SQL: Complex Queries

Shortly we write SQL Queries with complex **Where** criteria

- Usually these queries use
  - The **AND** operator
  - The **OR** operator
  - A **combination** of AND & OR operators

Eg:

```
SELECT   * FROM Lecturer
WHERE    Age >= 20
    AND  Age <= 29  ;

SELECT   * FROM Lecturer
WHERE    LecName = 'John Smith'
    OR   LecName = 'Jack Smith'

SELECT   * FROM Lecturer
WHERE    Age >= 20 AND  Age <= 29
    OR   LecId > 500
```

SQL: AND & OR operators

Each row in the table is **evaluated** against the criteria.
Only rows that **satisfy** the criteria are **returned** in the Result Set.

- Many people (especially non-programmers) are often **confused** about whether to use an **AND** or an **OR** operator

- Requirement: 'Look through all of our employee surnames.
  List all the Smiths and Nguyens"

- Which is the **correct** statement?

```
SELECT    Id, Firstname, Surname
FROM      Employee
WHERE     Surname = 'Smith'
    OR    Surname = 'Nguyen'
```

```
SELECT    Id, Firstname, Surname
FROM      Employee
WHERE     Surname = 'Smith'
    AND   Surname = 'Nguyen'
```

SQL: AND & OR operators

When dealing with an SQL statement with AND OR operators

- **Do not** consider the **entire** employee table.
- Only **consider** a **single row** of the table.

Suggestion: Consider only the **first row** in the table:

- Can the **first row have a surname** that meets the criteria

  Surname = 'Smith' AND Surname = 'Nguyen'

  No. Surname can only store a single value, it can never be both

- Can the **first row have a surname** that meets the criteria

  Surname = 'Smith' OR Surname = 'Nguyen'

  Yes. The Surname could be **one** of these values

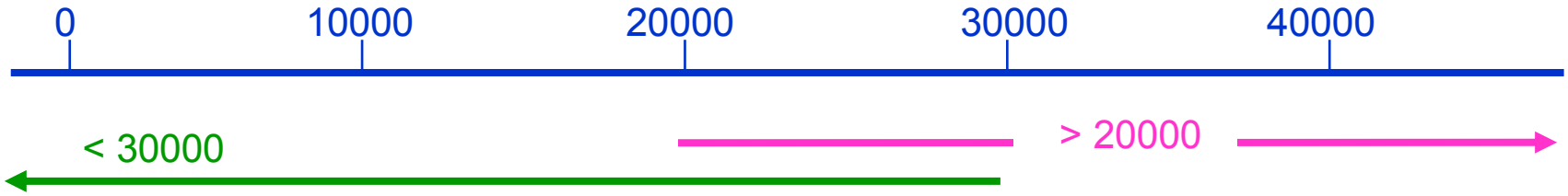
SQL: AND operator and numeric ranges

Sometimes a criteria will cover a range of values

- 'List employees whose salary is in the range 20000 to 30000'

- Which set of values is covered by:
  - salary > 20000 AND salary < 30000
  - salary > 20000 OR salary < 30000

- A diagram may help



- The **AND** alternative includes the range
  - **common to both lines**

- The **OR** alternative includes range
  - **covered by either line**.

SQL:Basic Operators

These are the basic **operators** that may be used in the WHERE clause

=	equals
<>	not equals
!=	not equals
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

SQL: Recap of WHERE clause

The WHERE clause has the syntax WHERE <*condition*> …

- A *condition* is an expression that evaluates to either **TRUE** or **FALSE**.
- Whenever the WHERE clause is used in a query:
  - The WHERE condition is evaluated for **each row** of the table
  - If the condition evaluates to **TRUE** then that row is added to the result set

- Consider the statement:

  SELECT * FROM employee WHERE age >20

Row	Name	Branch	Gender	Age	Condition True or False?	Is the row in the Result Set?
1	Dave	Haw	M	21	**TRUE**	**YES**
2	Sue	City	F	22	**TRUE**	**YES**
3	Rob	City	M	19	**FALSE**	**NO**
4	Jack	Kew	M	23	**TRUE**	**YES**

