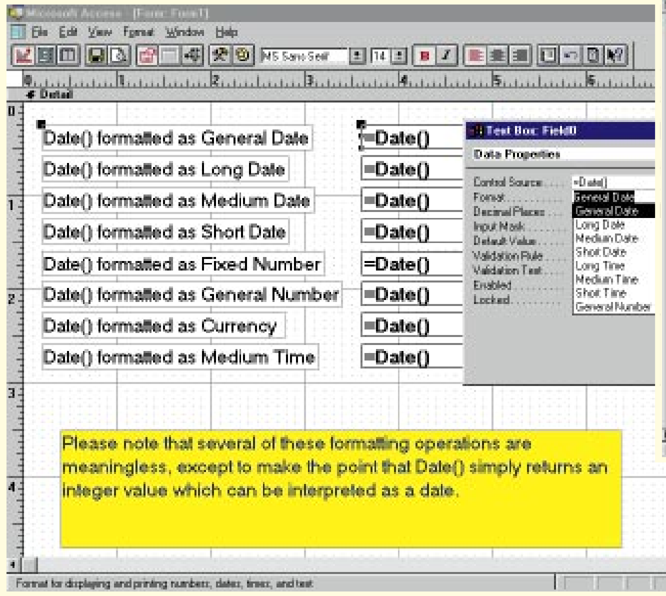
Do you have problems with **dates?**

Mark Whitehorn can't help you with your love life, but date/time types are another matter.

I have received several questions about handling dates in Access, particularly about ways in which specific dates (like the first day of the quarter) can be found. The following includes some elegant examples that I culled from the FAQs on the Microsoft Web site.

The Date/Time data type in Access is stored as a double-precision, floating-point number. The integer part represents the date and the decimal part represents the time. Clearly, we are only concerned with the integer part during this discussion.


Microsoft Access - [Form: Form1]
File Edit View Format Window Help
MS Sans Serif
14
Detail
Text Box: Field0
Data Properties

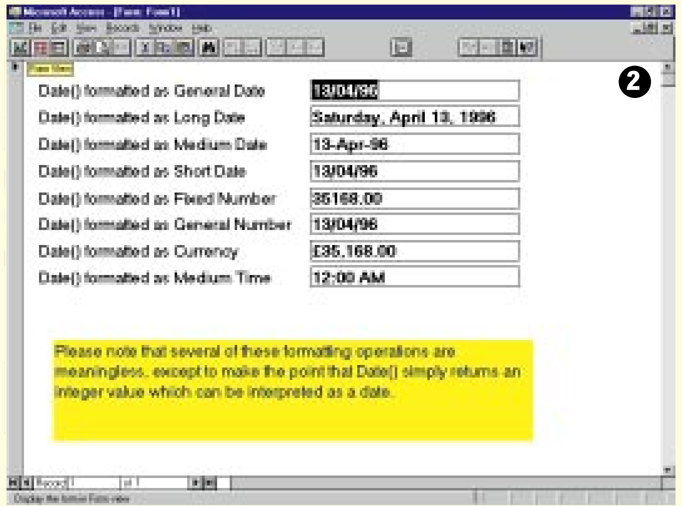
Date() formatted as General Date =Date()
Date() formatted as Long Date =Date()
Date() formatted as Medium Date =Date()
Date() formatted as Short Date =Date()
Date() formatted as Fixed Number =Date()
Date() formatted as General Number =Date()
Date() formatted as Currency =Date()
Date() formatted as Medium Time =Date()

Control Source: =Date()
Format: General Date
Decimal Places
Input Mask
Default Value
Validation Rule
Validation Text
Enabled
Locked

General Date
Long Date
Medium Date
Short Date
Long Time
Medium Time
Short Time
General Number

Please note that several of these formatting operations are meaningless, except to make the point that Date() simply returns an integer value which can be interpreted as a date.

