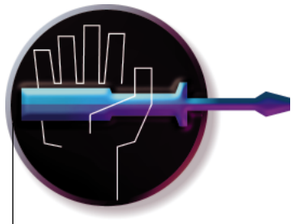

# Practical **joinery**

In part III of our SQL tutorial, Mark Whitehorn explains how multiple tables work together and highlights the distinction between left, right, inner and outer joins.

Last month I promised to continue dealing with the subject of working with multiple tables and how to use the SELECT statement to draw data from more than one. This month, I'll look at how it works and what it's doing.

The sample tables and the joins between them are shown in the two screenshots (Figs 1 & 2). In the sample Access database, which is included on our cover-mounted CD, I have removed the joins. Some of the SQL commands alter the sample tables so I have included extra versions of those, stored with the word SAFE after the name. Once you have run the SQL statement, you can delete the altered table and replace it with a copy of the "safe" version. This replacement process is easier if the joins are removed.

The only difference between these tables and the way they appeared in last month's issue is that John Greeves has lost his licence, so he is no longer allocated a company car. (This does not affect any of the examples shown in previous months.)

Note that in order to maintain consistency with my previous article, the first SQL statement this month is labelled as "Multi-Table 3" (not "Multi-Table 1") in the Access database provided on the cover-mounted CD-ROM. Last month, we looked at SQL which worked across multiple tables. The statement we finished with was:

```
SELECT SALES.Customer,
EMPLOYEES.LastName, SALES.Amount
FROM SALES, EMPLOYEES
WHERE SALES.EmployeeNo =
 EMPLOYEES.EmployeeNo;
```

which yields:

| Customer | LastName | Amount |
|---|---|---|
| Simpson | Groves | £235.67 |
| Johnson | Groves | £453.78 |
| Simpson | Groves | £235.67 |
| Jones | Groves | £453 |
| Smith | Greeves | £82.78 |
| Jones | Greeves | £3,421 |
| Smith | Smith | £235.67 |

In order to see how this is working, we can add the EmployeeNo fields, from the two tables, into the answer table and remove the WHERE statement. (I've used synonyms for the tables to reduce the size of the table headings.)

```
SELECT S.Customer, E.LastName,
S.Amount, S.EmployeeNo,
E.EmployeeNo
FROM SALES S, EMPLOYEES E;
```

See the table in Fig 3 (page 258).

Without a WHERE clause, the answer table contains every record in the SALES table matched against every record in the EMPLOYEE table, giving 4 x 7 = 28 records. The WHERE clause ensures that we see in the answer table only those records in which the EmployeeNo in SALES matches the EmployeeNo in EMPLOYEES. This is logically reasonable since we are using the value in SALES.EmployeeNo to indicate which employee made the sale.

It is possible to join more than two tables by adding to the WHERE clause. For example:

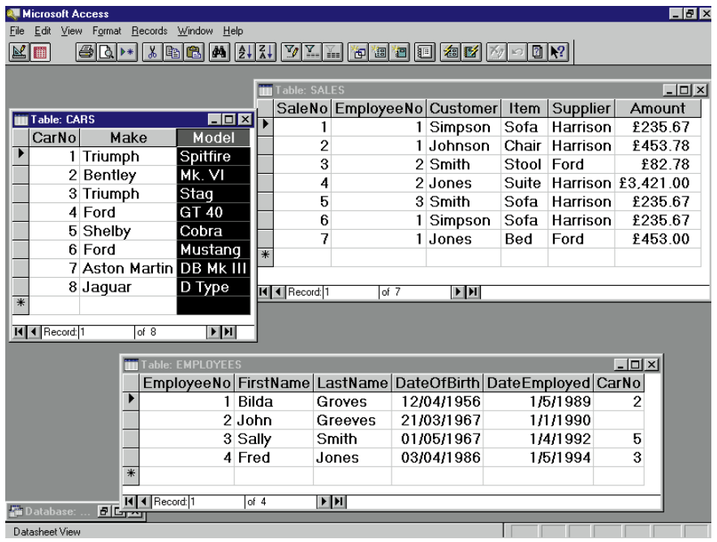


**Fig 1 The tables used in the examples. I have set the dates to show four-digit years in response to email from readers worried about the coming of the millennium. In fact, Access stores all dates as four-digit years: it is just the default format which doesn't show them**





