AGSO

Procedures to Access Point Spatial and Attribute Data in an Oracle Database from within the ARC/INFO GIS

by
Andrew Tucker

RECORD 1993/73

AUSTRALIAN GEOLOGICAL SURVEY ORGANISATION

AGSO
AUSTRALIAN GEOLOGICAL SURVEY ORGANISATION

Procedures to Access Point Spatial and Attribute Data in an Oracle Database from within the ARC/INFO GIS

RECORD 1993/73

by

Andrew Tucker

13 September 1993

Australian Geological Survey Organisation

Environment Geoscience and Groundwater Program

DEPARTMENT OF PRIMARY INDUSTRIES AND ENERGY

Minister for Resources: Hon. Michael Lee, MP
Secretary: Greg Taylor

AUSTRALIAN GEOLOGICAL SURVEY ORGANISATION

Executive Director: Harvey Jacka

ISSN: 1039-0073

ISBN: 0 642 19687 7

CONTENTS:

1) Introduction:

Borehole information and in particular waterbore information, is a major data source for hydrogeologists in the Groundwater component of the Environmental Geoscience and Groundwater (EGG) Program of the Australian Geological Survey Organisation (AGSO). Currently, several bore data bases have been developed using Oracle as the RDBMS: these databases are currently located either on an Intergraph workstation or on AGSO's corporate database server, but are accessible from networked personal computers and SUN workstations via terminal emulation software and/or SQL*Net.

The Oracle RDBMS offers many advantages (Zhou, 1992) over the PC database applications (Paradox and Knowledgeman) that were formerly used in the Groundwater section of AGSO:

. Oracle is available on a large number of platforms

. Oracle is widely used in the information technology industry, which suggests that applications developed using Oracle will have a relatively long lifetime

. Oracle offers cross-system communication - applications running on a variety of machines and operating systems can access data on an Oracle server across networks

The Environmental Systems Research Institute (ESRI) offers a database (INFO) as part of its ARC/INFO geological information system (GIS). The functionality of the INFO component of ARC/INFO is considerably less than that offered by the Oracle RDBMS products (ESRI, 1991), because ARC/INFO does not support:

. user specific "views" of the database

. equivalent functionality in comparison to the Oracle applications that support development of menus, forms and reports

. equivalent relational functionality in comparison to the Oracle RDBMS. For example, INFO is a tabular rather than a relational database, and doesn't support null values.

The points listed above present compelling reasons to implement attribute databases in the Oracle RDBMS, rather than with INFO or a PC database.

An ARC/INFO GIS, with attribute data stored in Oracle (or INFO) constitutes a hybrid GIS, with spatial and attribute data stored by separate mechanisms. Hybrid GIS's have evolved because of the need for high speed access to spatial data, and because the relational data model is inefficient for spatial data. However, point spatial data is something of an exception to other spatial data in that the point data will only consist of an x, a y and perhaps a z co-ordinate. The simplicity of the

spatial data requirements for point data mean that it is not unreasonable to maintain point spatial data with a relational DBMS.

In the Groundwater programs within AGSO, bore hole spatial data is maintained with the bore attribute data in an Oracle database, and is accessed and modified either by forms and menus developed with Oracle's application development software, or directly from the Oracle SQL command line.

This document describes procedures and programs to transfer point spatial data from the Oracle databases to ARC/INFO coverages, and procedures and programs that allow attribute data in an Oracle database to be accessed from within ARC/INFO. This document also describes methods for enhancing the speed of queries made using the ARC/Oracle hybrid GIS (Oracle version 3.0.11.1.2 and ARC/INFO version 6.1.1) . It is not the purpose of this document to needlessly duplicate the contents of ESRI's ARC/INFO manuals.

2) Transferring Point Spatial Data from Oracle Tables to ARC/INFO Coverages:

Many of the procedures discussed in this section are described at length in ESRI's (1991) "Managing Tabular Data" User's Guide.

Whenever bore location data is added or modified in the Oracle database, a new ARC/INFO bore location coverage must be generated, to include the new or modified spatial point data.

At least three pathways exist to transfer the bore locations from Oracle to ARC/INFO.

i) A text file containing unique bore numbers, and numeric x and y co-ordinates (e.g. latitudes and longitudes as decimal degrees) may be generated from within Oracle and transferred to an ARC/INFO workspace. The point coverage holding the locations of the bores may then be created with the ARC "generate" command. (Section 2.3.1)

ii) the bore locations may be placed in an ARC/INFO point coverage by accessing the Oracle database with DBMS cursors during an ARCEDIT session. (Section 2.3.2)

iii) the x and y co-ordinates (or latitude/longitude's) and unique point identifier integer values may be inserted into an ARC/INFO table with the ARC command "dbmsinfo", and the data in this file dumped to a text file. The point coverage holding the locations of the bores may then be created with the ARC "generate" command. (Section 2.3.3)

Note that ESRI (1991) refers to cursors used to access data held in a RDBMS table from within ARC/INFO as "DBMS cursors", though the term "RDBMS cursors" is often preferred.

2.1) An Example Oracle Table Containing Point Spatial Data:

In the remainder of this section, examples will be based on a specific bore location table maintained in Oracle, and accessed from ARC/INFO. This table is called TOPK, and is defined as follows:-

```
sql> create table TOPK (
        boreid          character(30)  not null,
        lat             number(9,6),
        lon             number(9,6),
        locacc          character(1),
        depth           number(6,2),
        unit            character(20),
        elev            number(6,2),
        source          char(6))
        tablespace tbspc
        storage (initial 100k next 10k);
```

The "lat" and "lon" fields hold the latitude and longitude of the bores (as numeric decimal degree values) respectively, the "boreid" field contains a character string identifying the bore, and the remainder of the fields hold other attribute data. The table is altered in the following section to assign a unique number to each bore. Note that in a production Oracle system, it may not be possible to change table or column definitions in this manner (Kucka, 1992).

2.2) Assigning Unique Numbers to Bores:

To relate attribute data (e.g. groundwater head) in an Oracle table to spatial data (for example, the locations of bores) in ARC/INFO, each point (or bore) in the coverage must have a unique identifier. This identifier must also be stored in the Oracle RDBMS to allow the spatial data in the GIS and the attribute data in Oracle to be linked. If all bore location data is to be maintained in Oracle, then it is reasonable (for this example) to add a field to the bore location table to store this identifier:-

```
sql> alter table TOPK add (uno          number(6,0)  unique);
```

It is possible to use data types other than integer types for the identifier, however integers can be used directly as the unique "user-id" field in the ARC/INFO point attribute data table (PAT) for the point coverage. The use of integer values can also speed attribute data retrieval from queries made from the GIS user interface. In a later section (3.7.1), the speed of queries using integer and character unique identifiers is compared.

It should be noted that if the unique integer identifier of a point in an Oracle table is transferred to the cover i.d. field in an ARC/INFO feature attribute table, the cover i.d. value will not change when a coverage is cleaned: it is not reasonable (or even possible) to apply the ARC/INFO "clean" command to point spatial data. Changes to coverage user i.d.'s in ARCEDIT can be controlled by the user, unlike the INFO record numbers.

Unique values can be inserted into the UNO field by creating an Oracle sequence, and updating the UNO numbers from the sequence. The UNO values should also be indexed:-

```
sql> create sequence topkunoseq;
sql> update topk set (uno) = (select nextval from topkunoseq) where topk.uno is null;
sql> create index topkuno on topk (uno) tablespace indxc storage (initial 10k next 2k);
```

Note that whenever new bore location records are added to the Oracle location table, the unique identifier for each new bore location record must be set.

2.3) Transferring Spatial Data From Oracle to ARC/INFO:

2.3.1) Transferring Spatial data from Oracle to ARC/INFO with Text files:

To generate a bore location text file (topk.lst) from the topk table, which can then be used as an input file for the ARC/INFO "generate" command, run the following Oracle commands (assuming that the latitudes of the bores are positive in a southwards direction):-

```
sql>set pagesize 0
sql>set verify off
sql>set feedback off
sql>set echo off
sql>spool topk.lst
sql>select uno||','||lon||','||(-1.*lat)  from topk;
sql>spool off
```

FTP or a similar file transfer utility may then be used to transfer the data to your ARC/INFO workspace.

To generate points for the bore locations, enter ARC/INFO, change to your workspace, and use the ARC/INFO generate command to load the point location data into an ARC/INFO coverage (called topk in this example):-

```
tucker@zircon> arc
arc: generate topk
generate: input topk.lst
generate: points
generate: quit
```

The point coverage should then be "built", and the unique identifier in the point attribute table (PAT) of the point coverage indexed, with the following arc commands:-

```
arc: build topk points
arc: indexitem topk.pat topk-id
```

Note that the topk-id item in the point coverage PAT corresponds to the UNO field in the Oracle table. The location data may then be projected with the ARC "project" command.

2.3.2) Using ARC/INFO DBMS Cursors to Access Point Spatial Data:

ARC/INFO DBMS cursors provide a means of accessing individual records (rows) and fields (columns) in a RDBMS table from within ARC/INFO. They also support one to many relationships between a single spatial item and several attribute data records in the RDBMS.

Conventional INFO cursors can also be used to access data held in RDBMS tables, however, the RDBMS table must be related to an existing INFO file or feature attribute table.

Unfortunately, it is difficult (though possible) to use DBMS cursors from the ARC/INFO command line. Therefore, an Arc Macro Language (AML) program called "boreloc.aml" that used DBMS cursors was developed to place points into an ARC/INFO point coverage from location data held in a remote Oracle table. The program accessed each point in the Oracle table through DBMS cursors. The AML ran within the ARCEDIT module, and allowed users to interactively select an Oracle table, as well as i.d., x and y co-ordinate fields.

The use of DBMS cursors allowed "boreloc" to detect when a bore location record had a null value for its unique i.d., or x or y co-ordinate fields - though these conditions can be enforced for data held within Oracle tables. Bore location records with a null i.d. or x or y co-ordinate fields were excluded from cursor processing by specifying that only records with non-null values for these fields be selected when declaring the DBMS cursor.

"boreloc" did not check for points that had duplicate user-ids - though again, it is possible to enforce unique record id's within an Oracle database.

"boreloc" performed adequately for small numbers of points (less than a few hundred, though this naturally depended on the hardware environment in which it was run), but had long execution times for large numbers of points. However, it did provide a straightforward means of accessing the spatial data held in the Oracle database. It was not necessary for "boreloc" to exploit the support offered by DBMS cursors for one to many relationships, but "boreloc" does demonstrate the use of DBMS cursors to access individual rows of a RDBMS table.

2.3.3) "bor2arc": an AML Program to Generate Bore Locations:

The "bor2arc" AML program transfers the data held in an Oracle table into an INFO table - removing the need for users to transfer the data themselves with ftp or a similar file transfer utility. The point i.d.'s, and x and y co-ordinate values are then stripped into a text file, and loaded into an ARC/INFO point coverage with the "generate" command.