Format for displaying and printing numbers, dates, times, and text

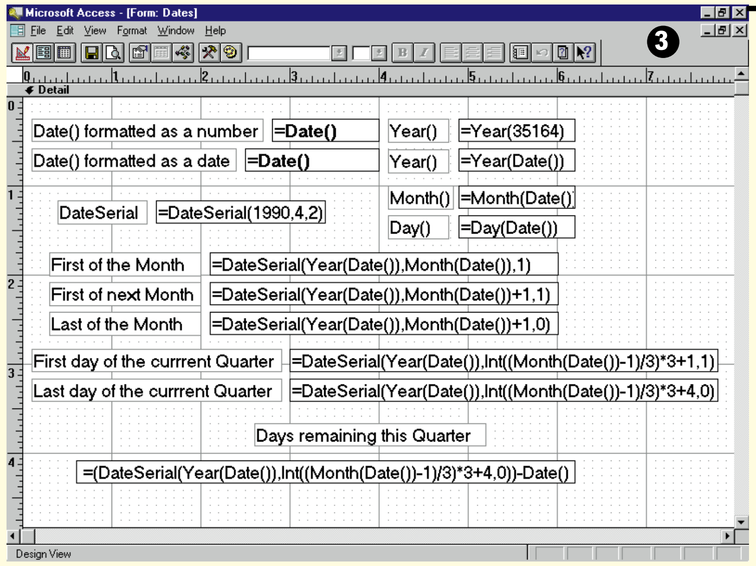

Microsoft Access - [Form: Form1]
File Edit View Records Window Help
Form View
Record: 1 of 1
Display the form in Form view

Date() formatted as General Date 13/04/96
Date() formatted as Long Date Saturday, April 13, 1996
Date() formatted as Medium Date 13-Apr-96
Date() formatted as Short Date 13/04/96
Date() formatted as Fixed Number 35168.00
Date() formatted as General Number 13/04/96
Date() formatted as Currency £35,168.00
Date() formatted as Medium Time 12:00 AM

Please note that several of these formatting operations are meaningless, except to make the point that Date() simply returns an integer value which can be interpreted as a date.

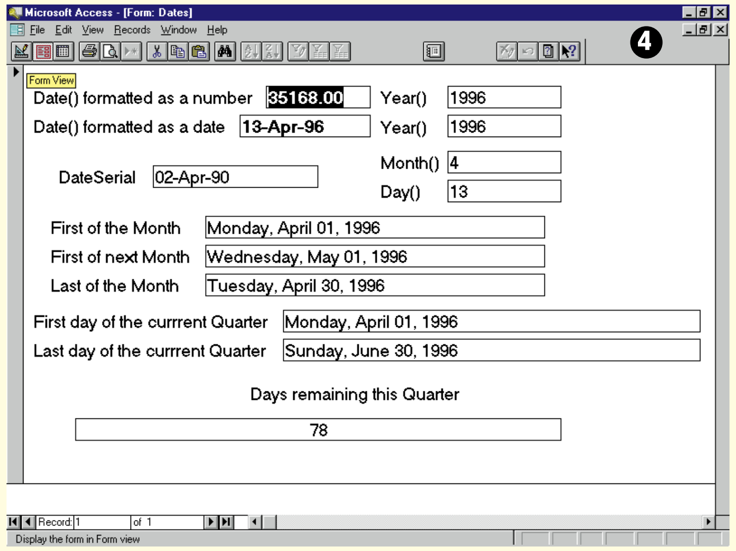
(1) *Setting the format properties to show how the value from Date() can be interpreted*

(2) *How the formats appear*


Microsoft Access - [Form: Dates]
File Edit View Format Window Help
Detail
Design View

Date() formatted as a number =Date() Year() =Year(35164)
Date() formatted as a date =Date() Year() =Year(Date())
Month() =Month(Date()
DateSerial =DateSerial(1990,4,2)
Day() =Day(Date())
First of the Month =DateSerial(Year(Date()),Month(Date()),1)
First of next Month =DateSerial(Year(Date()),Month(Date())+1,1)
Last of the Month =DateSerial(Year(Date()),Month(Date())+1,0)
First day of the currrent Quarter =DateSerial(Year(Date()),Int((Month(Date())-1)/3)*3+1,1)
Last day of the currrent Quarter =DateSerial(Year(Date()),Int((Month(Date())-1)/3)*3+4,0)
Days remaining this Quarter
=(DateSerial(Year(Date()),Int((Month(Date())-1)/3)*3+4,0))-Date()

(3) *The full set of date manipulations described in the text...*


Microsoft Access - [Form: Dates]
File Edit View Records Window Help
Form View
Record: 1 of 1
Display the form in Form view

Date() formatted as a number 35168.00 Year() 1996
Date() formatted as a date 13-Apr-96 Year() 1996
Month() 4
DateSerial 02-Apr-90
Day() 13
First of the Month Monday, April 01, 1996
First of next Month Wednesday, May 01, 1996
Last of the Month Tuesday, April 30, 1996
First day of the currrent Quarter Monday, April 01, 1996
Last day of the currrent Quarter Sunday, June 30, 1996
Days remaining this Quarter
78

(4) *and how they appear, at least how they did appear on the 13th April*

Access includes several useful functions for date manipulation. For example, Date() returns the current date (as a number of course). This can be formatted to appear in a variety of ways on-screen (say, in a form) by choosing the appropriate format from the properties box *(see Figs 1 and 2)*.