AND Operator

- When an **AND** operator is used, SQL evaluates the condition to the left and the condition to the right of the AND
  - A row is selected only when **BOTH** Conditions are **TRUE**

AND	Condition1 = TRUE	Condition1 = FALSE
Condition2 = TRUE	TRUE	FALSE
Condition2 = FALSE	FALSE	FALSE

- Consider the clause: **...WHERE gender = 'M' AND age >20**

Row	Name	Branch	Gender	Age	gender='M'	age>20	Comment	In Result Set?
1	Dave	Haw	M	21	**True**	**True**	True AND True is TRUE	**Y**
2	Sue	City	F	22	**False**		False AND any value is FALSE	**N**
3	Rob	City	M	19	**True**	**False**	True AND False is FALSE	**N**
4	Jack	Kew	M	23	**True**	**True**	True AND True is TRUE	**Y**

Multiple AND Operators

When more than one **AND** operator is used, deal with **one pair of conditions** at a time.

...WHERE **age >20 AND gender = 'M'** **AND branch = 'Haw'**

Evaluate this pair of conditions first

...WHERE **TRUE / FALSE AND branch = 'Haw'**

Evaluate this pair of conditions second

.

Row	Name	Branch	Gender	Age	Age>20	'M'	Result 1	'Haw'	Result 2	Selected?
1	Dave	Haw	M	21	T	T	T	T	T	✓
2	Sue	City	F	22	T	F	F			
3	Rob	City	M	19	F		F			
4	Jack	Kew	M	23	T	T	T	F	F	

SQL: OR Operator

When an **OR** operator is used, SQL evaluates the condition to the left and the condition to the right of the OR

- A row is selected if **either** of the two Conditions is **TRUE**

OR	Condition1 = TRUE	Condition1 = FALSE
Condition2 = TRUE	TRUE	TRUE
Condition2 = FALSE	TRUE	FALSE

- Consider the clause: **...WHERE gender = 'F' OR age >20 ...**

Row	Name	Branch	Gender	Age	gender=F	age > 20	comment	In Result Set?
1	Dave	Haw	M	21	**False**	**True**	False or True is TRUE	**Yes**
2	Sue	City	F	22	**True**		True or any value is TRUE	**Yes**
3	Rob	City	M	19	**False**	**False**	False or False is FALSE	**No**
4	Jack	Kew	M	23	**False**	**True**	False or True is TRUE	**Yes**

SQL: OR Operator (2)

When more than one **OR** operator is used, the system deals with one pair of conditions at a time.

…WHERE **age < 20** **OR** **gender = 'F'** **OR** **branch = 'Kew'**

Evaluate this pair of conditions first

…WHERE **TRUE / FALSE** **OR** **branch = 'Kew'**

Evaluate this pair of conditions second

Row	Name	Branch	Gender	Age	Age<20	'F'	Result 1	'Kew'	Result 2	In Result Set?
1	Dave	Haw	M	21	F	F	F	F	F	
2	Sue	City	F	22	F	T	T			✓
3	Rob	City	M	19	T		T			✓
4	Jack	Kew	M	23	F	F	F	T	T	✓

Mixing AND and OR operators

Consider:

…WHERE branch = 'City' OR branch = 'Haw' AND gender = 'M'

- **How many rows are in the result set?**

Row	Name	Branch	Gender	Age	Selected?
1	Dave	Haw	M	21	
2	Sue	City	F	22	
3	Rob	City	M	19	
4	Jack	Kew	M	23	

- When a WHERE clause contains both an AND & OR operator then the **AND** operators take precedence and must be evaluated first.

Mixing AND and OR operators

…WHERE branch = 'City' OR branch = 'Haw' AND gender = 'M'

- Does this mean 'select any males who live in Hawthorn or City'. NO

- Does this mean 'select **any** City person OR **any** males in Hawthorn' YES

- The **AND operator** always **takes precedence**.

WHERE branch = 'City' OR branch = 'Haw' AND gender = 'M'

- The conditions on either side of the **AND** are paired together as though there are parentheses surrounding the two conditions.