Before running "bor2arc" you must:

. connect to the RDBMS with the ARC/INFO 'connect' command (section

3.1).

- have access to the RDBMS table holding the
  - i.d. of the point (which must be of type integer)
  - x co-ordinate of the point (of numeric type)
  - y co-ordinate of the point (of numeric type)
- create the ARC/INFO coverage which will hold the points. This coverage must not have been built for polygons
- be in arc
- have set the station type (with the ARC/INFO '&station' command)

"bor2arc" requests you to enter:

- the name of the coverage to hold the points
- the name of the RDBMS table holding the point spatial data
- the name of the RDBMS field holding the i.d. of the point (integer)
- the name of the RDBMS field holding the x co-ordinate of the point (numeric)
- the name of the RDBMS field holding the y co-ordinate of the point (numeric)
- if the y direction is positive southwards, select the checkbox

"bor2arc" will attempt to place points even when a point's unique numeric i.d., or x or y co-ordinate is null. In these cases, zero will be used instead of the RDBMS null value. It is possible, however, to specify constraints to prevent null values being entered in a field when creating an Oracle table. Alternatively, a view of the table containing the point location data may be created that excludes rows containing null values for i.d., or x or y values: "bor2arc" supports selection of data from Oracle views as well as tables.

"bor2arc" allows users who are relatively unfamiliar with Oracle, ftp, INFO or the ARC "generate" command to interactively transfer point spatial data to ARC/INFO and to place those point locations into a point coverage. User specifications of field and table names are checked to ensure that they exist in the Oracle database. The "bor2arc" program requires that the unique identifier for the bore be an integer value, and that the x and y co-ordinates be of a numeric type (integer or floating point).

The coverage containing the points should be "built" after the "bor2arc" program has been run.

Typical use of "bor2arc" is shown in Figure 1. A listing of the "boreloc" program is given in Appendix i).

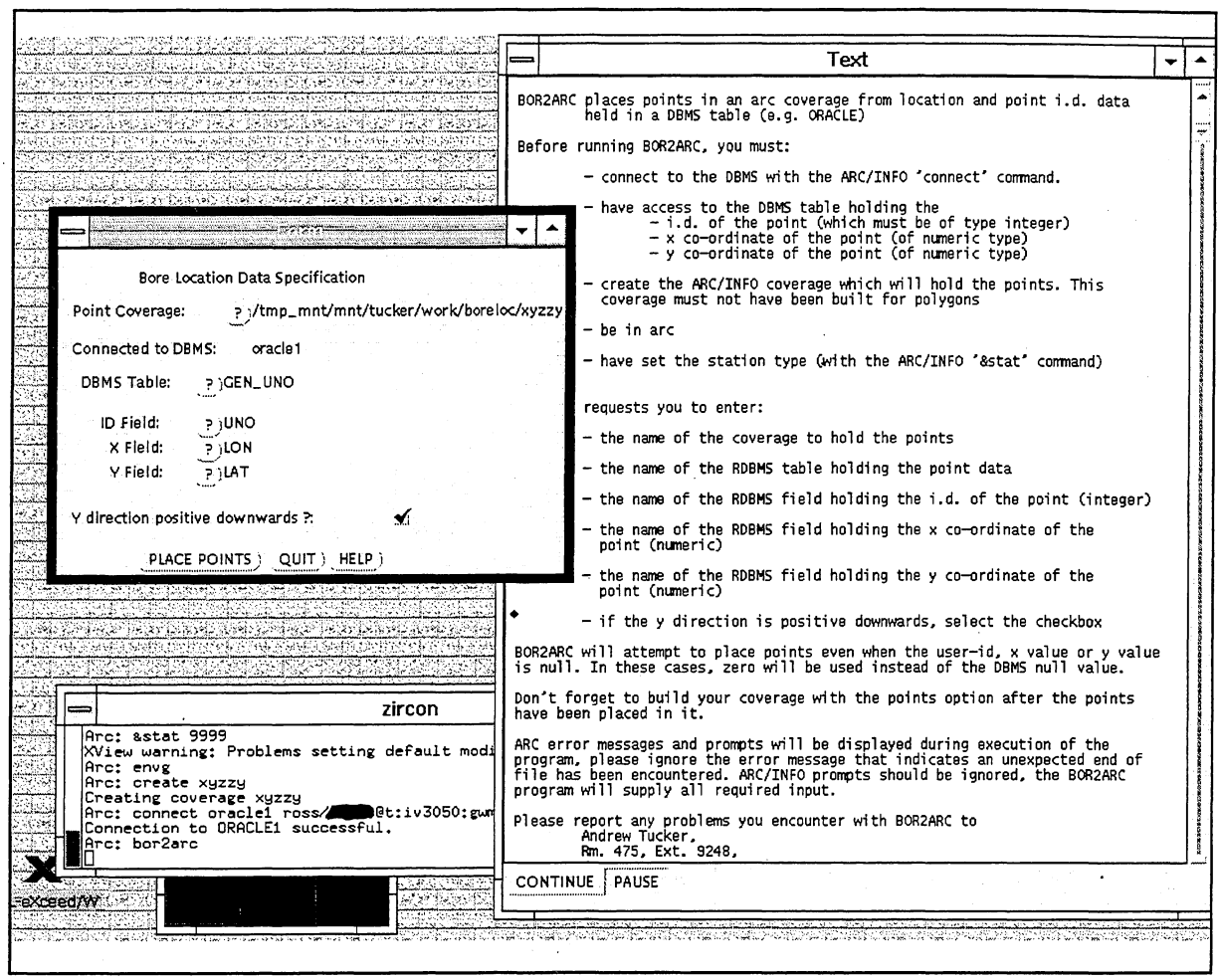

Figure 1: "bor2arc" can be run after you have defined the terminal type, created a coverage to hold the point spatial data, and you have connected to the relational database (the ARC commands to do this can be seen in the window with the "zircon" title).The data input form for bor2arc can be seen in the window titled "form". The help file for bor2arc is displayed in the popup window titled "text".

2.4) Discussion:

The method of transferring spatial data using text files (section 2.3.1) can prove inconvenient - it is necessary to become familiar with the ftp file transfer program and running SQL routines. It is also necessary to log on to two different computers to complete the transfer of the spatial data to ARC/INFO.

Transferring data with the "boreloc" AML program (section 2.3.2) has proved slow: the program takes some time to transfer the point data because:

- the co-ordinates of each point in the Oracle table must be accessed individually, and this information transferred to ARC/INFO each time a bore location is to be placed in a point coverage.
- the AML language is interpretive, and programs written in it can be slow for repetitive operations.

However, the "boreloc" program has proved useful when transferring small numbers of points, especially in situations where it is appropriate to display the point locations as they are being placed in an ARC/INFO coverage.

The "bor2arc" AML (section 2.3.3) has proved to be efficient, both in execution time and in the amount of effort required from users. The time required to transfer the borehole locations is at least an order of magnitude less than that required by "boreloc".

The examples presented in this section all assume that the points (or bores) can be uniquely identified with an integer value in both Oracle and INFO databases. It is certainly possible to use character strings to uniquely identify points in Oracle or INFO, but as will be shown in a later section, this can result in a considerable reduction in the efficiency of queries made on attribute data values.

3) Accessing Point Attribute Data in Oracle Tables from ARCPLOT

This section describes some of the ARC/INFO commands that can be used to access point data in an Oracle database. ESRI's database integrator for ARC allows users to execute SQL commands from within ARC, to link INFO tables and spatial data to Oracle tables, and to perform queries on attribute data with either ARC query syntax or the native query language of Oracle database. ESRI (1991) "Managing Tabular Data" ARC/INFO User's Guide should be referred to for a full discussion of the syntax and usage of commands described in this section.

3.1) Connecting to the Remote Oracle Database:

The ARC "connect" command is used to connect to the remote database from the ARC/INFO GIS. The appropriate software must be installed and configured before this command is issued. The parameters for the "connect" command are described in ESRI (1991). For an Oracle user named "horatio" with an Oracle password "squiggle" to access a borehole database (named "foodb") located on a workstation named "serial", the following connect statement would be executed:

arc: connect oracle horatio/squiggle@T:serial:foodb

This command must be run, either from the ARC command line, or from an AML, before any data on the remote Oracle database can be accessed.

3.2) Transferring Oracle Attribute Data to ARC/INFO Tables:

The ARC/INFO command DBMSINFO may be used to transfer the data held in Oracle tables or views into INFO tables. In many data development situations, this is probably the least satisfactory method of attribute data access as it makes it necessary to manage and update attribute data in two separate databases, and will also double media storage requirements.

3.3) Executing Native Oracle Commands:

The ARC/INFO command "dbmsexecute" may be used to run Oracle SQL queries on your Oracle database. For example, to list all data in the TOPK table, the following command could be executed:-

arc: dbmsexecute oracle select * from topk

Note that the SQL command is not terminated with a semi-colon.

The "dbmsexecute" command cannot be used to modify the attribute data in the Oracle database on the basis of spatial data held in the ARC point coverage.

However, it does support the SQL "rollback" and "commit" commands, making it possible to control transactions which have been made with other ARC database integrator commands.

3.4) Relating ARC/INFO Spatial Data to Oracle Attribute Data:

An ARC/INFO relate must be defined before the data access procedures described in the following two sections are used. A relate to link the point location data in the ARC/INFO coverage TOPK to the Oracle attribute data in table TOPK would be defined as:

```
arc: relate add
relation name: topk
table name: TOPK
database name: ORACLE
info item: topk-id
relate column: UNO
relate type: first
relate access: ro
relation name:
arc:
```

This relate specifies that the UNO field in the TOPK Oracle table will match the field called point coverage user i.d. field called TOPK-ID. The "first" relate type is the only relate type than can be used between ARC/INFO tables and RDBMS tables. If RDBMS data is not being modified from within ARC/INFO, it is preferable to specify that the relate access be read only ("ro") rather than read/write ("rw") as read/write access will lock all selected rows in the RDBMS table, preventing other users from updating them.

This example relate uses the coverage i.d. as the ARC/INFO relate item. If the point coverage is renamed, or copied, the relate will not be valid for the new point coverage, because the name of the ARC/INFO relate item will have changed: the user i.d. item name is based on the name of the coverage. If this is likely to be a problem, then one of at least two solutions can be applied:

- create new relates for the new point coverages

- add a new field to the original point attribute table, and transfer the values in the user i.d. field into it. This new field can be used as the ARC/INFO relate item. Whenever the coverage is copied or renamed, the ARC/INFO relate item name will be unchanged.

Relates may be saved for use in later ARC/INFO sessions with the "relate save" ARC/INFO command.

Limitations of ARC/INFO relates, both in the number of relates that can be used in a single query, and in situations where one to many or many to many relationships

exist between related tables are discussed at length in the ESRI (1991).

