Bureau of Mineral Resources, Geology & Geophysics

BMR Record 1991/30

R E C O R D

RTMAP
BMR Regolith Database
Users' Manual

by

Sonja Lenz
Information Systems Branch

BMR
GEOLOGY AND
GEOPHYSICS
AUSTRALIA

RTMAP
BMR Regolith Database
Users' Manual

by

Sonja Lenz
Information Systems Branch

BMR Record 1991/30

August 1991

BMR
GEOLOGY AND
GEOPHYSICS
AUSTRALIA

CONTENTS

1 INTRODUCTION

This manual describes the structure of RTMAP, the BMR regolith database, and how to insert, update or delete records in the database as well as how to retrieve data in the form of basic reports. It also describes how the database administrator (DBA) can make amendments to the structure should they be necessary.

Regolith is bedrock which has been altered by processes at or near the surface including weathering, erosion, transportation, and terrestrial sedimentation (Pain et al, 1991). For a further discussion of regolith concepts and mapping techniques see the companion publication to this manual, RTMAP - BMR Regolith Database Field Handbook (Pain et al, BMR Record 1991/29).

RTMAP is a database of information about the Australian regolith. It is a relational database on BMR's DG MV20000 computer using the ORACLE database management system and covers areas mapped by BMR's Regolith Group, with emphasis on regolith terrain units. Together with the spatial attributes of these mapping units the database will be used in a geographic information system (GIS). Data collected in the field are recorded in field books, and later entered into the database.

As a first step in the development of this database, the descriptions of many regolith and terrain attributes had to be structured into distinct, well-defined categories to make them more suitable to a relational database. This aim has not yet been completely achieved. In fact, all people involved in regolith mapping will have to take part in an ongoing process of definition and re-definition of these attributes and their classification until a system acceptable to the wider regolith-mapping community has been established. The current version of data attributes for regolith mapping projects within BMR is contained in the RTMAP Field Handbook.

2.1 THE RELATIONAL MODEL

ORACLE is a relational database management system (RDBMS). Because relational database terms are used inconsistently in the literature (Brathwaite, 1989, p.66) short definitions of the terms used in this manual are given here.

In the relational database model we talk about **data entities** and the **relationships** between these entities. The basic (and only) unit of data storage in a relational database is the **table**, a two-dimensional grid of **columns** and **rows**. An **entity** is any distinguishable object that is to be represented in the database; usually at least one table is set up to contain the attributes of each entity. Every table within the database is defined with a name and a set of columns. The **attributes** which characterise the entity are the columns in the table. Each column is given a name, a datatype, and a width. The distinct instances or occurrences of the entity, the so-called records or table rows, each have a certain set of attribute values.

TABLE (ENTITY)

	column 1	column 2	. . .	
row 1				record 1
row2				record 2
. . .				. . .
	attribute 1	attribute 2	. . .	

The **relationships** between entities are logical links between them which can be used to associate data in one table with that in another. This is usually done by "joining" two tables through data values which are common to both tables.

If one record in a table relates to a single record in another table we talk of a **one-to-one** relationship. Similarly, if one record in the first table corresponds to more than one in the second table, we are looking at a **one-to-many** relationship. Finally, a **many-to-many** relationship occurs when several records in the first table correspond to more than one record in the second table.

A **key** within a table is an attribute (column) or attributes whose values uniquely identify each record (row). For instance, in the RTMAP database the u_id (unit identifier) and site_id (site identifier) attributes are the keys to the UNIT and SITE tables, and combinations of u_id or site_id with other attributes are keys to most of the other tables.

Indexes can be used to guarantee uniqueness of records and/or to speed up execution of transactions.

2.2 DATABASE TABLES

RTMAP comprises a total of 35 tables (see Appendix A: Logical data structure). The database is made up of two parts: tables which contain information on field sites and tables which contain information on regolith terrain mapping units. Both parts have lookup tables in common.

The database schema (Appendix B) contains a detailed description of the tables, their attributes, and the relationships between them. Lists of possible values for certain attributes are also included in the description.

The schema also contains all the SQL commands for creating the RTMAP database in ORACLE. It can be run from within SQL*Plus should it be necessary to re-create the database.

2.2.1 Main Tables

The two most important entities within the regolith terrain database RTMAP are the regolith terrain mapping unit and the field site. Tables UNIT and SITE represent these entities. A regolith terrain mapping unit recognised in the field can extend over more than one landform type and can include several regolith types. Similarly, a field site will occur in one landform type but there can often be zone variations with depth. Tables LANDF_UNIT, REGT_LANDF and ZONE reflect these one-to-many relationships.

All other tables in the database relate directly or indirectly to these five main tables.

2.2.2 Lookup Tables

Some entities within the RTMAP database structure like drainage and landform patterns, tectonic structure elements, and geomorphic and weathering processes have a limited number of valid attribute values. These values have been listed in authority tables. Used as lookup tables for the description of field sites and/or mapping units they help reduce data redundancy, enforce data consistency and facilitate easier comparison of records.

The following lookup tables are currently used:

COMP	map compilers and field observers
DRAIN	drainage patterns
GPROC	geomorphic processes
KEYWORDS	keywords relating to the used references
LANDF	landform patterns
LITH	bedrock lithology types
MAPS	all maps used
PROV	regolith terrain provinces
REFS	all references used
REGTYPE	regolith types
STRAT	stratigraphic names
TECT	tectonic structure elements
WPROC	weathering processes

The contents of these tables are listed in the RTMAP Field Handbook (BMR Record 1991/29).

2.2.3 Other Tables

All other tables in the database are junction or intermediary tables. They relate mapping units or field sites to attributes like drainage patterns, and geomorphic or weathering processes in those cases where there is a many-to-many relationship between entity and attribute. Examples are the tables GPROC_ZONE and DRAIN_LANDF.

SQL*Forms is a full-screen interface tool used for creating, modifying and using forms for data entry and retrieval in an ORACLE database. Records may also be updated or deleted through SQL*Forms.

In order to access the database and use the RTMAP forms the prospective user must be a registered ORACLE user with a username and a password. On BMR's DG MV20000, all forms and files for accessing RTMAP reside in a directory called :ULD:REGOLITH. Any DG user may add this directory to his/her search list (at the DG prompt, type **SEA :ULD:REGOLITH [!SEA]**) and read or execute the forms, report files etc.

For more detailed information on the use of SQL*Forms refer to: SQL*Forms Designer's Reference and SQL*Forms Designer's Tutorial by ORACLE Corporation.

3.1 GENERAL COMMENTS

Keyboard keys are referred to by their SQL*Forms **function** in this manual - see Appendix C: Keyboard overlays for ORACLE. The available functions are different for designers and operators, and different again in query and entry mode. A listing of the available functions within SQL*Forms can be displayed on the screen by pressing the key called **<SHOW FUNCTION KEYS>.**

A menu-based system for accessing the database through SQL*Forms or SQL*Plus has been set up. We shall first look at database access through SQL*Forms, Chapter 4 describes the use of SQL*Plus. Following is a general description of how to use the query/entry forms. The individual forms are discussed under 3.2.1 to 3.2.3.

The query/entry forms are made up of **blocks.** Each block corresponds to a different table (= base table) in the database. The two main forms - UNIT and SITE - are each comprised of several blocks some of which take up a whole screen. Appendix A shows all the tables accessed by these two forms. The blocks in a multi-block screen are indicated by solid lines.

The **cursor** generally moves within the screen from left to right and from top to bottom. Watch the **message line** at the bottom of the screen closely as you move through the fields using the **<NEXT FIELD>** key to go forwards and the **<PREVIOUS FIELD>** key to go backwards. The message line displays help messages for data entry and error messages should something go wrong. There is also a **<DISPLAY ERROR>** key that gives additional information on errors which have occurred.

Some blocks display only one record per screen (e.g. the UNIT and SITE blocks), whereas others are multi-record blocks and display several records at a time (e.g. GPROC_LANDF and WPROC_ZONE blocks). Some fields are **mandatory** which means a valid value has to be entered before the cursor can move out of the field. Look at the help message or the database schema (Appendix B) if you are uncertain about the data type and/or the valid entries for a particular field. Full lists of valid values for attributes are also contained in the RTMAP Field Handbook (BMR Record 1991/29).

A number of fields are **display-only** fields which means they are not part of the base table for the particular block and the cursor does not enter them in entry mode. They are there

for checking purposes and display the values that correspond to the codes entered or retrieved from lookup tables.

Should you realize after leaving a field that you have made a mistake while entering data, you can always take the cursor back by pressing the **<PREVIOUS FIELD>** key for moving within a block or the **<PREVIOUS BLOCK>** key to take you back through blocks. Correct the mistake by typing over it.

It is generally recommended that users save the record/s in each block to the database ("commit" the data) before moving to the next one. In the main entry forms so-called triggers have been set up which move the cursor automatically to the next block as soon as the **<COMMIT>** key has been pressed. The message "**n records processed**" indicates that changes have been committed to the database.

3.2 SQL*FORMS WITH RTMAP

To run the forms, log on to the LAN and the DG, then type **RTMAP**.

You will be prompted for your ORACLE username and password.

The following menu will then come up on the screen:

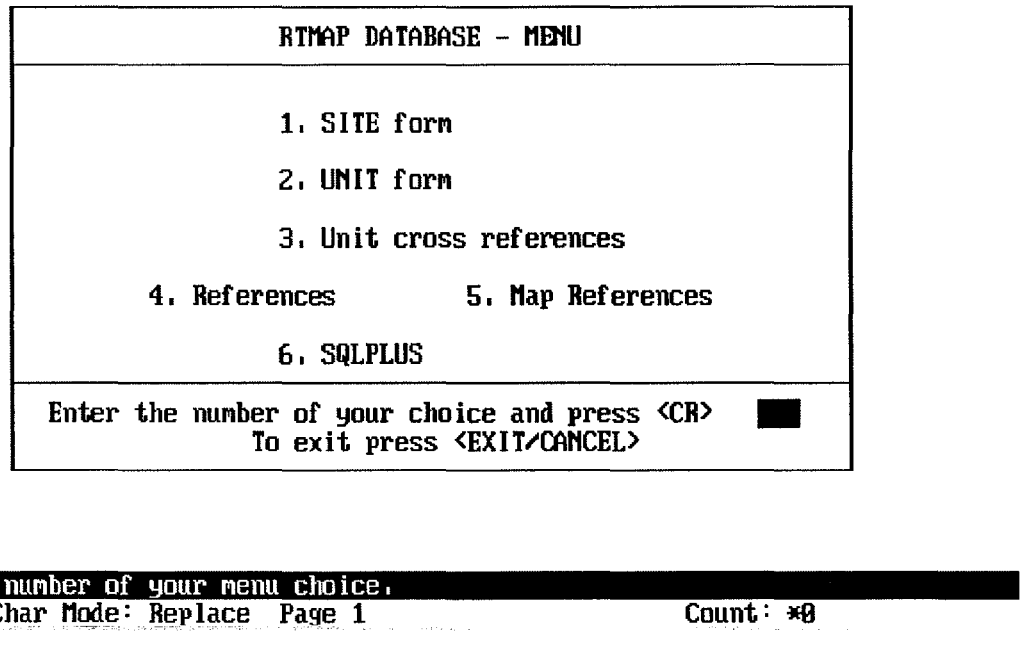


Figure 1. The RTMAP opening menu.

The DBA's menu also provides options for updating the lookup tables.

QUERYING THE DATABASE: Go into the appropriate form by entering the corresponding number on the menu. To retrieve all records in a particular block which satisfy certain conditions, press **<ENTER QUERY>** and then enter the values of the attributes (fields) which you are interested in, moving from field to field by using the **<NEXT FIELD>** and **<PREVIOUS FIELD>** keys. Pressing **<EXECUTE QUERY>** will result in the record/s being displayed on the screen. To scroll through the records which satisfy your query criteria use the **<NEXT RECORD>** and **<PREVIOUS RECORD>** keys.

The blocks within one form have been coordinated and on querying the database all data in subsequent blocks which are relevant to a particular site or unit displayed on the screen will be retrieved. Often there may be no data in one or more of the subsequent blocks which results in the message **"Query caused no records to be retrieved"** being displayed in the message line. Press any function key to acknowledge and remove that message, if necessary several times, to enable the cursor to move freely again.

In the UNIT form a query in the first block will only retrieve data in the blocks of that first screen. On entering the second screen, press **<EXECUTE QUERY>** again to retrieve all existing data in subsequent blocks.

DELETING RECORDS: If a record has to be deleted, use the above procedure to call it up on the screen and then press **<DELETE RECORD>**. Caution: Make sure the cursor is really positioned on the record you want to delete before pressing the delete key.

UPDATING RECORDS: To update or make changes to a record, use the above procedure to display it on the screen and then place the cursor in the field you want to update or change. Make any amendments to the field/s using the editing keys (**<LEFT ARROW>, <RIGHT ARROW>, <BACKSPACE>** and **<INSERT/REPLACE>**) and by typing over the existing field value.

ENTERING NEW RECORDS will be covered in more detail for individual forms on the following pages.

3.2.1 The SITE Form

The SITE form which consists of three screens has been set up for entry of data relating to field sites and the zones associated with them. Through this form, data can be entered into tables SITE, ZONE, GPROC_ZONE, WPROC_ZONE, SAMPLE, SIMSTRATA and AGE. The lookup tables LANDF, LITH, REGTYPE, GPROC, WPROC and STRAT can be accessed for codes from within the form (see Appendix A: Logical data structure for field sites). To call up the lookup table for a particular code press the **<NEXT KEY FIELD>** key and scroll through the values with the **<NEXT RECORD>** key.

To run the form, press **1**, from the RTMAP opening menu (see page 6).

The following form will come up on the screen (without data):

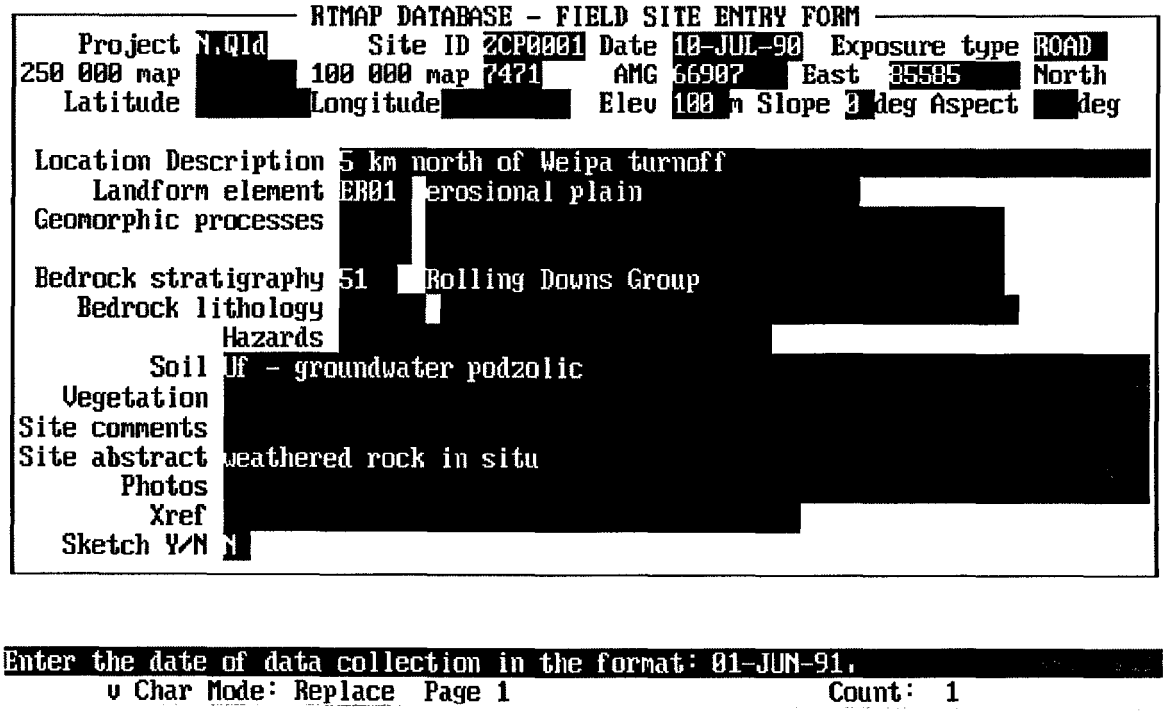


Figure 2. Site entry form, first screen—Site Description.

