University of BRISTOL

IT SERVICES

Access: using operators and functions in queries

Reference document

Aims and Learning Objectives

This document aims to cover all the query language elements (expressions, functions etc) that are available for you to use in your queries, and on your forms and reports.

Document information

This document is available on the web. To find this, go to www.bristol.ac.uk/is/learning/resources and in the **Keyword** box, type the document code given in brackets at the top of this page.

Related documentation

Other related documents are available from the web at:

http://www.bristol.ac.uk/is/learning/resources

Contents

Introduction

Unless specified, the following query elements are available in all versions of Access. Where a version is mentioned, then that element is only available from that version.

Null values

Null values are excluded from calculations. For example, if an average is requested, the value returned is the average of the ***non-null*** values only.

The reserved word ***null*** matches null values and is used with the 'is' operator, not '='

Note that null values come first in ascending order, last in descending order.

Operators

Arithmetic operators

***, +, -, /**	standard operators For example, 5/2 returns 2.5
****	round to integer For example, 5\2 returns 2
^	power of For example, 2^3 returns 8
mod	remainder of result For example, 5 mod 2 returns 1

Character operator

&	concatenation For example, [initials]&" "&[surname]

Logical operators

and, or, not	(join two criterion together) For example ***town ="Bristol" and sex ="m"***
eqv	if both expressions true or both expressions false then return true else return false
imp	if the first expressions implies the second expression then return true else return false eg ***a>b imp b>c***
xor	if both expressions true or both false then return true else return false

Relational operators

<, <=, =, > >=, <>	surname>"M" no_children>10
between	birth_date between #1/1/70# and #1/1/80# surname between "A" and "L" value betwen 1 and 3
in	in ("Bristol","Bath") returns either Bristol or Bath
exists	not exists (select * from courses where courses.course_code=students.course_code) find students doing unknown courses
like	like "A* wild card search (cannot use'=' since * is not part of the value so 'like' indicates it is using * as a pattern

Character matching

*****	any number of any character	***S****	any string starting with S
?	a single character	***S?***	any 2 character string starting with S
#	a single digit	***S#***	any 2 character string starting with S and followed by a digit
[]	give a list of values to use	***S[A-M]*** ***S[!A-M]***	any 2 character string starting with S and followed by a letter in the range A to M any 2 character string starting with S followed by a letter not in the range A to M (note the use of ! for not)
[]	enclose wild card character	****[*]****	find an asterisk somewhere in string

Character expressions are enclosed in double quotes and are used in pattern matching with the ***like*** operator. The following can be used in character expressions:

Functions

Aggregate functions

avg(numeric data/expression)	average ***avg ([year])***
count(any kind of data)	count ***count(*)***
max(any kind of data)	maximum ***max ([year])***
min(any kind of data)	minimum ***min ([year])***
sum(numeric data/expression)	total ***sum([number of students])***
first(any kind of data)	first record field value ***first([year])***
last(any kind of data)	last record field value ***last([year])***
stdev(numeric data/expression)	standard deviation for a sample ***stdev([year])***
stdevp (numeric data/expression)	standard deviation for a population ***stdevp([year])***
var(numeric data/expression)	variance for a sample ***var([year])***
varp(numeric data/expression)	variance for a population ***varp([year])***

Domain aggregate functions used for subset of records

Note that condition on text needs quotes around the text. Since double or single quotes can be used for a condition, it is easier to standardise on double quotes for the function arguments, and single quotes for a string condition to avoid confusion. Note also that condition is optional. Very useful on forms and reports if based on a table rather than a query. Less likely to be used in a query because the condition can be expressed easily.

davg(numeric data/expression,table, condition)	average for a given condition ***davg ("[year]","[students]", "[sex]='female'")*** find average year of female students (note extra quotes around text string)
dcount(any kind of data, table, condition)	count for a given condition ***dcount ("*", "[students]" ," [year]=1")*** find how many students in year 1 (numeric so no extra quotes)
dmax(any kind of data, table, condition)	maximum for a given condition ***dmax ("[year]", "[students]", "[year]<4")*** max year of students lower than year 4
dmin(any kind of data, table, condition)	minimum for a given condition ***dmin ("[year]", "[students]", "[year]>3 and [sex]='female'")***
dsum(numeric data/expression, table, condition)	total for a given condition ***dsum ("[year]", "[students]", "[sex]='"& [which sex]&"'")*** Example of a parameter query. Note of the single quotes around the parameter ***which sex***
dfirst(any kind of data, table, condition)	first record field value for a given condition ***dfirst([year], "student")*** (notice condition is optional)
dlast(any kind of data, table, condition)	last record field value for a given condition
dstdev(numeric data/expression, table,	standard deviation for a sample for

condition)	a given condition
dstdevp (numeric data/expression, table, condition)	standard deviation for a population for a given condition
dvar(numeric data/expression, table, condition)	variance for a sample for a given condition
dvarp(numeric data/expression, table, condition)	variance for a population for a given condition