3.5) INFO Style Selects for Oracle Table Data:

In ARCPLOT and ARCEDIT, bores may be selected on the basis of attribute data in the Oracle tables by using standard ARCPLOT and ARCEDIT selection syntax (or "INFO style selects"). To select and display the locations of all bores sourced from an organisation identified by the string "GSQ", the following ARCPLOT commands would be executed:

```
arcplot: mapextent topk
arcplot: markersymbol 2
arcplot: reselect topk points topk//source = 'GSQ'
arcplot: points topk noids
```

ESRI recommend that the relate item in the RDBMS table be indexed for ARC/INFO queries.

This method of accessing and selecting data has proved very slow in some circumstances. The procedure described in the next section has proven to be much faster for most queries.

3.6) Native Mode SQL Selects for Oracle Table Data:

In ARCPLOT and ARCEDIT, bores may be selected on the basis of attribute data values held in the Oracle tables by using standard ARCPLOT or ARCEDIT selection syntax and native SQL select commands. To select and display all bores sourced from an organisation abbreviated as "GSQ", the following ARCPLOT commands would be executed:

```
arcplot: mapextent topk
arcplot: markersymbol 2
arcplot: reselect topk points ^topk where source = 'GSQ'
arcplot: points topk noids
```

The relate item in the PAT of the point coverage must be indexed. Not only are native mode SQL selects almost always faster than INFO style selects, they are also potentially more useful because Oracle query syntax can be used in the reselect command. However, there are circumstances where it is only possible to use an INFO style query, for example where some of data on which the query is based must be held in an INFO table.

3.7) Speed Comparisons of Native Mode and INFO Style Selects:

M. Kucka (personal communication, 1993) has emphasised that AGSO ARC/Oracle GIS users should be aware of the workload they impose on the Oracle server. The corporate Oracle database server is a shared resource, and GIS users should try to minimise the "stress" they place on it.

In the following two sections, results of timing tests are presented that suggest methods of reducing the load on the Oracle server. All the usual limitations apply to the results: speed will vary with [database / network / GIS] [configuration / loading / hardware]. The tests were run with the AGSO corporate Oracle database server (a Data General AViiON) and the ARC/INFO GIS running on a SUN 4/670 SPARCstation. The tests were run at times when all components of the hybrid GIS were relatively lightly loaded - this may introduce some bias into the results, but also means that the tests were carried out under similar conditions. The times given are, at best, only accurate to within a second.

3.7.1) Comparing Integer/Character Relate Items and Native Mode and INFO Style Select Speeds:

The speed of point selections made on the basis of attribute data values held in an Oracle table using both integer and character relate items was compared. The times of both ARC/INFO and native mode sql queries were examined. The times required to perform the same query on data held in an INFO table were also examined.

Table 3.7.1 compares the times taken to select points on the basis of attribute data for 55,500 points. The query selected points on the basis of a floating point value in the related attribute table being greater than zero. The select set at the beginning and end of the query consisted of all the points in the data set. A one to one relationship existed between attribute and spatial data. All fields used to establish the relate in both the point coverage PAT and the related Oracle (or INFO) table were indexed.

The unique character identifier for each point consisted of a string up to 30 characters long. Both ARC/INFO and native mode queries were used to select the data in the Oracle table. ARC/INFO only supports ARC/INFO style queries to INFO tables.

Note that the integer item type in an INFO table stores integers as ASCII values, the INFO binary item type stores integers as numeric values.

PAT Relate Item Type	Related Attribute Table Database	Related Attribute Table Relate Item Type	Query Type	Select Time (seconds)
Binary	INFO	Binary	INFO	135
Integer	INFO	Integer	INFO	244
Character	INFO	Character	INFO	234
Binary	Oracle	Integer	INFO	2100 (35 minutes)
Character	Oracle	Character	INFO	1980 (33 minutes)
Binary	Oracle	Integer	Native SQL	47
Character	Oracle	Character	Native SQL	147

Table 3.7.1: Select times for integer and character relate items.

As can be seen from table 3.7.1, using an integer column as a relate item in an Oracle table instead of a character item (and a binary relate item in the PAT) produced faster select times for the native mode query. Although there is little difference between the character and integer relate item select speed for the INFO query to an Oracle table, it is clear that it would not be optimal to use an INFO query as the select takes far longer than the native mode query. Use of binary items as relate items in the PAT can also reduce ARC/INFO PAT file size, and also the size of indexes in the INFO database.

When the test was repeated using a small data set of five hundred points (with a one to one correspondence between attribute and spatial data), there was no discernible difference between select times using either integer or character relate items. It has also been reported that some users feel more comfortable if the relate item in the INFO table is meaningful and recognisable (R. Gallagher, personal communication, 1993), so perhaps in some situations a character item should be used as a relate item.

The difference in selection times between INFO and native sql queries to Oracle tables is significant - the use of an INFO query to perform similar operations on data sets of the size used in this test could not be recommended.

The select times for queries made to the related INFO attribute table are greater

than those made to the same data held in an Oracle table. The select times for the INFO table data using integer relate items is similar to the select times for character relate items - this can be expected from the method INFO uses to store integer item values.

3.7.2) Comparing Native Mode and INFO Style Query Selection Speeds when Reselecting from a Small Initial Subset of Points:

The development of the results presented in this section was motivated by a communication received from ESRI Technical Support which suggested that timing tests are required in some circumstances to determine whether selects made using native mode or INFO style queries are fastest.

The test data set used was the same as that in the previous section - 55,500 point values with a one to one correspondence between ARC/INFO point and Oracle attribute data. The relate item was an integer field in the Oracle table, and a binary item in the INFO PAT - both fields were indexed.

Before each logical select was made, the point spatial data was reduced to a known subset of the entire set of points. The select set size after the query was always 10 points.

The results of the tests are presented in table 3.7.2.

The native mode select always took the same time to complete, no matter what the size of the initial select set. The time required for the INFO style select was approximately proportional to the number of points in the initial select set. ESRI Technical Support has explained that this is due to the different ways INFO and native mode selects scan data in the Oracle table.

When a native mode select queries data in the Oracle table, all records in the Oracle table are tested to see if they match the selection criteria. The relate field of each matching row in the Oracle table is then passed to ARC/INFO to determine which points in the coverage match the selected Oracle rows. Therefore, the time taken by the select to find the features which satisfy the criteria is independent of the initial size of the spatial set. As the number of matching rows in the Oracle table increases, the time taken for the select to complete will increase, as all the matches in the Oracle table must be transferred to, and processed by, ARC/INFO. This is why the relate item in the feature attribute table must be indexed, so that ARC/INFO can rapidly access the feature attribute table row for each of the rows returned by Oracle.

When an INFO style select is made on an Oracle table, a separate query is made to the Oracle table for each active row of the point attribute table, i.e. for each point in the current select set. Therefore the time taken by the INFO style query to select the points is approximately proportional to the size of the initial select set. Hence, if the Oracle table is indexed by the relate item, the time taken to perform selects with

INFO style queries will decrease.

Generally, wherever possible, native mode queries should be used to access data held in Oracle tables. They are almost always much faster and more powerful than INFO queries.

Initial Select Set Size as Number of Points	INFO Select Time (seconds)	Native Mode Select Time (seconds)
56	3	3
113	5	3
555	21	3
1110	40	3
1665	61	3
2220	82	3
2775	102	3
3330	123	3
3885	145	3
4440	163	3
4995	190	3
5550	212	3

Table 3.7.2: Comparison of the speed of native mode and info style selects for various sizes of initial select sets.

3.8) "arcrdbms": an AML Program to Support Querying Oracle Attribute Data From ARCPLOT:

The "arcrdbms" AML program allows users to display a form containing data for a single bore (or point) from an Oracle database. The user must display the bore locations in ARCPLOT, then run the "arcrdbms" program, specify which Oracle table contains the data to be displayed, and then select the bore for which the data is to be displayed. The data for any number of bores may be viewed sequentially. If a bore has multiple records, each of the records may also be viewed sequentially.

Before running "arcrdbms" , you must:

- connect to Oracle with the ARC/INFO 'connect' command.
- have produced a point coverage with each point containing a field that identifies the records in Oracle that hold attribute data for that point.
- have read access to the Oracle table holding the data for the points in the point coverage
- be in ARCPLOT, and have displayed the points in the point coverage
- have set the station type with the ARC/INFO '&stat' command

"arcrdbms" requests you to enter:

- the name of the coverage holding the point spatial data
- the name of the Oracle table holding the point attribute data
- the name of the field in the PAT of the point coverage that links the point to the Oracle table's records for that point.
- the name of the field in Oracle table that links the attribute data to the point in the PAT of the point coverage
- if you want to build a help file for the Oracle table that includes the comments in the 'USER_TAB_COMMENTS' and 'USER_COL_COMMENTS' Oracle data dictionary tables, select the button labelled "Build help File"

If more than one Oracle record is available for a point, the first matching record is displayed, but the remaining records can be viewed with the 'NEXT' button.

"arcrdbms" will build a form to allow you to view the data held in the RDBMS. If the RDBMS is Oracle version 6.0 or later, "arcrdbms" will attempt to build a help file from the descriptive comments for the Oracle table and fields held in the Oracle data dictionary tables USER_TAB_COMMENTS and USER_COL_COMMENTS. These comments can be viewed by pressing the 'HELP' button.

The form displaying the Oracle data can only display up to 70 characters for each field. The maximum number of fields displayed is dependent on the terminal type.

"arcrdbms" is independent of the structure of the Oracle database, its only requirement is that the Oracle table holds an integer field that can be used to link to an item in the point attribute table of the ARC/INFO coverage.

Typical use of the "arcrdbms" aml is displayed in figures 2 and 3.

"arcrdbms" uses ARC DBMS cursors to fetch the data for each bore (or point) selected by the user, and supports data display even when one to many relationships exist between spatial and attribute tables. If more than one record (or row) of attribute data exists for a point, each of the records may be displayed sequentially. If a relate was used to match the database rows to the spatial feature, only the first row that satisfied the query could be accessed and displayed.

