

Logical Operators

After completing this module, you will be able to:

- **Use Logical Operators in writing queries.**
- **Link Logical Operators together with AND – OR – BETWEEN – IN and NOT IN.**
- **Identify unsatisfiable conditions.**
- **Use parentheses to show order of precedence and improve readability.**
- **Incorporate NULL syntax into queries correctly.**
- **Identify issues introduced by adding NULL into WHERE conditions.**

The “AND” Condition

Retrieve all employees with last name of “Brown”.

```
SELECT Last_Name, First_Name,  
       Department_number Dept#  
FROM Employee  
WHERE Last_Name = 'brown';
```

last_name	first_name	dept#
-----	-----	-----
Brown	Alan	401
Brown	Allen	801
Brown	Cary	567

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	Cary	567
Smith	Mary	900
Jones	Jimmy	401

Retrieve information for employee “Alan Brown”.

```
SELECT Last_Name, First_Name, Dept#  
FROM Employee  
WHERE Last_Name = 'brown'  
AND     First_Name = 'alan';
```

last_name	first_name	dept#
-----	-----	-----
Brown	Alan	401

All conditions linked with AND must evaluate “True” in the same row for the predicate to evaluate “True”.

The “OR” Condition

Contrast the AND with the OR, below:

Retrieve employees with last name “Brown” and first name “Mary”.

```
SELECT *  
FROM Employee  
WHERE Last_Name = 'brown'  
AND First_Name = 'mary';
```

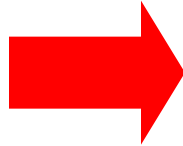
*** Query completed. No rows found.

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	Cary	567
Smith	Mary	900
Jones	Jimmy	401

Retrieve employees with last name “Brown” or first name “Mary”.

```
SELECT *  
FROM Employee  
WHERE Last_Name = 'brown'  
OR First_Name = 'mary';
```



last_name	first_name	dept#
-----	-----	-----
Smith	Mary	900
Brown	Alan	401
Brown	Allen	801
Brown	Cary	567

Mixing AND and OR

Contrast the 2 queries below.

Retrieve employees with last name “Brown” and work in department 401.

```
SELECT *  
FROM Employee  
WHERE Last_Name = 'brown'  
AND Department_number = 401;
```

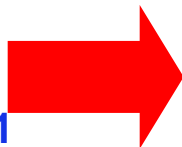
last_name	first_name	dept#
-----	-----	-----
Brown	Alan	401

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	Cary	567
Smith	Mary	900
Jones	Jimmy	401

Retrieve employees with last name “Brown” and Work in department 401, OR first name “Mary”. (“AND” is evaluated first)

```
SELECT *  
FROM Employee  
WHERE Last_Name = 'brown'  
AND Department_number = 401  
OR First_Name = 'mary';
```



last_name	first_name	dept#
-----	-----	-----
Smith	Mary	900
Brown	Alan	401

Parentheses and the Predicate

The use of the parentheses in this query illustrate the order of evaluation of “AND” and “OR” for the query on the previous page.

Retrieve employees with
last name “Brown” and
work in department 401,
or first name is “Mary”.

```
SELECT *  
FROM Employee  
WHERE (Last_Name = 'brown'  
AND Department_number = 401)  
OR (First_Name = 'mary');
```

last_name	first_name	dept#
-----	-----	-----
Smith	Mary	900
Brown	Alan	401

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	Cary	567
Smith	Mary	900
Jones	Jimmy	401

Rearranging the parentheses changes the business question.

Retrieve employees with last name of “Brown”,
AND first name of “Mary” or work in department 401.

```
SELECT *  
FROM Employee  
WHERE (Last_Name = 'brown')  
AND (Department_number = 401  
OR First_Name = 'mary');
```



last_name	first_name	dept#
-----	-----	-----
Brown	Alan	401

The IN Operator

Retrieve employee whose first names are in the following list (as a set).

```
SELECT *  
FROM Employee  
WHERE First_Name IN ('alan', 'allen');
```

last_name	first_name	dept#
-----	-----	-----
Brown	Alan	401
Brown	Allen	801

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	Cary	567
Smith	Mary	900
Jones	Jimmy	401

Rewrite Equivalent:

```
WHERE  
First_Name = 'allen'  
OR  
First_Name = 'alan';
```

The NOT IN Operator

Retrieve employee whose first names are NOT IN the following list (*as a set*).

```
SELECT *  
FROM Employee  
WHERE First_Name NOT IN ('alan', 'allen');
```

last_name	first_name	dept#
-----	-----	-----
Brown	Cary	567
Smith	Mary	900
Jones	Jimmy	401