This is a one-block screen corresponding to one table, the SITE table.

The Site ID for each site has to be a unique combination of a letter for the organisation (it is 'Z' for BMR), the compiler's two initials and a four digit sequential number (e.g. ZCP0001).

Enter the data for the site moving between fields by pressing **<NEXT FIELD>** or **<PREVIOUS FIELD>.** When you are satisfied with the data in the first block, press **<COMMIT>** to add the record to the database and move on to the Zone Description block.

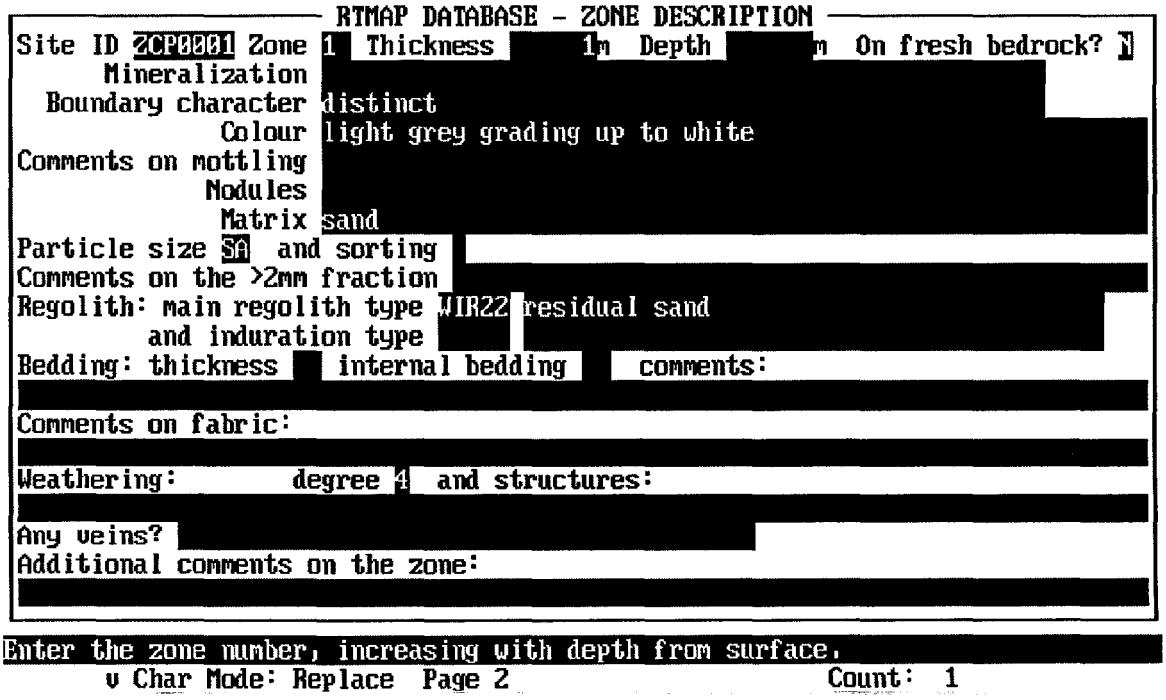


Figure 3. Site entry form, second screen—Zone Description.

The whole screen is again based on one table, the ZONE table. The Site ID which is a mandatory field is copied to this block from the previous screen. The only other field that is mandatory in this block is the Zone number.

Fill in the fields for which there are data and commit the record to the database and move to the second screen of the zone description by pressing **<COMMIT>.**

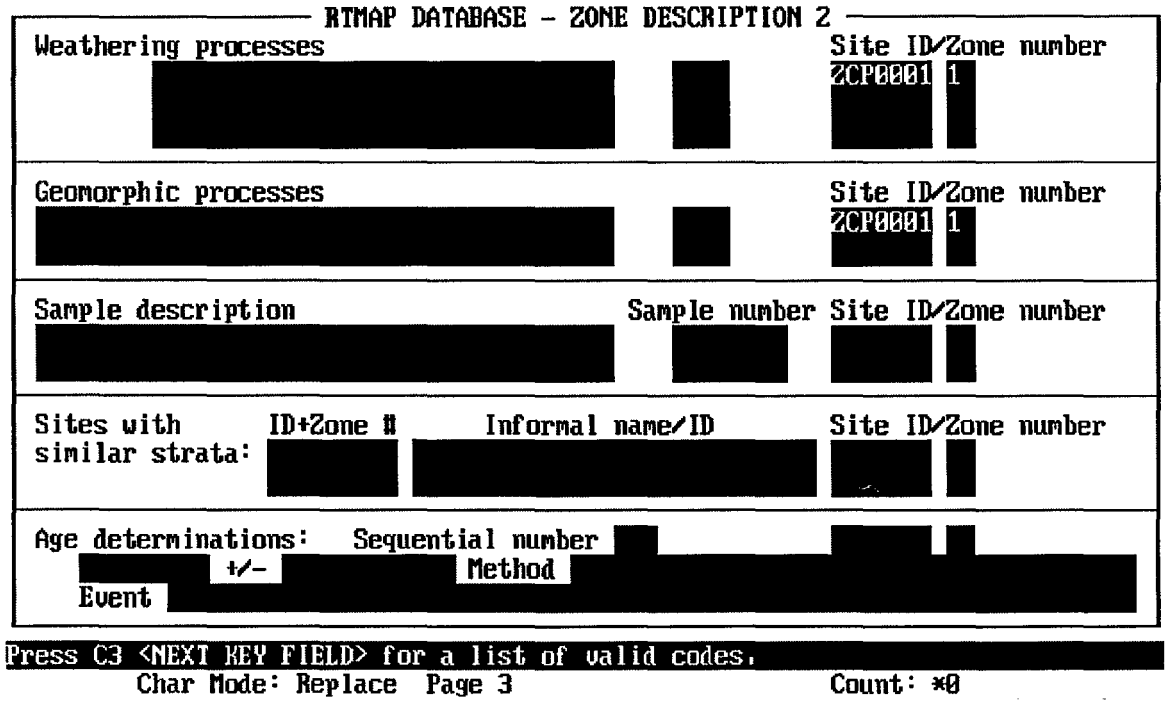


Figure 4. Site entry form, third screen—Zone Description 2.

As the cursor moves from one block to the next, the Site ID and Zone number get automatically copied into their respective fields. Enter codes for weathering processes, then press **<NEXT FIELD>** to move the cursor down within the block and add another code. Press **<COMMIT>** and **<NEXT BLOCK>** to enter the second block. Repeat the procedure for geomorphic processes.

If there are data to be entered for samples, sites with similar strata, or age determinations press **<NEXT BLOCK>** to get into the Sample, Simstrata and Age blocks, enter the data and press **<COMMIT>.**

Pressing **<F5>** from anywhere in this third entry screen will take the cursor back to the first Zone Description screen (which is the second screen) where data can be entered for the next zone for the current site after pressing **<CREATE RECORD>.**

To return to the first site entry screen, press **<NEXT BLOCK>** after committing the data entered. To remove the previous record from the screen and prepare for a new site entry, press **<CREATE RECORD>.**

3.2.2 The UNIT Form

The UNIT form is used for entering data relating to regolith terrain mapping units after processing field and published data. This form also consists of three screens. Data are entered into tables UNIT, COMP_UNIT, TECT_UNIT, PROV_UNIT, MAPS_UNIT, REFS_UNIT, LANDF_UNIT, DRAIN_LANDF, GPROC_LANDF, WPROC_LANDF, LITH_LANDF and REGT_LANDF. Lookup tables COMP, TECT, PROV, MAPS, REFS, LANDF, LITH, STRAT, DRAIN, GPROC, WPROC and REGTYPE can be accessed for codes from within this form (see Appendix A: Logical data structure for mapping units). Pressing **<NEXT KEY FIELD>** with the cursor positioned in the code field calls up the appropriate lookup table which can then be scrolled through to find the appropriate code.

To run the UNIT form, press **2**, at the RTMAP opening menu (see page 6).

The following form will come up on the screen (without data):

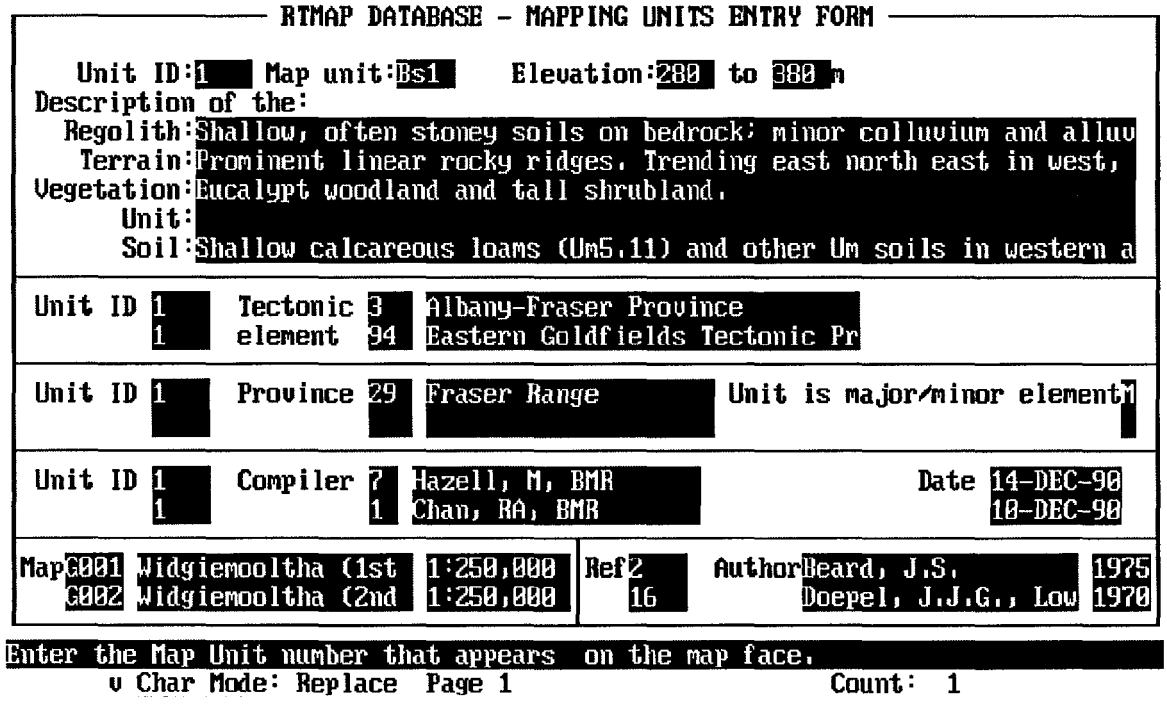


Figure 5. Unit form, first screen—Unit Description.

The cursor is positioned in the Map Unit field. The Map Unit is the identifying number/code that is on the map face. The cursor does not enter the Unit ID field as this is a sequential number automatically generated by the system before inserting the new record into the database.

Enter all the available data for the first block moving from field to field by pressing **<NEXT FIELD>** or **<PREVIOUS FIELD>**. Press **<COMMIT>** when all the data have been entered. This will also move the cursor to the next block.

Repeat the procedure for all blocks on this entry screen. Skip through blocks for which there are no data by pressing **<NEXT BLOCK>**, if necessary several times. Pressing **<COMMIT>** from the References block in the bottom right corner takes the cursor to the Landform Unit block in the second entry screen.

Pressing **<F5>** from anywhere in the first screen will take you to the second screen and position it at the top of the Landform Unit block.

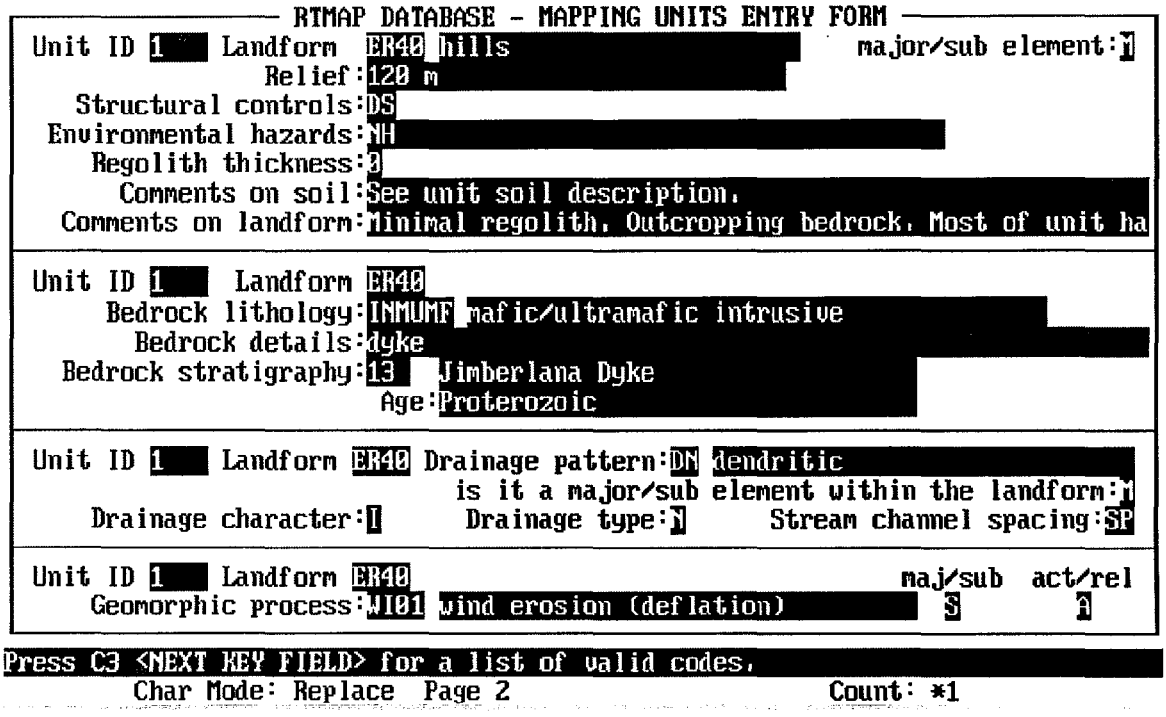


Figure 6. Unit form, second screen—Landform Unit Description.

In blocks where more than one record is to be entered but only one record is displayed on the screen, pressing **<NEXT FIELD>** in the last field of the block will move the cursor back to the top of the block and clear it for another record to be entered. When all the records belonging to a block have been entered, press **<COMMIT>** to commit the data to the database and move to the next block.

