

SQL Programming

Boolean / Logic Operations

When we have a single condition the predicate expression is easily formed. For example:

I'm looking for a new car, and since yellow is my favorite color, that's all I'll consider.

color = 'YELLOW'

But when we have a number of conditions or options, how do we translate those conditions into a predicate expression? For example:

I'm looking for a new car, and I want a color that won't show dirt so much. So, either white or silver will do.

color = 'WHITE'

color = 'SILVER'

The comparison operators, by themselves, allow the programmer to phrase only a single condition.

We need something that will allow us to link together multiple predicate expressions, and these linking elements are known as the *logical operators*.

There are three logical operators available to the SQL programmer:

AND

OR

NOT

The first two of these, AND and OR, are binary operators. This means that they require 2 operands, one operand on the left side, and one operand, or piece, on the right side.

(color = 'WHITE')	OR	(color = 'SILVER')
LEFT SIDE		RIGHT SIDE

Each operand must be a predicate expression that evaluates to: TRUE, FALSE, or UNKNOWN. In this regard, operands might be simple conditions or more complex compound conditions.

I am using parenthesis as a stylistic conventions to make it easier for you to see the scope of the OR operation.

The AND operator is used when the programmer wants to ensure that both pieces are TRUE.

Now remember our context. Simple conditions, written as predicate expressions, are used in the WHERE clause to identify which rows in the base table should be carried over into our result table.

We're looking at compound conditions that will also be expressed in WHERE clauses, to identify which rows in the base table get included in the result table.

Consider, for example, the problem in which our user community needs a report showing which of our clients has worked in both theater and film.

The conditions are:

theatre: Yes

film: Yes

Module 04: Logical Operations Page B-3 Problem 5-1 Code & Design

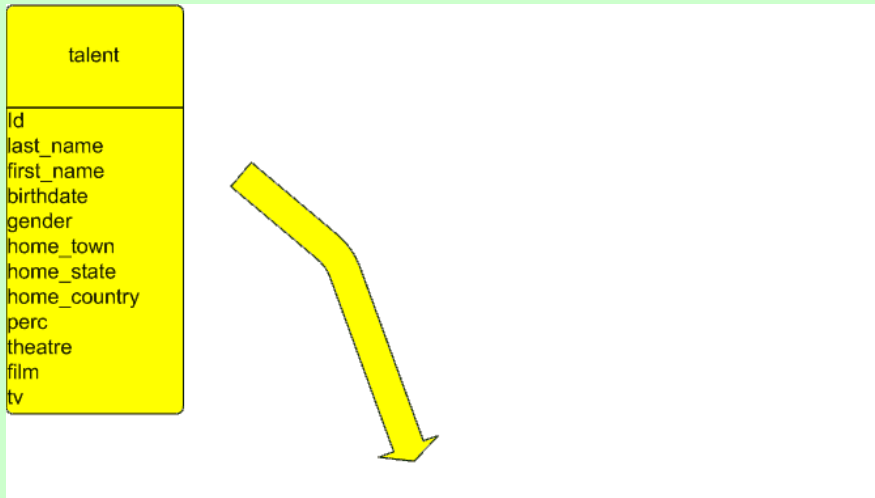
So let's put this together.

Step 1: Build the Table Build Chart (TBC)

Step 2: Double check your TBC solution

Step 3: Transform the TBC into code.

Notice how the multiple criteria are specified in the TBC.



Column Name/Expression	last_name	first_name	theatre	film
Table Name	talent	talent		
Alias				
Criteria			= 'Yes'	= 'Yes'
Display				

```
SELECT last_name, first_name, theatre, film
FROM    talent
WHERE   (theater = 'Yes') AND (film = 'Yes')
```

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox ...