Number functions

exp	base of natural logarithm - complement to log
log	natural log
sqr	square root
atn	arctangent
cos	cosine
sin	sine
tan	tangent
fix	like int for positive values but returns first negative less than number. ***fix(-8.4)*** returns -8, ***fix(8.9***) returns 8
int	truncates. ***int(-8.4)*** returns -9, ***int(8.9)*** returns 8
abs	absolute. ***abs(-100)*** returns 100
sgn	returns value indicating sign of number(negative returns -1, zero 0, positive 1) ***sgn(-3)*** returns -1
rnd	random number- single real number between 0 and 1

Character functions

asc	ASCII value of char. ***asc("A")*** returns 65
chr(n)	character with given ASCII value. ***chr(65)*** returns "A"
instr(n, char1, char2, type)	position of char2 in char1, starting at n. type identifies if case sensitive (0) or not (1). ***instr("fred","d")*** returns 4
left(char,n)	first n letters. ***left("fred",2)*** returns "fr"
len(char)	length of char. ***len("fred")*** returns 4
lcase(char)	force to lower case. ***lcase("FRED")*** returns "fred"
ltrim(char)	remove leading spaces. ***ltrim(" fred")*** returns "fred"
mid(char,m,n)	middle letters starting at m length n. ***mid("fred",2,2)*** returns "re
space (number)	string with specified number of spaces. ***space(5)*** returns 5 spaces
strcomp (char1,char2,n)	compare two strings where n=0 for case-sensitive, 1 for not case-sensitive (default), 2 using database sort order. Returns -1 true, 0 equal, or 1 false. ***strcomp("a","b")*** returns -1
string (n,char)	return strings first character a specified number of times. ***string(5,"*")*** returns "*****"
trim (char)	strip leading and trailing spaces
right(char,n)	return n rightmost chars. ***right("fred",2)*** returns "ed"
rtrim(char)	remove trailing spaces

ucase (char)	force to uppercase. ***ucase("fred")*** returns "FRED"
nz (field, char)	return char if field is null (Access 97 upwards). ***nz(fee,"no fee given")*** returns either the value of the fee or the given text
hyperlinkpart (field, part)	return part of hyperlink field (Access 97 upwards). part = 0 for hyperlink (default), 1 for display text, 2 for address, 3 for subaddress
strconv (field,3)	convert text to first letter of each word capitalised (Access 97 upward). ***strconv("joe bloggs",3)*** returns "Joe Bloggs"

Date functions

dateadd(interval,n,date)	date plus n intervals (eg "m"). Intervals described in handling dates section. ***dateadd("d",3,date())*** adds 3 days to today's date
datediff(interval,d1, d2)	number of intervals (described in handling dates section)between dates d1 and d2 where d1 is earlier date. ***datediff("m",sent_date,date())*** gives the number of months betwen the two dates (***date()-sent_date*** gives the difference in days)
datepart(interval,date)	return specific part of date. ***datepart("yyyy",birth_date)*** is equivalent to ***year(birth_date)***
dateserial(year,month,day)	returns date. ***dateserial(1,12,2)*** returns 2nd December 2001. If year>29 then the previous century is returned
datevalue(char)	returns date. e.g datevalue("1 feb") returns "01/02/2003"
day(date)	returns day of month(1-31)

month(date)	returns month number(1-12)
year(date)	returns year (100-9999)
weekday(date)	returns weekday number(1-7)
hour(date)	returns hour(0-23)
cdate (char)	returns string as date (in current century if only 2 year digit and year<=29) (Access 97 up). 30th December 1899 returned if can not convert
minute(date)	returns minute(0-59)
second(date)	returns second(0-59)
timeserial(hour,minute,second)	returns time, date set to 30 December 1899 e.g ***timeserial(1,2,3)*** returns 01:02:03
timevalue(time)	returns time, date set to 30 December 1899 e.g ***timevalue(now( ))*** returns 16:31:53

Conversion functions

hex(number)	number to hexadecimal. ***hex(5)*** returns A
oct(number)	number to octal. ***oct(8)*** returns 10
ccur(number)	double to currency. ***ccur(2)*** returns £2.00
cdbl(number)	currency to double
cint(number)	double to integer (rounds)
clng(number)	single to long
isdate(expression)	true if can convert to date

isnull(expression)	true if expression contains no data
isnumeric(expression)	true if can convert to number
csng(number)	double to single (rounds)
cstr(number)	double to string(reserves leading space for sign)
str(number)	number to string(reserves leading space for sign)
rowidtochar(rowid)	row identifier to character string
format(x, fmt)	date or number x to specified format (see date and time handling section for list of formats available