Year() Month() and Day() are three functions which will extract the relevant part from any date. Without wishing to over-stress the point, this means that these functions will extract that information from a number since dates are stored as numbers. Clearly, these functions can be given an actual number (such as 31234), or they can be given Date() which in turn will provide them with "today's" number.

DateSerial() can be used to manipulate the day, month, and year components of a date. It takes three arguments and returns a serial version of the date. Thus:

```
DateSerial(1990,4,2)
```

returns

```
02/04/90
```

This can be presented on screen in different ways by playing with the format.

I realise that so far this list of functions and their abilities must sound a little tedious, not to say boring. However, given a working knowledge of these five functions, you can combine them in such ways as heaven's wonders to perform.

For example, to find the first day of the current month, you can use:

```
=DateSerial(Year(Date()),
 Month(Date()), 1)
```

The first of the next month:

```
=DateSerial(Year(Date()),
 Month(Date()) + 1, 1)
```

The last day of the current month (a clever one this!):

➪

Fig 5

EMPLOYEES

Employee No	First Name	Last Name	Date Of Birth	Date Employed
1	Bilda	Groves	12/04/56	1/5/89
2	John	Greeves	21/03/67	1/1/90
3	Sally	Smith	1/5/67	1/4/92

Fig 6

SALES

Sale No	Employee No	Customer	Item	Supplier	Amount
1	1	Simpson	Sofa	Harison	£ 235.67
2	1	Johnson	Chair	Harrison	£ 453.78
3	2	Smith	Stool	Ford	£ 82.78
4	2	Jones	Suite	Harisonn	£3421.00
5	3	Smith	Sofa	Harrison	£ 235.67
6	1	Simpson	Sofa	Harrison	£ 235.67
7	1	Jones	Bed	Ford	£ 453.00

Fig 7

SALES2

Sale No	Employee No	Customer	Item	Supplier	Amount
3	2	Smith	Stool	Ford	£ 82.78
5	3	Smith	Sofa	Harrison	£ 235.67
213	3	Williams	Suite	Harisonn	£3421.00
216	2	McGreggor	Bed	Ford	£ 453.00
217	1	Williams	Sofa	Harrison	£ 235.67
218	3	Aitken	Sofa	Harison	£ 235.67
225	2	Aitken	Chair	Harrison	£ 453.78

```
=DateSerial(Year(Date()),
 Month(Date()) + 1, 0)
```

The first day of the current quarter:

```
=DateSerial(Year(Date()),
 Int((Month(Date()) - 1) / 3) *
 3 + 1, 1)
```

The last day of the current quarter:

```
=DateSerial(Year(Date()),
 Int((Month(Date()) - 1) / 3) *
 3 + 4, 0)
```

Number of days remaining in this quarter:

```
=(DateSerial(Year(Date()),Int((Month
Date())-1)/3)*3+4,0))-Date()
```

and so on (*see Figs 3 and 4, page 293*).

The possibilities are almost endless…

Gang screens

I don't want anyone to think I am obsessed with gang screens, but…

Windows 95

Right click on the DESKTOP, select NEW, FOLDER and name it "and now, the moment you've all been waiting for". Press Enter, right click on the folder and rename it to "we proudly present for your viewing pleasure". Press Enter again, right click and rename the folder once more to "The Microsoft Windows 95 Product Team!", now open the folder. (Just type in the words, not the inverted commas.)

Excel 95

First open a new Excel workbook and go to row 95. Select the entire row, then press tab once to move the cursor into column B (the entire row should remain selected). Pop down Help, About Excel, hold down CTRL and SHIFT together and select the Technical Support button. A new window will open. Walk forwards slightly, turn around 180 degrees, walk up to the wall and type "excelkfa". A secret door will open and I leave it up to you to navigate across the top of the wall to the next room.

What has this to do with databases? Er, the gang screens present data about the people who wrote the products, so the gang screens themselves must be databases. Sounds reasonable to me. *(But not to me — Ed.)*

SQL tutorial

Last month I started looking at SQL and began with the operators that it uses. We covered Restrict (aka Select), Union, Difference and Intersection. That only leaves two major ones, Product and Projection.

Once more, the sample tables are presented here *(Figs 5, 6 & 7)*.

Product

The product of two tables is a third which contains all the records in the first one, added to each of the records in the second. Thus, if the first table has 3 records and the second has 7, the product will have 21 records. The product of EMPLOYEES times SALES is shown in *Fig 8*.