File Edit View Favorites Tools Help Address Go

ORACLE iSQL*Plus [Password](#) [Log Out](#) [Help](#)

Script Location:

Enter statements:

```
SELECT last_name, first_name, theatre, film
FROM talent
WHERE (theatre = 'Yes') AND (film = 'Yes');
```

Output:

LAST_NAME	FIRST_NAME	THE	FIL
McKellen	Ian	Yes	Yes

Done Internet

Module 04: Logical Operations

Page B-5: Problem 5-1 Analysis

The screenshot shows the iSQL*Plus web interface in a Microsoft Internet Explorer browser window. The browser title is "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox ...". The address bar shows "http://plus.cox.net". The page has a header with the "ORACLE iSQL*Plus" logo and links for "Password", "Log Out", and "Help". Below the header is a "Script Location:" field with a "Browse..." button and a "Load Script" button. The main area is labeled "Enter statements:" and contains a SQL query:

```
SELECT last_name, first_name, theatre, film
FROM talent
WHERE (theatre = 'Yes') AND (film = 'Yes');
```

 Below the query is an "Execute" button and an "Output:" dropdown menu set to "Work Screen". To the right of the dropdown are "Clear Screen" and "Save Script" buttons. The results are displayed in a table with four columns: "LAST_NAME", "FIRST_NAME", "THE", and "FIL". The first row of data shows "McKellen", "Ian", "Yes", and "Yes".

LAST_NAME	FIRST_NAME	THE	FIL
McKellen	Ian	Yes	Yes

The predicate expression in the WHERE clause is evaluated by SQL as it examines the rows in the base table, one at a time, for inclusion in the result table.

In this example, only the rows that have both a Yes value in the theatre column and a Yes value in the film column will be displayed in the result table.

Both sides of the AND operation must be TRUE for this compound expression to evaluate TRUE. If either side is FALSE, the entire expression evaluates as FALSE

I like to use parenthesis to highlight the operands of a logical operation. I generally enclose each operand in a pair of parenthesis.

```
SELECT  last_name, first_name, theatre, film
FROM    talent
WHERE   (theater = 'Yes') AND (film = 'Yes')
```

And, as we do with other SQL keywords, AND is typed in uppercase characters.

The OR operator is used when the programmer will accept either condition.

In an academic setting we might receive a request from the dean who wants a report showing the names and gpa for English majors and Music majors.

We can identify these conditions in the problem statements:

major = 'ENGLISH'

major = 'MUSIC'

And the resultant WHERE clause might be written as:

```
WHERE (major = 'ENGLISH') OR (major = 'MUSIC')
```

When SQL evaluates a compound condition, ANDs take precedence over ORs.

How will this expression be evaluated?

```
WHERE (make = Mercedes)
      AND (color = Black)
      OR (year = 2003)
```

Only black mercedes, OR 2003 model year cars will be included.

How about this one?

```
WHERE (make = Toyota)
      OR (country = US)
      AND (year = 2003)
```

I want any Toyota, or any any 2003 model year American car.

Module 04: Logical Operations Page B-9: Overriding Precedence

The programmer can override the default rules for precedence by using parenthesis to indicate which conditions should be evaluated first.

How will this expression be evaluated?

```
WHERE (make = Mercedes)
      AND ((color = Black)
           OR (year = 2003))
```

Only black Mercedes, OR 2003 model year Mercedes will be included.

Notice how a little style helps the reader understand your intentions:

```
WHERE (make = Mercedes)
      AND ((color = Black) OR (year = 2003))
```

Module 04: Logical Operations Page B-10: Overriding Precedence(cont)

How about this one?

```
WHERE ((make = Toyota)
      OR (country = US))
      AND (year = 2003)
```

I only want to see American cars or Toyotas that were manufactured in 2003.

The same where clause, only 'prettier'

```
WHERE ((make = Toyota) OR (country = US))
      AND (year = 2003)
```

The user community would like a report showing the names, addresses and gpa's for all students on the honor roll (gpa \geq 3.5)

```
SELECT name, address, phone
FROM students
WHERE gpa  $\geq$  3.5
```

The computing club needs a roster of computer science majors

```
SELECT name, address, phone  
FROM   students  
WHERE  major = 'CompSci'
```

The computing club wants to recognize computer science majors who are on the honor roll.

```
SELECT name, address, phone  
FROM students  
WHERE (major = 'CompSci') AND (gpa >= 3.5)
```

In this example the user of parentheses is not required, but I use them nonetheless to make the program a little bit easier to read.