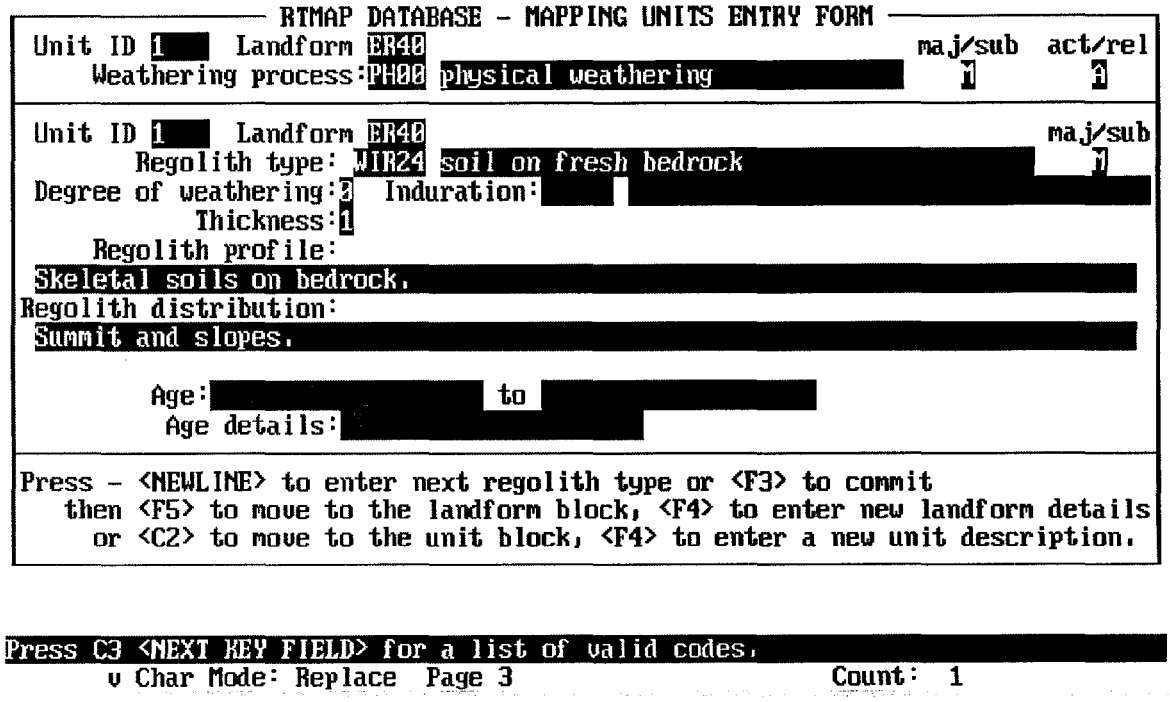


Figure 7. Unit form, third screen—Regolith Type Description.

The Regolith Type Description block is the final block in this entry form. Pressing **<COMMIT>** after entering all data in this block will commit the data to the database but the cursor will not move to another block.

At this point the operator has the option of pressing **<NEXT BLOCK>** to get back to the first block of the UNIT entry form or pressing function key **<F5>** to get into the Landform block if details of another landform type are to be entered. Press **<CREATE RECORD>** to clear the blocks of previous data and prepare for entering new data.

To create several records which have attribute values in common, it can be time-saving to use the record or field duplicating facilities: Call up the record whose values are to be duplicated and clear the block by pressing **<CLEAR RECORD>**, then press **<DUPLICATE RECORD>** for the whole record to be copied to the block again or **<DUPLICATE FIELD>** for copying field values into each field the cursor is positioned in. Edit the values as necessary and **<COMMIT>** the record to the database.

When all the data for a particular unit have been entered and committed the operator is likely to be in the Regolith Type block and wanting to return to the start of the form. To do this press **<NEXT BLOCK>** to display the first screen and **<CREATE RECORD>** to clear the blocks of previous data and prepare the screen for entering data for a new unit.

3.3.3 Other Forms

Simple forms have been set up for entering data into all the lookup tables and the sequential numbers table SEQNOS. Only users with DBA rights can make changes to the lookup tables, either from within the SITE or UNIT forms after calling up the lookup table by pressing **<NEXT KEY FIELD>**, or from the menu. To run the forms from the menu, choose their corresponding option on the DBA's menu.

As you enter the form, all records are retrieved and a screenful is displayed.

This is an example of a screen for data entry into a lookup table, in this case the Stratigraphic Names table STRAT:

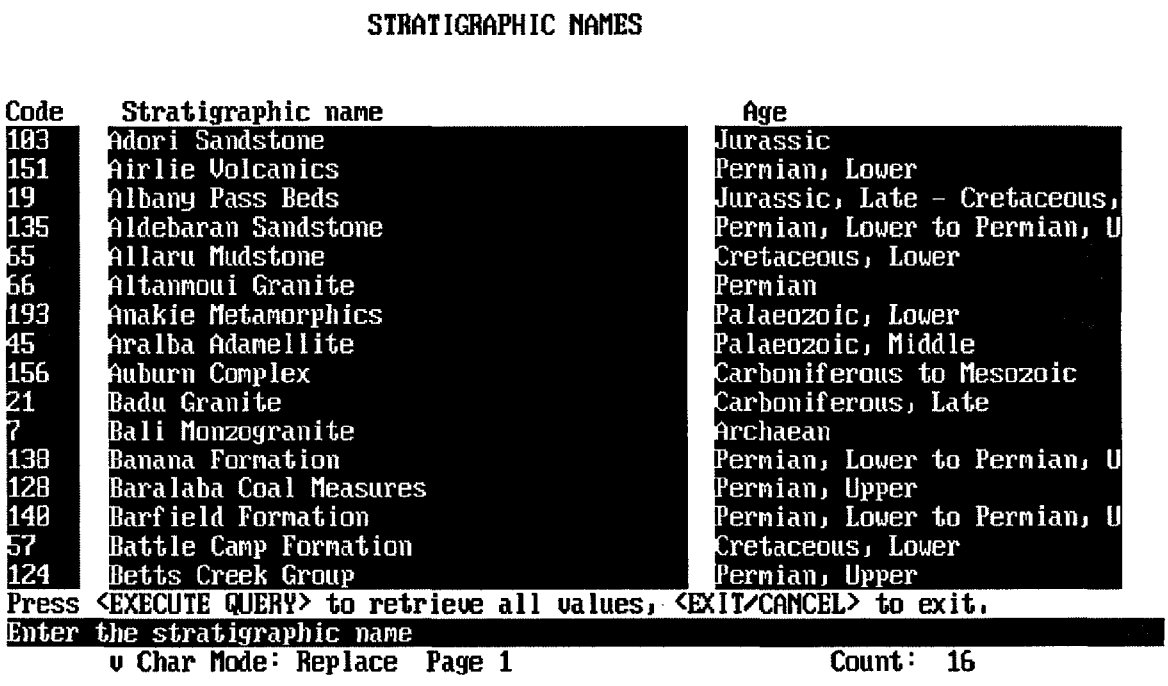


Figure 8. Lookup table for stratigraphic names.

To **add a new record:**

- either use **<NEXT RECORD>** to get to the bottom of the retrieved list and enter the new record in the first empty line, or
- press **<CREATE RECORD>** and enter the new record.

Repeat the procedure for any additional records. Press **<COMMIT>** to commit the new data to the database and **<EXIT/CANCEL>** to exit the form.

To **delete a record,** position the cursor on the record to be deleted and then press **<DELETE RECORD>**. Press **<EXIT/CANCEL>** to exit the form.

To **update a record**, position the cursor on the record and edit the field/s to be updated. Press **<EXIT/CANCEL>** to exit the form.

The ORACLE relational database management system may be accessed via the Structured Query Language (**SQL**). There are four different types of SQL statement:

- data definition e.g. CREATE TABLE...
- data control e.g. GRANT SELECT...
- data manipulation e.g. INSERT...
- data query e.g. SELECT * FROM...

The data definition and data control statements are used when setting up, or changing the structure of, the database, and for controlling data access. These are mainly tasks performed by the DBA. Data manipulation and data query statements are used for inserting, updating and deleting records as well as for data retrieval.

Data manipulation and query, as well as some of the data control statements, are invoked from within SQL*Forms without the user having to learn the SQL syntax.

SQL*Plus is an extended version of SQL which allows for limited report formatting in addition to the standard SQL functions.

RTMAP data on the DG can be viewed by any ORACLE user as the public has been granted SELECT privileges to all tables. Public synonyms have been created for all tables in RTMAP which allows them to be called by the names used in this manual without having to prefix them with the name of the owner ('RTMAP.'). Only members of the Regolith Group have the right to manipulate (INSERT, UPDATE, DELETE) data in the database.

For more detailed information on the use of SQL*Plus refer to: SQL*Plus User's Guide, and SQL*Plus Reference Guide by ORACLE Corporation.

4.1 QUERYING THE DATABASE

Any registered ORACLE user can query the database as it is publicly available for viewing.

To run SQL*Plus, choose option 6 on the RTMAP opening menu (see page 6) or, alternatively, log on to the LAN and the DG, then into SQL*Plus by typing:

SQLPLUS

You will be prompted for your ORACLE username, then your password.

The general SQL statement for retrieving data is

SELECT column FROM tablename

WHERE condition; (optional)

For instance, to retrieve all the records in the lookup table of drainage patterns, at the SQL> prompt type:

SELECT * FROM DRAIN;

or to retrieve some attributes of all the units in a certain landform type:

```
SELECT U_ID, RELIEF,REGT_THICK
    FROM LANDF_UNIT
    WHERE L_CODE ='ER40';
```

Combinations of conditions are also possible:

```
SELECT U_ID,RELIEF,REGT_THICK
    FROM LANDF_UNIT
    WHERE L_CODE = 'ER40'
    AND MM_CODE = 'M';
```

as are combinations of columns from several tables ("table joins"):

```
SELECT SITE.SITE_ID,S_DATE,EXP_TYPE,MAP1,Z_NO
    FROM SITE,ZONE
    WHERE L_CODE = 'ER40'
    AND SITE.SITE_ID = ZONE.SITE_ID;
```

The last line in this statement shows which attributes in the two tables are to be used for "joining" them. If this condition is left out a so-called "Cartesian join" is done which combines every record in one table with every record in the other table. This has the potential of creating a huge output listing - probably not what the user was looking for. Care should therefore be taken to ensure that the conditions imposed on the retrieval are tight enough to retrieve the required information.

Type **BYE** to exit from SQL*Plus.

See the SQL*Plus User's Guide and the Reference Guide for further information on SQL*Plus syntax.

4.2 UPDATING THE DATABASE

Regolith Group members have update rights to the database which means that they can insert new records, delete records and change data, either through SQL*Forms or from within SQL*Plus.

To run SQL*Plus, choose option 4 on the RTMAP opening menu (see page 6) or, alternatively, log on to the LAN and the DG, then into SQL*Plus by typing:

```
SQLPLUS
```

You will be prompted for your ORACLE username, then your password.

The general SQL syntax for the three operations mentioned above is:

```
INSERT INTO tablename
    VALUES (...,...,...);
DELETE FROM tablename
    WHERE condition;          (optional)
```

UPDATE tablename

SET column = value

WHERE condition; (optional)

Take the following examples:

1 You want to add 'dust deposition' as a geomorphic process to zone 2 of site ZCP0010:

INSERT INTO GPROC_ZONE

VALUES ('ZCP0010',2,'WI03');

Note that the values have to be in the same order as specified when creating them which is the order they appear in when you ask for a description of the table: **DESCRIBE tablename.**

2 You want to delete all records from the landform units table which are only subordinate landform components of unit 220:

DELETE FROM LANDF_UNIT

WHERE U_ID = 220

AND M_CODE = 'S';

3 You want to change a type description in the regolith type lookup table:

UPDATE REGTYPE

SET REGT_DESC = 'scree deposits'

WHERE REGT_CODE = 'SDC01';

If you do **not** want to commit the changes you have made to the database,

type **ROLLBACK**

and **BYE** to exit SQL*Plus.

5 REPORTS

Simple reports can be written using the SQL*Plus language which adds some report formatting features to the basic SQL commands. For information on the use of SQL*Plus refer to the SQL*Plus User's Guide and the SQL*Plus Reference Guide by ORACLE Corporation. An example of such a retrieval and its output is attached as Appendix D.

More sophisticated reports using the **S**tructured **Q**uery **R**eport Writer (SQR) for retrieving all information on regolith terrain mapping units within a certain project area or just selected data as a summary of the units have been set up. They are called UNITREP.SQR and UNITSUM.SQR. Similarily, an SQR report called SITEREP.SQR has been written to retrieve all information on individual sites within a certain project area. Copies of the reports can be edited in SLATE or SED on the DG, or alternatively, in your wordprocessor as a non-document file, to retrieve only the records which fulfil a certain set of conditions of your choice. Contact the DBA for help.

To run a report, type **SQR reportname**

You will be prompted for your ORACLE username and then your password.

After running a report and retrieving the required information the result is stored in a .LIS file in your DG directory. The file can be printed out as it is (it is in ASCII format) or, alternatively, imported into the wordprocessing or desktop publishing package of your choice for further enhancing before publication.

An example of an SQR report and its output is attached as Appendix E.

For further information on the use of SQR refer to the SQR User Guide by SQ Software.

6 DATA EXPORT, IMPORT AND BACKUP

Export and **Import** are two ORACLE utilities for moving ORACLE data to and from operating system files. The files can be used for archiving data (= data backup) or moving data between operating systems or ORACLE databases. The following types of ORACLE data can be stored in this way: table definitions, table data, indexes, space definitions, grants, synonyms, and view definitions.

With the Export utility data in the database are copied to a special kind of backup file. This export file is in a special format and, therefore, no attempt to edit it should be made. It can only be used by the Import utility which restores the exported data into an ORACLE database - each table is re-created and its data loaded into it.

On the DG, full backups of all databases are done on a regular basis by the Informations Systems Branch computer operators. In the case of a system failure or human error with consequent data loss or corruption, the databases can be restored. Contact the database administrator in Informations Systems Branch if you need this kind of help.

When using the database on a stand-alone PC (e.g. a laptop in the field) it is imperative to make regular exports of the tables containing your data as a backup on to floppy discs.

Export and Import are interactive utilities - you are asked questions and the utility proceeds according to your answers. To run the utilities, type

EXP or **IMP**

and respond to the questions (you will be prompted for your ORACLE username and then your password first).

For details and further information refer to ORACLE Utilities User's Guide.

7 CHANGING THE DATABASE STRUCTURE

As the database is used it will most certainly become apparent that the present design is not perfect and tables might have to be added or dropped, columns added or their width or datatype changed. The DBA has the right to make changes like these to the structure of the database.

The following sets out the basic procedures for making those changes. First log on to the LAN and DG, then into SQL*Plus by typing:

SQLPLUS

or, alternatively, choose option 4 on the RTMAP opening menu (see page 6).

You will be prompted for your ORACLE username, then your password.

7.1 ADDING/DROPPING A TABLE

To add a table, enter at the SQL> prompt,

```
CREATE TABLE tablename
    (column1 datatype1(size1),
    column2 datatype2(size2),
    etc)
    SPACE space name;
```

You must also specify whether the columns are mandatory (NOT NULL) or not (see the next example).

The space name must be a valid space definition that has already been created on the system. To list the available space definitions, type

SELECT * FROM SPACES;

There should be one contained in the list that suffices for the future space requirements of your new table, otherwise a new space definition must be created. For more information on space definitions refer to the ORACLE Database Administrator's Guide.For instance, the SQL statements for creating the lookup table for landform patterns are:

```
CREATE TABLE LANDF
    (L_CODE CHAR(4)  NOT NULL,
    L_DESC CHAR(30) NOT NULL)
    SPACE SPB02;
```

For further examples refer to Appendix B: Database Schema.

