only if both $\boxed{\alpha}$ is true and $\boxed{\beta}$ is true

		α	
		false	*true*
β	*false*	*false*	*false*
	true	*false*	*true*

PHP also includes and/or operators (subtle differences)

Two-Criterion WHERE Conditions cont'd

- What if we wanted all students with Cork as their hometown or who have at least 400 points.

```
SELECT last_name, first_name
FROM students
WHERE hometown = 'Cork' OR points >= 400;
```

- Result contains rows for which **either**
  - hometown value is 'Cork' **or**
  - points value is at least 400 (just one "clause" needs to be satisfied)
-

students

. . .	*hometown*	. . .	*points*	
· · ·	Tralee	· · ·	350	
· · ·	Cork	· · ·	350	✓
· · ·	Limerick	· · ·	550	✓
· · ·	Cork	· · ·	500	✓

The Logical Operator OR

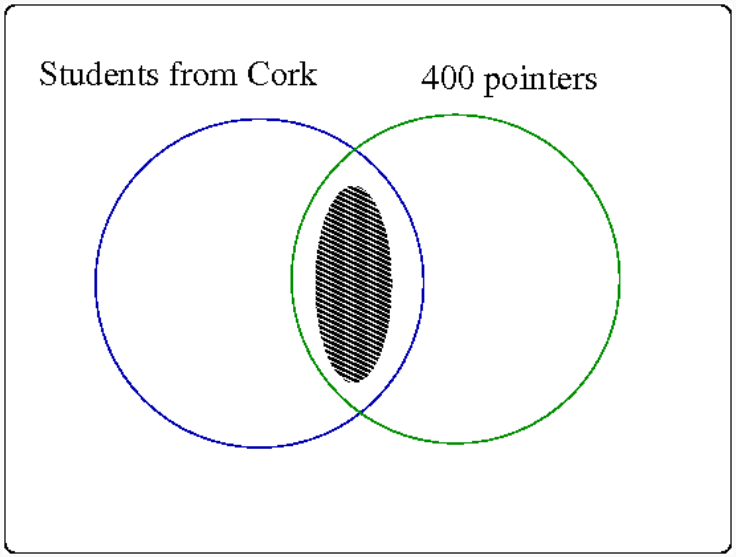
- Suppose $\boxed{\alpha}$, $\boxed{\beta}$ are two SQL conditions (with true/false answers).
- Then the combined condition

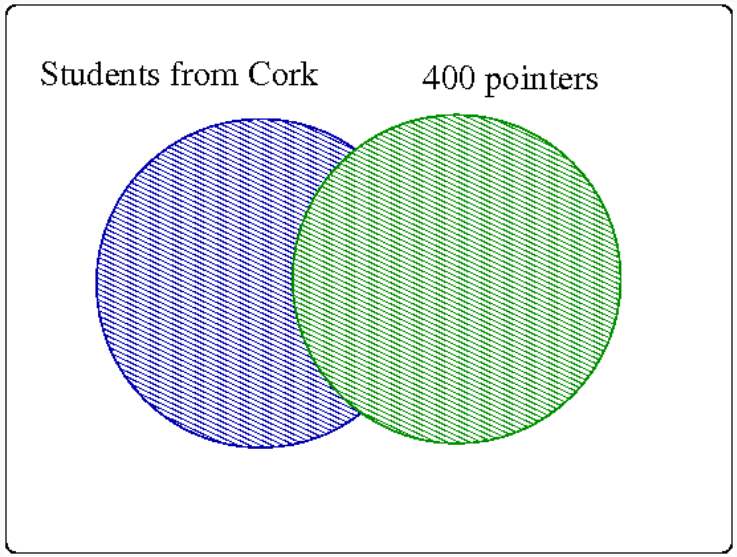
$$\boxed{\alpha} \ \textbf{OR} \ \boxed{\beta}$$

is true if and only if one or other or both of $\boxed{\alpha}$, $\boxed{\beta}$ is true

		α	
		false	*true*
β	*false*	*false*	*true*
	true	*true*	*true*

AND/OR operators






```
SELECT last_name, first_name
FROM students
WHERE hometown = 'Cork' AND
      points >= 400;
```

```
SELECT last_name, first_name
FROM students
WHERE hometown = 'Cork' OR
      points >= 400;
```