A local Hemet computing professional is funding a scholarship program for local (Hemet residents). The computing club wants to get the word out to qualified students.

Problem rephrased:

Local hemet residents – zip code

Qualified students:

Major = CompSci

gpa \geq 3.5

Try this before you view the solution –
I don't mean for you to type these in against the database – just sketch them out on paper and see if your solutions resemble mine.

Module 04: Logical Operations

A local Hemet computing professional is funding a scholarship program for local (Hemet residents). The computing club wants to get the word out to qualified students.

Problem rephrased:

Local hemet residents – zip code

Qualified students:

Major = CompSci

gpa >= 3.5

Page B-15: Scenario - 4

```
SELECT name, address, phone
FROM students
WHERE (zip = 92543 OR zip = 92544)
      AND (major = 'CompSci' AND gpa >= 3.5)
```

A local Hemet computing professional is funding a scholarship program for women in the sciences. Female computing science majors with a 3.2 gpa or better should be notified, as well as chemistry majors with a 3.6 gpa or better.

Problem rephrased:

Gender = female

Qualified students:

Major = CompSci and gpa \geq 3.2

major = Chem and gpa \geq 3.6

Try this before you view the solution

Module 04: Logical Operations

Page B-17: Scenario - 5

A local Hemet computing professional is funding a scholarship program for women in the sciences. Female computing science majors with a 3.2 gpa or better should be notified, as well as chemistry majors with a 3.6 gpa or better.

Problem rephrased:

Gender = female

Qualified students:

Major = CompSci and gpa \geq 3.2

major = Chem and gpa \geq 3.6

```
SELECT  name, address, phone
FROM    students
WHERE   (gender = 'F')
        AND (( major = 'CompSci' AND gpa >= 3.2)
            OR (major = 'Chem'    AND gpa >= 3.6))
```

I'm looking for a used car.

```
SELECT make, model, year, color, price  
FROM   used_cars;
```

I'm looking for a used Toyota.

```
SELECT make, model, year, color, price  
FROM   used_cars  
WHERE  make = 'TOYOTA';
```

I'm looking for a used Toyota or Jeep Cherokee.

```
SELECT make, model, year, color, price
FROM   used_cars
WHERE  make = 'TOYOTA'
       OR (make = 'JEEP' AND model = 'CHEROKEE');
```

I'm looking for a blue or silver Toyota or a late model black Jeep Cherokee.

Problem rephrased:

toyota blue or silver

black jeep cherokee \geq 2001

Try this before you view the solution

Module 04: Logical Operations

I'm looking for a blue or silver Toyota or a late model black Jeep Cherokee.

Problem rephrased:

toyota blue or silver

black jeep cherokee >= 2001

Page B-20: Scenario - 9

```
SELECT make, model, year, color, price
FROM   used_cars
WHERE
    (make = 'TOYOTA'
     AND (color = 'BLUE' OR color = 'SILVER'))
OR
    (make = 'JEEP' AND model = 'CHEROKEE'
     AND color = 'BLACK' AND year >= 2001);
```


The Table Build Charts (TBC) are somewhat lacking when it comes to modeling complex compound conditions.

They are most useful when it comes to identifying base tables, and the columns that are available, and also when trying to layout the columns in the result table.

Digression (of sort)

Total Quality Management is a philosophy of work, and the systems that are employed in the work place.

One of the key tenets of TQM is that every process in the business is worthy of study and is a candidate for improvement. In TQM you will never hear the phrase:

“If it ain’t broke, don’t fix it”

Rather, you are more likely to hear something along the lines of”

“If it ain’t broke, how can we improve it”

TQM is self evaluating. By this I mean that not only does TQM evaluate the business processes it also evaluates the TQM processes themselves that are used to evaluate the business processes.

We've seen this double speak before when we talked about meta data.

And now that we've moved on to discuss meta processes, let me point out one of the most important applications of meta processes.

Do you evaluate the way you do your work?

As a student it's easy to evaluate your work, you can simply look at your course grades and get some idea as to what kind of student you are.

But do you evaluate the way you study. Which study techniques are most profitable for you? For you, which study techniques work best for different subject matter.