To drop a table, type

DROP TABLE tablename; e.g.

DROP TABLE RTCLASS;

Existing indexes on the table will be dropped at the same time.

Caution: Before dropping a table make sure that it does not contain data which are still needed as it can not be restored (there is no UN-DROP command).

7.2 CREATING AN INDEX

Indexes on tables are a means of speeding up retrieval times on big tables. However, too many indexes will slow down update activity. An index can be concatenated which means that a combination of columns is used for indexing.

To create an index in SQL*Plus, specify its name and the table with its column/s that contain/s the information to go into the index:

CREATE INDEX indexname

ON tablename (column1,column2,...);

If you specify that the index is a **unique** index ORACLE will make sure that there are no two entries (records) in the table with the same value/s in the specified column or combination of columns.

For example, the SQL statement to create a unique index on the unit identifier in the UNIT table is:

CREATE UNIQUE INDEX UNIT1 ON UNIT(U_ID);

For further examples refer to Appendix B: Database Schema.

7.3 MODIFYING A COLUMN

You can change a column's width or datatype in SQL*Plus by typing:

ALTER TABLE tablename

MODIFY (column datatype(size));

For instance, to change the column for easting values to a 7-digit number field of which 2 digits are decimals, you type:

ALTER TABLE SITE

MODIFY (EAST NUMBER(7,2));

To modify more than one column at a time, use commas within the parentheses to separate each column and its definitions from the next.

To change a mandatory field to a non-mandatory one, add the NULL clause to the end of the column specification. A non-mandatory field can only be changed to a mandatory one (NOT NULL) if there are no null values in the column.

7.4 ADDING A COLUMN

A column can be added to an existing table with the commands:

ALTER TABLE tablename

ADD (column datatype(size));

To add more than one column, use commas within the parentheses to separate each column and its definitions from the next.

For example, to add two fields for map name and scale to the UNIT table, type:

```
ALTER TABLE UNIT
        ADD (M_NAME CHAR(5),
        SCALE NUMBER(7));
```

All fields in a new column initially have a value of null. Therefore, a new column added to an existing table cannot be defined as NOT NULL when the table already contains records. If a new column is to be made mandatory, add the column, then give every record a non-null value, and finally, use the ALTER TABLE command with the MODIFY clause to change it to NOT NULL.

Type **BYE** to exit from SQL*Plus.

Refer to SQL*Plus User's Guide and SQL*Plus Reference Guide for further information on SQL*Plus syntax.

8 MODIFYING/CREATING FORMS

Only the DBA can modify an existing form or create a new one for use within the RTMAP database.

The need for modifying an existing form arises when the underlying table (= base table) is changed, i.e. column/s added, column size/s and/or data type/s changed (see 7). If the form is not modified after such changes are made to the table structure it may not be possible to use the form for correct data entry. Indeed, sometimes it can not be used at all and the attempted use will only create error messages.

Likewise, a form may have to be modified if a new table is added to the database or a table is dropped. Sometimes a completely new form might have to be created in these cases.

Refer to the SQL*Forms Designer's Reference or Designer's Tutorial for full instructions on creating or modifying a form.

9 REFERENCES

Braithwaite, K.S., 1989, Systems design in a database environment, McGraw-Hill Book Company.

Pain, C., Chan, R., Craig, M., Hazell, M., Kamprad, J. & Wilford, J., 1991, RTMAP BMR Regolith Database Field Handbook, BMR Record 1991/29.

SQL*Forms V.2.0 Designer's Reference, Oracle Corporation.

SQL*Forms V.2.0 Designer's Tutorial, Oracle Corporation.

SQL*Plus V.2.0 User's Guide, Oracle Corporation.

SQL*Plus V.2.0 Reference Guide, Oracle Corporation.

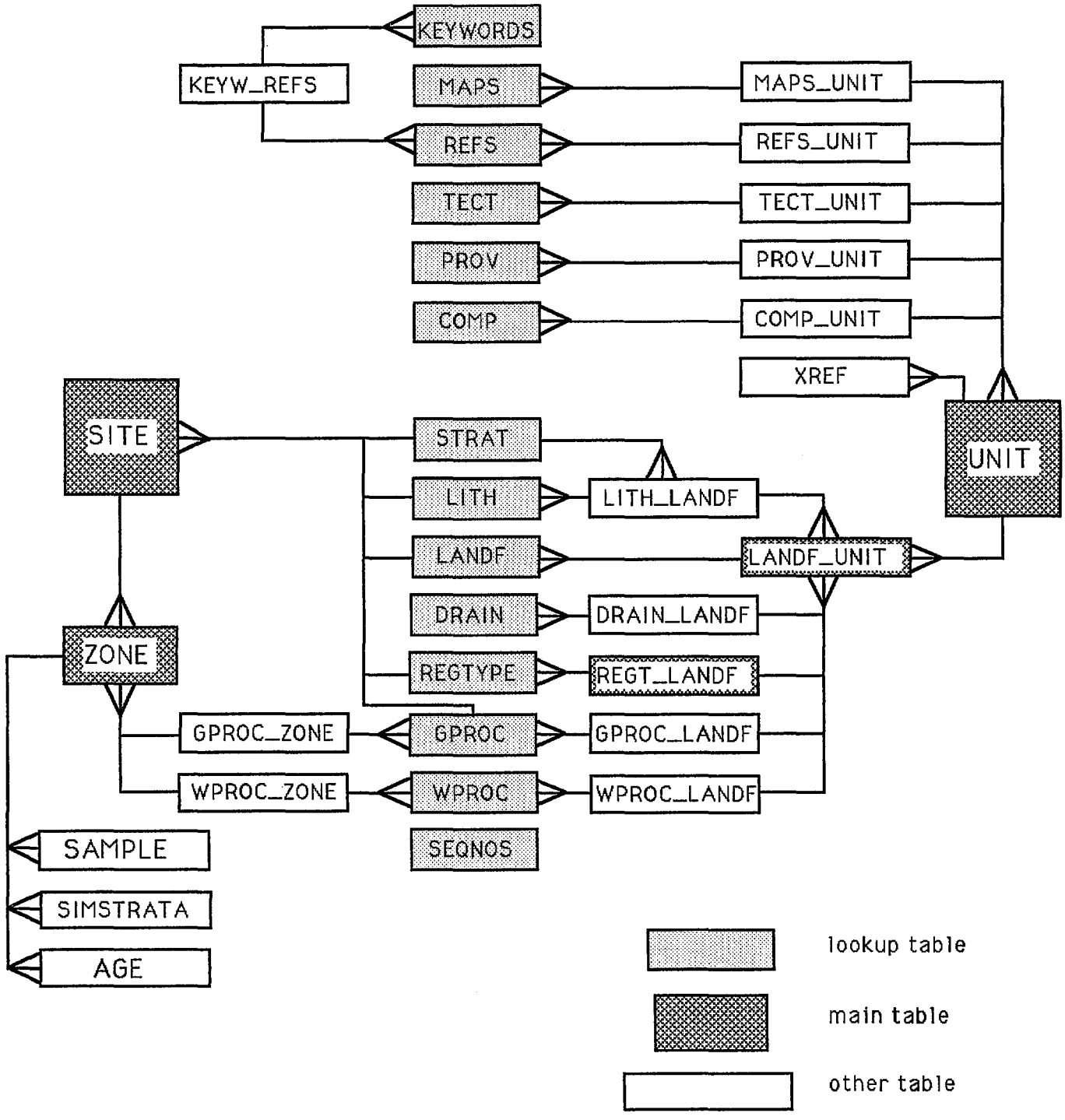
ORACLE V.5.1 Database Administrator's Guide, Oracle Corporation.

ORACLE V.5.1 Utilities User's Guide, Oracle Corporation.

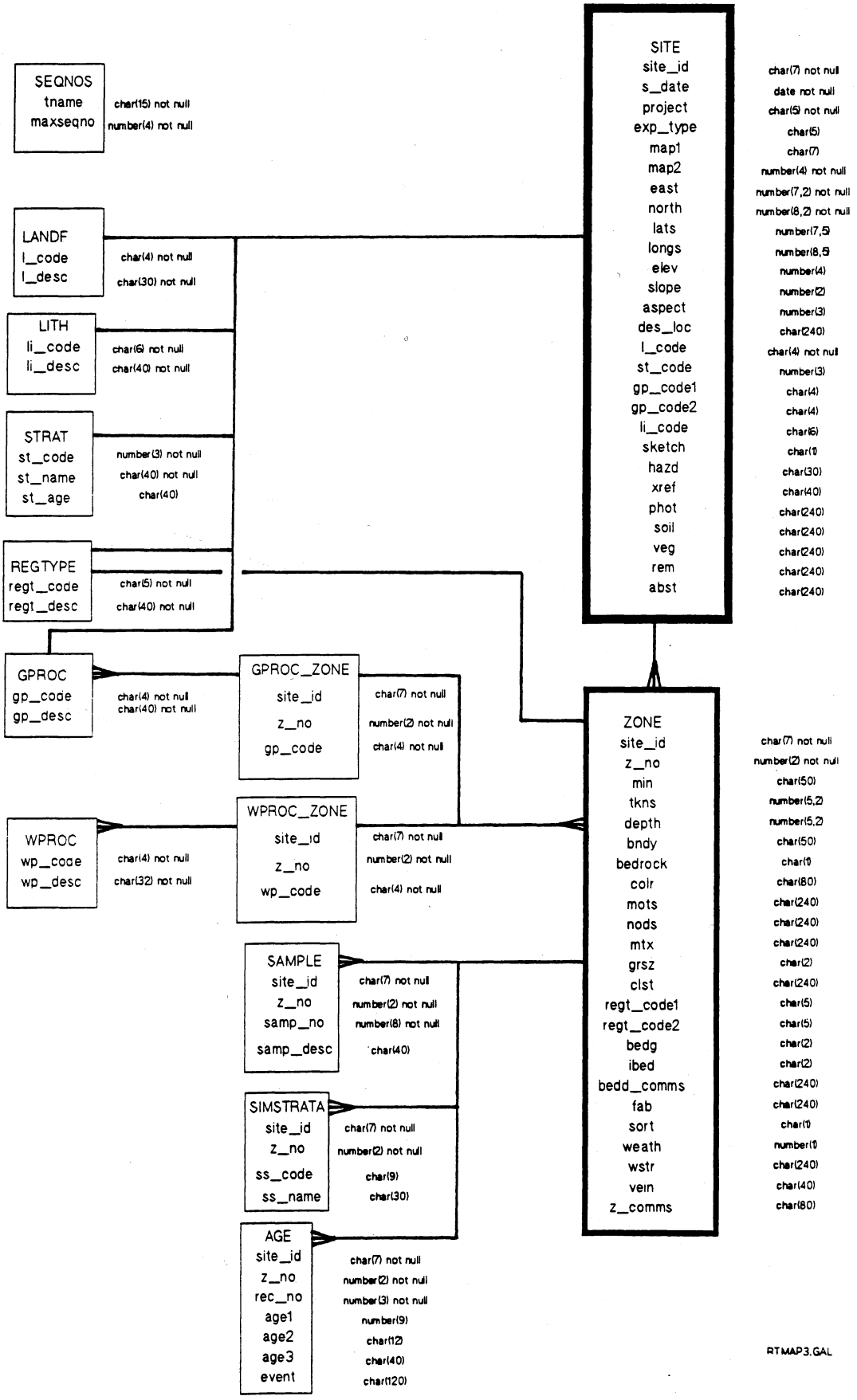
SQR Structured Query Report Writer User Guide, 1987, SQ Software.