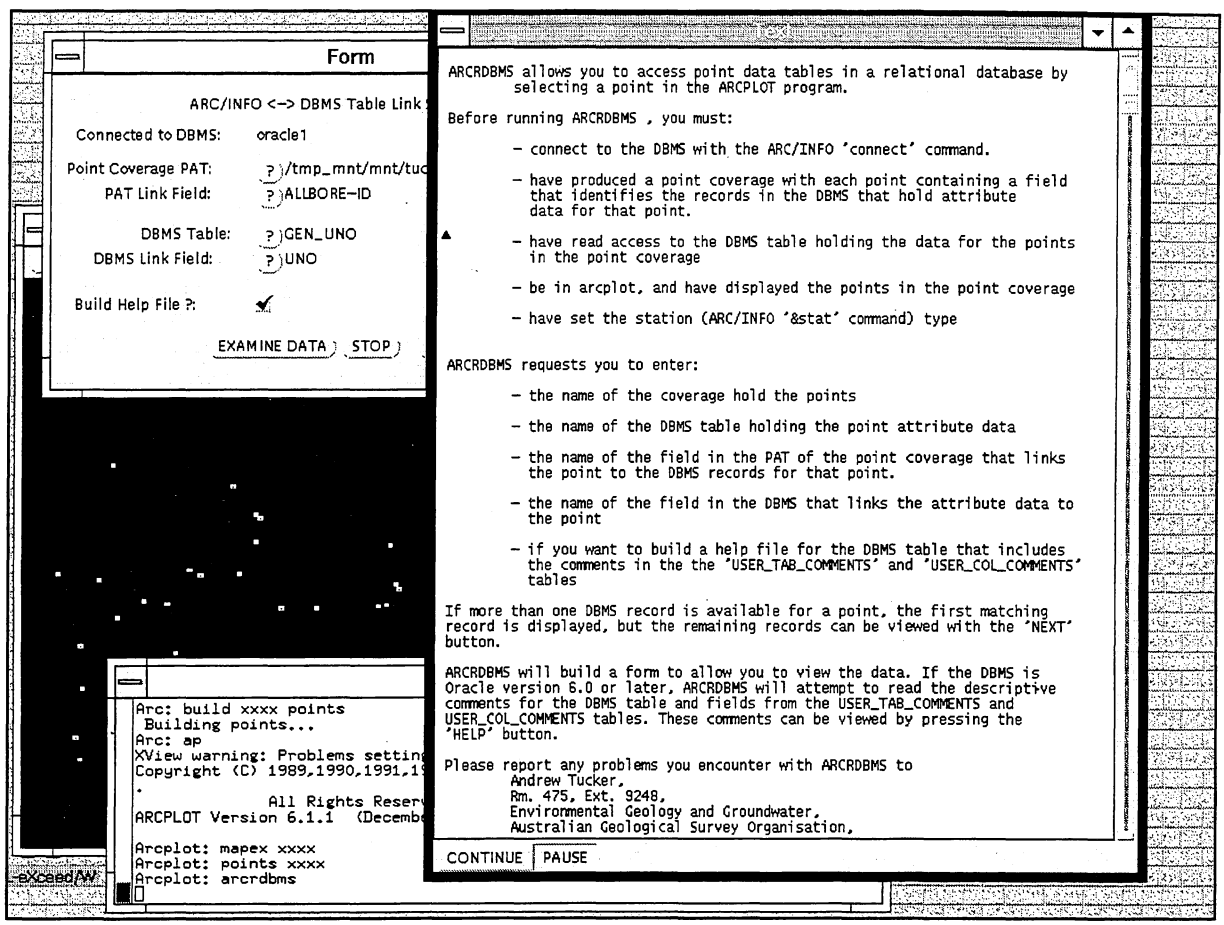

Figure 2: The commands used to display a point coverage in ARCPLOT can be seen in the lowermost window. The "arcrdbms" data input form for specifying the ARC/INFO point coverage and the RDBMS table parameters can be seen in the window titled "form". The "arcrdbms" help file is displayed in the window titled "text".

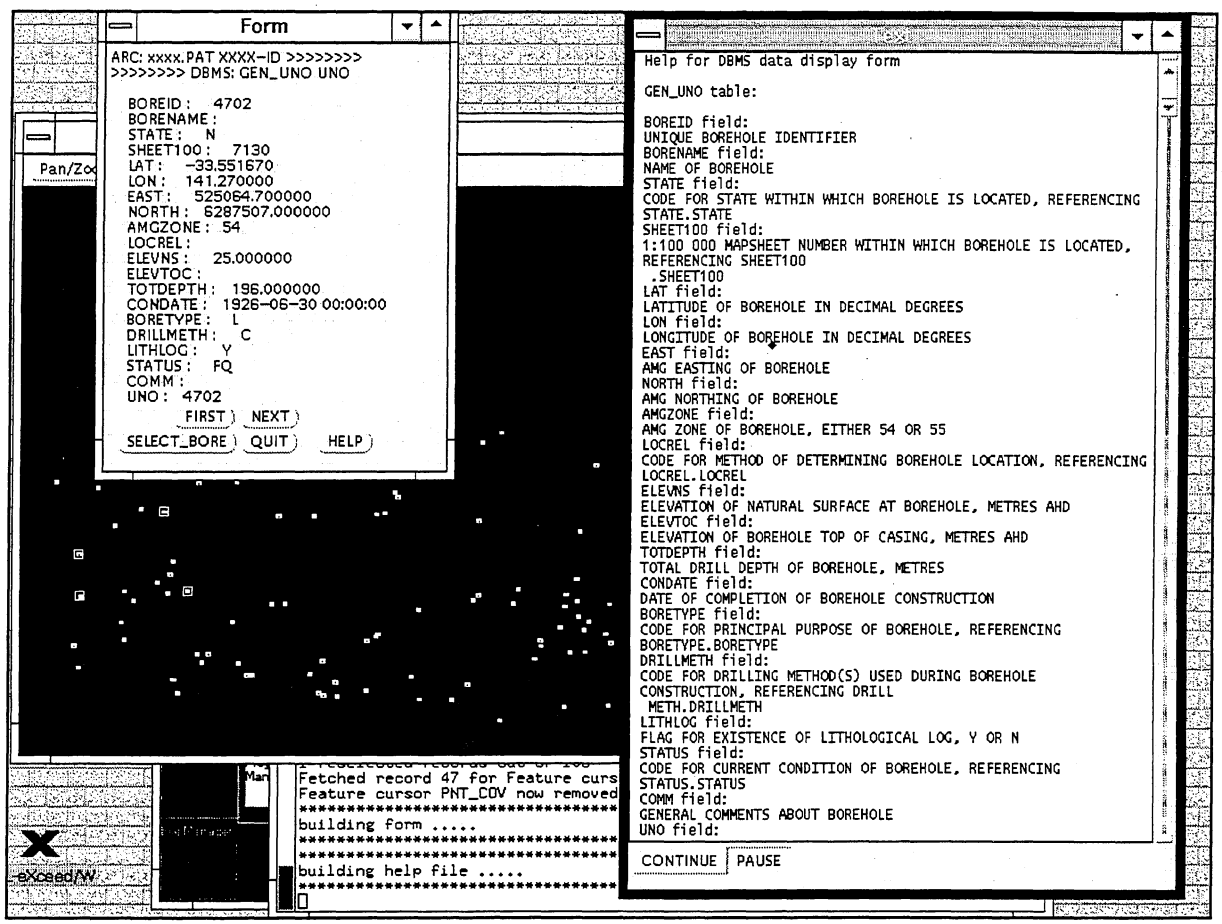

Figure 3: After a point has been selected, the "arcrdbms" program displays the attribute data for the point in the window titled "form". A help file containing the information in the data dictionary tables can also be displayed if the attribute data is held in an Oracle database.

4) Conclusion:

In this document I have described several methods of accessing data held in Oracle data files from within the ARC/INFO GIS. Some of these procedures can be applied to accessing line and polygon attribute data, but the spatial data for these feature types is generally not suitable for storage in a relational database.

The results of some relatively simple timing tests indicate that storing attribute data tables in an Oracle database can produce superior results to locating this data in INFO tables. Generally, native mode SQL queries offer greater speed and functionality when compared to INFO style queries.

It should be readily apparent from this document that establishing links between attribute and spatial data (required by the hybrid GIS data model used in ARC/INFO) is not a transparent process. It is certainly appropriate to consider the requirements of establishing and maintaining these links when designing and implementing the attribute and spatial components of a GIS database.

The AML programs described in this document can be obtained from:

Andrew Tucker,
Environmental Geoscience and Groundwater Program,
Australian Geological Survey Organisation,
GPO Box 378,
Canberra, ACT 2601,
Australia

Acknowledgments:

Ross Brodie, Keith Porritt, and Karen Ivković all examined and commented on an earlier version of this document. Ross and Karen also helped test the procedures and programs. The final version was reviewed by Mirek Kucka, Ross Brodie of AGSO, and Robyn Gallagher of GISolutions, who all provided useful comments. I discussed issues relevant to ARC/Oracle usage with Robyn, who also passed on to me information she received from ESRI Technical Support. Dr M.A. Habermehl edited the final version of the document.

References:

ESRI, 1991, *Managing Tabular Data: Design, Construction, and Management of Feature Attribute Databases*, ARC/INFO User's Guide 6.0, Environmental Systems Research Institute, Inc., Redlands, USA

M. Kucka, 1992, *Dual Oracle Environment and Change Control Management*, Australian Geological Survey Organisation, Record 1992/85

Zhou, Q., 1992, *Relational Database Techniques*, in course notes from "A Short Course In Geographical Information Systems: Advanced Geographical Information Systems", School of Geography, University of New South Wales, Sydney, Australia

Appendix i) The "bor2arc" AML Program:

The program consists of an ARC AML program, an ARC menu file, and a text help file. Listings of each of these are shown on the following pages.

The main routine in bor2arc:

- checks that the user is running the aml from ARC
- calls "init_variables" to initialise table and field variables from the values used in a previous run of "bor2arc". These values are held in ARC aml global variables. The values are only restored if "bor2arc" has been run from the current ARC/INFO session.
- calls "chk_connect" to ensure that the user has connected to a RDBMS
- the routine then loops, until the user selects quit with a menu button:
    - the variable "data_ok" is set true, if any of the subsequent checks on user data entry fail, this value is reset to false, and no further processing is undertaken with the current variable values.
    - a menu is displayed allowing the user to specify which RDBMS table and fields hold the spatial co-ordinates and unique bore identifier and also the name of the point coverage which is to hold the point locations.
    - if the user opts to quit, the field values are saved in global values, and the program returns the user to the ARC command line.
    - calls "chk_cover" to check that the point coverage has not been built for polygons - polygon and point spatial features are mutually exclusive within a single coverage
    - calls "chk_table" to ensure a valid table name has been entered
    - calls "chk_fields" to verify that all required fields have been specified by the user, and that these fields are of the correct type. The numeric fields must be of decimal, smallfloat, double, money, number, integer, smallint or tinyint types.
    - calls "place_points" which in turn calls routines to copy the RDBMS table to a temporary info table (routine "copy_dbms_info"), to generate an ascii file holding x and y co-ordinates and the unique numeric bore i.d. (routine "copy_info_ascii"), and to call the "generate" command to load the ascii data file into a point coverage (routine "copy_ascii_cover"). If the y co-ordinate values are to be multiplied by

-1 (i.e. the original y co-ordinates were positive southwards), the multiplication is carried out within the ARC/INFO tables module (routine "copy_info_ascii").

The menu file "bor2arc.men" supports specification of all data and program control parameters through buttons, check boxes or menus. Table, data field and point coverage names are all selected from menus - these menus are activated by buttons labelled with a question mark. This avoids users having to know that "control + mouse button 2" will display a list of menu options - this appears to be the usual ARC method of accessing menus of this nature.

