Oracle Developer

What are the advantages — and disadvantages — of a powerful mainframe relational database implemented on a PC? Mike Liardet gets to grips with the complex and expensive Oracle Developer.

Relational databases are currently one of the great growth areas in mini and mainframe computing. A number of relational database management systems (RDBMS for short) are available on PCs, and traditional micro database companies such as Ashton-Tate are frantically (and ill-temperedly, according to many accounts) implementing their own versions as well.

In this article we are going to look at one particular PC version of a RDBMS, Oracle Developer, but before going any further we should pose the question of why the reader should be interested in any RDBMS on a PC. Why should he give up his elegant and simple PC database system for all the complexity (and expense) of an RDBMS?

There are two ways of analysing this RDBMS/PC marriage. For a PC user, we should consider the possible benefits of an RDBMS; and for an RDBMS user, the benefits of a PC.

For the established PC user, the main benefit of switching to an RDBMS, the one that finally might prise him away from his favourite PC database product, is that an RDBMS offers solid multi-user capability via networked PCs (and other hardware). All data can be shared, with all staff having instant access to the latest information in the organisation. This can even benefit traditionally off-line activities like spreadsheeting, by guaranteeing that the various budgets and financial models are based on the most recent figures and not on some floppy disk copied several weeks previously.

Another benefit to the PC user is greater expansion capability. If an application were ultimately to outgrow the PC, it could instantly be transferred lock, stock and barrel, programs and data, onto a larger machine without a major rewrite. Also RDBMS are not based on *ad hoc* database techniques but all conform to well known standards (ANSI and IBM), and accordingly have a strong measure of compatibility with one another. Thus an organisation opting for a particular RDBMS on a PC is not really putting all its eggs in one basket, but has a number of future options to change either hardware or software relatively painlessly.

*'The SQL*Plus component of Oracle allows the user to query and change the database via a complex and powerful set of commands.'*

Looking at the marriage from the other viewpoint, PCs also have a role for established RDBMS users, arising from the PC's capability as an inexpensive calculating engine. An RDBMS can eat up mainframe power at an alarming rate, and some of its functionality (notably the 'forms' that handle user interaction) can readily be off-loaded onto PCs. And, of course, application development falls into this category too. If an application can run unchanged across all different types of hardware, then why tie up expensive mainframe resources during a project development? Instead, let the developers work on PCs, running their resource-hungry compilers and debuggers, and then only transfer the application to its ultimate destination towards the end of the project. This is precisely where Oracle Developer comes in.

Getting started

Oracle appears highly intimidating on first inspection. There is a plethora of manuals, and it comes on no fewer than 18 disks. It's a big system, with a lot in it!

Unfortunately, because of Oracle's size it will not run on an ordinary PC, but requires the power of an AT or compatible and at least 896k of extended memory in addition to the machine's 640k fitted as standard. Not surprisingly, with 18 disks of software a hard disk is mandatory, because Oracle will eat up 8Mbytes if all the facilities are installed — and that's before you start storing data.

Because it uses extended memory the 640k of ordinary non-extended memory is barely touched by Oracle, so any