This product operation has been applied quite correctly; however, the astute reader will note that the table in *Fig 8* contains seven rows which appear to be "meaningful" and 14 which are not. Note that we are dealing with a raw operator which takes no account of the values in fields, nor of any meaning that those values may imply or indicate.

In practice, the product operation may need to be modified by further operations in order to yield a meaningful answer.

The even more astute reader will have noticed that *Fig 8* contains two fields with identical field names. This state of affairs is not permitted in a table, and in practice an RDBMS will have to cope with this in some way, perhaps by renaming one of the fields.

However, as has been said before, these relational operators are the "primitives" from which more complex systems are constructed and it's usually the job of these higher-level constructs to cope with problems like this.

Projection

Projection selects one or more fields from a table and generates a new table which contains all the records, but only the selected fields. Thus, if we project EMPLOYEES on [FirstName] and [LastName] the result is as *Fig 9*.

This seems straightforward, but if we project SALES on [EmployeeNo] and [Customer] the result is as *Fig 10*.

Despite the fact that [SALES] has seven records, the answer table has only six. This is because one of them:

```
1        Simpson
```

would be duplicated in the answer table, and tables cannot contain duplicated records.

Fig 8

Employee No	First Name	Last Name	Date Of Birth	Date Employed	Sale No	Employee No	Customer	Item	Supplier	Amount
1	Bilda	Groves	12/04/56	1/5/89	1	1	Simpson	Sofa	Harison	£ 235.67
1	Bilda	Groves	12/04/56	1/5/89	2	1	Johnson	Chair	Harrison	£ 453.78
1	Bilda	Groves	12/04/56	1/5/89	3	2	Smith	Stool	Ford	£ 82.78
1	Bilda	Groves	12/04/56	1/5/89	4	2	Jones	Suite	Harisonn	£3421.00
1	Bilda	Groves	12/04/56	1/5/89	5	3	Smith	Sofa	Harrison	£235.67
1	Bilda	Groves	12/04/56	1/5/89	6	1	Simpson	Sofa	Harrison	£ 235.67
1	Bilda	Groves	12/04/56	1/5/89	7	1	Jones	Bed	Ford	£ 453.00
2	John	Greeves	21/03/67	1/1/90	1	1	Simpson	Sofa	Harison	£ 235.67
2	John	Greeves	21/03/67	1/1/90	2	1	Johnson	Chair	Harrison	£ 453.78
2	John	Greeves	21/03/67	1/1/90	3	2	Smith	Stool	Ford	£ 82.78
2	John	Greeves	21/03/67	1/1/90	4	2	Jones	Suite	Harisonn	£3421.00
2	John	Greeves	21/03/67	1/1/90	5	3	Smith	Sofa	Harrison	£235.67
2	John	Greeves	21/03/67	1/1/90	6	1	Simpson	Sofa	Harrison	£ 235.67
2	John	Greeves	21/03/67	1/1/90	7	1	Jones	Bed	Ford	£ 453.00
3	Sally	Smith	1/5/67	1/4/92	1	1	Simpson	Sofa	Harison	£ 235.67
3	Sally	Smith	1/5/67	1/4/92	2	1	Johnson	Chair	Harrison	£ 453.78
3	Sally	Smith	1/5/67	1/4/92	3	2	Smith	Stool	Ford	£ 82.78
3	Sally	Smith	1/5/67	1/4/92	4	2	Jones	Suite	Harisonn	£3421.00
3	Sally	Smith	1/5/67	1/4/92	5	3	Smith	Sofa	Harrison	£235.67
3	Sally	Smith	1/5/67	1/4/92	6	1	Simpson	Sofa	Harrison	£ 235.67
3	Sally	Smith	1/5/67	1/4/92	7	1	Jones	Bed	Ford	£ 453.00

If we projected SALES on [SaleNo], [EmployeeNo] and [Customer] then the answer table *(Fig 11)* will contain seven records because in the original table the values in [SalesNo] are unique.

Summary

The following is not rigorous, nor is it detailed, but if you have read and understood the previous section it should provide a quick reference to remind you what the operators are and what they do.

Two of the operators (**Restriction** and **Projection**) operate on single tables.

● **Restriction** (aka Select) extracts records.

● **Projection** extracts fields.

The remaining four operators (**Union, Difference, Intersection** and **Product**) all perform operations on two tables.

● **Union** adds the records from two tables together.

● **Difference** subtracts the records in one table from that in another.

● **Intersection** locates the records that are common to two tables.