Listing of "bor2arc.aml"

```
/* copyright Australian Geological Survey Organisation, 1992
/*****************************************************************************
/*
/* name: bor2arc.aml
/* by:   AGTucker
/* date: 28/9/92
/*
/* function: to make an ARC/INFO point coverage from x/y locations held
/*           in a RDBMS table
/*
/*****************************************************************************
/*
/* check that the user has gone into arc
&if %:program% <> 'ARC' &then &do
   &call usage
   &return
   &end
/*
/* initialise all variables if they need it
&call init_variables
/*
/* check that the user has connected to a DBMS, clear off if not
&s data_ok = .true.
&call chk_connect
&if not %data_ok% &then &return
/*
/* loop, until the user wants out
&do &while .true.
   /*
   /* set the data valid flag to true just for now
   &s data_ok = .true.
   /*
   /* display the form
   &menu bor2arc.men
   /*
   /* clear off if the user wants out, but save the table and fields for later use
   &if not %.ok2con% &then &do
      &s .table_name = %table_name%
      &s .id_field = %id_field%
      &s .x_field = %x_field%
      &s .y_field = %y_field%
      &s .cov_name = %cov_name%
      &s .invert_y = %invert_y%
      &return
      &end
   /*
   /* check the input coverage (returns data_ok for success/failure)
   &call chk_cover
   /*
   /* check the dbms table
   &if %data_ok% &then &call chk_table
   /*
   /* check the fields
   &if %data_ok% &then &call chk_fields
   /*
   /* place the points marking the locations of the fields
   &if %data_ok% &then &call place_points
   /*
   /* end of loop
   &end
/*
/* returns in body of procedure
/*
/*****************************************************************************
/*
&routine init_variables
/*
/* initialise all variables if they need it
/*
&if [variable .table_name] &then &s table_name = %.table_name%
&else &s table_name
```

```
&if [variable .id_field] &then &s id_field = %.id_field%
&else &s id_field
&if [variable .x_field] &then &s x_field = %.x_field%
&else &s x_field
&if [variable .y_field] &then &s y_field = %.y_field%
&else &s y_field
&if [variable .cov_name] &then &s cov_name = %.cov_name%
&else &s cov_name
&if [variable .invert_y] &then &s invert_y = %.invert_y%
&else &s invert_y = .false.
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine chk_connect
/*
/* check that the user has connected to a DBMS
/*
&s connect_str = [show connects]
&if [length %connect_str%] = 0 &then &do
   &s error_str = 'You must connect to a DBMS before running boreloc'
   &call error_msg
   &s data_ok = .false.
   &return
   &end
/*
/* if the user has only connected to one DBMS, use that one
&s connect_str = [translate %connect_str% ',' ' ']
&if [length [extract 2 %connect_str%]] = 0 &then &do
   &s conn_dbms = %connect_str%
   &end
&else &do
   &s conn_dbms = [getchoice %connect_str% -prompt 'select the DBMS:' -sort]
   &end
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine chk_cover
/*
/* check that the coverage specified by the user exists, and has not been
/* built  for polygons - data_ok set to false if not
/*
/* check that a cover name has been entered
&if [length %cov_name%] = 0 &then &do
   &s error_str = 'You must enter a coverage name !'
   &call error_msg
   &s data_ok = .false.
   &return
   &end
/*
/* check that the input coverage exists
/* !!!!!!!!- this check is not required if using generate !!!!!!!!
/*&if not [exists %cov_name% -cover] &then &do
/*   &s error_str = 'The cover does not exist !'
/*   &call error_msg
/*   &s data_ok = .false.
/*   &end
/*
/* check that the coverage is not a polygon coverage
&if [exists %cov_name%.pal -info] or [exists %cov_name%.pff -info] &then &do
   &s error_str = 'The cover has already been built for polygons !'
   &call error_msg
   &s data_ok = .false.
   &end
/*
/* clear off
&return
/*
/*****************************************************************************
```

```
/*
&routine chk_table
/*
/* check that the table exists - data_ok set to false if not
/*
/* check that a cover name has been entered
&if [length %table_name%] = 0 &then &do
   &s error_str = 'You must enter a table name !'
   &call error_msg
   &s data_ok = .false.
   &return
   &end
/*
/* check that the table exists
&s table_list = [show tables %conn_dbms%]
&s valid_table = .false.
&do one_table &list %table_list% &while not %valid_table%
   &if %one_table% = %table_name% &then &s valid_table = .true.
   &end
&if not %valid_table% &then &do
   &s error_str = 'The table does not exist !'
   &call error_msg
   &s data_ok = .false.
   &end
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine chk_fields
/*
/* check that the fields exist - data_ok set to false if not
/*
/* check that a id field has been entered
&if [length %id_field%] = 0 &then &do
   &s error_str = 'You must enter an id field name !'
   &call error_msg
   &s data_ok = .false.
   &return
   &end
/*
/* check that an x coordinate field name has been entered
&if [length %x_field%] = 0 &then &do
   &s error_str = 'You must enter an x co-ordinate field name !'
   &call error_msg
   &s data_ok = .false.
   &return
   &end
/*
/* check that an y coordinate field name has been entered
&if [length %y_field%] = 0 &then &do
   &s error_str = 'You must enter an y co-ordinate field name !'
   &call error_msg
   &s data_ok = .false.
   &return
   &end
/*
/* get a list of all fields in the table
&s field_list = [show columns %conn_dbms% %table_name%]
&s fields_ok = .true.
&s id_field_ok = .false.
&do one_field &list %field_list%
   &if %one_field% = %id_field% &then &s id_field_ok = .true.
   &end
&if not %id_field_ok% &then &do
   &s error_str = 'The id field does not exist !'
   &call error_msg
   &s fields_ok = .false.
   &end
&s x_field_ok = .false.
&do one_field &list %field_list%
   &if %one_field% = %x_field% &then &s x_field_ok = .true.
   &end
```

```
&if not %x_field_ok% &then &do
   &s error_str = 'The x co-ordinate field does not exist !'
   &call error_msg
   &s fields_ok = .false.
   &end
&s y_field_ok = .false.
&do one_field &list %field_list%
   &if %one_field% = %y_field% &then &s y_field_ok = .true.
   &end
&if not %y_field_ok% &then &do
   &s error_str = 'The y co-ordinate field does not exist !'
   &call error_msg
   &s fields_ok = .false.
   &end
&if %fields_ok% &then &do
   /*
   /* attempt to open a cursor into the table
   dbmscursor ext_tab declare %conn_dbms% ~
      select * from %table_name%
   /*
   /* check the type of the id field - must be integer
   &s id_field_ok = .false.
   &if [value :ext_tab.%id_field%.aml$type] = DECIMAL or ~
       [value :ext_tab.%id_field%.aml$type] = SMALLFLOAT or ~
       [value :ext_tab.%id_field%.aml$type] = DOUBLE or ~
       [value :ext_tab.%id_field%.aml$type] = MONEY or ~
       [value :ext_tab.%id_field%.aml$type] = NUMBER &then &do
      &if [value :ext_tab.%id_field%.aml$precision] = 0 &then &do
         &s id_field_ok = .true.
         &end
      &end
   &if [value :ext_tab.%id_field%.aml$type] = INTEGER or ~
       [value :ext_tab.%id_field%.aml$type] = SMALLINT or ~
       [value :ext_tab.%id_field%.aml$type] = TINYINT &then &do
      &s id_field_ok = .true.
      &end
   &if not %id_field_ok% &then &do
      &s error_str = 'The id field is of the wrong type !'
      &call error_msg
      &end
   /*
   /* check the type of the x field - must be real or integer
   &s x_field_ok = .false.
   &if [value :ext_tab.%x_field%.aml$type] = DECIMAL or ~
       [value :ext_tab.%x_field%.aml$type] = SMALLFLOAT or ~
       [value :ext_tab.%x_field%.aml$type] = DOUBLE or ~
       [value :ext_tab.%x_field%.aml$type] = MONEY or ~
       [value :ext_tab.%x_field%.aml$type] = NUMBER or ~
       [value :ext_tab.%x_field%.aml$type] = INTEGER or ~
       [value :ext_tab.%x_field%.aml$type] = SMALLINT or ~
       [value :ext_tab.%x_field%.aml$type] = TINYINT &then &do
      &s x_field_ok = .true.
      &end
   &if not %x_field_ok% &then &do
      &s error_str = 'The x field is of the wrong type !'
      &call error_msg
      &end
   /*
   /* check the type of the y field - must be real or integer
   &s y_field_ok = .false.
   &if [value :ext_tab.%x_field%.aml$type] = DECIMAL or ~
       [value :ext_tab.%x_field%.aml$type] = SMALLFLOAT or ~
       [value :ext_tab.%x_field%.aml$type] = DOUBLE or ~
       [value :ext_tab.%x_field%.aml$type] = MONEY or ~
       [value :ext_tab.%x_field%.aml$type] = NUMBER or ~
       [value :ext_tab.%x_field%.aml$type] = INTEGER or ~
       [value :ext_tab.%x_field%.aml$type] = SMALLINT or ~
       [value :ext_tab.%x_field%.aml$type] = TINYINT &then &do
      &s y_field_ok = .true.
      &end
   &if not %y_field_ok% &then &do
      &s error_str = 'The y field is of the wrong type !'
      &call error_msg
      &end
```

```
   /*
   /* remove the dbms cursor
   dbmscursor ext_tab remove
   &end
/*
/* update the value of the data valid flag
&if not %fields_ok% &then &do
   &s data_ok = .false.
   &end
&else &do
   &if not %id_field_ok% or not %x_field_ok% or not %y_field_ok% &then &do
      &s data_ok = .false.
      &end
   &end
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine place_points
/*
/* place the points in the arc/info coverage at the locations specified in
/* the dbms table, and with the ids also from the dbms table
/*
/* copy the dbms table into an info table
&call copy_dbms_info
/*
/* generate an ascii file containing the points
&call copy_info_ascii
/*
/* generate the point coverage holding the data
&call copy_ascii_cover
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine copy_dbms_info
/*
/* copy the dbms table into an info table
/*
/* build a temporary info file name
&s temp_count = 1
&do &while [exists temp%temp_count% -info]
   &s temp_count = %temp_count% + 1
   &end
&s temp_info = temp%temp_count%
/*
/* copy the data over
dbmsinfo %conn_dbms% %table_name% %temp_info% define
%id_field% %id_field% 8 8 I
%x_field% %x_field% 8 30 F 15
%y_field% %y_field% 8 30 F 15
END
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine copy_info_ascii
/*
/* generate an ascii file containing the points
/*
/* do the work in tables
tables
/*
/* build a temporary ascii file name
&s temp_count = 1
&do &while [exists temp%temp_count%.dat -file]
   &s temp_count = %temp_count% + 1
   &end
```

```
&s temp_ascii = temp%temp_count%.dat
/*
/* select the table
select %temp_info%
/*
/* if the user wants to invert the y co-ordinates (e.g. latitudes), do it now
&if %invert_y% &then &do
   calculate %y_field% = %y_field% * -1.0
   &end
/*
/* download the selected data fields
unload %temp_ascii% %id_field% %x_field% %y_field% delimited
/*
/* get out of tables
q stop
/*
/* delete the temporary info file
&s x = [delete %temp_info% -info]
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine copy_ascii_cover
/*
/* generate the point coverage holding the data
/*
/* generate the point coverage
&severity &error &ignore
generate %cov_name%
input %temp_ascii%
points
quit
&severity &error &fail
/*
/* delete the temporary ascii file
&s x = [delete %temp_ascii% -file]
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine usage
/*
/* display a usage message
&type
&type BOR2ARC: places bore locations in an ARC/INFO point coverage from
&type          co-ordinate data held in an ORACLE table
&type
&type You must be in ARC to run BOR2ARC
&type
&type You must connect to a DBMS before running BOR2ARC
&type
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine error_msg
/*
/* display an error message for the recalcitrant user
/*
/* does the error message form exist ?, if not then create it
&if not [exists errormsg.men -file] &then &do
   /*
   /* open the menu file
   &s error_unit = [open errormsg.men openstat -W]
   /*
   /* write the menu fields and buttons
   &s write_str = '7 - program generated error message form'
   &s write_stat = [write %error_unit% %write_str%]
```

```
   &s write_str = '/* program generated error message form '
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = ' '
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = ' ERROR MESSAGE:'
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = ' '
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = ' ERROR: %1'
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = ' '
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = ' %exit'
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = ' '
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = '%1 display error_str 40 '
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = '%exit button cancel ''EXIT'' &return'
   &s write_stat = [write %error_unit% %write_str%]
   &s close_stat = [close %error_unit%]
   &end
/*
/* display the form containing the error message
&menu errormsg.men
/*
/* clear off
&return
/*
/*****************************************************************************
/*
```