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	Cary	567
Smith	Mary	900
Jones	Jimmy	401

Rewrite Equivalent:

```
WHERE  
First_Name <> 'allen'  
AND  
First_Name <> 'alan';
```

Exclusive OR

An exclusive OR is where we want one condition or the other condition to be true, but not both to be true.

```
SELECT *  
FROM Employee  
WHERE Last_Name = 'Brown'  
OR First_Name = 'Alan'
```

Typical “OR”.

last_name	first_name	dept#
-----	-----	-----
Brown	Allen	801
Brown	Alan	301
Adams	Alan	401

Employee Table

Last Name	First Name	Dept#
Adams	Alan	401
Brown	Allen	801
Murphy	Cary	567
Blair	Mary	900
Brown	Alan	301

```
SELECT *  
FROM Employee  
WHERE (Last_Name = 'Brown'  
AND First_Name <> 'Alan')  
OR  
(Last_Name <> 'Brown'  
AND First_Name = 'Alan');
```

The “XOR”.

Exclusive “OR” result

last_name	first_name	dept#
-----	-----	-----
Brown	Allen	801
Adams	Alan	401

(Note: “Alan Brown” does not appear in this result as he did in the previous result.)

The BETWEEN Operator

You can use BETWEEN to perform range constraints.
BETWEEN is inclusive.

```
SELECT *  
FROM Employee  
WHERE Department_number BETWEEN 401 and 501;
```

last_name	first_name	dept#
-----	-----	-----
Jones	Jimmy	401
Brown	Alan	401
Brown	?	...

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	?	567
Smith	Mary	?
Jones	Jimmy	401

NULL

- NULL represents something of unknown quantity or value.
- NULL is an SQL keyword.
- NULL is neither a data type nor a characteristic of data.
- When doing an ascending sort, NULLs sort before numbers and characters (*including spaces and zeroes*).

Any arithmetic operation involving a NULL operand (*literal, column, or expression*) computes a NULL result.

Col_A [+ - * /] **Col_B** = NULL

Any logical conditional expression involving a NULL operand (*literal, column, or expression*) evaluates as “unknown” (*Not True AND Not False*).

Col_A [>, >=, <, <=, =, <>] **Col_B** ➔ result is unknown

Conditional Expressions and NULL

Conditional expressions may involve columns or literals.

Checking whether a column IS, or IS NOT, NULL:

- `WHERE Salary = NULL` -- *Incorrect Usage*
- `WHERE Salary = "` -- *(two single quotes) - Incorrect Usage*
- `WHERE Salary NOT= NULL` -- *Incorrect Usage*
- `WHERE Salary <> NULL` -- *Incorrect Usage*
- `WHERE Salary IS NULL` -- *Correct Usage*
- `WHERE Salary IS NOT NULL` -- *Correct Usage*

Comparing two columns, either of which may contain a null.

- `WHERE Old_Salary [operator] New_Salary`

Logically and semantically, there is a subtle, but distinct, difference between:

- Comparing two columns, where one or both contain a null. (e.g., `WHERE Col1 = Col2`)
- Asking if a column or expression result is null. (i.e., `WHERE Col1 IS NULL`)

What Gets Returned

In SQL, the database only projects column values for those rows that contain data that satisfy all of the WHERE CONDITIONS.

If columns for a row are not projected, it is not because the conditions evaluated FALSE, but instead because they did not evaluate TRUE.

We will build on this logic throughout the course.

Observe:

For this
C1 value:

X

The following “T/F/Unknown” evaluations occur.

WHERE C1 = 'X' → “T” (row projected)

WHERE C1 = 'Y' → “F” (row not projected)

WHERE C1 = NULL → Unknown (row not projected)

WHERE C1 <> 'X' → “F” (row not projected)

WHERE C1 <> 'Y' → “T” (row projected)

WHERE C1 <> NULL → Unknown (row not projected)

NULL and the Business Question

Retrieve employees whose
job_code is not known.

```
SELECT *  
FROM Employee  
WHERE job_code IS NULL;
```

Retrieve employees whose job_code is unknown,
or work in department numbers less than 500.

```
SELECT *  
FROM Employee  
WHERE Department_number < 500  
OR job_code IS NULL;
```

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	?	567
Smith	Mary	?
Jones	Jimmy	401

```
SELECT *  
FROM Employee  
WHERE job_code = NULL;
```

***** Failure The user must use IS NULL
or IS NOT NULL to test for NULL
values.**

NOT NULL and the Business Question

Retrieve all employees who have job_code.

```
SELECT *  
FROM Employee  
WHERE job_code IS NOT NULL;
```