APPENDIX A: LOGICAL DATA STRUCTURE

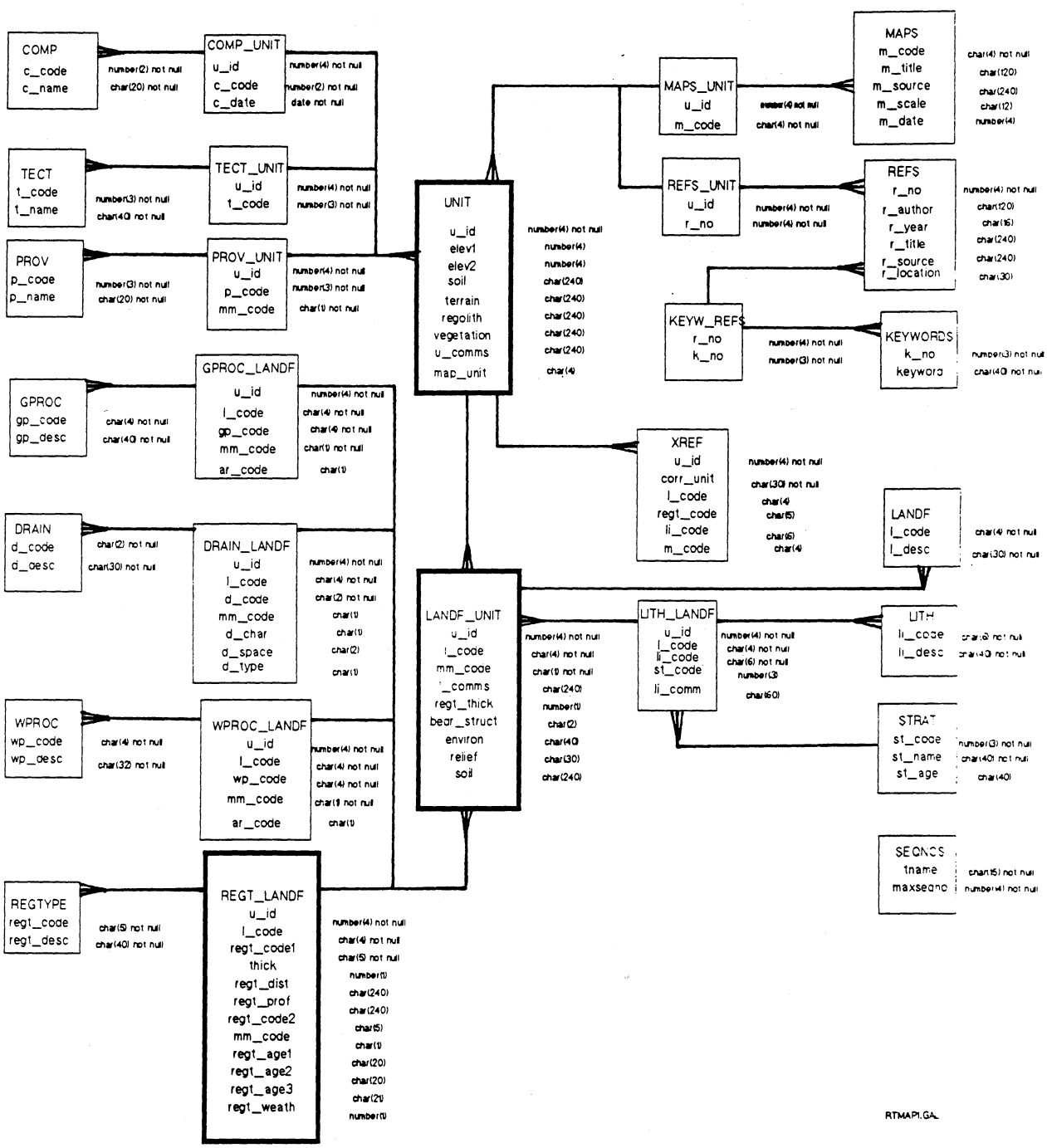
Tables



Field sites


SEQNOS
tname
maxseqno
char(15) not null
number(4) not null
LANDF
l_code
l_desc
char(4) not null
char(30) not null
LITH
li_code
li_desc
char(6) not null
char(40) not null
STRAT
st_code
st_name
st_age
number(3) not null
char(40) not null
char(40)
REGTYPE
regt_code
regt_desc
char(5) not null
char(40) not null
GPROC
gp_code
gp_desc
char(4) not null
char(40) not null
GPROC_ZONE
site_id
z_no
gp_code
char(7) not null
number(2) not null
char(4) not null
WPROC
wp_code
wp_desc
char(4) not null
char(32) not null
WPROC_ZONE
site_id
z_no
wp_code
char(7) not null
number(2) not null
char(4) not null
SAMPLE
site_id
z_no
samp_no
samp_desc
char(7) not null
number(2) not null
number(8) not null
char(40)
SIMSTRATA
site_id
z_no
ss_code
ss_name
char(7) not null
number(2) not null
char(9)
char(30)
AGE
site_id
z_no
rec_no
age1
age2
age3
event
char(7) not null
number(2) not null
number(3) not null
number(9)
char(12)
char(40)
char(120)
SITE
site_id
s_date
project
exp_type
map1
map2
east
north
lats
longs
elev
slope
aspect
des_loc
l_code
st_code
gp_code1
gp_code2
li_code
sketch
hazd
xref
phot
soil
veg
rem
abst
char(7) not null
date not null
char(5) not null
char(5)
char(7)
number(4) not null
number(7,2) not null
number(8,2) not null
number(7,5)
number(8,5)
number(4)
number(2)
number(3)
char(240)
char(4) not null
number(3)
char(4)
char(4)
char(6)
char(1)
char(30)
char(40)
char(240)
char(240)
char(240)
char(240)
char(240)
ZONE
site_id
z_no
min
tkns
depth
bndy
bedrock
colr
mots
nods
mtx
grsz
clst
regt_code1
regt_code2
bedg
ibed
bedd_comms
fab
sort
weath
wstr
vein
z_comms
char(7) not null
number(2) not null
char(50)
number(5,2)
number(5,2)
char(50)
char(1)
char(80)
char(240)
char(240)
char(240)
char(2)
char(240)
char(5)
char(5)
char(2)
char(2)
char(240)
char(240)
char(1)
number(1)
char(240)
char(40)
char(80)
RTMAP3.GAL

Mapping units



```
rem     The two most important entities within the regolith terrain database
rem     RTMAP are the regolith terrain mapping unit (UNIT) and the field site
rem     (SITE). The regolith terrain mapping unit can combine horizontally
rem     more than one terrain form (=landform type) and several regolith types.
rem     The field site occurs in one landform type and one regolith type but
rem     can be made up of several zones vertically. Therefore the tables UNIT,
rem     LANDF_UNIT, REGT_LANDF, SITE and ZONE are central to this database and
rem     all other tables relate directly or indirectly to these five.

rem     Create the tables for units, sites and zones and their indexes.

create table UNIT               (u_id           number(4) not null,
                                 map_unit       char(4),
                                 elev1          number(4),
                                 elev2          number(4),
                                 u_comms        char(240),
                                 terrain        char(240),
                                 regolith       char(240),
                                 vegetation     char(240),
                                 soil           char(240))
                                        space SPB20;
create unique index UNIT1 on UNIT (u_id);

rem     This table contains topographic information on the regolith terrain
rem     unit and descriptive information on terrain, regolith and vegetation.
rem     The values of the unit identifier (u_id) are automatically created
rem     4-digit sequential numbers.
rem     The fields elev1 and elev2 are number fields for the lower and the
rem     upper values of the elevation range.
rem     Fields terrain, regolith and vegetation are free-text fields for any
rem     information on those three aspects (including land-use in field
rem     vegetation) pertaining to the unit as a whole.
rem     Soil field is for comments or description of the the soil within the
rem     unit as a whole. It is to be used where the soil data available for
rem     the unit is not explicit enough to allow delineation between landform
rem     elements.


create table SITE               (site_id        char(7) not null,
                                 s_date         date not null,
                                 project        char(5) not null,
                                 exp_type       char(5),
                                 map1           char(7),
                                 map2           number(4) not null,
                                 east           number(7,2) not null,
                                 north          number(8,2) not null,
                                 lats           number(7,5),
                                 longs          number(8,5),
                                 elev           number(4),
                                 slope          number(2),
                                 aspect         number(3),
                                 des_loc        char(240),
                                 l_code         char(4) not null,
                                 st_code        number(3),
                                 gp_code1       char(4),
                                 gp_code2       char(4),
                                 li_code        char(6),
                                 sketch         char(1),
                                 hazd           char(30),
                                 xref           char(40),
                                 phot           char(240),
                                 soil           char(240),
                                 veg            char(240),
                                 rem            char(240),
                                 abst           char(240))
                                        space SPB30;
create unique index SITE1 on SITE (site_id);
```

```
rem     This table describes a field site and contains identifying information,
rem     locational data and descriptive information on various aspects of the
rem     field site.
rem     The site_id is a unique identifier made up of three parts: the first
rem     character gives the organisation ('Z' for BMR), the next 2 characters
rem     are the compiler's initials, the remaining 4 characters make up a
rem     number which has to be unique for each compiler.
rem     The date of data collection in the field is entered into field s_date.
rem     An abbreviation for the project name goes into field project.
rem     Field exp_type records the type of site from the following list:
rem             Code    Exposure type
rem             ----------------------------------
rem             AUGER   Auger hole (soil or otherwise)
rem             CANAL   Canal
rem             CLIFF   Cliff
rem             CORE    Core
rem             COST    Costean
rem             CUTTI   Cuttings
rem             DAM     Dam
rem             FLOAT   Float
rem             GRAVE   Gravel scrape
rem             GULLY   Gully (for gullies/washouts)
rem             MINE    Mine
rem             OUTCR   Outcrop
rem             PROSP   Prospect
rem             QUARR   Quarry
rem             RAILW   Railway
rem             ROAD    Road/highway cutting
rem             RUBBL   Rubble
rem             SOIL    Soil
rem             STREA   Stream (for creeks/rivers)
rem             TRENC   Trench
rem
rem     Fields map1 and map2 are for the 1:250,000 map (format: S55/13) and
rem     the 1:100,000 map number (4 digits).
rem     The AMG readings in metres are entered into fields east and north.
rem     Values for latitude and longitude go into fields lats and longs.
rem     Elevation in metres, slope angle in degrees (0-90) and aspect of the
rem     site in degrees (0-360) can be entered in fields elev, slope and
rem     aspect, respectively.
rem     Any description of the site location can be entered in field des_loc.
rem     A code for the landform element is entered in field l_code (out of
rem     lookup table LANDF).
rem     The code for the bedrock stratigraphic unit which underlies the
rem     regolith at this site, if known, goes into field st_code. It comes
rem     out of the lookup table STRAT.
rem     Into fields gp_code1 and gp_code2 go codes for the main and a less
rem     dominant geomorphic process. They come out of the lookup table GPROC.
rem     A code for bedrock lithology type which comes out of the lookup table
rem     LITH is entered in field li_code.
rem     If there was a sketch made of this site during field data collection,
rem     'Y' is entered in field sketch.
rem     Field hazd is for comments on environmental hazards. The following
rem     abbreviations can be used:
rem             NH      no recognised hazards
rem             AV      snow avalanche
rem             CO      coastal erosion
rem             CP      coastal progradation
rem             FF      flash flood
rem             FL      flood
rem             LA      land slide
rem             NH      no recognised hazards
rem             RO      rockfall
rem             SA      salinity
rem             SC      solution cavities
rem             SD      sand drift
rem             SO      soil erosion
rem             ST      storm surge
rem             SU      subsidence
rem             TS      tsunami
rem             VE      volcanic eruption.
```

```
rem     References to similar sites can be entered in field xref.
rem     The photos taken at this site are recorded in field phot.
rem     A description of the soil/s at this site goes into field soil.
rem     A description of the vegetation can be entered in field veg.
rem     Any other comments pertaining to the site as a whole can be input into
rem     field rem.
rem     Field abst is for an abstract of the whole site description, including
rem     brief comments about the zones.


create table ZONE               (site_id        char(7) not null,
                                 z_no           number(2) not null,
                                 min            char(50),
                                 tkns           number(5,2),
                                 depth          number(5,2),
                                 bndy           char(50),
                                 bedrock        char(1),
                                 colr           char(80),
                                 mots           char(240),
                                 nods           char(240),
                                 mtx            char(240),
                                 grsz           char(2),
                                 clst           char(240),
                                 regt_code1     char(5),
                                 regt_code2     char(5),
                                 bedg           char(2),
                                 ibed           char(2),
                                 bedd_comms     char(240),
                                 fab            char(240),
                                 sort           char(1),
                                 weath          number(1),
                                 wstr           char(240),
                                 vein           char(40),
                                 z_comms        char(80))
                                        space SPB35;
create unique index ZONE1 on ZONE (site_id,z_no);
create index ZONE2 on ZONE (tkns);
create index ZONE3 on ZONE (weath);

rem     This table contains information pertaining to a zone within a field
rem     site.
rem     The site identifier (site_id) comes out of table SITE. The zone number
rem     (z_no) is a 2-digit number in the format 01,02,03 etc., increasing with
rem     depth from surface.
rem     Any noteworthy mineralisation should go into field min using REGMAP's
rem     mineral table.
rem     Tkns is a number field for the average thickness of the zone, in the
rem     format: 3 digits, decimal point, 2 digits, given in metres.
rem     The depth of the lower boundary of the zone in metres (2 decimals
rem     allowed) goes into field depth.
rem     Field bndy is for boundary character (e.g. smooth, wavy, irregular,
rem     sharp, abrupt, clear, gradual, diffuse, weathering, conformable,
rem     angular unconformity, disconformity, paraconformity, nonconformity etc).
rem     Field bedrock will contain a 'Y' if there is fresh bedrock immediately
rem     below this zone.
rem     Field colr is for any free-text description of colour, colour changes
rem     or combinations.
rem     Comments about any mottling present, including size, abundance,
rem     contrast with surrounding material, and strength or induration can be
rem     entered in field mots.
rem     The same applies to nodules and field nods.
rem     Field mtx is a text field for a description of the matrix of the
rem     zone.
rem     Field gnsz is for particle size:  CL = clay
rem                                       SI = silt
rem                                       SA = sand (< 2 mm)
rem                                       GR = gravel (2 - 60 mm)
rem                                       CO = cobbles (60 - 200 mm)
rem                                       ST = stones (200 - 600 mm)
rem                                       BO = boulders (> 600 mm)
rem     Field clst is a text field for describing other characteristics of
rem     the > 2mm fraction, particularly whether particles are clast or matrix
```

```
rem     supported, their abundance, strength, lithology etc.
rem     Fields regt_code1 and regt_code2 are for the main regolith type and for
rem     degree and type of induration, respectively. They both come out of the
rem     lookup table REGTYPE.
rem     Field bedg describes the bedding thickness:
rem             LA = laminated          (< 1cm)
rem             VN = very thin beds     (1 - 3 cm)
rem             TN = thin beds          (3 - 10 cm)
rem             MB = medium beds        (10 - 30 cm)
rem             TK = thick beds         (30 - 100 cm)
rem             VK = very thick beds    (> 100 cm)
rem     and ibed the internal bedding:
rem             MA = massive
rem             LA = laminations
rem             XX = cross bedding
rem             BD = bidirectional bedding
rem             NG = normal graded bedding
rem             RG = reverse graded bedding
rem             HO = horizontal bedding
rem             BL = blanket bedding
rem             OT = other bedding types.
rem     Field bedd_comms is a free-text field for any comments on the bedding.
rem     Comments on the fabric (orientation, flow direction) go into field fab.
rem     Field sort describes the particle sorting: W = well sorted,
rem             M = moderately sorted, P = poorly sorted, B = bimodal sorting,
rem             U = unsorted.
rem     The degree of weathering is entered in field weath:
rem             0 = unknown
rem             1 = unweathered
rem             2 = slightly weathered
rem             3 = moderately weathered
rem             4 = highly weathered
rem             5 = very highly weathered
rem             6 = completely weathered.
rem     Comments on weathering characteristics go into field wstr.
rem     Field vein is for any comments on existing veins.
rem     Field z_comms is for any free-text additional comments pertaining
rem     to the zone.


rem     Create a table for the samples.

create table SAMPLE             (site_id        char(7) not null,
                                 z_no           number(2) not null,
                                 samp_no        number(8) not null,
                                 samp_desc      char(40))
                                        space SPB07;
create unique index SAMPLE1 on SAMPLE (samp_no);
create index SAMPLE2 on SAMPLE (site_id);

rem     This table contains all the samples taken in the field. The site
rem     identifier (site_id) and zone number (z_no) are carried over from
rem     table ZONE. The first four digits of the sample number (samp_no)
rem     comprise the BMR identification for sample numbers (2 digits for the
rem     current year and the next two digits identify the Regolith Group: 99).
rem     The next four digits must uniquely identify the sample.
rem     Any comments or description of the sample can be entered in field
rem     samp_desc.


rem     Create a table for similar strata.

create table SIMSTRATA          (site_id        char(7) not null,
                                 z_no           number(2) not null,
                                 ss_code        char(9),
                                 ss_name        char(30))
                                        space SPB05;
create unique index SIMSTRATA1 on SIMSTRATA (site_id,z_no,ss_code);

rem     This table relates a zone within a field site to zones with similar
rem     strata for comparisons. The field ss_code is a combination of site_id
rem     and z_no of the zone with similar strata.
```

```
rem     Field ss_name can contain an informal name or identifier for the
rem     similar strata.


create table AGE                (site_id        char(7) not null,
                                 z_no           number(2) not null,
                                 rec_no         number(3) not null,
                                 age1           number(9),
                                 age2           char(12),
                                 age3           char(40),
                                 event          char(120))
                                        space SPB05;
create unique index AGE1 on AGE (rec_no);

rem     This table records determined ages of distinguishable events within
rem     one zone. Fields site_id and z_no come out of the ZONE table, the
rem     unique record number rec_no is created automatically. The actual dating
rem     in millions of years goes into field age1 (3 digits, decimal point, two
rem     digits), the plus/minus range goes into field age2, and the method of
rem     dating into field age3. The dated event is described in field event.


rem     Create all intermediary tables and necessary indexes.

create table DRAIN_LANDF        (u_id           number(4) not null,
                                 l_code         char(4) not null,
                                 d_code         char(2) not null,
                                 d_char         char(1),
                                 d_type         char(1),
                                 d_space        char(2),
                                 mm_code        char(1))
                                        space SPB05;
create unique index DRAIN_LANDF1 on DRAIN_LANDF (u_id,l_code,d_code);
create index DRAIN_LANDF2 on DRAIN_LANDF (d_code);

rem     This table relates units to drainage patterns. The values of u_id,
rem     l_code and d_code come out of LANDF_UNIT and DRAIN tables.
rem     Values for d_char (drainage character) are restricted to:
rem     D = dry;        I = intermittent        P = perennial;
rem     values for d_type (drainage type) can be:
rem     N = normal      A = anticedent          S = superimposed
rem     C = captured    D = diverted            R = reversed
rem     U = underground;

rem     values for d_space (stream channel spacing) can be:
rem             AB = absent or very rare          > 2500 m
rem             SP = sparse                     1500 - 2500 m
rem             VW = very widely spaced         1000 - 1500 m
rem             WS = widely spaced               625 - 1000 m
rem             MS = moderately spaced           400 -  625 m
rem             CS = closely spaced              250 -  400 m
rem             VC = very closely spaced         150 -  250 m
rem             NU = numerous                      < 150 m

rem     The mm_code (major/minor code) can be M = major or S = subordinate,
rem     meaning the particular drainage pattern is a major/minor element in
rem     the landform within the unit.


create table COMP_UNIT          (u_id           number(4) not null,
                                 c_code         number(2) not null,
                                 c_date         date not null)
                                        space SPB05;
create unique index COMP_UNIT1 on COMP_UNIT (u_id,c_code,c_date);
create index COMP_UNIT2 on COMP_UNIT (c_code);

rem     This table relates units to compilers. Values for unit identifier
rem     (u_id) and compiler code (c_code) are taken from tables UNIT and COMP.
rem     The date (c_date) will be entered automatically.
```

```
create table TECT_UNIT          (u_id           number(4) not null,
                                 t_code         number(3) not null)
                                        space SPB05;
create unique index TECT_UNIT1 on TECT_UNIT (u_id,t_code);
create index TECT_UNIT2 on TECT_UNIT (t_code);

rem     This table relates units to tectonic structure elements. The values for
rem     unit identifier (u_id) and tectonic structure element code (t_code)
rem     come out of tables UNIT and TECT.


create table LANDF_UNIT         (u_id           number(4) not null,
                                 l_code         char(4) not null,
                                 mm_code        char(1) not null,
                                 bedr_struct    char(2),
                                 relief         char(30),
                                 environ        char(40),
                                 l_comms        char(240),
                                 soil           char(240),
                                 regt_thick     number(1))
                                        space SPB15;
create unique index LANDF_UNIT1 on LANDF_UNIT (u_id,l_code);
create index LANDF_UNIT2 on LANDF_UNIT (l_code);

rem     This table relates units to landform types. Unit identifier (u_id) and
rem     landform (= terrain form ) code (l_code) come out of tables UNIT and
rem     LANDF. The values for the major/minor code (mm_code) can be M = major
rem     or S = subordinate, depending on whether the landform is a major/minor
rem     element in the unit. The type of structural control is entered in field
rem     bedr_struct:
rem             NS = no structural control, CU = cuesta forms, DI = dip slope,
rem             SA = strike aligned, AN = anticline, SN = syncline,
rem             MN = monocline, FT = faulted, BF = block faulting,
rem             AD = anti dip slope, DB = dipping beds, JN = jointing,
rem             HG = horst/graben, HO = horizontal bedding, DS = dyke/sill.
rem     Field environ is for comments on environmental hazards. The same
rem     abbreviations as for field hazd in table SITE can be used here.
rem     The comments field (l_comms) is for free-text description of the
rem     landform within the unit, or related data.
rem     Soil is a comments/description field for the soil within the landform
rem     element.
rem     The regt_thick field refers to the total observed or assumed thickness
rem     of the regolith profile under this landform:
rem             0   unknown
rem             1   < 0.5 m
rem             2   < 2 m
rem             3   > 2 m
rem             4   > 5 m
rem             5   > 15 m
rem             6   > 50 m.
rem     Relief field is for the relief of the landform element.


create table PROV_UNIT          (u_id           number(4) not null,
                                 p_code         number(3) not null,
                                 mm_code        char(1) not null)
                                        space SPB05;
create unique index PROV_UNIT1 on PROV_UNIT (u_id,p_code,mm_code);
create index PROV_UNIT2 on PROV_UNIT (p_code);

rem     This table relates units to regolith terrain provinces. Unit identifier
rem     (u_id) and regolith terrain province code (p_code) come out of
rem     tables UNIT and PROV. The value for the major/minor code (mm_code) can
rem     be M = major or S = subordinate, depending on whether the regolith
rem     terrain unit is a major or minor element within the regolith terrain
rem     province.


create table GPROC_LANDF        (u_id           number(4) not null,
                                 l_code         char(4) not null,
                                 gp_code        char(4) not null,
                                 mm_code        char(1) not null,
```

```
                                ar_code         char(1))
                                        space SPB05;
create unique index GPROC_LANDF1 on GPROC_LANDF (u_id,l_code,gp_code,mm_code);
create index GRP0OC_LANDF2 on GPROC_LANDF (l_code);
create index GPROC_LANDF3 on GPROC_LANDF (gp_code);

rem     This table relates landforms within units to geomorphic processes. Unit
rem     and landform codes (u_id and l_code) come out of the LANDF_UNIT table
rem     and the geomorphic process code (gp_code) comes out of the table
rem     GPROC. The value for the major/minor code (mm_code) can be M = major
rem     or S = subordinate, depending on whether the geomorphic process has a
rem     major or minor part in the formation of the regolith terrain unit.
rem     ar_code refers to whether the geomorphic processes that have acted on
rem     the landform are still active (A) or are relict (R).

create table WPROC_LANDF        (u_id           number(4) not null,
                                 l_code         char(4) not null,
                                 wp_code        char(4) not null,
                                 ar_code        char(1),
                                 mm_code        char(1) not null)
                                        space SPB05;
create unique index WPROC_LANDF1 on WPROC_LANDF (u_id,l_code,wp_code,mm_code);
create index WPROC_LANDF2 on WPROC_LANDF (l_code);
create index WPROC_LANDF3 on WPROC_LANDF (wp_code);

rem     This table relates weathering processes to landforms within a unit.
rem     The values for u_id and l_code come out of table LANDF_UNIT and the
rem     code for weathering type comes out of the lookup table WPROC. The value
rem     for major/minor code (mm_code) can be M = major or S = subordinate,
rem     depending on whether the weathering process has a major or minor part
rem     in the formation of the regolith terrain unit. The ar_code refers to
rem     whether the weathering processes acting on the landform are active (A)
rem     or relict (R).


create table REGT_LANDF         (u_id           number(4) not null,
                                 l_code         char(4) not null,
                                 regt_code1     char(5) not null,
                                 thick          number(1),
                                 regt_dist      char(240),
                                 regt_prof      char(240),
                                 regt_code2     char(5),
                                 regt_age1      char(20),
                                 regt_age2      char(20),
                                 regt_age3      char(21),
                                 mm_code        char(1),
                                 regt_weath     number(1))
                                        space SPB15;
create unique index REGT_LANDF1 on REGT_LANDF (u_id,l_code,regt_code);
create index REGT_LANDF2 on REGT_LANDF (l_code);

rem     This table relates regolith types to landforms within a unit. The
rem     values for unit and landform codes (u_id and l_code) come out of
rem     the LANDF_UNIT table and the regolith type code (regt_code) comes out
rem     of the REGTYPE lookup table. The regt_code is used to select the type
rem     of induration (regt_code2) and the main regolith type (regt_code1).
rem     The range of regolith depth within the unit is recorded in field thick:
rem                     0       unknown
rem                     1       < 0.5 m
rem                     2       < 2 m
rem                     3       > 2 m
rem                     4       > 5 m
rem                     5       > 15 m
rem                     6       > 50 m
rem     The regt_dist field is for a description of the regolith type's lateral
rem     distribution in the landscape.
rem     The regt_prof field is for a description of the profile position of the
rem     regolith type.
rem     The regt_weath field is for entering the degree of weathering of the
rem     sedimentary and volcanic regolith types. The degree of weathering is
rem     entered as a number code as follows:
rem                     0 = unknown degree of weathering
```

```
rem                        1 = unweathered
rem                        2 = slightly weathered
rem                        3 = moderately weathered
rem                        4 = highly weathered
rem                        5 = very highly weathered
rem                        6 = completely weathered
rem     Regt_age1 is for informal ages. It can be for lower age in a range or if
rem     there is no age range then the single age is entered in this field.
rem     Regt_age2 is is for the upper value in the age range.
rem     Regt_age3 is for comments about the source of the date, eg dating
rem     technique or inference.

create table LITH_LANDF          (u_id           number(4) not null,
                                  l_code         char(4) not null,
                                  li_code        char(6) not null,
                                  st_code        number(3),
                                  li_comm        char(60))
                                        space SPB02;
create unique index LITH_LANDF1 on LITH_LANDF (u_id,l_code,li_code,st_code);
create index LITH_LANDF2 on LITH_LANDF (l_code);

rem     This table relates lithology and stratigraphy to the landform type
rem     within the unit.
rem     Fields u_id and l_code come out of the LANDF_UNIT table.
rem     Fields li_code and st_code come out of lookup tables LITH and STRAT.
rem     The li_comm field is a descriptive that further defines the
rem     lithology.


create table XREF                (u_id           number(4) not null,
                                  corr_unit      char(30) not null,
                                  l_code         char(4),
                                  regt_code      char(5),
                                  li_code        char(6),
                                  m_code         char(4))
                                        space SPB05;
create unique index XREF1 on XREF (u_id,corr_unit,l_code,regt_code,li_code,
                                   m_code);
create index XREF2 on XREF (u_id);
create index XREF3 on XREF (m_code);

rem     This table is for cross-referencing units with corresponding units in
rem     other maps/publications. If this only refers to one landform, regolith
rem     type or lithology type within the unit the corresponding codes are
rem     entered in fields l_code, regt_code or li_code (lookup tables LANDF,
rem     REGTYPE and LITH). The value for u_id comes out of table UNIT.
rem     Field corr_unit is for a description of the corresponding unit or its
rem     name.
rem     Field m_code is for the map code of the map containing the unit, and
rem     comes out of lookup table MAPS.


create table GPROC_ZONE          (site_id        char(7) not null,
                                  z_no           number(2) not null,
                                  gp_code        char(4) not null)
                                        space SPB05;
create unique index GPROC_ZONE1 on GPROC_ZONE (site_id,z_no,gp_code);

rem     This table relates geomorphic processes to zones within a field site.
rem     The values for the site identifier (site_id) and the zone number (z_no)
rem     are copied across from the ZONE table; the code for the geomorphic
rem     process comes out of the lookup table GPROC.


create table WPROC_ZONE          (site_id        char(7) not null,
                                  z_no           number(2) not null,
                                  wp_code        char(4) not null)
                                        space SPB05;
create unique index WPROC_ZONE1 on WPROC_ZONE (site_id,z_no,wp_code);

rem     This table relates weathering processes to zones within a field site.
rem     The values for the site identifier (site_id) and the zone number (z_no)
```