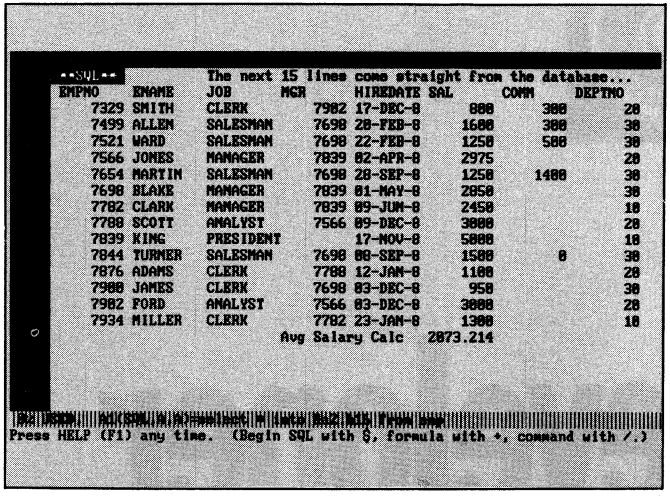

Fig 1 Oracle provides a spreadsheet-type facility for viewing tables of data

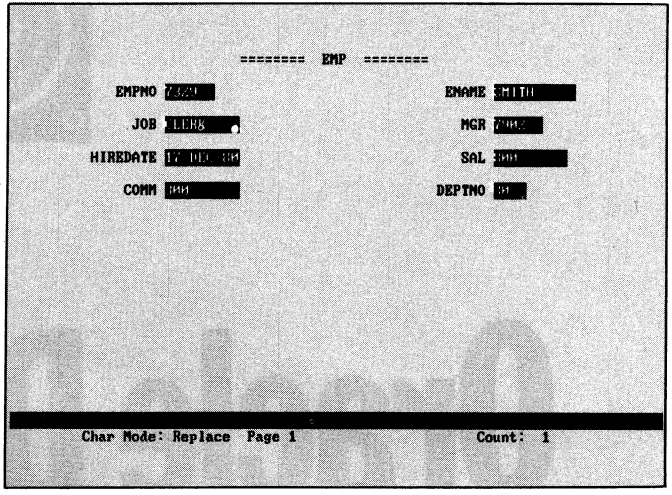

Fig 2 A simple screen-based form for entering or viewing information

other applications on the PC should still run as normal.

The machine used for this review was a '286 with a 20Mbyte disk and 1Mbyte of extended memory. Oracle was installed by simply running the installation program contained on one of the supplied disks. This program directs the whole installation operation, with simple, clearly worded instructions and explanations. It can even automatically adjust your system's configuration files should they not be correctly set up for Oracle's requirements. The whole process took about 30 minutes of tedious swapping in and out of disks, and set up demonstration tables in the database, help files and absolutely everything you subsequently need.

Before any Oracle work can proceed the kernel must first be loaded into extended memory. There is a special program to do this, which could easily be run from the AUTOEXEC.BAT file when

the system is first switched on. Thereafter, the Oracle facilities are run directly from DOS — there is no overall menu for the system.

*SQL*Plus*

The SQL*Plus component of Oracle allows the user to query and change the database via a complex and powerful set of commands. As the name implies, it is an implementation of SQL, but with additions to the ANSI standard, such as the ability to modify the structure of database tables after they have been created.

SQL stands for Structured Query Language, but do not be misled by this; it can handle more than just queries, and it is not really that highly structured. It is at least a language — a command language in fact — and at the heart of nearly all RDBMSs. I cannot go into great detail about SQL here, but I'll give the reader a sample of its flavour, with some observations on Oracle's implementation of it.

The key concept behind relational databases and SQL, the usual language used to manipulate them, is that the data is organised into 'tables', where each row in the table represents a 'record' of some item, and each column represents a property or feature (a 'field') applying to all these records.

On this page there are three tables with which we shall be working throughout this article. They represent part of a personnel database. To read the EMP table observe that each row represents an employee, and that the columns represent various employee attributes — for example, the JOB column of EMP specifies what jobs the employees do and so on. Notice that the

EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SALARY	COMM	DEPT NO
7329	SMITH	CLERK	7902	17-DEC-80	800.00	300.00	20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600.00	300.00	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250.00	500.00	30
7566	JONES	MANAGER	7839	02-APR-81	2975.00		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250.00	1400.00	30
7698	BLAKE	MANAGER	7830	01-MAY-81	2850.00		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450.00		10
7788	SCOTT	ANALYST	7566	09-DEC-82	3000.00		20
7839	KING	PRESIDENT		17-NOV-81	5000.00		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500.00	0.00	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100.00		20
7900	JAMES	CLERK	7698	03-DEC-81	950.00		30
7902	FORD	ANALYST	7566	03-DEC-81	3000.00		20
7934	MILLER	CLERK	7782	23-JAN-82	1300.00		10

DEPT

DEPT NO	DNAME	LOC
10	ACCOUNTING	MELBOURNE
20	RESEARCH	SYDNEY
30	SALES	BRISBANE
40	OPERATIONS	ADELAIDE

SALGRADE

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

different tables cross-reference one another, with more information on a particular employee's department becoming available if his or her department number is looked up in the DEPT table. Similarly, an employee's grade can be ascertained by checking his salary against LOSALs and HISALs in SALGRADE.