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	?	567
Smith	Mary	?
Jones	Jimmy	401

```
SELECT *  
FROM Employee  
WHERE job_code <> NULL;
```

*** Failure The user must use IS NULL or IS NOT NULL to test for NULL values.

Negating Conditions and Operators

To retrieve employees other than those in department number 401, you can perform either of the following:

Negate the operator -

```
SELECT *  
FROM Employee  
WHERE Department_number <> 401
```

Or negate the condition -

```
SELECT *  
FROM Employee  
WHERE NOT (Department_number = 401);
```

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	?	567
Smith	Mike	?
Jones	Jimmy	401

= 401	NE 401	
T	F	Not True
F	T	
F	T	
?	?	Not True
T	F	Not True

The IN Operator and NULL

Retrieve employees whose first names are in the following list (*as a set*).

```
SELECT *  
FROM Employee  
WHERE First_Name IN ('alan', 'allen');
```

The rewrite of this query shows an alternate method for writing this.

last_name	first_name	dept#
Brown	Alan	401
Brown	Allen	801

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	?	567
Smith	Mary	?
Jones	Jimmy	401

=

T

T

?

F

F

Not True

Not True

Not True

The NOT IN Operator and NULL

Retrieve employee whose first names are NOT IN the following list (as a set).

```
SELECT *  
FROM Employee  
WHERE First_Name NOT IN ('alan', 'allen');
```

last_name	first_name	dept#
-----	-----	-----
Smith	Mary	?
Jones	Jimmy	401

Employee Table

Last Name	First Name	Dept#	IN	NOT (IN)
Brown	Alan	401	T	F Not True
Brown	Allen	801	T	F Not True
Brown	?	567	?	? Not True
Smith	Mary	?	F	T
Jones	Jimmy	401	F	T

“<>” With OR vs. “<>” With AND

Contrast the previous (*NOT IN*) query result against these two query results.

```
SELECT *
FROM Employee
WHERE First_Name <> 'alan'
AND First_Name <> 'allen';
```

last_name	first_name	dept#
-----	-----	-----
Smith	Mary	?
Jones	Jimmy	401

```
SELECT *
FROM Employee
WHERE First_Name <> 'alan'
OR First_Name <> 'allen';
```

last_name	first_name	dept#
-----	-----	-----
Smith	Mary	?
Brown	Alan	401
Jones	Jimmy	401
Brown	Allen	801

Employee Table

Last Name	First Name	Dept#	OR	AND
Brown	Alan	401	T	F
Brown	Allen	801	T	F
Brown	?	567	?	?
Smith	Mary	?	T	T
Jones	Jimmy	401	T	T

T ↔ F Not True
 T ↔ F Not True
 ? ↔ ? Not True
 T ↔ T
 T ↔ T

Remember that when using “OR”, if only one condition evaluates “TRUE”, the predicate evaluates “TRUE”!

For the bottom query, this means:

For Alan Brown → “Alan” <> “Allen”

For Allen Brown → “Allen” <> “Alan”

NULL Literal in an IN-List

Recall the difference between comparing two columns where one or both may contain a null, (*e.g., WHERE Col1 = Col2*) vs. asking if a column or expression is null (*i.e., WHERE Col1 IS NULL*).

```
SELECT *  
FROM Employee  
WHERE Department_number IN (401, 403, null);
```

last_name	first_name	dept#
-----	-----	-----
Jones	Jimmy	401
Brown	Alan	401

Not included in
the result.



Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	?	567
Smith	Mary	?
Jones	Jimmy	401

Including NULL to an IN-List

Note the difference between this EXPLAIN and that for the query on the previous page.

```
SELECT *  
FROM Employee  
WHERE Department_number IS NULL  
OR Department_number IN (401, 403);
```

last_name	first_name	dept#
-----	-----	-----
Smith	Mary	?
Brown	Alan	401
Jones	Jimmy	401

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	?	567
Smith	Mary	?
Jones	Jimmy	401



NULL Literal in a NOT IN-List

This page is the reason for the immediately preceding pages.

No rows are returned due to the result of everything being linked with AND together with the outlined condition never being able to be true.

```
SELECT *  
FROM Employee  
WHERE Department_number NOT IN (401, 403, null);
```

*** Query completed. No rows found.

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	?	567
Smith	Mary	?
Jones	Jimmy	401

Three Versions of NOT IN

The following 3 queries return the same result.

```
SELECT *  
FROM Employee  
WHERE First_Name NOT IN ('alan', 'allen');
```

```
SELECT *  
FROM Employee  
WHERE NOT (First_Name = 'alan'  
OR      First_Name = 'allen');
```

```
SELECT *  
FROM Employee  
WHERE First_Name <> 'alan'  
AND    First_Name <> 'allen';
```

All three return the same.