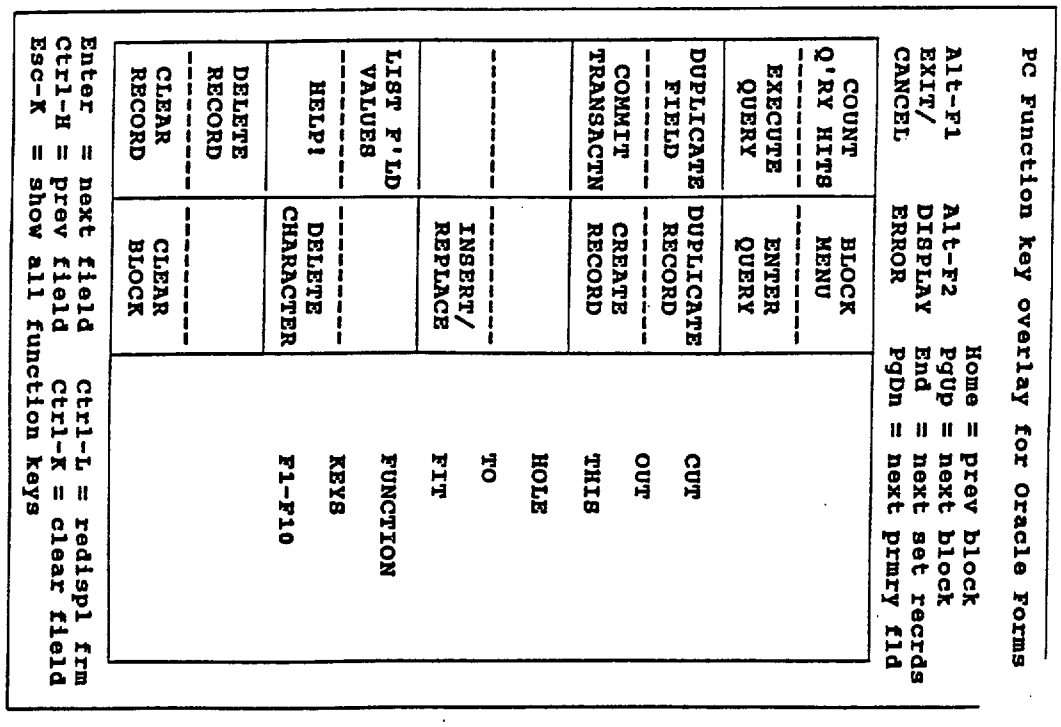
```
rem     are copied across from the ZONE table; the code for the weathering
rem     process comes out of the lookup table WPROC.


create table MAPS_UNIT          (u_id           number(4) not null,
                                 m_code         char(4) not null)
                                        space SPB05;
create unique index MAPS_UNIT1 on MAPS_UNIT (u_id,m_code);

rem     This table relates units to the map references. The values for u_id
rem     and m_code come out of tables UNIT and MAPS.

create table REFS_UNIT          (u_id           number(4) not null,
                                 r_no           number(4) not null)
                                        space SPB05;
create unique index REFS_UNIT1 on REFS_UNIT (u_id,r_no);

rem     This table relates units to all other references. The values for u_id
rem     and r_no come out of tables UNIT and REFS.


create table KEYW_REFS          (r_no           number(4) not null,
                                 k_no           number(3) not null)
                                        space SPB02;
create unique index KEYW_REFS1 on KEYW_REFS (r_no,k_no);
create index KEYW_REFS2 on KEYW_REFS (r_no);
create index KEYW_REFS3 on KEYW_REFS (k_no);

rem     This table relates keywords to references.
rem     Fields r_no and k_no come out of the tables REFS and KEYWORDS.


rem     Create all the lookup tables and any necessary indexes (most tables
rem     will contain less than 100 records and therefore it is not deemed
rem     necessary to index them).


create table DRAIN      (d_code         char(2) not null,
                         d_desc         char(30) not null)
                                space SPB02;

rem     This table contains a list of drainage patterns.
rem     The values of d_code are in upper case.
rem     Field d_desc is for a full description of the drainage pattern.
rem     Source: Speight, J.G. 1984: Landform. In: Australian Soil and Land
rem     Survey Field Handbook, McDonald et al. (Eds), Inkarta Press, Mel-
rem     bourne. (Some additions).


create table COMP       (c_code         number(2) not null,
                         c_name         char(20) not null)
                                space SPB05;

rem     This table lists all the people working as map compilers or field
rem     observers on regolith terrain mapping projects and it will grow as the
rem     projects increase in numbers. Compiler names are in the format:
rem     surname in upper/lower case, comma, space, initials, comma, space,
rem     affiliation (e.g. Pain, C F, BMR).


create table TECT       (t_code         number(3) not null,
                         t_name         char(40) not null)
                                space SPB02;

rem     This table lists 93 tectonic structure elements (after
rem     Palfreyman, 1984).


create table LANDF      (l_code         char(4) not null,
                         l_desc         char(30) not null)
                                space SPB02;
```