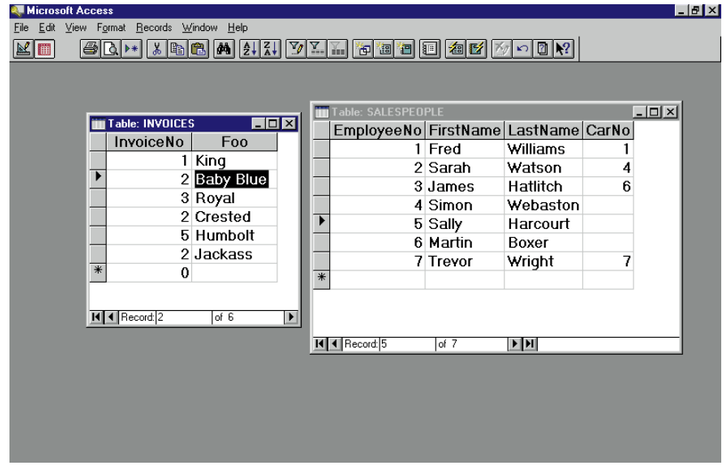


**Fig 2 Two tables used in a couple of the later examples.The Foo field is simply a shorthand representation of the boring information that would usually be displayed in an invoice**

```
SELECT SALES.Customer,
EMPLOYEES.FirstName, CARS.Make,
CARS.Model
FROM CARS, EMPLOYEES, SALES
WHERE EMPLOYEES.EmployeeNo =
SALES.EmployeeNo
AND EMPLOYEES.CarNo = CARS.CarNo;
```

| Customer | FirstName | Make | Model |
|---|---|---|---|
| Simpson | Bilda | Bentley | Mk. VI |
| Johnson | Bilda | Bentley | Mk. VI |
| Simpson | Bilda | Bentley | Mk. VI |
| Jones | Bilda | Bentley | Mk. VI |
| Smith | Sally | Shelby | Cobra |

Note that this query is finding the car driven by the sales person who dealt with a given customer, so it isn't supposed to present particularly meaningful information.

The most recent ISO standard for SQL (SQL-92) includes a new way of expressing joins such that:

```
SELECT SALES.Customer,
EMPLOYEES.LastName, SALES.Amount
FROM SALES, EMPLOYEES
WHERE SALES.EmployeeNo =
EMPLOYEES.EmployeeNo;
```

| Customer | LastName | Amount |
|---|---|---|
| Simpson | Groves | £235.67 |
| Johnson | Groves | £453.78 |
| Simpson | Groves | £235.67 |
| Jones | Groves | £453 |
| Smith | Greeves | £82.78 |
| Jones | Greeves | £3,421 |
| Smith | Smith | £235.67 |

can be replaced by:

```
SELECT SALES.Customer,
EMPLOYEES.LastName, SALES.Amount
FROM SALES INNER JOIN EMPLOYEES
ON EMPLOYEES.EmployeeNo =
SALES.EmployeeNo;
```

This produces the same answer table and is generally considered to be more readable. However, it does raise another question: what is this INNER business?

### Inner (natural) joins

Suppose your boss says: "Give me a list of all the cars and the sales person to whom each is currently allocated."

You are immediately tempted to use the SQL statement:

```
SELECT CARS.Make, CARS.Model,
EMPLOYEES.FirstName,
EMPLOYEES.LastName
FROM CARS INNER JOIN EMPLOYEES
ON CARS.CarNo = EMPLOYEES.CarNo;
```

but this will give the answer:

| Make | Model | FirstName | LastName |
|---|---|---|---|
| Bentley | Mk. VI | Bilda | Groves |
| Triumph | Stag | Fred | Jones |
| Shelby | Cobra | Sally | Smith |

which doesn't list all the cars because that delectable D-type Jaguar, for instance, hasn't been allocated to anyone.

In fact, your boss has phrased the question badly, since the original question assumes that every car is allocated to an employee and this is not the case. However, voicing your opinion about the inexact use of English is likely to be a CLM (Career Limiting Move). Better to keep quiet and find a query that will list all the cars and show what has been allocated to which lucky employees.

But before that, we'll have a look at what's wrong with the query shown above. By default, a join combines the two tables via fields that have identical values. This is known as a "Natural" or "Inner" join. However, if one or both of the fields contain unique values (I am using the term "unique" to mean that the values are found in one table but not the other) then the join ignores the records that are associated with these values. Thus, the table CARS has a delightful Aston Martin, CarNo = 7, but since there is no corresponding value in EMPLOYEES.CarNo, this fine automobile never appears in the answer table.

So instead of a Natural join, what you need to use here is an Unnatural join. Okay, I admit it, that was just to see if you were awake. It is really known as an "Outer" join.

### Outer joins

There are two distinct types of Outer join, Left and Right.