● **Product** multiplies the records in the two tables together.

Fig 9

ANSWER

FirstName	LastName
Bilda	Groves
John	Greeves
Sally	Smith

Fig 10

SALES

Employee No	Customer
1	Johnson
1	Jones
1	Simpson
2	Jones
2	Smith
3	Smith

Fig 11

ANSWER

SaleNo	Employee No	Customer
1	1	Simpson
2	1	Johnson
3	2	Smith
4	2	Jones
5	3	Smith
6	1	Simpson
7	1	Jones

Fig 13

Employee No	First Name	Last Name	Date Of Birth	Date Employed	Sale No	Customer	Item	Supplier	Amount
1	Bilda	Groves	12/04/56	1/5/89	1	Simpson	Sofa	Harison	£ 235.67
1	Bilda	Groves	12/04/56	1/5/89	2	Johnson	Chair	Harrison	£ 453.78
1	Bilda	Groves	12/04/56	1/5/89	6	Simpson	Sofa	Harrison	£ 235.67
1	Bilda	Groves	12/04/56	1/5/89	7	Jones	Bed	Ford	£ 453.00
2	John	Greeves	21/03/67	1/1/90	3	Smith	Stool	Ford	£ 82.78
2	John	Greeves	21/03/67	1/1/90	4	Jones	Suite	Harisonn	£3421.00
3	Sally	Smith	1/5/67	1/4/92	5	Smith	Sofa	Harrison	£235.67

Assuming that each operation is performed on a pair of tables with 20 and 10 records respectively, the number of records in the answer table will have:

● **Union** — between 20 and 30

● **Difference** — between 20 and 10 (assuming that we subtract the table with 10 records from that with 20)

● **Intersection** — between 0 and 10

● **Product** — 200

Join

Join is often used as a relational operator and it can be built up from the simpler ones described earlier. Think of it as a mixture of the product and restriction operators, sometimes with an added dash of projection.

Suppose that you want to examine the sales that have been made by your employees. In order to do this, you need information from both the EMPLOYEES and SALES tables. (In fact, the table SALES2 contains information about more sales, and if we wanted to include this information we would first use the union operator. However, for the sake of brevity, we will assume that we are only interested in the sales recorded in SALES.)

Fig 12

Employee No	First Name	Last Name	Date Of Birth	Date Employed	Sale No	Employee No	Customer	Item	Supplier	Amount
1	Bilda	Groves	12/04/56	1/5/89	1	1	Simpson	Sofa	Harison	£ 235.67
1	Bilda	Groves	12/04/56	1/5/89	2	1	Johnson	Chair	Harrison	£ 453.78
1	Bilda	Groves	12/04/56	1/5/89	6	1	Simpson	Sofa	Harrison	£ 235.67
1	Bilda	Groves	12/04/56	1/5/89	7	1	Jones	Bed	Ford	£ 453.00
2	John	Greeves	21/03/67	1/1/90	3	2	Smith	Stool	Ford	£ 82.78
2	John	Greeves	21/03/67	1/1/90	4	2	Jones	Suite	Harisonn	£3421.00
3	Sally	Smith	1/5/67	1/4/92	5	3	Smith	Sofa	Harrison	£235.67

Go-slow on speed

I have said that I'd look at the speed of the different SQL solutions to the long-running meter problem. However, last month I published another solution and asked for comments on both its match to the relational model and its speed potential. There is a delay between my writing this column and you reading it, such that as I write this month's you still haven't read last month's. So, I'll delay the speed issue one more month, and then let you know what I found.

The first job is to perform a projection on these tables. Next we need to perform a selection which removes the records where EMPLOYEES.EmployeeNo is not equal to SALES.EmployeeNo. (Finally, we might optionally remove some fields from the answer table.)

We might express a join in this form:

```
EMPLOYEES JOIN (EMPLOYEES.EmployeeNo
      = SALES.EmployeeNo) SALES
```

and the result would be as shown in *Fig 12*.

To be a little more accurate, the table in *Fig 12* is the result of what is known as an **equijoin**. The table in *Fig 13* is the result of a **natural** join.

The simplistic difference is that one of the fields used in the join has been removed from the answer table.

There is slightly more to this than meets the eye, however. Joins come in several flavours and you will hear people talking about natural, equi, theta, outer and semi-joins.

While it is true that all of these joins differ in usefulness, they nevertheless all find their way into discussions about SQL. So, we'll have a look at them in more detail next month.

PCW Contacts

Mark Whitehorn welcomes readers' correspondence and ideas for the Databases column. He's on **m.whitehorn@dundee.ac.uk**