SQL – Logical Operators and aggregation

Chapter 3.2

V3.0

Dr Gordon Russell

NAPIER UNIVERSITY
EDINBURGH

Logical Operators

- Combining rules in a single WHERE clause would be useful
- AND and OR allow us to do this
- NOT also allows us to modify rule behaviour

- When these are combined together, problems in rule ordering can occur.
- This is solved using parentheses.

AND

- AND combines rules together so that they ALL must be true.
- Lets revisit the CAR table:

REGNO	MAKE	COLOUR	PRICE	OWNER
F611 AAA	FORD	RED	12000	Jim Smith
J111 BBB	SKODA	BLUE	11000	Jim Smith
A155 BDE	MERCEDES	BLUE	22000	Bob Smith
K555 GHT	FIAT	GREEN	6000	Bob Jones
SC04 BFE	SMART	BLUE	13000	

SELECT regno from car
where colour = ‘BLUE’

REGNO
J111 BBB
A155 BDE
SC04 BFE

SELECT regno from car
WHERE regno LIKE ‘%5’

REGNO
A155 BDE
K555 GHT

```
SELECT regno from car
WHERE colour = 'BLUE' and regno LIKE '%5%'
;
```

REGNO
A155 BDE

Multiple AND rules

- You can have as many rules as you like ANDed together.
- For example:

```
SELECT regno
FROM car
WHERE colour = 'BLUE'
AND regno like '%5%'
AND owner like 'Bob %'
;
```

OR

- OR is like ‘either’. So long as one of the rules is true then the filter is true.
- Looks for cars which are EITHER red or blue…

```
SELECT regno,colour from CAR
WHERE colour = ‘RED’ OR colour = ‘BLUE’
```

REGNO	COLOUR
F611 AAA	RED
J111 BBB	BLUE
A155 BDE	BLUE
SC04 BFE	BLUE

NOT

- NOT inverts the rule it is put in front of:
- WHERE colour = ‘RED’
- This could be inverted as:
  - WHERE colour != ‘RED’
  - WHERE NOT colour = ‘RED’

- NOT is not really useful in this example, but comes into its own in more complex rulesets.

Precedence

- Precedence is the order in which the rules are evaluated and combined together.
- It is NOT in the order they are written.
- Rules are combined together firstly at AND, then OR, and finally at NOT.
- Consider : Car has a 5 in reg and is either red or blue.

```
SELECT regno,colour from car
WHERE colour = 'RED'          -- Line 1
OR    colour = 'BLUE'         -- Line 2
AND   regno LIKE '%5%'        -- Line 3
```

Brackets

- Rewrite as:

```
SELECT regno,colour from car
WHERE (colour = ‘RED’
OR       colour = ‘BLUE’)
AND     regno LIKE ‘%5%’
```

- Might be clearer as:

```
SELECT regno,colour from car
WHERE ( colour = ‘RED’ OR colour = ‘BLUE’ )
AND     regno LIKE ‘%5%’
```

DISTINCT

- Find all the colours used in cars.

SELECT colour from car;

COLOUR
RED
BLUE
BLUE
GREEN
BLUE

DISTINCT

SELECT DISTINCT colour from car;

COLOUR
RED
BLUE
GREEN

ORDER BY

- It would be nice to be able to order the output using a sort.
- SELECT make from car;

MAKE
FORD
SKODA
MERCEDES
FIAT
SMART

ASCending order

- Sort by alphabetical or numeric order: ASC
- ORDER BY … ASC is the default.

SELECT make from car
ORDER BY make;

MAKE
FORD
FIAT
MERCEDES
SKODA
SMART

DESCending order

- Sort by reverse alphabetical or numeric order: DESC
- ORDER BY … DESC must be selected.

```
SELECT make from car
ORDER BY make DESC;
```

MAKE
SMART
SKODA
MERCEDES
FIAT
FORD

Multi Column Sort

- ORDER BY can take multiple columns.

SELECT make,colour FROM car
ORDER BY colour,make;

MAKE	COLOUR
SKODA	BLUE
SMART	BLUE
MERCEDES	BLUE
FIAT	GREEN
FORD	RED

IN

- When you have a list of OR, all on the same attribute, then IN could be a simpler way:

- Rather Than:

```
SELECT regno,make FROM car
WHERE make = ‘SKODA’ or make = ‘SMART’
```

- Have

```
SELECT regno,make FROM car
WHERE make in (‘SKODA’,’SMART’);
```

Aggregate Functions

- Aggregate functions allow you to write queries to produce statistics on the data in the database.
- These functions are sometimes also called SET functions.

- These include:
  - AVG (calculate the average)
  - SUM
  - MAX
  - MIN
  - COUNT

AVERAGE

SELECT price FROM car;

PRICE
12000
11000
22000
6000
13000

SELECT avg(price) FROM car;

AVG(PRICE)
12800

SUM

- Add up all the values in a column

SELECT sum(price) FROM car;

SUM(PRICE)
64000

MAX

- What is the maximum value in a column

SELECT max(price) FROM car;

MIN(PRICE)
22000

MIN

- What is the minimum value in a column

SELECT min(price) FROM car;

MIN(PRICE)
22000

COUNT

- How many rows make up a column

SELECT count(price) FROM car;

COUNT(PRICE)
5

- Count(*) is similar, but also counts when price is NULL.

SELECT count(*) FROM car;

COUNT DISTINCT

- Sometimes you do not want to count how many rows are in a column, but how many different values could be found in that column.
- There is a special variant of count which does this:

SELECT count(colour) from car;

COUNT(PRICE)
5

SELECT count(DISTINCT colour) from car;

COUNT(PRICE)
3

GROUP BY

- Aggregation functions so far have only been shown in queries with only the single aggregation function on the select line.
- You can combine functions and non-functions on the select line.
- To do this you need GROUP BY.

- Question: What is the most expensive car for each colour.
- Intuitively the following seems right, but will not execute!

```
SELECT colour,max(price)
FROM car;
```

```
SELECT colour,price
FROM car;
```

COLOUR	PRICE
RED	12000
BLUE	11000
BLUE	22000
GREEN	6000
BLUE	13000

```
SELECT colour,max(price)
FROM car
GROUP BY colour;
```

COLOUR	PRICE
RED	12000
BLUE	22000
GREEN	6000

HAVING

- WHILE allows rules for each row.
- HAVING allows rules for each group of a GROUP BY.

- Consider the problem “Who has more than 1 car”.

- We would like to say:

SELECT owner from car where count(owner) > 1

- Aggregate functions are not allowed in WHERE.
- They are allowed in HAVING.

```
SELECT owner,count(regno)
FROM car
GROUP BY owner
HAVING count(regno) > 1
```

OR

```
SELECT owner
FROM car
GROUP BY owner
HAVING count(regno) > 1
```

count(*) works just as well in this case.