WHERE branch = 'City' OR (branch = 'Haw' AND gender = 'M')

Mixing AND and OR operators


```
WHERE   branch = 'City'  OR  branch = 'Haw'  AND gender = 'M'
```

Part 1 — Part 2

- This is now treated exactly the same as the OR example shown earlier

If either **Part1 is True** OR **Part2 is True**, then the row is **selected**.

Row	Name	Branch	Gender	Age	Part1	Part2 A	Part2 B	Part 2	Selected?
1	Dave	Haw	M	21	F	T	T	T	✓
2	Sue	City	F	22	T				✓
3	Rob	City	M	19	T				✓
4	Jack	Kew	M	23	F	F		F	

Mixed AND and OR operators

Statements with Ands & Ors are confusing.

Follow these steps to reduce confusion.

1. Decide if the select statement has a **mixture** of AND and Ors
   If NO, then apply the truth tables as previously discussed.

   WHERE Gender = 'F' OR Gender = 'M' AND Branch = 'City'

2. Find an AND clause in the statement

3. Find the expression to the **left** of the AND clause

4. Insert a **open bracket** to the **left** of the expression

   WHERE Gender = 'F' OR **(** Gender = 'M' AND Branch = 'City'

Mixed AND and OR operators

5. Find the expression to the **right** of the AND clause

6. Insert a **close bracket** to the **right** of the expression

```
WHERE   Gender = 'F'  OR  ( Gender = 'M' AND  Branch = 'City'  )
```

7. For each additional AND operator in the statement, repeat steps 2 – 6.

8. Now treat all bracketed expressions as a single expression

```
WHERE   Gender = 'F'  OR  ( Gender = 'M' AND  Branch = 'City'  )
```

The statement now has all expressions are separated by OR operators.

If any expression is True, then the row will be selected.

Mixing AND and OR operators

Consider this statement:

```
WHERE   gender = 'F'   OR  gender = 'M'  AND  age < 23
```

1. Does the statement have a mixture of ANDs and Ors?

   Yes

2. Where do the brackets have to be placed?

```
WHERE   gender = 'F'   OR  ( gender = 'M'  AND  age < 23 )
```

Mixing AND and OR operators

Consider this statement:

```
WHERE    age < 23
    OR    age > 30
    AND   gender = 'F'
```

1. Does the statement have a mixture of ANDs and Ors?

   Yes

2. Where do the brackets have to be placed?

```
WHERE    age < 23
    OR    ( age > 30
    AND     gender = 'F'  )
```

Mixing AND and OR operators

Consider this statement:

```
WHERE    dept = 'A'  OR   qty >= 50
   AND   qty <= 100  OR   branch = 'City'
```

1. Does the statement have a mixture of ANDs and ORs?

   Yes

2. Where do the brackets have to be placed?

```
WHERE    dept = 'A'  OR   ( qty >= 50
   AND   qty <= 100   )  OR   branch = 'City'
```

Mixing AND and OR operators

Consider this statement:

```
WHERE    dept = 'B'  AND   qty >= 100
   AND   qty <= 200  OR   branch = 'Haw'
```

1. Does the statement have a mixture of ANDs and ORs?

   Yes

2. Where do the brackets have to be placed?

```
WHERE    ( ( dept = 'B'  AND   qty >= 100 )
   AND   qty <= 200 ) OR   branch = Haw'
```

Mixing AND and OR operators

How many rows are selected?

```
WHERE   age =  21   AND   gender = 'M'  OR
        age =   23   AND   gender = 'F'
```

Row	Name	Branch	Gender	Age				Selected?
1	Dave	Haw	M	21				
2	Sue	City	F	22				
3	Rob	City	M	19				
4	Jack	Kew	M	23				

Mixing AND and OR operators

How many rows are selected?

```
WHERE   age =  21   AND   gender = 'M'  OR
        age =  23   AND   gender = 'F'
```

Row	Name	Branch	Gender	Age				Selected?
1	Dave	Haw	M	21				
2	Sue	City	F	22				
3	Rob	City	M	19				
4	Jack	Kew	M	23				