The following SQL statement

```
SELECT CARS.Make, CARS.Model,
EMPLOYEES.FirstName,
EMPLOYEES.LastName
FROM CARS LEFT JOIN EMPLOYEES
ON CARS.CarNo = EMPLOYEES.CarNo;
```

yields:

| Make | Model | FirstName | LastName |
|---|---|---|---|
| Triumph | Spitfire | | |
| Bentley | Mk. VI | Bilda | Groves |
| Triumph | Stag | Fred | Jones |
| Ford | GT 40 | | |
| Shelby | Cobra | Sally | Smith |
| Ford | Mustang | | |
| Aston Martin | DB Mk III | | |
| Jaguar | D Type | | |

p258 ➢

Essentially, the substitution of LEFT JOIN for INNER JOIN has made all the difference.

The other sort of outer join is RIGHT, which simply ensures that every record in the table on the right-hand side of the join is included in the answer table, so

```
SELECT CARS.Make, CARS.Model,
EMPLOYEES.FirstName,
EMPLOYEES.LastName
FROM CARS RIGHT JOIN EMPLOYEES
ON CARS.CarNo = EMPLOYEES.CarNo;
```

yields:

| Make | Model | FirstName | LastName |
|---|---|---|---|
| | | John | Greeves |
| Bentley | Mk. VI | Bilda | Groves |
| Triumph | Stag | Fred | Jones |
| Shelby | Cobra | Sally | Smith |

It is important to note that

```
SELECT CARS.Make, CARS.Model,
EMPLOYEES.FirstName,
EMPLOYEES.LastName
FROM EMPLOYEES LEFT JOIN CARS
ON CARS.CarNo = EMPLOYEES.CarNo;
```

produces exactly the same answer table, namely:

| Make | Model | FirstName | LastName |
|---|---|---|---|
| | | John | Greeves |
| Bentley | Mk. VI | Bilda | Groves |
| Triumph | Stag | Fred | Jones |
| Shelby | Cobra | Sally | Smith |

In other words, the LEFT and RIGHT simply refer to the tables as named in the SQL statement. So

```
FROM EMPLOYEES LEFT JOIN CARS
```

and

```
FROM CARS RIGHT JOIN EMPLOYEES
```

will include all the employees and some of the cars:

```
FROM CARS LEFT
JOIN EMPLOYEES
```

and

```
FROM EMPLOYEES
RIGHT JOIN CARS
```

will include all the cars and some of the employees.

The upshot is that you can have all of the cars some of the time, and indeed, you can have all of the people some of the time. But what you really want to know is, can we have all of the cars and all of the people all of the time?

The answer, not surprisingly, is "Yes". But in order for that to happen, we need to make use of UNION and I'll be covering this and other topics in next month's column.

■ You will find the Access sample file in the Resources section on this month's cover-mounted CD.

## Fig 3

| Customer | LastName | Amount | S.EmployeeNo | E.EmployeeNo |
|---|---|---|---|---|
| Simpson | Groves | £235.67 | 1 | 1 |
| Johnson | Groves | £453.78 | 1 | 1 |
| Smith | Groves | £82.78 | 2 | 1 |
| Jones | Groves | £3,421 | 2 | 1 |
| Smith | Groves | £235.67 | 3 | 1 |
| Simpson | Groves | £235.67 | 1 | 1 |
| Jones | Groves | £453 | 1 | 1 |
| Simpson | Greeves | £235.67 | 1 | 2 |
| Johnson | Greeves | £453.78 | 1 | 2 |
| Smith | Greeves | £82.78 | 2 | 2 |
| Jones | Greeves | £3,421 | 2 | 2 |
| Smith | Greeves | £235.67 | 3 | 2 |
| Simpson | Greeves | £235.67 | 1 | 2 |
| Jones | Greeves | £453 | 1 | 2 |
| Simpson | Smith | £235.67 | 1 | 3 |
| Johnson | Smith | £453.78 | 1 | 3 |
| Smith | Smith | £82.78 | 2 | 3 |
| Jones | Smith | £3,421 | 2 | 3 |
| Smith | Smith | £235.67 | 3 | 3 |
| Simpson | Smith | £235.67 | 1 | 3 |
| Jones | Smith | £453 | 1 | 3 |
| Simpson | Jones | £235.67 | 1 | 4 |
| Johnson | Jones | £453.78 | 1 | 4 |
| Smith | Jones | £82.78 | 2 | 4 |
| Jones | Jones | £3,421 | 2 | 4 |
| Smith | Jones | £235.67 | 3 | 4 |
| Simpson | Jones | £235.67 | 1 | 4 |
| Jones | Jones | £453 | 1 | 4 |

**PCW Contacts**

Mark Whitehorn welcomes readers' correspondence. He is at **m.whitehorn@dundee.ac.uk**