Listing of "bor2arc.men"

```
7 get the name of the oracle table with the bore locations, and the point cover
/*************************************************************************
/* name: bor2arc.men
/* by:   AGTucker
/* date: 29/9/92
/*
/* function: all parameters required to create points in an ARC/INFO
/*           coverage from data held in an oracle table
/*
/*************************************************************************

        Bore Location Data Specification

 Point Coverage: %1 %2

 Connected to DBMS: %conn

  DBMS Table: %3 %4

     ID Field: %5 %6
      X Field: %7 %8
      Y Field: %9 %10

 Y direction positive downwards ?: %11

        %place_points %quit %help

%conn display conn_dbms 15
%1 button cancel keep '?' ~
   &s cov_name = ~
      [getcover * -all 'Select name of point coverage:' -sort]
%2 display cov_name 40
%3 button cancel keep '?' ~
   &s table_name = [getchoice [show tables %conn_dbms%] ~
                   -prompt 'Select the location table:' -sort] ; ~
   &s id_field ; &s x_field ; &s y_field
%4 display table_name 25
%5 button cancel keep '?' ~
   &if [null %table_name%] &then &s msg = 'Specify a table name first' ; ~
   &else &s id_field = [getchoice [show columns %conn_dbms% %table_name%] ~
                    -prompt 'Select the bore i.d. field:' -sort]
%6 display id_field 25
%7 button cancel keep '?' ~
   &if [null %table_name%] &then &s msg = 'Specify a table name first' ; ~
   &else &s x_field = [getchoice [show columns %conn_dbms% %table_name%] ~
                    -prompt 'Select the x co-ordinate field:' -sort]
%8 display x_field 25
%9 button cancel keep '?' ~
   &if [null %table_name%] &then &s msg = 'Specify a table name first' ; ~
   &else &s y_field = [getchoice [show columns %conn_dbms% %table_name%] ~
                    -prompt 'Select the y co-ordinate field:' -sort]
%10 display y_field 25
%11 checkbox invert_y
%place_points button return 'PLACE POINTS' &return ; &s .ok2con = .true.
%quit button cancel 'QUIT' &return ; &s .ok2con = .false.
%help button help 'Display help window' return 'HELP' ~
   &popup bor2arc.hlp
%formopt setvariables immediate messagevariable msg
%forminit &s x ; &if not [variable table_name] &then &s table_name
```

Listing of "bor2arc.hlp" text file:

```
BOR2ARC places points in an arc coverage from location and point i.d. data
        held in a DBMS table (e.g. ORACLE)

Before running BOR2ARC, you must:

        - connect to the DBMS with the ARC/INFO 'connect' command.

        - have access to the DBMS table holding the
                - i.d. of the point (which must be of type integer)
                - x co-ordinate of the point (of numeric type)
                - y co-ordinate of the point (of numeric type)

        - create the ARC/INFO coverage which will hold the points. This
          coverage must not have been built for polygons

        - be in arc

        - have set the station type (with the ARC/INFO '&stat' command)


BOR2ARC requests you to enter:

        - the name of the coverage to hold the points

        - the name of the RDBMS table holding the point data

        - the name of the RDBMS field holding the i.d. of the point (integer)

        - the name of the RDBMS field holding the x co-ordinate of the
          point (numeric)

        - the name of the RDBMS field holding the y co-ordinate of the
          point (numeric)

        - if the y direction is positive downwards, select the checkbox

BOR2ARC will attempt to place points even when the user-id, x value or y value
is null. In these cases, zero will be used instead of the DBMS null value.

Don't forget to build your coverage with the points option after the points
have been placed in it.


Please report any problems you encounter with BOR2ARC to
        Andrew Tucker,
        Rm. 475, Ext. 9248,
        Environmental Geology and Groundwater,
        Australian Geological Survey Organisation,
        GPO Box 378,
        Canberra, ACT 2601
        Australia
```

Appendix ii) The "arcrdbms" AML Program:

Listings of the aml program, menu file and help text file are shown on the following pages. The program creates a menu called "dbmsform.men" (in the current workspace) that is used to display the data values in the Oracle table. A help file is also created in the current workspace to hold the field and table comments for the attribute data table - this file is called "dbmsform.hlp".

The main routine in "arcrdbms" :

- checks that the user has called the program from within ARCPLOT
- calls "chk_connect" to ensure that the user has connected to a RDBMS
- calls "init_variables" to initialise table and field variables from the values used in a previous run of "bor2arc". These values are held in ARC aml global variables. The values are only restored if "bor2arc" has been run from the current ARC/INFO session.
- the routine then loops, until the user selects quit with a menu button:
    - the variable "data_ok" is set true, if any of the subsequent checks on user data entry fail, this value is reset to false, and no further processing is undertaken with the current variable values.
    - a menu is displayed allowing the user to specify which RDBMS table is to be accessed, the point coverage holding the point location data, and the fields in the table and PAT which are to be used to link the attribute and spatial data.
    - if the user opts to quit, the form field values are saved in global variables, and the program returns the user to the ARC command line.
    - calls "chk_cover" to check that the point coverage has not been built for polygons - polygon and point spatial features are mutually exclusive within a single coverage
    - calls "chk_pat" to check that a point coverage attribute table (PAT) exists for the user specified coverage.
    - calls "chk_pat_link" to check that the relate item exists in the user specified PAT
    - calls "chk_dbms" to verify that the RDBMS table exists
    - calls "chk_dbms_link" to ensure that the user specified relate item

exists in the RDBMS table

. calls "display_dbms_form" to perform all other processing.

The routine "display_dbms_form" performs the following tasks:

. calls "select_point" to allow the user to select a point

. calls "declare_dbms_cursor" to open a DBMS cursor into the DBMS table rows holding the data for the point

. calls "init_form_variables" to define and initialise AML variables holding the names and widths of all the fields in the Oracle attribute table

. calls "build_dbms_form" to create an ARC/INFO menu form called "dbmsform.men" to display the data in the Oracle attribute table

. calls "build_dbms_help" to create a help file describing the function of each of the buttons on the "dbmsform.men" menu. If the user has requested that RDBMS field descriptions be added to the help file, the field and table comments are added from the Oracle data dictionary tables (for Oracle version 6.0 and later versions only)

. the routine then loops, displaying the attribute data in the RDBMS table and allowing the user to select another point. When the user selects quit, the DBMS cursor is removed, and control returns to the main "arcrdbms" routine.

The menu file "arcrdbms.men" supports user specification the point coverage PAT, the field in the PAT which links the point to the attribute data table, the attribute data table, and the field in the attribute data table that links to the point attribute row. Buttons allowing the user to select help on the program, to quit the program and to examine the data in the attribute table are also displayed.