SQL*Plus enables commands to be either typed in and executed directly from the keyboard or run non-interactively from batch files. The listing opposite is of a fairly typical (brief) interactive session with SQL*Plus. The session starts from DOS command level — as I have said, the Oracle system has no overall menu control. Notice that the user has to log in to the system first, with a password.

Oracle maintains tight security on all aspects of the database, and the security system can restrict individual users to access only certain tables, or even just parts of tables, or to be denied the option to make changes and so on. Clearly, the security is essential in multi-user applications but of rather less importance for a single-user developer.

Following the log-in, the SQL> prompt is displayed to indicate that SQL*Plus is ready for user input. The commands typed alongside it were keyed in directly at the keyboard, and numbered within comments (/* Statement x */) so that they can be referred to here. The system's responses followed immediately after each command and generally disappeared off the top of the screen before I could read them — that is, until I discovered the SQL*Plus Set Pagesize and Set Pause commands.

Nine SQL commands were used, briefly summarised as follows:

1 Display the entire contents of the EMP table;

2 Display the entire contents of the SALGRADE table;

3 Display everything on employees earning between $800 and $1100;

4 Display the employee number and name, for grade three employees;

5 Change employee number 7329's department to 30;

6 Visually check that it really happened to precisely one employee.

7, 8 For the whole organisation display the management structure under the heading HIERARCHY (in a 'sawtooth' format, so we can see that Jones, Blake and Clark report to King; Allen and Ward report to Blake, and so on along with each employee's level from the top of the organisation, his number, manager and department.

9 Finish and return to DOS.

This simple session only scratches the

A typical Oracle session

```
SQL> select * from emp;

  EMPNO ENAME      JOB           MGR HIREDATE      SAL     COMM  DEPTNO
------- ---------- --------- ------- --------- ------- ------- -------
   7329 SMITH      CLERK        7902 17-DEC-80     800     300      20
   7499 ALLEN      SALESMAN     7698 20-FEB-81    1600     300      30
   7521 WARD       SALESMAN     7698 22-FEB-81    1250     500      30
   7566 JONES      MANAGER      7839 02-APR-81    2975              20
   7654 MARTIN     SALESMAN     7698 28-SEP-81    1250    1400      30
   7698 BLAKE      MANAGER      7839 01-MAY-81    2850              30
   7782 CLARK      MANAGER      7839 09-JUN-81    2450              10
   7788 SCOTT      ANALYST      7566 09-DEC-82    3000              20
   7839 KING       PRESIDENT         17-NOV-81    5000              10
   7844 TURNER     SALESMAN     7698 08-SEP-81    1500       0      30
   7876 ADAMS      CLERK        7788 12-JAN-83    1100              20
   7900 JAMES      CLERK        7698 03-DEC-81     950              30
   7902 FORD       ANALYST      7566 03-DEC-81    3000              20
   7934 MILLER     CLERK        7782 23-JAN-82    1300              10

14 records selected.

SQL> select * from salgrade;

  GRADE   LOSAL   HISAL
------- ------- -------
      1     700    1200
      2    1201    1400
      3    1401    2000
      4    2001    3000
      5    3001    9999

SQL> select * from emp where sal between 800 and 1100;

  EMPNO ENAME      JOB           MGR HIREDATE      SAL     COMM  DEPTNO
------- ---------- --------- ------- --------- ------- ------- -------
   7329 SMITH      CLERK        7902 17-DEC-80     800     300      20
   7876 ADAMS      CLERK        7788 12-JAN-83    1100              20
   7900 JAMES      CLERK        7698 03-DEC-81     950              30

SQL> select empno, ename from emp, salgrade
  2  where grade=3 and sal between losal and hisal;

  EMPNO ENAME
------- ----------
   7499 ALLEN
   7844 TURNER

SQL> update emp set deptno=30 where empno=7329;

1 record updated.

SQL> select * from emp where empno=7329;

  EMPNO ENAME      JOB           MGR HIREDATE      SAL     COMM  DEPTNO
------- ---------- --------- ------- --------- ------- ------- -------
   7329 SMITH      CLERK        7902 17-DEC-80     800     300      30

SQL> update emp set deptno=30 where empno=7329;

1 record updated.

SQL> column hierarchy format a21;

SQL> select lpad(' ',2*level) || ename hierarchy, level, empno, mgr, deptno
  2  from emp
  3  connect by prior empno = mgr
  4  start with ename = 'KING';


HIERARCHY               LEVEL   EMPNO     MGR  DEPTNO
--------------------- ------- ------- ------- -------
  KING                      1    7839              10
    JONES                   2    7566    7839      20
      SCOTT                 3    7788    7566      20
        ADAMS               4    7876    7788      20
      FORD                  3    7902    7566      20
        SMITH               4    7329    7902      30
    BLAKE                   2    7698    7839      30
      ALLEN                 3    7499    7698      30
      WARD                  3    7521    7698      30
      MARTIN                3    7654    7698      30
      TURNER                3    7844    7698      30
      JAMES                 3    7900    7698      30
    CLARK                   2    7782    7839      10
      MILLER                3    7934    7782      10

14 records selected.
```