Mixing AND and OR operators

How many rows are selected?

```
WHERE   age =  19   OR   branch = 'Haw'  AND  branch = 'Kew' OR
        age =  22   AND   gender = 'M'  OR gender = 'F'
```

Row	Name	Branch	Gender	Age					Selected?
1	Dave	Haw	M	21					
2	Sue	City	F	22					
3	Rob	City	M	19					
4	Jack	Kew	M	23					

Parentheses ( )

Operators within parentheses ( ) are always evaluated **first**.
These similar looking statements may produce different results

WHERE gender = 'F' OR Age <20 AND branch = 'Haw'

WHERE (gender = 'F' OR Age <20) AND branch = 'City'

Row	Name	Branch	Gender	Age	Alternative 1	Alternative 2
1	Ralph	Haw	M	18	**T**	**F**
2	Emma	City	F	21	**T**	**T**
3	Julie	Haw	F	19	**T**	**F**
4	Jon	City	M	22	**F**	**F**

Parentheses (2)

Parentheses can also be used just for **readability**.

```
WHERE   age >18   AND  gender = 'M'  OR  age <= 17  AND  gender = 'F'
```

```
WHERE  (age >18   AND  gender = 'M') OR (age <= 17  AND  gender = 'F')
```

- The parentheses in the 2nd example **do not alter** the meaning of the SQL.
  - Parentheses make the SQL **more readable**
  - Easier for the **Author / Programmer**
  - Easier to **read / change / debug**
  - Easier for the person **marking** your assignment or exam!

NOT Operator

The **NOT** operator **negates** the **expression** immediately to its **right**.

- If the <expression> is TRUE, then NOT <expression> is FALSE
- If the <expression> is FALSE, then NOT <expression> is TRUE

- Select all people who work in the City branch

  SELECT * FROM Emp WHERE branch = 'City'

- Select all people from all other branches

  SELECT * FROM Emp WHERE NOT (branch = 'City')

- Select all people who work in either City or Haw

  SELECT * FROM Emp WHERE branch = 'City' OR branch='Haw'

- Select all people from all branches except City or Haw

  SELECT * FROM Emp WHERE NOT (branch='City' OR branch='Haw')

NOT Operator and parentheses

- The **NOT** operator negates the **expression <u>immediately</u> to its right**.
- **Be careful:** If you have a pair of conditions, then they must be surrounded by parentheses / brackets.
  If the brackets are omitted, you may get unexpected results!
- These **three** queries all return **different** results

A. SELECT * FROM emp WHERE Dept = 1 OR Dept = 2

B. SELECT * FROM emp WHERE NOT (Dept = 1 OR Dept = 2)

C. SELECT * FROM emp WHERE NOT Dept = 1 OR Dept = 2

Row	Name	Dept	Result A	Result B	Result C
1	Amy	1	✓		
2	Bruce	2	✓		✓
3	Carol	3		✓	✓
4	Dave	4		✓	✓

SQL: Order of precedence

1. Relational Operators (<, >, = *etc*.)
2. Brackets
3. NOT
4. AND
5. OR

SQL: UPPER ( ) & LOWER ( )

- Syntax: **UPPER** (<character expression>)
- The Upper() function takes a <**character expression**> and **returns a value** where all characters are in **upper case.**
- Syntax: **LOWER** (<character expression>)
- The Lower() function takes a <**character expression**> and **returns a value** where all characters are in **lower case.**
- Upper() and Lower() functions **DO NOT change** the database data
  - Each simply **returns** a value to be used **within** the query
- Characters outside the range A-Z / a-z are **not affected**
  - E.g. numbers & punctuation not affected

SQL: UPPER ( ) & LOWER ( )

- Example

```
SELECT  UPPER(empid), UPPER(fname),
        UPPER(sname), UPPER(address)
FROM    employee
```

Table Data

EmpId	Fname	Sname	Address
N98	Jenny	Smith	23 RED RD.
V123	Leonardo	DiCaprio	4 HIGH STREET
N226	Dave	Smith	unit 2, 7 green street

Result Set

EmpId	Fname	Sname	Address
N98	JENNY	SMITH	23 RED RD.
V123	LEONARDO	DICAPRIO	4 HIGH STREET
N226	DAVE	SMITH	UNIT 2, 7 GREEN STREET