Listing of "arcrdbms.aml"

```
/* copyright Australian Geological Survey Organisation, 1992
/*****************************************************************************
/*
/* name: arcrdbms.aml
/* by:   AGTucker
/* date: 28/9/92
/*
/* function: to link an ARC/INFO point coverage to a DBMS table, and to
/*           allow the user to view the DBMS table data
/*
/* typical usage:
/*           user enters arcplot, and displays the points in a point coverage
/*           user issues the ARC 'connect' command to the remote database
/*           user sets the display type with the '&stat' command
/*           user starts arcrdbms by typing :
/*                  &r arcrdbms
/*           user defines the name of the ARC/INFO point coverage, the INFO
/*              link field, the ORACLE table, and the oracle link field
/*           the program then builds a form to hold data in the oracle table
/*           the user then selects a point and the program displays the
/*              the data for that point from the ORACLE table
/*
/*****************************************************************************
/*
/* check that the user has gone into arcplot
&if %:program% <> 'ARCPLOT' &then &do
   &call usage
   &return
   &end
/*
/* initialise all variables if they need it
&call init_variables
/*
/* check that the user has connected to a DBMS, clear off if not
&s data_ok = .true.
&call chk_connect
&if not %data_ok% &then &return
/*
/* loop, until the user wants out
&do &while .true.
   /*
   /* set the data valid flag to true just for now
   &s data_ok = .true.
   /*
   /* display the form
   &menu arcrdbms.men
   /*
   /* clear off if the user wants out, but save variables are globals
   /* for next time
   &if not %.ok2con% &then &do
      &s .cov_name = %cov_name%
      &s .pat_name = %pat_name%
      &s .pat_link = %pat_link%
      &s .dbms_name = %dbms_name%
      &s .dbms_link = %dbms_link%
      &s .build_help = %build_help%
      &return
      &end
   /*
   /* check the ARC/INFO pat info file, and get the name of the ARC coverage
   &call chk_pat
   /*
   /* check the link field in the pat coverage
   &if %data_ok% &then &call chk_pat_link
   /*
   /* check the DBMS table
   &if %data_ok% &then &call chk_dbms
   /*
   /* check the DBMS link field
   &if %data_ok% &then &call chk_dbms_link
   /*
```

```
   /* display the form and data for the user's edification
   &if %data_ok% &then &call display_dbms_form
   /*
   /* loop ends
   &end
/*
/* returns in body of procedure
/*
/*****************************************************************************
/*
&routine init_variables
/*
/* initialise all variables if they need it
/*
&if [variable .cov_name] &then &s cov_name = %.cov_name%
&else &s cov_name
&if [variable .pat_name] &then &s pat_name = %.pat_name%
&else &s pat_name
&if [variable .pat_link] &then &s pat_link = %.pat_link%
&else &s pat_link
&if [variable .dbms_name] &then &s dbms_name = %.dbms_name%
&else &s dbms_name
&if [variable .dbms_link] &then &s dbms_link = %.dbms_link%
&else &s dbms_link
&if [variable .build_help] &then &s build_help = %.build_help%
&else &s build_help = .false.
/*
/* if not help path variable has been defined, define one to a null state
&if not [variable .help_path ] &then &s .help_path
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine chk_connect
/*
/* check that the user has connected to a DBMS
/*
&s connect_str = [show connects]
&if [length %connect_str%] = 0 &then &do
   &s error_str = 'You must connect to a DBMS before running boreloc'
   &call error_msg
   &s data_ok = .false.
   &return
   &end
/*
/* if the user has only connected to one DBMS, use that one
&s connect_str = [translate %connect_str% ',' ' ']
&if [length [extract 2 %connect_str%]] = 0 &then &do
   &s conn_dbms = %connect_str%
   &end
&else &do
   &s conn_dbms = [getchoice %connect_str% -prompt 'select the DBMS:' -sort]
   &end
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine chk_pat
/*
/* check that the pat specified by the user exists
/*
/* check that a pat name has been entered
&if [length %pat_name%] = 0 &then &do
   &s error_str = 'You must enter a point coverage PAT file name !'
   &call error_msg
   &s data_ok = .false.
   &return
   &end
/*
/* check that the PAT exists
```

```
&if not [exists %pat_name% -info] &then &do
   &s error_str = 'The point coverage PAT does not exist !'
   &call error_msg
   &s data_ok = .false.
   &return
   &end
/*
/* calculate the name of the point coverage
&s cov_name = [before [after %pat_name% [unquote 'arc!']] .pat]
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine chk_pat_link
/*
/* check that the link field in the pat is ok
/*
/* check that an PAT link field name has been entered
&if [length %pat_link%] = 0 &then &do
   &s error_str = 'You must enter an PAT link field name !'
   &call error_msg
   &s data_ok = .false.
   &return
   &end
/*
/* does the PAT link field exist in the info table ?
&s field_list = [show columns info %pat_name%]
&s pat_field_ok = .false.
&do one_field &list %field_list%
   &if %one_field% = %pat_link% &then &s pat_field_ok = .true.
   &end
&if not %pat_field_ok% &then &do
   &s error_str = 'The PAT link field does not exist !'
   &call error_msg
   &s data_ok = .false.
   &end
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine chk_dbms
/*
/* check that the table exists - data_ok set to false if not
/*
/* check that a cover name has been entered
&if [length %dbms_name%] = 0 &then &do
   &s error_str = 'You must enter a DBMS table name !'
   &call error_msg
   &s data_ok = .false.
   &return
   &end
/*
/* check that the DBMS table exists
&s table_list = [show tables %conn_dbms%]
&s valid_table = .false.
&do one_table &list %table_list% &while not %valid_table%
   &if %one_table% = %dbms_name% &then &s valid_table = .true.
   &end
&if not %valid_table% &then &do
   &s error_str = 'The DBMS table does not exist !'
   &call error_msg
   &s data_ok = .false.
   &end
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine chk_dbms_link
```

```
/*
/* check the DBMS link field
/*
/* check that a dbms link field has been entered
&if [length %dbms_link%] = 0 &then &do
   &s error_str = 'You must enter an dbms link field !'
   &call error_msg
   &s data_ok = .false.
   &return
   &end
/*
/* get a list of all fields in the table
&s field_list = [show columns %conn_dbms% %dbms_name%]
&s dbms_link_ok = .false.
&do one_field &list %field_list% &while not %dbms_link_ok%
   &if %one_field% = %dbms_link% &then &s dbms_link_ok = .true.
   &end
&if not %dbms_link_ok% &then &do
   &s error_str = 'The dbms link field does not exist !'
   &call error_msg
   &s data_ok = .false.
   &end
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine display_dbms_form
/*
/* display a form holding data from the DBMS table
/*
/* get the user to select the point
&s got_point = .true.
&call select_point
&if not %got_point% &then &return
/*
/* declare the cursor for the DBMS table and the info point
&call declare_dbms_cursor
/*
/* initialise the form variables
&call init_form_variables
/*
/* build the form to display the values
&call build_dbms_form
/*
/* build the a help file for the dbms form
&call build_dbms_help
/*
/* loop, display the values for each selected bore until the user quits
&do &while .true.
   /*
   /* display the form
   &menu dbmsform.men
   /*
   /* allow the user to select a new point, and display the information for it
   &if %select_bore% &then &do
      /*
      /* remove the old cursor
      dbmscursor ot1 remove
      /*
      /* get the user to select a point
      &call select_point
      /*
      /* declare the cursor for the DBMS table and the info point
      &call declare_dbms_cursor
      &end
   &else &do
      /*
      /* remove the dbms cursor
      dbmscursor ot1 remove
      /*
      /* clear off
      &return
```

```
      &end
   &end
/*
/* returns in body of procedure
/*
/*****************************************************************************
/*
&routine select_point
/*
/* get the user to select a point
/*
   /*
   /* display the form allowing the user to select a point
/*   &menu getpoint.men
   /*
   /* stop if the user wants to quit
/*   &if not %.ok2con% &then &stop
/*
/* select all points in the coverage
&s msg = Select a point
clearselect %cov_name% point
/*
/* declare a cursor for the point coverage
cursor pnt_cov declare %cov_name% point ro
/*
/* let the user specify a point
reselect %cov_name% point one *
/*
/* open a cursor for the point coverage
cursor pnt_cov open
/*
/* check that the user has only selected one bore
&if %:pnt_cov.aml$nsel% <> 1 &then &do
   &s got_point = .false.
   cursor pnt_cov remove
   &s error_str = 'You did not select a point !'
   &call error_msg
   &end
&else &do
   /*
   /* get the value of the link field
   &s info_link_val = [value :pnt_cov.%pat_link% ]
   /*
   /* close the info cursor
   cursor pnt_cov remove
   &end
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine declare_dbms_cursor
/*
/* declare the cursor for the DBMS table
/*
/* declare the cursor, and only select records with a matching link value
&if [null %info_link_val%] &then &do
   dbmscursor ot1 declare %conn_dbms% ~
   select * from %dbms_name% ~
      where %dbms_link% is null
   &end
&else &do
   dbmscursor ot1 declare %conn_dbms% ~
   select * from %dbms_name% ~
      where %dbms_link% = %info_link_val%
   &end
/*
/* open the cursor
dbmscursor ot1 open
/*
/* clear off
&return
/*
```

```
/*************************************************************************
/*
&routine init_form_variables
/*
/* initialise the form variables
/*
/* declare/initialise the display fields
&do cnt := 1 &to %:ot1.aml$ncol%
   /*
   /* determine the name of the field
   &s dispnam%cnt% = [value :ot1.%cnt%.aml$name ]
   /*
   /* determine the length of the display field (to a maximum of 70, and
   /* minimum of 20)
   &s displen%cnt% = [max [min [value :ot1.%cnt%.aml$length ] 70] 20]
   &end
/*
/* clear off
&return
/*
/*************************************************************************
/*
&routine build_dbms_form
/*
/* build the form to display the values
/*
/* let the user know whats is going on
&type *************************************************************************
&type building form .....
&type *************************************************************************
/*
/* open the file to hold the form
&s dbms_unit = [open dbmsform.men openstat -w]
/*
/* write the form header to the menu file
&s write_stat = [write %dbms_unit% [quote 7 - program generated form for DBMS table data display ]]
/*
/* write the form title to the menu file
&s write_str = ~
[unquote 'ARC: '] %cov_name%.PAT %pat_link% [unquote '>>>>>>>>']
&s write_stat = [write %dbms_unit% [quote %write_str%]]
&s write_str = ~
[unquote '>>>>>>>> DBMS:'] %dbms_name% %dbms_link%
&s write_stat = [write %dbms_unit% [quote %write_str%]]
&s write_str
&s write_stat = [write %dbms_unit% [quote %write_str%]]
/*
/* define the position of the display fields
&do cnt := 1 &to %:ot1.aml$ncol%
   &s write_str = [quote [unquote ' '] [value dispnam%cnt%] : [unquote '%']%cnt% ]
   &s write_stat = [write %dbms_unit% %write_str%]
   &end
/*
/* define the position of the form buttons
&s write_stat = [write %dbms_unit% '       %first %next ']
&s write_stat = [write %dbms_unit% ' %select_bore %cancel %help ']
/*
/* define the definitions of each of the display fields
&do cnt := 1 &to %:ot1.aml$ncol%
   &s write_str = [quote [unquote '%']%cnt% display :ot1.%cnt% [value displen%cnt%]]
   &s writestat = ~
      [write %dbms_unit% %write_str%]
   &end
/*
/* define each of the buttons
/*
/* next DBMS record button
&s write_str = ~
   '%next button return ''NEXT'' dbmscursor ot1 next ; &if not %:ot1.aml$next% &then &s msg = no more
records'
&s write_stat = [write %dbms_unit% %write_str% ]
/*
/* first DBMS record button
&s write_str = ~
```

```
   '%first button return ''FIRST'' dbmscursor ot1 first ; &if not %:ot1.aml$next% &then &s msg = no more
records'
&s write_stat = [write %dbms_unit% %write_str% ]
/*
/* select another bore button
&s write_str = ~
   '%select_bore button return ''SELECT_BORE'' &return ; &s select_bore = .true.'
&s write_stat = [write %dbms_unit% %write_str% ]
/*
/* QUIT button
&s write_str = ~
   '%cancel button cancel ''QUIT'' &return ; &s select_bore = .false.'
&s write_stat = [write %dbms_unit% %write_str% ]
/*
/* HELP button
&s write_str = ~
   '%help button cancel keep ''HELP'' &popup dbmsform.hlp '
&s write_stat = [write %dbms_unit% %write_str% ]
/*
/* formoptions
&s write_str = ~
   '%forminit &s x ; &if not %:ot1.aml$next% &then &s msg = No records found'
&s write_stat = [write %dbms_unit% %write_str% ]
&s write_str = ~
   '%formopt setvariable immediate messagevariable msg'
&s write_stat = [write %dbms_unit% %write_str% ]
/*
/* close the file
&s file_stat = [close %dbms_unit%]
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine build_dbms_help
/*
/* build the help file for the dbms form
/*
/* let the user know whats is going on
&type *****************************************************************************
&type building help file .....
&type *****************************************************************************
/*
/* open the file to hold the help file
&s help_unit = [open dbmsform.hlp openstat -w]
/*
/* write the title of the help file
&s write_str  = 'Help for DBMS data display form'
&s write_stat = [write %help_unit% %write_str%]
/*
/* write a space
&s write_stat = [write %help_unit% ' ']
/*
/* write the name of the table
&s write_str  = [quote %dbms_name% table: ]
&s write_stat = [write %help_unit% %write_str%]
/*
/* if the user doesn't want help on the table fields, then don't try to find
/* the field descriptions
&if %build_help% &then &do
   /*
   /* write the description of the table
   /*
   /* check that the DBMS user_tab_comments table exists
   /* &s table_list = [show tables %conn_dbms%]
   /* &s valid_table = .false.
   /* &do one_table &list %table_list% &while not %valid_table%
   /*    &if %one_table% = USER_TAB_COMMENTS &then &s valid_table = .true.
   /*    &end
   &s valid_table = .true.
   &if %valid_table% &then &do
      /*
      /* get the data out of the table
```

```
      dbmscursor tab_com declare %conn_dbms% ~
      select * from user_tab_comments ~
         where [quote %dbms_name%] = table_name
      dbmscursor tab_com open
      &if %:tab_com.aml$next% &then &do
         &if not [null %:tab_com.comments%] &then &do
            &s com_str = [quote %:tab_com.comments%]
            &s write_str = [quote [substr %com_str% 1 80]]
            &s write_stat = [write %help_unit% %write_str% ]
            &if [length %com_str%] > 80 &then &do
               &s write_str = [quote [substr %com_str% 81 80]]
               &s write_stat = [write %help_unit% %write_str% ]
               &end
            &if [length %com_str%] > 160 &then &do
               &s write_str = [quote [substr %com_str% 161 80]]
               &s write_stat = [write %help_unit% %write_str% ]
               &end
            &if [length %com_str%] > 240 &then &do
               &s write_str = [quote [substr %com_str% 241 80]]
               &s write_stat = [write %help_unit% %write_str% ]
               &end
            &end
         &end
      dbmscursor tab_com remove
      &s write_stat = [write %help_unit% ' ']
      &end
   /*
   /* build the help strings for the fields in the table
   /*
   /* check that the DBMS user_col_comments table exists
   /* &s table_list = [show tables %conn_dbms%]
   /* &s valid_table = .false.
   /* &do one_table &list %table_list% &while not %valid_table%
   /*    &if %one_table% = USER_COL_COMMENTS &then &s valid_table = .true.
   /*    &end
   &s valid_table = .true.
   &if %valid_table% &then &do
      &do cnt = 1 &to %:ot1.aml$ncol%
         /*
         /* get the data out of the table
         dbmscursor col_com declare %conn_dbms% ~
         select * from user_col_comments ~
            where table_name = [quote %dbms_name%] and ~
                  column_name = [quote [value :ot1.%cnt%.aml$name]]
         dbmscursor col_com open
         &if %:col_com.aml$next% &then &do
            &if not [null %:col_com.comments%] &then &do
               &s write_str  = [quote [value :ot1.%cnt%.aml$name] field: ]
               &s write_stat = [write %help_unit% %write_str%]
               &s com_str = [quote %:col_com.comments%]
               &s write_str = [substr %com_str% 1 80]
               &s write_stat = [write %help_unit% %write_str% ]
               &if [length %com_str%] > 80 &then &do
                  &s write_str = [substr %com_str% 81 80]
                  &s write_stat = [write %help_unit% %write_str% ]
                  &end
               &if [length %com_str%] > 160 &then &do
                  &s write_str = [substr %com_str% 161 80]
                  &s write_stat = [write %help_unit% %write_str% ]
                  &end
               &if [length %com_str%] > 240 &then &do
                  &s write_str = [substr %com_str% 241 80]
                  &s write_stat = [write %help_unit% %write_str% ]
                  &end
               &end
            &end
         dbmscursor col_com remove
         &end
      &end
   &end
/*
/* write descriptions of each of the buttons
&s write_stat = [write %help_unit% ' ']
&s write_str  = [quote NEXT :]
```

```
&s write_stat = [write %help_unit% %write_str%]
&s write_str  = [quote Select the next DBMS record for the current bore]
&s write_stat = [write %help_unit% %write_str%]
&s write_stat = [write %help_unit% ' ']
&s write_str  = [quote FIRST :]
&s write_stat = [write %help_unit% %write_str%]
&s write_str  = [quote Select the first DBMS record for the current bore]
&s write_stat = [write %help_unit% %write_str%]
&s write_stat = [write %help_unit% ' ']
&s write_str  = [quote SELECT_BORE:]
&s write_stat = [write %help_unit% %write_str%]
&s write_str  = [quote Select a new bore]
&s write_stat = [write %help_unit% %write_str%]
&s write_stat = [write %help_unit% ' ']
&s write_str  = [quote QUIT :]
&s write_stat = [write %help_unit% %write_str%]
&s write_str  = [quote return to the ARC<->DBMS specification form]
&s write_stat = [write %help_unit% %write_str%]
/*
/* close the help file
&s file_stat = [close %help_unit%]
/*
/* clear off
&return
/*
/*
/*****************************************************************************
/*
&routine usage
/*
/* display a usage message
&type
&type ARCRDBMS: display bore data held in an DBMS table for a user
&type           specified point in an ARC point coverage
&type
&type You must be in ARCPLOT to run ARCRDBMS
&type
&type You must connect to a DBMS before running ARCRDBMS
&type
&type You must set the station type before running ARCRDBMS
&type
/*
/* clear off
&return
/*
/*****************************************************************************
/*
&routine error_msg
/*
/* display an error message for the recalcitrant user
/*
/* does the error message form exist ?, if not then create it
&if not [exists errormsg.men -file] &then &do
   /*
   /* open the menu file
   &s error_unit = [open errormsg.men openstat -w]
   /*
   /* write the menu fields and buttons
   &s write_str = '7 - program generated error message form'
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = '/* program generated error message form '
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = ' '
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = ' ERROR MESSAGE:'
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = ' '
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = ' ERROR: %1'
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = ' '
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = ' %exit'
   &s write_stat = [write %error_unit% %write_str%]
```

```
   &s write_str = ' '
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = '%1 display error_str 40 '
   &s write_stat = [write %error_unit% %write_str%]
   &s write_str = '%exit button cancel ''EXIT'' &return'
   &s write_stat = [write %error_unit% %write_str%]
   &s close_stat = [close %error_unit%]
   &end
/*
/* display the form containing the error message
&menu errormsg.men
/*
/* clear off
&return
/*
/*****************************************************************************
/*
```