surface of SQL*Plus. There are a number of other basic SQL commands to create and delete tables, add or delete fields from tables, and insert and delete records. The Select command is even more powerful and flexible than we have shown here. And there are a number of more exotic SQL*Plus facilities, typically restricted to a skilled 'database administrator' — commands to improve database performance, assign new users and monitor existing users.

There is also a Help facility in SQL*Plus. Simply type 'help' followed by the topic required, and you are treated to a screen version of a section from the SQL*Plus Reference Guide. As the Reference Guide is arranged alphabetically for ease of reference, and has a bigger page size than the PC, it is simpler to just use it directly and ignore Help altogether.

To an experienced PC user, the lack of menus, windows and function key facilities in SQL*Plus may seem a little barbaric, but be reassured: this is only Oracle revealing its mainframe origins. To be fair, we should say here that in a number of areas Oracle Corporation is now striving to attain the higher standards of user interaction pioneered by PC software. And one of its newer products, which may overcome some of these objections, is Easy*SQL. Easy*SQL is not included with the Developer, but will shortly be available as an option for the PC.

SQL*Calc

SQL*Calc is a fairly standard spreadsheet system, working much like, say, Lotus 1-2-3, but without the graphics, and without what Lotus refers to as 'database'. Of course there was no way Oracle Corporation would get involved with implementing this level of rather puny RAM-only 'database' facility!

As with all Oracle programs, SQL*Calc cannot be run without first logging into the database. This feels slightly odd in this case as it is quite feasible to use SQL*Calc as a standalone spreadsheet system without touching the database. However, once logged in all the normal spreadsheet interaction and facilities become available.

Formulae, text and numbers can be entered into cells on a 256 by 8192 worksheet. As with all spreadsheets you can't use the entire worksheet area on offer because PC memory is insufficient for this amount of data. If you try and use the bottom right-hand corner of the area (cell IV8192), recalculation becomes alarmingly slow, even for trivial two or three-cell models. It's best to keep to the top left.

The interesting part of SQL*Calc is its interface to the Oracle database, which allows data in the database to be read into the spreadsheet and subsequently written back. This is achieved by entering SQL statements into the cells, distinguished by a leading '$'. To get data into the worksheet a Select statement would be used, for example:

```
$select * into &a2 from emp
```

This statement reads all the data in the 'emp' table into the worksheet starting at cell A2. The column headings are included, and the table is loaded into the worksheet in a fairly obvious fashion, each worksheet row corresponding to a row in the table. Once loaded, calculations can be performed with the data in the normal way.

The screen dump in Fig 1 shows the 'emp' table loaded, and then the average of all the salaries calculated by:

```
+ avg(f3.f16)
```

Of course, the Select statement can be used in its full generality with search conditions, sorting and calculations, and so on. SQL*Calc also allows Update, Insert, Delete and Lock SQL commands to be placed in the spreadsheet. All SQL commands are distinguished from other entries by the preceding '$' and they are displayed as ****SQL****.