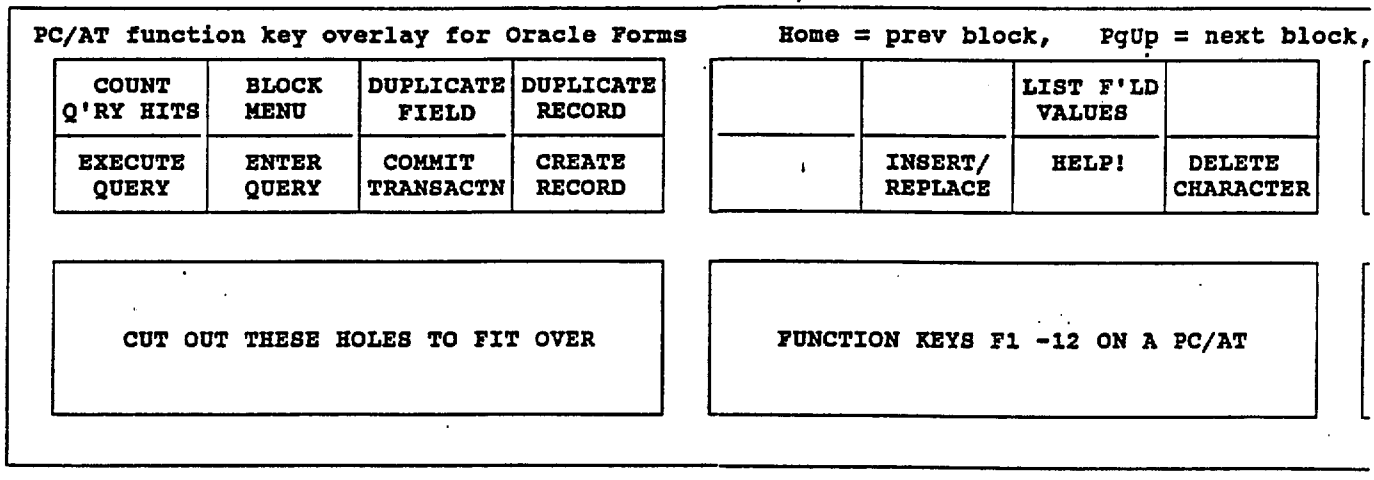
```
rem     This table contains a list of landform patterns according to Speight.
rem     Landform patterns are areas more than 600m across, and are made up of
rem     landform elements. Values for l_code are 2 upper case letters and 2
rem     digits.
rem     Source: Speight, J.G. 1984: Landform. In: Australian Soil and Land
rem     Survey Field Handbook, McDonald et al. (Eds), Inkarta Press, Melbourne.
rem     (Some additions).

create table PROV       (p_code         number(3) not null,
                         p_name         char(20) not null)
                                space SPB05;
create unique index PROV_NAME on PROV (p_name);

rem     This table will list all the regolith terrain provinces, 30 to begin
rem     with. It will have a much faster growth rate than most of the other
rem     lookup tables.

create table GPROC      (gp_code        char(4) not null,
                         gp_desc        char(40) not null)
                                space SPB02;

rem     This table gives a list of the main geomorphic processes which
rem     contribute to the present-day regolith terrains. The values for
rem     gp_code are 2 upper case letters and 2 digits.
rem     Source: Speight, J.G. 1984: Landform. In: Australian Soil and Land
rem     Survey Field Handbook, McDonald et al. (Eds), Inkarta Press,
rem     Melbourne. (Some additions).

rem     The existing tables of 1:250 000 and 1: 100 000 map sheet names and
rem     numbers called QMAPS and HMAPS which are publicly available can be
rem     used as lookup tables for the RTMAP database.

create table STRAT              (st_code        number(3) not null,
                                 st_name        char(40) not null,
                                 st_age         char(40))
                                        space SPB05;
create unique index STRAT1 on STRAT (st_name);
create unique index STRAT2 on STRAT (st_code);

rem     This is the table for stratigraphic names. The stratigraphic name code
rem     (st_code) is a system-generated sequential number. Field st_name is
rem     a character field for the fully spelled-out stratigraphic names. Care
rem     should be taken to get the spelling of stratigraphic names right
rem     according to the guidelines of the Stratigraphic Nomenclature
rem     Committee. The age of the stratigraphic unit in the format: Devonian,
rem     Lower goes into field st_age.

create table LITH               (li_code        char(6) not null,
                                 li_desc        char(40) not null)
                                        space SPB02;

rem     This table lists the main types of bedrock lithology. The lithology
rem     code (li_code) values are 2 upper case letters and 2 digits.
rem     Field li_desc contains the lithology type.
rem     Source: Speight, J.G. and R.F. Isbell 1984: Substrate Material.
rem     In: Australian Soil and Land Survey Field Handbook, McDonald et al.
rem     (Eds), Inkarta Press, Melbourne. (Some additions - PETCHEM).

create table REGTYPE            (regt_code      char(5) not null,
                                 regt_desc      char(40) not null)
                                        space SPB02;

rem     This table contains a list of regolith types. Regolith type codes
rem     (regt_code) are 3 upper case letters and 2 digits.
rem     Source: Speight, J.G. and R.F. Isbell in press: Substrate Material.
rem     In: Australian Soil and Land Survey Field Handbook, 2nd Edition,
```

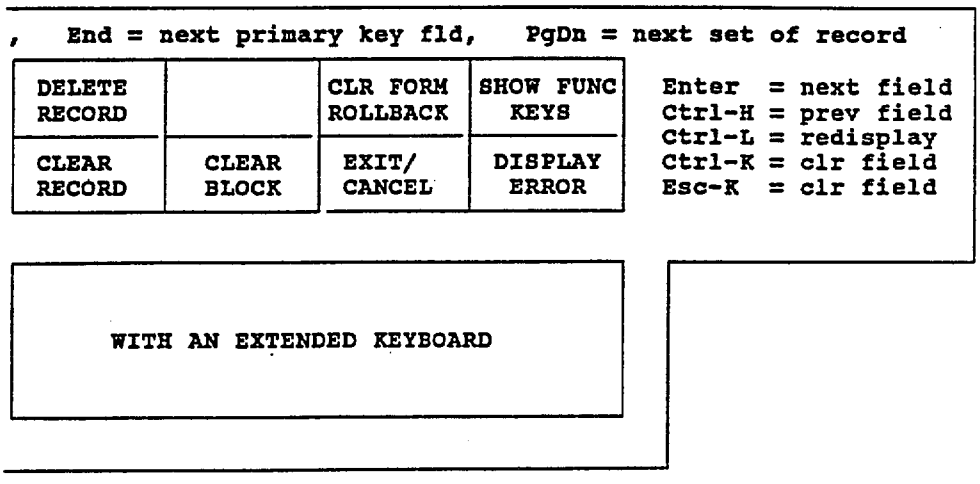
```
rem     McDonald et al. (Eds), Inkarta Press, Melbourne. (Some additions).

create table WPROC              (wp_code        char(4) not null,
                                 wp_desc        char(32) not null)
                                        space SPB02;

rem     This table contains contains a list of weathering processes. The values
rem     for the weathering process code (wp_code) are 2 upper case letters and
rem     2 digits.
rem     Source: Taken largely from Ollier, C.D. 1984: Weathering, 2nd Edition,
rem     Longman.

create table KEYWORDS           (k_no           number(3) not null,
                                 keyword        char(40) not null)
                                        space SPB07;
create unique index KEYWORD1 on KEYWORDS (k_no);
create unique index KEYWORD2 on KEYWORDS (keyword);

rem     This table contains a listing of useful keywords for referencing the
rem     references contained in the REFS table.
rem     Source: Thesaurus of earth sciences and related terms, Australian
rem     Mineral Foundation.

create table REFS               (r_no           number(4) not null,
                                 r_author       char(120),
                                 r_year         char(16),
                                 r_title        char(240),
                                 r_source       char(240),
                                 r_location     char(30))
                                        space SPB10;
create unique index REFS1 on REFS (r_no);
create index REFS2 on REFS (r_author);

rem     This table contains all the references used for regolith terrain
rem     mapping projects. The reference number (r_no) is a system-generated
rem     sequential number. Field r_author contains the name of the author/s
rem     in the format: surname in upper/lower case, comma, initials in upper
rem     case with full stops. Field r_year is a character field to also
rem     accommodate comments like 'in press'; year of publication is entered
rem     as a 4-digit number.
rem     Fields r_title and  r_source are character fields. Journal names,
rem     publishers, and volume and part numbers should be entered in r_source.

create table MAPS               (m_code         char(4) not null,
                                 m_title        char(120),
                                 m_scale        char(12),
                                 m_source       char(240),
                                 m_date         number(4))
                                        space SPB07;
create unique index MAPS1 on MAPS (m_code);
create index MAPS2 on MAPS (m_title);

rem     This table contains all the maps used for regolith terrain mapping
rem     projects. Values for m_code consist of a letter symbol identifying the
rem     category of map used (T = topographic; S = soils; G = geologic;
rem     V = vegetation; L = land systems; O = other) plus a sequential number
rem     within each category. The fields m_title, m_scale, m_source and m_date
rem     are for map title, scale, author/s, and year of publication,
rem     respectively. If the year of publication is unknown, 9999 is input.

rem     Create a table for generating sequential numbers:

create table SEQNOS             (tname          char(15) not null,
                                 maxseqno       number(4) not null)
                                        space SPB02;
```

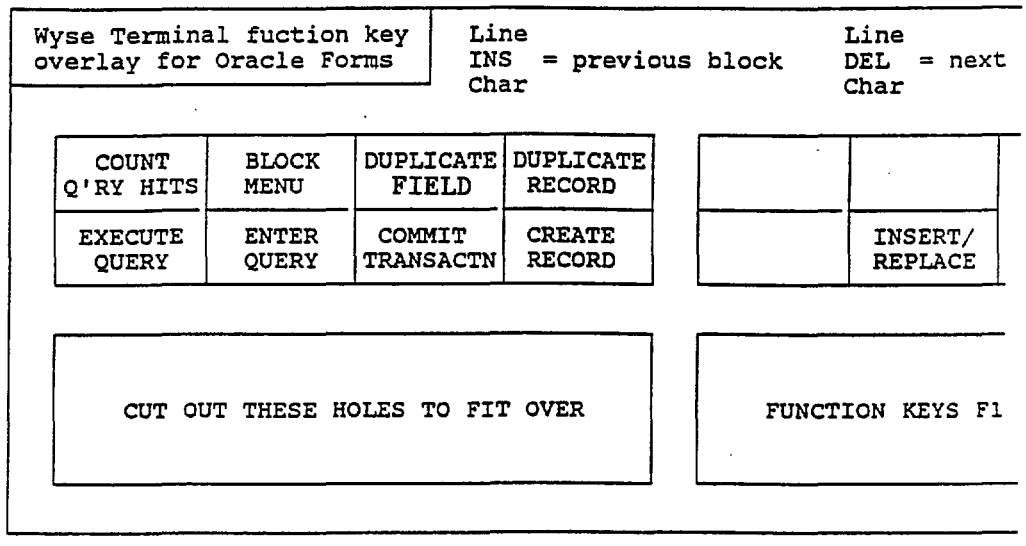
```
rem     For every table that contains a sequential number field whose value is
rem     to be generated automatically, an entry must be made into this table.
rem     Field tname contains the table name, maxseqno the maximum sequential
rem     number for that table. Triggers to increment the values in maxseqno
rem     must be set up in the entry forms.
rem     Following is a list of the tables and fields which need entries in the
rem     SEQNOS table:
rem             UNIT            u_id
rem             COMP            c_code
rem             TECT            t_code
rem             PROV            p_code
rem             REFS            r_no
rem             STRAT           st_code
rem             KEYWORDS        k_no
rem             AGE             rec_no.
rem     The two-step pre-insert block trigger for sequential number generation
rem     in SQL*Forms is:
rem     1. update SEQNOS set maxseqno = maxseqno + 1 where tname = 'COMP'
rem     2. select maxseqno into :c_code from SEQNOS where tname = 'COMP'
```