SQL: UPPER ( ) & LOWER ( )

- Example

```
SELECT  LOWER(empid), LOWER(fname),
        LOWER (sname), LOWER(address)
FROM    employee
```

Table Data

EmpId	Fname	Sname	Address
N98	Jenny	Smith	23 RED RD.
V123	Leonardo	DiCaprio	4 HIGH STREET
N226	Dave	Smith	unit 2, 7 green street

Result Set

EmpId	Fname	Sname	Address
n98	jenny	smith	23 red rd.
v123	leonardo	dicaprio	4 high street
n226	dave	smith	unit 2, 7 green street

SQL: UPPER() in where clause

- The **Upper() and Lower()** functions are often used in the **Where** clause.

```
SELECT  *
FROM    employee
WHERE   UPPER(sname) = 'SMITH';
```

Table Data

EmpId	Fname	Sname	Address
N98	Jenny	Smith	23 Red Rd.
V123	Leonardo	DiCaprio	4 High street
N226	Dave	Smith	Unit 2, 7 Green Street

Result Set

EmpId	Fname	Sname	Address
N98	Jenny	Smith	23 Red Rd.
N226	Dave	Smith	Unit 2, 7 Green Street

Note: …WHERE UPPER(sname) = 'Smith' ✘ - ***wrong!***

The result of UPPER(sname) can NEVER match lowercase letters

SQL:Between Operator

[NOT] <column-name> **BETWEEN** <value/exp> AND <value/exp>

WHERE age BETWEEN 30 and 40

Equivalent of WHERE age >= 30

AND age <= 40

The range in **inclusive** (includes 30 and 40)

- **Not widely used**
- Doesn't have equivalent command in most other programming languages.

SQL: Like Operator

The like operator allows you to have **partial matches** for text values

- You must use the **Like** keyword
  - Do **not** use the = operator.
- The % can substitute for 1 or more character(s)
- You can use more than one % symbol in your expression

[NOT] < column-name > LIKE <string with % wildcards>

… WHERE UPPER(name) LIKE 'J%'

	'J%';	'%J'	'%J%'	'%JO%'	'%JON%'	'%J%-%'
Johnson	✓		✓	✓		
Jonas	✓		✓	✓	✓	
Jones-Taylor	✓		✓	✓	✓	✓
Taylor-Jones			✓	✓	✓	
Major-Timms			✓	✓		✓
Pankaj		✓	✓			

SQL:Special Operators

The IN operator allows you to specify a **list of values** that may match values in the database

- Each value is separated by a comma

[NOT] < column-name > **IN** (<expression1> [,…])

WHERE upper(name) IN ('JONES', 'BROWN', 'LEE', 'SOO')

same as WHERE upper(name) = 'JONES' OR upper(name) = 'BROWN' OR …

WHERE age IN (24, 26, 28, 30)

same as WHERE age = 24 OR age = 26 OR age = 28 OR age = 30

SQL:Special Operators

The **IS NULL** operator allows you search for rows that have a value that contains a NULL value

[NOT] < column-name > **IS NULL / IS NOT NULL**

WHERE height IS NULL

WHERE name IS NOT NULL

- Do NOT use the equals sign — where height = Null ✗
- Do NOT use quotes — where height Is 'Null' ✗

SQL: Distinct keyword

Student Table contents:

firstname	country
Dave	Australia
Peter	England
Daniel	USA
Sue	China
Dave	Australia
Raj	India
Jim	Australia
Helen	India
Tina	China
Sue	China
Dave	Australia

```
SELECT  country
FROM    student
```

Result Set:

country
Australia
England
USA
China
Australia
India
Australia
India
China
China
Australia

Some rows in the result set are duplicated

SQL: Distinct keyword

The keyword **DISTINCT** removes **duplicate** rows from a **result set**

```
SELECT  country
FROM    student
```

country
Australia
England
USA
China
Australia
India
Australia
India
China
China
Australia

```
SELECT  DISTINCT country  FROM student
```

country
Australia
England
USA
China
India