There is an 'Oracle' option on the main menu which contains all the other database-specific spreadsheet facilities. Its options are:

- **Table** A facility to create or recreate tables. The column names for the table and the data itself can be read straight out of the worksheet.
- **Rollback/Commit** Standard database management facilities to undo (rollback) or finalise (commit) any changes to the database during the session, or rather since the last 'commit'.
- **Logon** It is possible to log in to the database as an alternative user, without quitting and restarting the spreadsheet session.
- **Options** When a table is loaded (by a $select . . .) the system checks that there is room for it in the area specified, and warns the user if there is not. The Options option enables this checking to be switched off, or for excess rows to be ignored.
- **Show-info** Lists the database tables available to the particular user currently logged into the database.
- **Execute** The system does not re-do any of the SQL statements when it completes a spreadsheet recalculation, as this would not normally be required. Instead, these statements must be re-executed under separate control, from the Oracle/Execute option. Since it is not normally required to execute the whole lot together, each different type of statement can be re-executed independently.

SQL*Forms

SQL*Forms is the Oracle facility for designing and running screen layouts to interact with the end user. Without forms, data entry and modification to the database would have to be in SQL, using Select, Update, Insert and so on. This would just about be acceptable for a skilled programmer, but quite worthless for less expert users who will typically be running Oracle applications.

At its simplest, it is quite feasible to set up a form in less than two minutes. Such a form might enable a user to view, add, delete and modify records in a single table, but would not involve any further complexities. The first form, shown in Fig 1, was created in roughly this amount of time, and allows the user to work with the EMP table.

To create a form, the forms designer facility must first be used. When a form has been designed a 'generator' compiles the design into a state that can be used by the third program, the 'forms processor'.

The forms designer is menu-driven with a complex range of options, and its function keys take a little getting used to. This is mainly because the documentation describes a generalised version of the system, and it is necessary constantly to cross-reference a list of keyboard assignments which tell you that, for example, [Accept] in the documentation means function key F2 on the PC keyboard.

To set up a simple form is just a matter of running the designer, specifying a name for a form and a name for the table it will access, and asking for the default setup. This automatically designs the second form, shown in Fig 2. The prompts on the screen, and the areas where data will actually appear (the fields) are all put in place by the system making appropriate choices of defaults.

However, with the designer it is possible to be considerably more complicated than we have shown here.

- Validation can be specified for each field so that any data entered for a particular field can be checked for range, type and so on.
- Forms can be spread across several pages if the quantity of information is too great to fit on a single screen.
- In multi-table applications forms can be split up into 'blocks', where each block handles a different table.
- 'Triggers' provide a powerful processing capability, where following an 'event' (usually a user data entry), arbitrary calculations and other consequences can be invoked.
- Forms can also be multi-record, where a number of records can be tabulated beneath the column headings.

Normally the generator program can be invoked as an option within the designer program, but due to memory limitations this is not possible on the PC, and the designer program must be exited and the generator run independently. When this has been done the form can be used. (The usual Oracle security arrangements apply before any information in the database can be accessed.)

Conclusion

Oracle Developer has a mainframe 'feel' about it, in terms of size, scope and (on the minus side) user interaction. It contains an amazing breadth of facilities — too many to mention here — and a great deal of power once you can get to grips with it.

The Oracle software is really a family of products and tools, all of which relate to the central relational database in one way or another. Oracle Developer does not encompass the full range.

The price of Oracle Developer is pitched at a PC level, but it is really the squid to catch the marlin, and everything that follows is priced in a higher bracket. Don't forget that the Developer can only be used to develop applications, and that to actually run them involves purchasing Oracle Professional, which costs considerably more.

The above points indicate that a considerable commitment is being made when purchasing Oracle Developer. It is not just a matter of buying a piece of PC software, because ultimately a great deal more money will be spent if the system is to be used as intended: staff training, outside professional expertise, network hardware, maintenance agreements, and further Oracle software purchases are all likely to follow as a natural consequence of the initial purchase.

END

Oracle Developer costs $599; Oracle PC Professional, $2590; Lotus 1-2-3 add-in, $499; Maintenance $1150 per year. The software is available from Oracle Systems, tel (02) 959 5080.

'We feed the address into the computer and it shows us a map of the area together with the exact location of the fire.'