Need to frame conditions carefully to capture exactly what you intend

Example

- List all students whose home town is Cork, Limerick or Tralee
-

```
SELECT first_name, last_name, hometown
FROM students
WHERE hometown = 'Cork' OR
      hometown = 'Limerick' OR
      hometown = 'Tralee';
```

- Cannot simplify to

```
...
WHERE hometown = 'Cork' OR 'Limerick' OR 'Tralee'
```

Example

- List all students with over 500 points whose home town is Cork, Limerick or Tralee
-

```
SELECT first_name, last_name, hometown
FROM students
WHERE points >= 500 AND
      (  hometown = 'Cork' OR
         hometown = 'Limerick'  OR
         hometown = 'Tralee');
```

- Note use of parentheses for grouping; What if we omitted these?

Example cont'd

-

```
SELECT first_name, last_name, hometown
FROM students
WHERE points >= 500 AND
      hometown = 'Cork' OR
      hometown = 'Limerick' OR
      hometown = 'Tralee';
```

- Relationship between **AND** and **OR** akin to $*$ and $+$ in arithmetic expressions: **AND** has higher precenence than **OR** (evaluated first)
- Captures students in any of these three categories
  - Students from Cork with 500 points or more
  - Students from Limerick
  - Students from Tralee

SQL's In Keyword

- Could re-express previous query as

```
SELECT first_name, last_name, hometown
FROM students
WHERE points >= 500 AND
      hometown IN ('Cork', 'Limerick', 'Tralee');
```

- Here ('Cork', $\cdots$) denotes a *set* of values; as long as hometown is in the set, the `hometown IN (...)` clause is satisfied

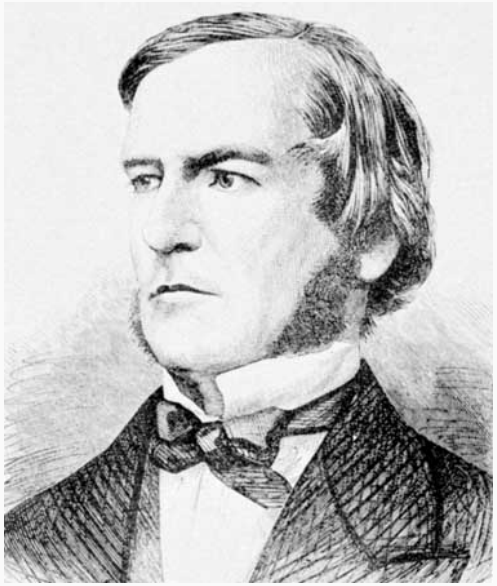
The NOT Operator

- SQL also has a NOT keyword which negates the meaning of the condition

```
SELECT first_name, last_name
FROM students
WHERE NOT points >= 475;
```

- Lists all students whose points are *not* greater than or equal to 475
- (Or equivalently those with points less than 475)

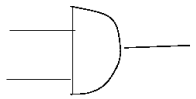
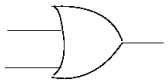
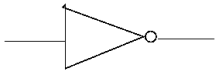
George Boole



George Boole (1815–1864)

- Discovered Boolean Algebra–algebraic system based on
  - True/False values and variables
  - AND, Or and NOT operators
- Important CS concept; foundation for computer circuit theory
- Self-taught mathematician first Professor of Mathematics UCC (then QCC) 1849–1864; buried Blackrock

Boolean Logic

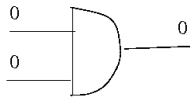
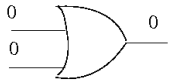
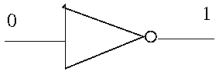
Operator	Symbol	Truth table	
AND			Output 1, if both inputs are 1
OR			Output 1, if either one of inputs is 1
NOT		Negates input	Negates input

	0	1
0	0	0
1	0	1

	0	1
0	0	1
1	1	1

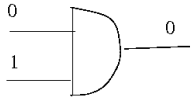
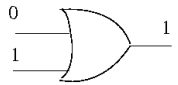
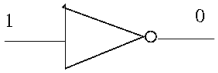
Logic gates implementing AND, OR and NOT operators (0 as False, 1 as True) provided foundations for most computer circuitry. For more on circuits see cs1101.