If you only evaluate your work product (eg. your grade) , you're not going far enough if you hope to improve upon that work product.

TBCs are only a tool to help you with the design and coding SQL programs.

I've pointed out a shortcoming in this tool.

How are you going to proceed?

1. Throw it out, obviously it's not good enough.
2. Keep on using it until the instructor gives you something else
3. Think about it, and perhaps develop a better, more robust tool.

One of the most effective tools a programmer can use is a Programming Journal.

Back in the day, when I as a wee young programmer, I kept a journal that noted what I did well, and the things I did poorly on any programming project.

The things that worked were things I kept on doing, and the journal helped me reinforce these positive behaviors.

As to the things I did poorly, well, with the journal I was able to notice some patterns in my problems solving. And lo and behold there were some areas that I was overlooking, some mistakes that I was repeating. Once I realized these patterns I could modify my behavior.

There's more to the notion of a programming journal than these brief comments can describe, but the point of it all is this: evaluate your work and the way you do your work.

You can lead a horse to water, but you can't make it drink.

Throughout these modules I will share ideas and tips with you. I hope you take them to heart, I hope they benefit you in some fashion. But in the end it all boils down to you.

What are you going to do?

- Disregard them – they're obviously not going to be on any exam.
- Wait until lightning strikes – if it's important it'll pop up again in another discussion.
- Think about it, and find a way to exploit it for an advantage.

The last of the logical operators that we need to discuss is NOT.

NOT returns TRUE when given a FALSE value.

NOT returns FALSE when given a TRUE value.

Unlike AND and OR, NOT is not a binary operation that requires two operands. NOT is a unary operation, and it requires only one operand.

As a general rule, the use of the NOT keyword precedes the expression that is being negated.

Module 04: Logical Operations

Page D-2: NOT with Equality

See how the NOT operator both precedes the comparison and is 'outside' of the relational expression.

Also note, parentheses are not required in this instance, they are used to make the code more readable.

The screenshot shows the iSQL*Plus web interface in a Microsoft Internet Explorer browser window. The browser title is "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox ...". The address bar shows "http://plus.cox.net". The page has a header with the "ORACLE iSQL*Plus" logo and links for "Password", "Log Out", and "Help". Below the header, there is a "Script Location:" field with a "Browse..." button and a "Load Script" button. The main area is labeled "Enter statements:" and contains a text box with the following SQL query:

```
SELECT last_name, first_name
FROM talent
WHERE NOT (theatre = 'Yes')
```