Other functions

iif(expression,x,y)	return x if expression true else y. ***iif(sex="m", "male", "female")*** ***iif(isnull(Forms![Student Form]![course_code]), "don'tknow",Forms![Student Form]![course_code])***
dlookup (field, table, condition)	look up the field value from another table ***dlookup("[coursename]", "[subjects]","[subjects]. [subject]= '" & [students]. [subject] & "'")***

Special values

time()	current system time
timer ()	number of seconds since midnight
date()	today (no time)
now()	today (including time)
null	null value
user()	current username

Handling dates and times

The normal format for dates is DD-MM-YY, but dates may be given as DAY MONTH YEAR. The only exception is when importing dates when you must change the date field to be data type text. You can then change the data type back after the import.

- Press Alt/Tab to switch to Program Manager.
- Click on the International icon.
- Check that country is United Kingdom.
- Check the date format. This is the date format that will be used for all Windows applications.
- Change the long date format to include the day of the week.
- Change the year to display as 4 digits. (Access 2 assumes a 2 digit year as current century. Access 97 assumes 00 - 29 as current century, 30-99 as last.)

At least two components of the date must be given. If day number is omitted then 1 will be the default. If year is omitted then the current year will be default. If a date alone is stored, a time of 00:00 is assumed. Dates can range in value from 1 January 100 to 31 December 9999

The special value ***date*** and ***now*** return the current date and time.

Arithmetic can be performed on dates using the datediff.

Intervals used in the date functions dateadd, datepart, datediff are:

yyyy	year
q	quarter
m	month
y	day of year
d	day
w	weekday
ww	week
h	hour
n	minute
s	second

The date can be displayed in different formats using functions, for example:

format(birth_date, 'dddd d MMMM yyyy')

displays the date as, for example, 'Monday 23 January 1989'.

Elements of dates which may be used to specify the date format are listed below. The case of the output follows the case used in the format, for example ***MMMM*** produces FEBRUARY, ***mmmm*** produces February.

Element	meaning	sample output
YYYY	Year	1990
YY	Last 2 digits of year	90
Q	Quarter of year	1 2 3 4
M, or MM	Month	01 1
MMM M or MMM	Name of month or first 3 letters	February Feb
D	Day of week	7
DDDD or DDD	Name of day, or first 3 letters	Monday Mon
am/pm	Meridian indicator	am pm
AM/PM	Meridian indicator	AM PM
a/p	Meridian indicator	a p
A/P	Meridian indicator	A P
H or ***HH***	Hour of day (0-23)	2 02 10 22
N or NN	Minute	08 31
S or SS	Second	05 40

It may be worth considering using two fields for data and time to make validation easier (but not if subtracting two dates over midnight).

Character strings can be converted to dates using the ***format*** function. For example, if the 'first_built' column of the CAR table had been declared as char, and the data was entered

in a consistent format such as '01/01/00 12:00', it could be converted to dates when required by:

format (birth_date, 'dd/mm/yy hh:mm')

To convert a number to a date, it must first be converted to a char using the ***format*** function: For example, if the 'first_built' column had been declared as a 6 digit number, the actual values would be 5 or 6 digits (depending on the day of the month). These could be converted first to a 6 character string then to a date by:

format (format (birth_date, '099999'), 'ddmmyy')

Other format elements

Element	meaning	sample output
#	digit or blank	***format(1,"##")*** returns 1, ***format(1000,"#,###"***) returns 1,000
0	digit or 0	***format(1,"0#")*** returns 01
%	percent	***format(1,"#.00%"***) returns 100.00%
<	lowercase	***format("FRED","<")*** returns fred
>	uppercase	***format("fred",">")*** returns FRED

Financial functions

ddb (cost, salvage, life,* period*, factor)	return double declining balance
fv (rate, no-period, payment, present-value, type)	return future value
ipmt (rate, pay-period, no-period, present-value, future-value, type)	return interest payment
irr (array ofcash flow values, guess)	return interest rate of return
mirr (array of values, finance-rate, reinvest-rate)	return modified interest rate
hper (rate, payment, present value, future value, type)	return number of periods for an annuity
npv (rate, array of values)	return net value of investment
pmt (rate, no-period, present value, future value, type)	return payment for an annuity
pv (rate, no-period, payment, future value, type)	return present value
rate (no-period, payment, present value, future value, type, guess)	return interest rate for period
sln (cost, salvage, life)	return straight line depreciation
syd (cost, salvage, life, period)	return sum of years depreciation