Boolean Logic

Operator	Symbol	Truth table			
AND			0	1	Output 1, if both inputs are 1
		0	0	0	
		1	0	1	
OR			0	1	Output 1, if either one of inputs is 1
		0	0	1	
		1	1	1	
NOT		Negates input			Negates input

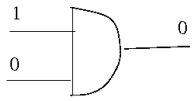
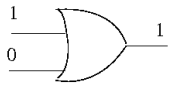
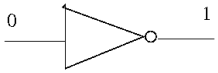
Logic gates implementing AND, OR and NOT operators (0 as False, 1 as True) provided foundations for most computer circuitry. For more on circuits see cs1101.

Boolean Logic

Operator	Symbol	Truth table			
AND			0	1	Output 1, if both inputs are 1
		0	0	0	
		1	0	1	
OR			0	1	Output 1, if either one of inputs is 1
		0	0	1	
		1	1	1	
NOT		Negates input			Negates input

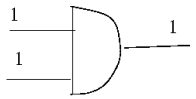
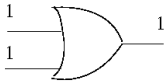
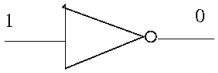
Logic gates implementing AND, OR and NOT operators (0 as False, 1 as True) provided foundations for most computer circuitry. For more on circuits see cs1101.

Boolean Logic

Operator	Symbol	Truth table			
AND			0	1	Output 1, if both inputs are 1
		0	0	0	
		1	0	1	
OR			0	1	Output 1, if either one of inputs is 1
		0	0	1	
		1	1	1	
NOT		Negates input			Negates input

Logic gates implementing AND, OR and NOT operators (0 as False, 1 as True) provided foundations for most computer circuitry. For more on circuits see cs1101.

Boolean Logic

Operator	Symbol	Truth table			
AND			0	1	Output 1, if both inputs are 1
		0	0	0	
		1	0	1	
OR			0	1	Output 1, if either one of inputs is 1
		0	0	1	
		1	1	1	
NOT		Negates input			Negates input

Logic gates implementing AND, OR and NOT operators (0 as False, 1 as True) provided foundations for most computer circuitry. For more on circuits see cs1101.

Some Beauty Tips

Reordering Results

- SQL provides some tools to control the appearance of the result tables of your queries
- ORDER BY clause presents results in order of some attribute

```
SELECT *
FROM students
ORDER BY points;
```

  Lists students in increasing order of points (lowest to highest)
- Use `ORDER BY points DESC` to order from highest to lowest

Prettifying Results

- Can also specify alternative column headings for greater readability
-

```
SELECT
    id_number AS 'Student Id',
    first_name AS 'Given Name',
    last_name AS 'Surname',
    points AS 'CAO Points'
FROM students
ORDER BY points;
```

Student Id	*Given Name*	*Surname*	*CAO Points*
112356489	Ciara	Callaghan	425
112467389	Barry	Barry	450
112561728	Eimear	Early	475
112836467	Fionn	Fitzgerald	485
112345678	Aoife	Ahern	500
112986347	Declan	Duffy	550

- By default result table uses same column names as students table, but `id_number AS 'Student Id'` substitutes 'Student Id' in place of 'id_number' etc.

Prettifying cont'd

- Can use SQL *functions* to modify values appearing in results

```
SELECT
    UPPER(last_name) AS 'Surname',
    first_name  AS 'Given Name',
    DATE_FORMAT(date_of_birth, '%W, %d %M %Y') AS 'Born On'
FROM students
ORDER BY date_of_birth DESC;
```

- Generates

Surname	*Given Name*	*Born On*
FITZGERALD	Fionn	Monday, 13 June 1994
DUFFY	Declan	Wednesday, 03 November 1993
EARLY	Eimear	Sunday, 18 July 1993
CALLAGHAN	Ciara	Sunday, 14 March 1993
AHERN	Aoife	Monday, 25 January 1993
BARRY	Barry	Monday, 30 June 1980

- UPPER and DATE_FORMAT[1] are *functions*; former transforms text into upper-case
- (Advanced feature; don't worry about this one for now, just be aware of existence)

[1] MySQL

Notes and Acknowledgements