Incorrect Sequencing of the BETWEEN

Contrast this query with the one following it:

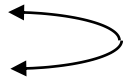
```
SELECT *  
FROM Employee  
WHERE Department_number BETWEEN 567 AND  
401;
```



The optimizer rewrites the previous query to the following.

How many rows can satisfy this condition?

```
SELECT *  
FROM Employee  
WHERE Department_number >= 567  
AND Department_number <= 401;
```



Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	?	567
Smith	Mary	?
Jones	Jimmy	401

This condition is deemed unsatisfiable by the optimizer.
This is not a failure!

***** Query completed. No rows found.**

Unsatisfiable BETWEEN with AND Residual

Since all AND'ed conditions must evaluate true for projection, this condition is also unsatisfiable.

```
SELECT *  
FROM Employee  
WHERE Department_number  
BETWEEN 567 AND 401  
AND Last_Name = 'brown';
```

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	?	567
Smith	Mary	?
Jones	Jimmy	401

*** Query completed. No rows found.

Unsatisfiable BETWEEN with OR Residual

Unsatisfiable conditions combined with OR need only satisfy the OR condition.

```
SELECT *  
FROM Employee  
WHERE Department_number BETWEEN 567  
AND 401  
OR Last_Name = 'brown';
```

last_name	first_name	dept#
Brown	Alan	401
Brown	Allen	801
Brown	?	567

Employee Table

Last Name	First Name	Dept#
Brown	Alan	401
Brown	Allen	801
Brown	?	567
Smith	Mary	?
Jones	Jimmy	401

Precedence of Operators

The following is the order of precedence for predicates (*WHERE conditions*).

Evaluation Precedence:

1. Parenthesis evaluated first
2. NOT operators
3. AND operators
4. OR operators
5. Operators of equal precedence evaluated from left to right

```
SELECT  Last_Name, Salary_Amount,  
        Manager_Employee_Number,  
        Job_Code, Department_Number  
FROM Employee  
WHERE  
  Manager_Employee_Number  
    NOT IN (1011, 801, 1017, 1019, 1005, 1003)  
OR Salary_Amount  
  BETWEEN 10000 AND 20000  
AND Department_Number IN (301, 401)  
OR Job_Code IN (111100, 211100)
```

With
Parentheses

```
SELECT  Last_Name, Salary_Amount,  
        Manager_Employee_Number,  
        Job_Code, Department_Number  
FROM Employee  
WHERE  
  (Manager_Employee_Number NOT IN  
   (1011, 801, 1017, 1019, 1005, 1003))  
OR (Salary_Amount  
   BETWEEN 10000 AND 20000  
AND Department_Number IN (301, 401))  
OR (Job_Code IN (111100, 211100))
```

Module 2: Summary

- Operators =, <>, <=, >=, <, >, IN, NOT IN and BETWEEN can be used as qualifiers.
- All conditions found of the WHERE clause must be satisfied for results to be projected.
- All AND'ed conditions in a list must evaluate true.
- Only one condition of an OR'ed list need evaluate true.
- Parentheses can be used to (reorder an) order of operations.
- NULL, when used in a condition, evaluates unknown.
- BETWEEN is inclusive when used to qualify on a range of values.
- IN, and NOT IN lists can be used as a short cut to replace long “AND/OR” linked conditions.

Module 2: Review Questions

True or False:

1. The IN operator is a short-cut for replacing a list of OR'ed conditions.
2. When using BETWEEN, only numeric data may be compared.
3. A NULL is treated like a zero (0) or a space(' ').
4. “Unsatisfiable” conditions are those which can never be true.
5. The following are equivalent operator conditions. **C1 > 500 vs. C1 >= 501**
6. You can include NULL inside an IN or NOT IN list
7. The items inside an IN list must be in order from lowest to highest.

Module 2: Lab Exercise

- 1) List last names, department numbers for employee in department 301, 401, and 501.
- 2) Project the last names of employees whose salary is greater than or equal to \$28,078.
- 3) Request a report of employees who have not been assigned to a department.
- 4) Modify #1 to include those employee who have a job code of either 512102, 432101.
- 5) Modify #4 to show only those whose salary amounts are between \$50,000 and \$60,000.
- 6) Although we only discussed “less than” in this module, project a distinct list of job codes which have been assigned to people and are greater than 510000 and sort the result descending.