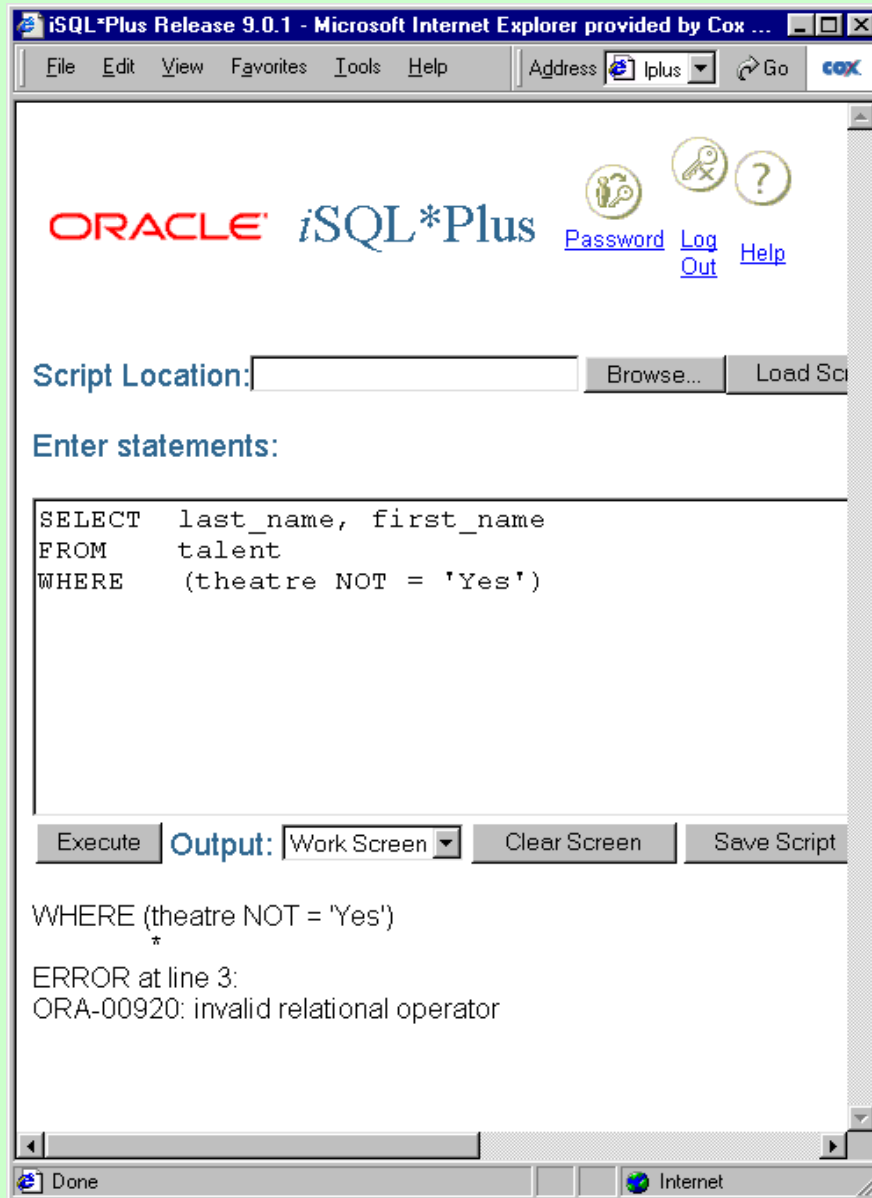
Below the text box, there are buttons for "Execute", "Output:" (with a dropdown menu set to "Work Screen"), "Clear Screen", and "Save Script". The results are displayed in a table with two columns: "LAST_NAME" and "FIRST_NAME".

LAST_NAME	FIRST_NAME
Willis	Bruce
Cruise	Tom
Kidman	Nicole
Redford	Robert
Ditt	Prod

The browser status bar at the bottom shows "Done" and "Internet".

Module 04: Logical Operations

Page D-3: NOT with Equality



Notice the error that SQL throws when the NOT keyword is moved 'inside' the comparison condition.

"Hey programmer, you've got an invalid relational operator in here"

And indeed there is.

NOT is not a relational operator, it's a logical operator, and as such, it is used outside of the relational comparison.

Module 04: Logical Operations

Page D-4: NOT with Inequality

ORACLE iSQL*Plus [Password](#) [Log Out](#) [Help](#)

Script Location:

Enter statements:

```
SELECT last_name, first_name
FROM talent
WHERE NOT (theatre <> 'Yes')
```

Output:

LAST_NAME	FIRST_NAME
McKellen	Ian

NOT may be used with any of the comparison operators, but that doesn't mean it should be.

Above all you want to write correct and meaningful code.

Correct in the sense that it performs according to specifications.

Meaningful in the sense that someone else can interpret what you've written.

Can you think of a more meaningful way of expressing this code????

When you find these uses of NOT in your code, consider rewriting them.

<u>NOT Usage</u>	<u>Alternative</u>
NOT (equality)	< >
NOT (inequality)	=
NOT (greater than)	< =
NOT (greater than or equal)	<
NOT (less than)	> =
NOT (less than or equal)	>

On occasion you will read or hear about Boolean operations. Boolean operations are the same as the logical operations we've been discussing.

We use the term *Boolean operations* in homage to George Boole, mathematician and logician whose work helped lay the foundation for all computing technologies.

"The Mathematical Analysis of Logic"

"An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities"

A comparison operation is an action that compares two items. This operation will be written in a predicate expression that uses a special comparison operator/symbol, in lieu of spelling out the action in English.