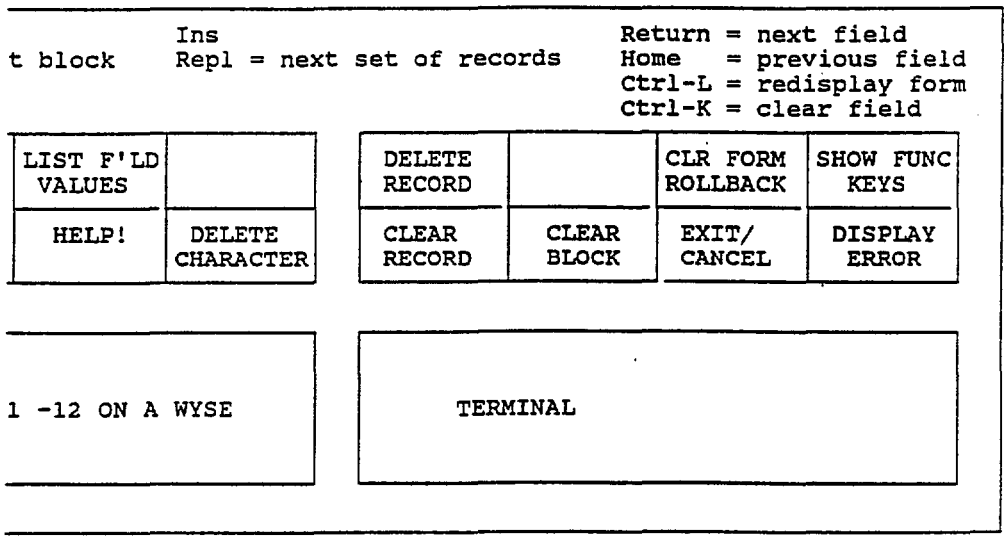
APPENDIX C: KEYBOARD OVERLAYS FOR ORACLE

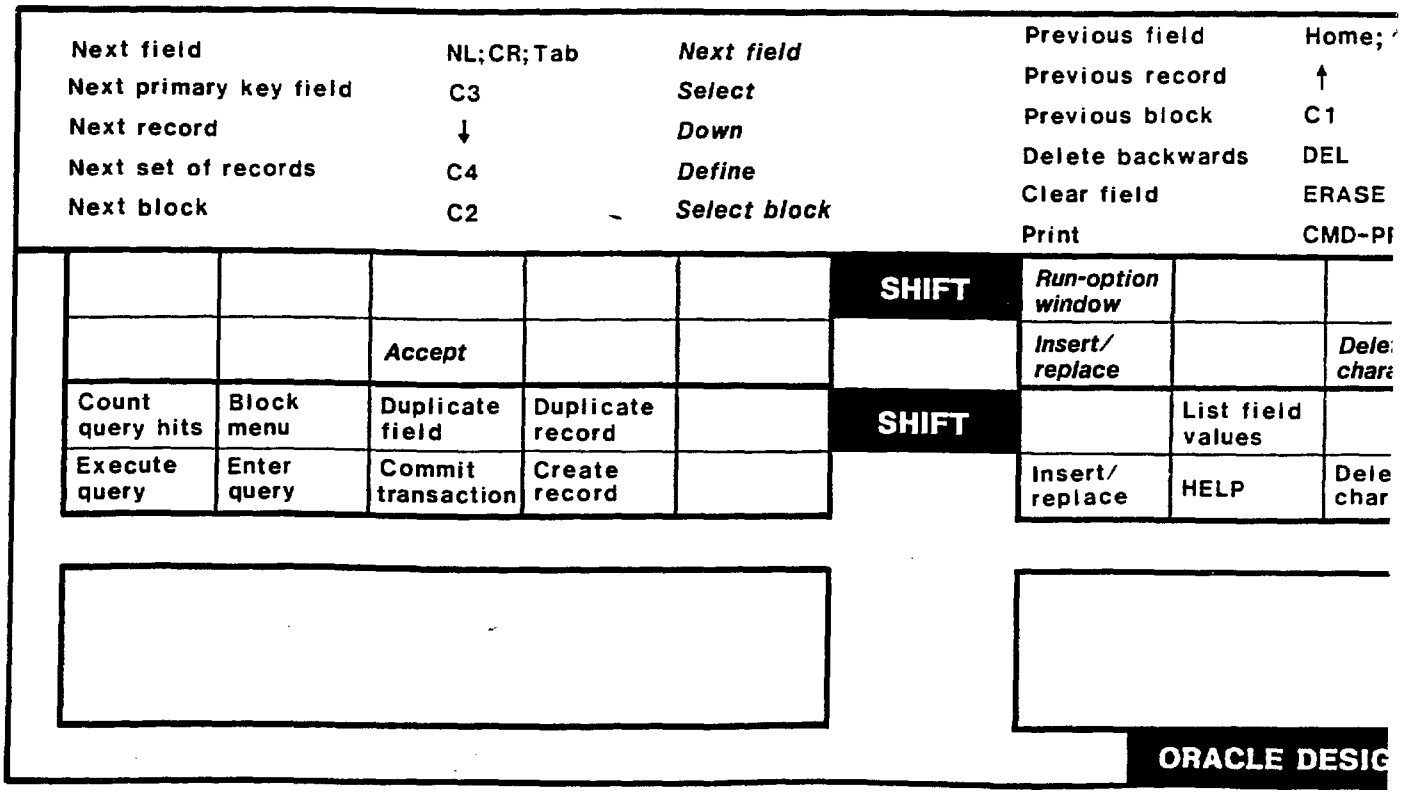


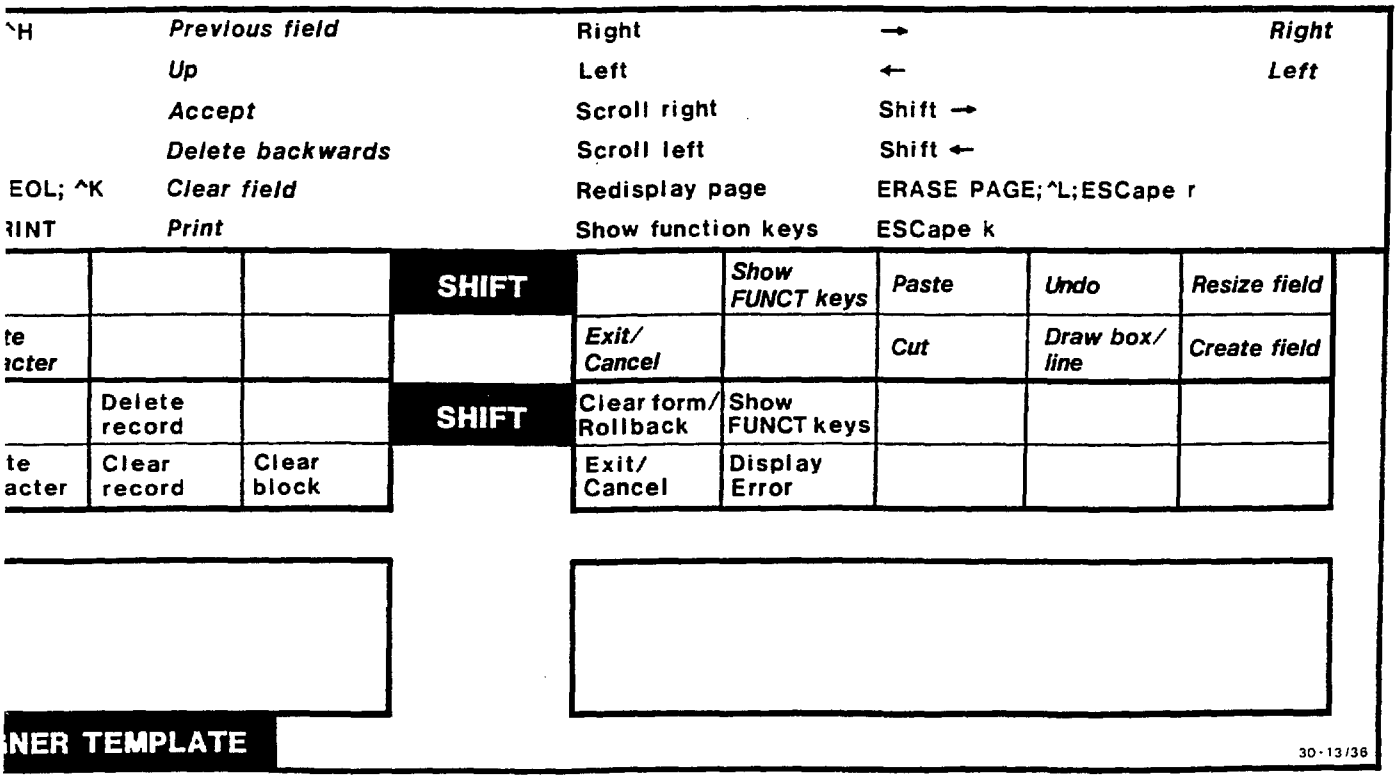





Wyse Terminal fuction key overlay for Oracle Forms
Line INS Char = previous block
Line DEL Char = next
COUNT Q'RY HITS
BLOCK MENU
DUPLICATE FIELD
DUPLICATE RECORD
EXECUTE QUERY
ENTER QUERY
COMMIT TRANSACTN
CREATE RECORD
INSERT/ REPLACE
CUT OUT THESE HOLES TO FIT OVER
FUNCTION KEYS F1


t block
Ins
Repl = next set of records
Return = next field
Home = previous field
Ctrl-L = redisplay form
Ctrl-K = clear field
LIST F'LD VALUES
HELP!
DELETE CHARACTER
DELETE RECORD
CLEAR RECORD
CLEAR BLOCK
CLR FORM ROLLBACK
EXIT/ CANCEL
SHOW FUNC KEYS
DISPLAY ERROR
1 -12 ON A WYSE
TERMINAL


Next field
NL;CR;Tab
Next field
Next primary key field
C3
Select
Next record
↓
Down
Next set of records
C4
Define
Next block
C2
Select block
Previous field
Home;
Previous record
↑
Previous block
C1
Delete backwards
DEL
Clear field
ERASE
Print
CMD-P
SHIFT
Run-option window
Insert/ replace
Accept
Count query hits
Block menu
Duplicate field
Duplicate record
Execute query
Enter query
Commit transaction
Create record
SHIFT
List field values
Insert/ replace
HELP
ORACLE DESIG


^H
Previous field
Right
→
Right
Up
Left
←
Left
Accept
Scroll right
Shift →
Delete backwards
Scroll left
Shift ←
EOL; ^K
Clear field
Redisplay page
ERASE PAGE; ^L; ESCape r
RINT
Print
Show function keys
ESCape k
SHIFT
Show FUNCT keys
Paste
Undo
Resize field
Exit/ Cancel
Cut
Draw box/ line
Create field
Delete record
SHIFT
Clear form/ Rollback
Show FUNCT keys
Clear record
Clear block
Exit/ Cancel
Display Error
NER TEMPLATE
30-13/36

```
rem     This file called EXAMPLE1.SQL is an example of an SQL retrieval.
rem     To run it from within SQL*Plus, type START EXAMPLE2 at the SQL prompt.

rem     Set up the report format:

set pagesize 66
set linesize 132
column site_id heading 'Site ID'
column s_date  heading 'Date'
column project heading 'Project'            format a7
column map2    heading 'Map#'
column l_desc  heading 'Landform'           format a20
column gp_desc heading 'Geomorphic process' format a40
column li_desc heading 'Lithology'          format a30

rem     Retrieve the data and send it to a file called EXAMPLE1.LIS:

spool EXAMPLE1.LIS
select  site_id,s_date,project,map2,l_desc,gp_desc,li_desc
from    SITE,LANDF,LITH,GPROC
        where site_id between 'ZCP0001' and 'ZCP0010'
        and   SITE.l_code = LANDF.l_code
        and SITE.gp_code1 = GPROC.gp_code
        and  SITE.li_code = LITH.li_code
order by site_id;
spool off
```

Site ID	Date	Project	Map#	Landform	Geomorphic process	Lithology
ZCP0002	10-JUL-90	N.Qld	7472	flood plain	channelled stream flow	sandstone
ZCP0003	10-JUL-90	N.Qld	7472	plateau	sheet flow,sheet wash,surface wash	siltstone
ZCP0004	11-JUL-90	N.Qld	7471	flood plain	channelled stream flow	sandstone
ZCP0005	11-JUL-90	N.Qld	7471	erosional plain	sheet flow,sheet wash,surface wash	sandstone
ZCP0006	11-JUL-90	N.Qld	7471	erosional plain	channelled stream flow	sandstone
ZCP0007	11-JUL-90	N.Qld	7471	erosional plain	channelled stream flow	sandstone
ZCP0008	11-JUL-90	N.Qld	7471	hills	sheet flow,sheet wash,surface wash	sandstone
ZCP0009	11-JUL-90	N.Qld	7471	flood plain	channelled stream flow	sandstone
ZCP0010	11-JUL-90	N.Qld	7471	plateau	sheet flow,sheet wash,surface wash	sandstone

9 records selected.

APPENDIX E: EXAMPLE OF AN SQR REPORT AND ITS OUTPUT

```
!       This file called EXAMPLE2.SQR retrieves summary information on
!       specified regolith terrain mapping units.

!       To run the report, type SQR EXAMPLE2 username/password (your Oracle
!       username/password combination).

!       Written by S. Lenz, BMR Information Systems Branch, 10.07.1991

!       Specify the report format:

begin-setup
  page-size 1000 120
end-setup

!       This is the body of the report which calls the main procedure:

begin-report
  do print_all
end-report

!       Specify the page heading:

begin-heading 2
  print 'Unit'                    (+1,1)
  print 'Regolith description'   (0,6)
  print 'Terrain description'    (0,32)
  print 'Landforms'              (0,58)
  print 'Bedrock lithology'      (0,85)
  print 'Regolith/Indur'         (0,105)
end-heading

!       The main procedure calls sub-procedures reflecting the one-to-many
!       relationships between the unit and landforms etc.

begin-procedure print_all
  columns 1 6 32 58 85 105              !       Specify column sizes.

  move 0 to #topcount                   !       These counters keep track of
  move 0 to #botcount                   !       where to print within columns.
  move 0 to #col4a                      !       They have to be initialized at
  move 0 to #bot2                       !       the beginning of a run.

begin-select
u_id
  use-column 1
map_unit                          (+2,1)
  move #current-line to #topcount
  use-column 2
regolith                          (0,1) wrap 24 12
  move #current-line to #botcount
  position (#topcount)
  use-column 3
  position (-2)
terrain                           (0,1) wrap 24 12
  if #current-line > #botcount
    move #current-line to #botcount
  end-if
  do landforms

  position (#botcount)
  position (-2)

from unit
where u_id between 1 and 3
order by map_unit

end-select
end-procedure
```

```
begin-procedure landforms

begin-select
  use-column 4
  if #topcount > #col4a
    position (#topcount)
    position (-3)
  else
    position (#bot2)
    position (-1)
  end-if
landf_unit.u_id
l_code
  do description                          !       this is a sub-procedure
  move #current-line to #col4a
regt_thick
  evaluate &regt_thick
    when = 0
    print 'unknown thickness'    (+1,1)
    when = 1
    print '< 0.5m thick'         (+1,1)
    when = 2
    print '< 2m thick'           (+1,1)
    when = 3
    print '> 2m thick'           (+1,1)
    when = 4
    print '> 5m thick'           (+1,1)
    when = 5
    print '> 15m thick'          (+1,1)
    when = 6
    print '> 50m thick'          (+1,1)
  end-evaluate
relief                           (+1,1)
  move #current-line to #bot2

  if #current-line > #botcount
    move #current-line to #botcount
  end-if
  use-column 5
  position (#col4a)
  position (-3)
  do lith                                 !       another sub-procedure
  if #current-line > #botcount
    move #current-line to #botcount
  end-if
  use-column 6
  position (#col4a)
  position (-3)
  do regtypes                             !       another sub-procedure
  if #current-line > #botcount
    move #current-line to #botcount
  end-if

from landf_unit
where landf_unit.u_id = &u_id
order by l_code
end-select
  if #current-line > #botcount
    move #current-line to #botcount
  end-if
end-procedure
```

```
begin-procedure lith

begin-select
lith_landf.u_id
lith_landf.l_code
lith.li_code
  print '*'                          (+1,1)
li_desc                             (0,0) wrap 18 2

from lith_landf,lith
where lith_landf.li_code = lith.li_code
and lith_landf.u_id = &u_id
and lith_landf.l_code = &l_code

end-select
  if #current-line > #bot2
    move #current-line to #bot2
  end-if
end-procedure

begin-procedure regtypes

begin-select
regt_code1                          (+1,1)
  print ','                         (0,0)
regt_code2                          (0,+1)

from regt_landf
where regt_landf.u_id = &u_id
and regt_landf.l_code = &l_code
order by regt_code1

end-select
  if #current-line > #bot2
    move #current-line to #bot2
  end-if
end-procedure

begin-procedure description

begin-select
l_desc                              (+1,1)
from landf
where landf.l_code = &l_code

end-select
end-procedure
```

Unit	Regolith description	Terrain description	Landforms	Bedrock lithology	Regolith/Indur
Bp1	Less than 0.5m residual soils on rises, calcareous, powdery to clayey; abundant large lmst frags; some calcrete nodules. 1-3 m colluvial calcareous clay in depressions; lmst & calcrete cobbles; often gilgai structure.	Flat to gently undulating plain with rises & depressions. Parellel chains of elongated or circular hummocky clay flats in sth. Large shallow depressions (dongas) & numerous small scattered claypans in nth. Some rock holes & blow holes.	etchplain < 0.5m thick 6m occasionally 9m karst > 2m thick 3m max in northern dongas.	*sandstone *limestone *calcarenite	WIR21, IDS20 WIR21, WIR24, SDC00, WIR21, IDS20 WIR21,
Bs1	Shallow, often stoney soils on bedrock; minor colluvium and alluvium.	Prominent linear rocky ridges. Trending east north east in west, north east in east.	hills unknown thickness 120 m	*gneiss *metagabbro *mafic/ultramafic intrusive *granulite *mafic/ultramafic intrusive	SDA10, SDC00, WIR24,
Sp1	Evaporite interbedded with clay & sand in playas. Sand,silt & gypsum in stabilized dunes adjacent to playas. Associated colluvial & alluvial deposits between & fringing playas. Variable subsurface & pavement duricrust.Deep weathered bedrock	Partially infilled palaeodrainage valleys with playas (often with saline lakes), claypans, kopi dunes & sand dunes & lunettes (preferentially on eastern side of playas), & fringing flats.	playa plain unknown thickness 10 m	*mafic/ultramafic intrusive *mafic igneous extrusive *metasediment *banded iron formation *sandstone *metasediment *sandstone *siliciclastic sedimentary rocks *calcarenite *sedimentary rocks, chemical/biogenic *sandstone *siliciclastic sedimentary rocks *regional metamorphic rock *granite	EVA01, EVA02, IDU00, SDA00, SDC00, SDE01, SDE03, SDL00, WMU00,