Listing of "arcrdbms.men"

```
7 get the data required to link a Point pat to an oracle table
/*************************************************************************
/* name: arcrdbms.men
/* by:   AGTucker
/* date: 29/9/92
/*
/* function: get the name of a point coverage PAT, and its link field,
/*           and the name of an oracle table and its link field
/*
/*************************************************************************

             ARC/INFO <-> DBMS Table Link Specification

  Connected to DBMS: %conn

 Point Coverage PAT: %1 %2
     PAT Link Field: %3 %4

         DBMS Table: %5 %6
    DBMS Link Field: %7 %8

  Build Help File ?: %9

               %examine_data %cancel %help

%conn display conn_dbms 15
%1 button cancel keep '?' ~
   &s pat_name = [getfile *.pat -info -sort -prompt ~
                  'Select the point coverage PAT name'] ; ~
   &s pat_link
%2 display pat_name 40
%3 button cancel keep '?' ~
   &if [null %pat_name%] &then &s msg = 'Specify a PAT table name first' ; ~
   &else &s pat_link = [getchoice [show columns info %pat_name%] ~
                    -prompt 'Select the PAT link field:' -sort]
%4 display pat_link 25
%5 button cancel keep '?' ~
   &s dbms_name = [getchoice [show tables %conn_dbms%] ~
                    -prompt 'Select the DBMS table:' -sort] ; ~
   &s dbms_link
%6 display dbms_name 25
%7 button cancel keep '?' ~
   &if [null %dbms_name%] &then &s msg = 'Specify a DBMS table name first' ; ~
   &else &s dbms_link = [getchoice [show columns %conn_dbms% %dbms_name%] ~
                    -prompt 'Select the DBMS link field:' -sort]
%8 display dbms_link 25
%9 checkbox build_help keep
%examine_data button return 'EXAMINE DATA' &return ; &s .ok2con = .true.
%cancel button cancel 'STOP' &return ; &s .ok2con = .false.
%help button help 'Display help window' return 'HELP' ~
   &popup %.help_path%arcrdbms.hlp
%formopt setvariables immediate messagevariable msg
%forminit &s x ; &if not [variable dbms_name] &then &s dbms_name ; ~
                 &if not [variable pat_name ] &then &s dbms_link ; ~
                 &if not [variable pat_name ] &then &s pat_name  ; ~
                 &if not [variable pat_name ] &then &s pat_link
```

Listing of "arcrdbms.hlp"

```
ARCRDBMS allows you to access point data tables in a relational database by
        selecting a point in the ARCPLOT program.

Before running ARCRDBMS , you must:

        - connect to the DBMS with the ARC/INFO 'connect' command.

        - have produced a point coverage with each point containing a field
          that identifies the records in the DBMS that hold attribute
          data for that point.

        - have read access to the DBMS table holding the data for the points
          in the point coverage

        - be in arcplot, and have displayed the points in the point coverage

        - have set the station (ARC/INFO '&stat' command) type


ARCRDBMS requests you to enter:

        - the name of the coverage hold the points

        - the name of the DBMS table holding the point attribute data

        - the name of the field in the PAT of the point coverage that links
          the point to the DBMS records for that point.

        - the name of the field in the DBMS that links the attribute data to
          the point

        - if you want to build a help file for the DBMS table that includes
          the comments in the 'USER_TAB_COMMENTS' and 'USER_COL_COMMENTS'
          tables

If more than one DBMS record is available for a point, the first matching
record is displayed, but the remaining records can be viewed with the 'NEXT'
button.

ARCRDBMS will build a form to allow you to view the data. If the DBMS is
Oracle version 6.0 or later, ARCRDBMS will attempt to read the descriptive
comments for the DBMS table and fields from the USER_TAB_COMMENTS and
USER_COL_COMMENTS tables. These comments can be viewed by pressing the
'HELP' button.

Please report any problems you encounter with ARCRDBMS to
        Andrew Tucker,
        Rm. 475, Ext. 9248,
        Environmental Geology and Groundwater,
        Australian Geological Survey Organisation,
        GPO Box 378,
        Canberra, ACT 2601
        Australia
```