<u>Operation</u>	<u>Operator/Symbol</u>
Equality	=
Inequality	< >
Less than	<
Less than or equal to	< =
Greater than	>
Greater than or equal to	> =

Generally, most operations have a short cut symbol, but there are a few that don't.

The SQL standard does not provide a set symbols for the Boolean actions. ☹

Most other programming languages have special symbols/operators for the Boolean actions. For example:

<u>Operation</u>	<u>Operator/Symbol</u>
AND	&
OR	
NOT	!

Once again, the SQL standard does ***not*** require the use of these symbols, and you cannot be sure that these symbols will work in your database environment.

In all SQL implementations, the use of Boolean *verbs* does work.

The BETWEEN verb, or operator, gives the programmer a simple way of specifying an inclusive range of values.

BETWEEN is a ternary operator, hence requires 3 operands. The value being compared is placed on the left-hand side of the keyword (BETWEEN) and the operands that specify the range of values are placed on the right-hand side of the keyword. The two operands that specify the range of values are separated from one another with the AND keyword.

For example

perc BETWEEN 8 AND 10

As you grapple with trying to understand the BETWEEN phrase, all you need to do is remember that it is a short cut for the programmer. Any BETWEEN phrase can be rewritten as a compound condition.

The condition:

value BETWEEN range-1 AND range-2

Is equivalent to:

value \geq range-1 AND value \leq range-2

In this regard, BETWEEN provides two services to the programmer:

1. Simplifies the way some compound conditions can be expressed
2. Clearly highlights in the program exactly what is happening.

BETWEEN specifies an inclusive range of values. This means that when a data point matches either endpoint, or falls within that range of values, the expression is evaluated as TRUE, and that record will be included in the result table.

For example, in evaluating this BETWEEN phrase

perc BETWEEN 8 and 10

if the value of perc is 8, 9, or 10, then this statement evaluates as TRUE, and this row will be included in the result table.

Module 04: Logical Operations

Page E-4: Numeric ranges

The screenshot shows the iSQL*Plus web interface. The title bar reads "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox ...". The address bar shows "Address: iplus" and a "Go" button. The main content area is titled "Enter statements:" and contains the following SQL query:

```
SELECT last_name, first_name, perc
FROM talent
WHERE perc BETWEEN 7 AND 8
```

Below the query editor are buttons for "Execute", "Output: Work Screen", "Clear Screen", and "Save Script". The "Execute" button has been clicked, and the results are displayed in a table with the following data:

LAST_NAME	FIRST_NAME	PERC
Harris	Ed	7
Clooney	George	8
McKellen	Ian	8
Bloom	Orlando	7
Ford	Harrison	8
Ryder	Winona	7
Moore	Demi	7
Pacino	Al	7
Costner	Kevin	8
Jackson	Samuel L.	8

Below the table, it states "10 rows selected." The status bar at the bottom shows "Done" and "Internet".

Consider this problem:

For which of our clients do we charge the industry average percentage rate.

Rephrased:

Industry average 7% - 8%

This example demonstrates how BETWEEN can be used with numeric data types.

Module 04: Logical Operations

Page E-5: Date ranges

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Internet

File Edit View Favorites Tools Help Address <http://cisdb02.msjc.edu/isqlplus> Go

ORACLE iSQL*Plus [Password](#) [Log Out](#) [Help](#)

Script Location:

Enter statements:

```
SELECT last_name, first_name, birthdate
FROM talent
WHERE birthdate BETWEEN '01-JAN-1955' AND '31-DEC-1955'
```

Output:

LAST_NAME	FIRST_NAME	BIRTHDATE
Willis	Bruce	19-MAR-55
Costner	Kevin	18-JAN-55

Done Internet

Consider this problem:

Which of our clients were born in 1955?

Rephrased:

Which of our clients was born between January 1, 1955 and Dec 31, 1955?

BETWEEN seems to work with DATE data types as well!

Module 04: Logical Operations

Page E-6: Syntax

Be careful to insure that the low value is the first value listed in the range of values for the BETWEEN phrase.

If the 2nd value is less than the 1st value, the expression will always evaluate to FALSE.

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High Speed Internet

File Edit View Favorites Tools Help Address <http://cisdb02.msjc.edu/isqlplus> Go

ORACLE iSQL*Plus [Password](#) [Log Out](#) [Help](#)

Script Location:

Enter statements:

```
SELECT last_name, first_name, birthdate
FROM talent
WHERE birthdate BETWEEN '31-DEC-1955' AND '01-JAN-1955'
```

Output:

no rows selected

Done Internet

Module 04: Logical Operations

Page E-7: Syntax (cont)

This won't work either.

And SQL isn't smart enough to warn you about this obvious programming gaffe.

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer provided by Cox High ...

File Edit View Favorites Tools Help Address http://cisd Go

ORACLE iSQL*Plus Password Log Out Help

Script Location: Browse... Load Script

Enter statements:

```
SELECT last_name, first_name, perc
FROM talent
WHERE perc BETWEEN 10 AND 8
```

Execute Output: Work Screen Clear Screen Save Script

no rows selected

Done Internet

Module 04: Logical Operations

Page S-1: Summary

Comparison Operations:

A comparison operation is an action that compares two items. This operation will be written in a predicate expression that uses a special comparison operator/symbol, in lieu of spelling out the action in English.

<u>Operation</u>	<u>Operator/Symbol</u>
Equality	=
Inequality	< >
Less than	<
Less than or equal to	< =
Greater than	>
Greater than or equal to	> =

Generally, most operations have a short cut symbol, but there are a few that don't. For example:

BETWEEN

Boolean Operations:

<u>Operation</u>	<u>Operator/Symbol</u>
AND	&
OR	
NOT	!

Once again, the SQL standard does ***not*** require the use of these symbols, and you cannot be sure that these symbols will work in your database environment.

Module 04: Logical Operations

Page T-1: Terminology

Boolean operations, Boolean operators / verbs

Logic operations, logic operations / verbs
George Boole

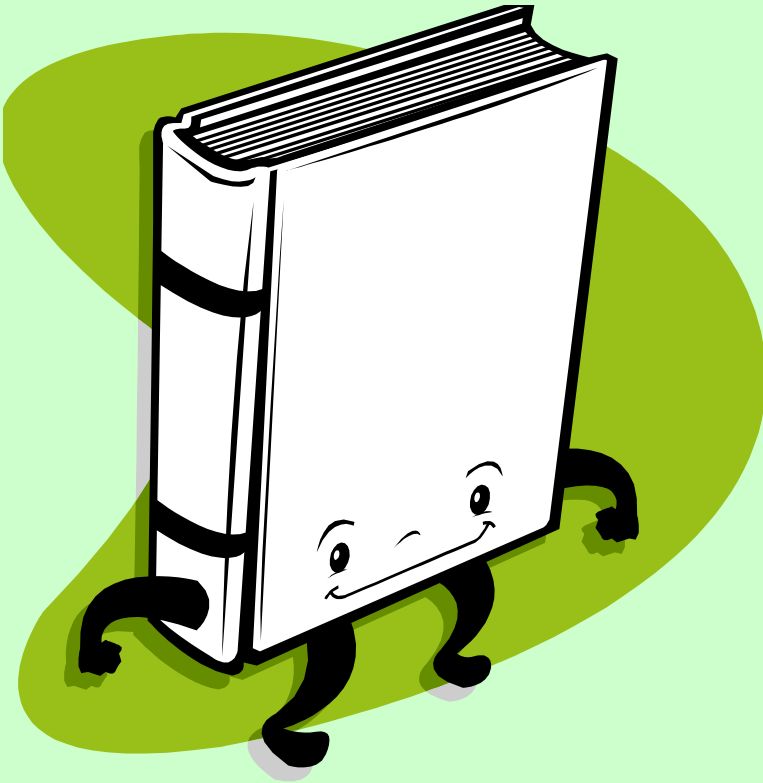
AND, OR, NOT
Compound conditions

Binary operators, unary operators, ternary operations

Precedence of operations

BETWEEN

inclusive



Please drop me an email if you noticed any errors in this module. I'd also appreciate reading your comments, criticisms, and or suggestions as to how this module could be improved.

Thanks,

bil



That's All