AGSC

AGSO Library's World Wide Web Catalogue: development and maintenance

John Newton, Rod Ryburn and Lynton Bond

AGSO Record 1997/50

AGSO

AUSTRALIAN GEOLOGICAL SURVEY ORGANISATION

Australian Geological Survey Organisation

AGSO Record 1997/50

AGSO Library's World Wide Web Catalogue : development and maintenance

John Newton, Rod Ryburn and Lynton Bond

DEPARTMENT OF PRIMARY INDUSTRIES AND ENERGY

Minister for Primary Industries and Energy: Hon. J. Anderson, M.P.
Minister for Resources and Energy: Senator the Hon. W. R. Parer
Secretary: Paul Barratt

AUSTRALIAN GEOLOGICAL SURVEY ORGANISATION

Executive Director: Neil Williams

ISSN: 1039-0073
ISBN: 0 642 27315 4

CONTENTS

Table of Figures

ABSTRACT

This Record describes the development and use of the AGSO Library's catalogue accessible from the AGSO World Wide Web site. The Library's in-house catalogue of books and serials is maintained on the Datatrek Professional Series library management system and access to it was only available in the Library itself. Also, at the time of development the Datatrek system provided no access, either directly or indirectly, from the Internet. It was determined that, in the interests of making the Library catalogue more readily available to AGSO's staff and to AGSO's clients, a version of the Datatrek catalogue would be developed as an Oracle database and that that version would be made available on the World Wide Web (WWW) of the Internet with a forms-based search interface that could be used by any Web browser.

Specifications for the tender to provide the WWW catalogue required the development of an Oracle database, a program to convert data in the Datatrek catalogue into a form which could be inserted into the Oracle database tables, and a SQL querying and reporting function with a WWW browser interface.

Although the Datatrek system uses a proprietary database structure, it allows for importing and exporting in USMARC format - a standard format for the representation and communication of bibliographic and related information in machine readable form. Exporting all or part of the Datatrek catalogue database is a simple process and this, combined with a specially written Microsoft Visual Basic program (MARCvrt), Oracle SQL scripts for processing the converted data, and SQL querying and reporting from/to the WWW interface, has resulted in a WWW catalogue that is fairly easy to maintain and provides a friendly user interface.

The AGSO Library's WWW catalogue is at the following URL:
http://www.agso.gov.au/information/structure/isd/library/catalogue/

1. INTRODUCTION

The Australian Geological Survey Organisation (AGSO) Library's World Wide Web catalogue is an Oracle database running on AGSO's corporate database server. The books and serials catalogue contains data derived from the Library's Datatrek library management system.

The in-house AGSO Library catalogue is part of the Datatrek Professional Series library management system. It includes a cataloguing module (in a proprietary database format) which can import and export cataloguing data for books, serials, and other published formats in the USMARC (United States Machine-Readable Catalogue) format.

In December 1995, AGSO decided that the catalogue should be made publicly available via the World Wide Web (WWW) as a searchable database. Lynton Bond won the contract to develop an Oracle database on the AGSO AViiON database server, a USMARC conversion program to replicate the Datatrek catalogue database in the Oracle system, procedures to load either all or part of the Datatrek catalogue data into the Oracle database, and a WWW query procedure for the Oracle database which would allow Internet users to search and retrieve bibliographic data for books, serials and other material formats held in the AGSO Library. The conversion program (MARCvrt) selectively converts the external USMARC format of the catalogue to text files suitable for loading into Oracle (or most other database systems).

The AGSO Library's WWW catalogue is at the following URL:
http://www.agso.gov.au/information/structure/isd/library/catalogue/

2. DATATREK PROFESSIONAL SERIES LIBRARY MANAGEMENT SYSTEM

The Datatrek library management system consists of several linked modules which provide functions for cataloguing, OPAC (online public access catalogue), circulation, serials control, and acquisitions. The cataloguing module is based on the USMARC format for recording of bibliographic data. The AGSO Library's cataloguing procedure for books, serials, some maps, computer files, and audio-visual materials is, firstly, to purchase catalogue records in USMARC format from the National Library of Australia's Australian Bibliographic Network (ABN) database. Where no catalogue record for a particular title exists, AGSO Library cataloguers create a catalogue record in the ABN system. These catalogue records are imported into the Datatrek system and subsequently converted from USMARC format into Datatrek's proprietary data format. The Datatrek system provides for the reversing of this process - converting all or part of the Datatrek database into USMARC format records and exporting them to a file.

3. USMARC FORMAT

The USMARC formats were initially devised in 1982 and revised in 1989 by the American Library Association in consultation with Library of Congress (1994) and the Canadian National Library and bibliographic networks as standards for the representation and communication of bibliographic and

related information in machine readable form. The structure of USMARC records is an implementation of US and international standards (e.g. Bibliographic Information Interchange (ANSI Z39.2) and Format for Bibliographic Information Interchange on Magnetic Tape (ISO 2709)). The formats include specifications for bibliographic, holdings and authority data types; bibliographic materials included in the specifications include books, archival and manuscripts control, computer files, maps, music, visual materials and serials. The *USMARC specifications for record structure, character sets, and exchange media* (Library of Congress Network Development and MARC Standards Office, 1994) contains details of the specification. The MARCvrt conversion program is capable of reading the Australian Bibliographic Network variation of the USMARC standard.

4. EXPORTING DATA FROM THE DATATREK CATALOGUE

A complete export from Datatrek needs only to be performed initially, in theory. Although Oracle forms-based procedures for deleting records in the Oracle database were provided as part of the contract, in practice, it has been found that a complete export from Datatrek and thence a complete replacement of the data in the Oracle database is the most time effective and accurate way of ensuring that the Datatrek and Oracle databases are nearly exact in content. A complete export from Datatrek is simply a matter of selecting the appropriate menu choices in the Datatrek system and providing an export filename. It takes about 4 hours to export the 20,000+ records in the Datatrek database into a USMARC data file (over 20Mb in size); the MARCvrt program processes this number of records in about 20 minutes; the final step, a SQL*Loader procedure to fully replace the data in the existing WWW Oracle database, takes about two hours.

At the time of writing, the WWW catalogue is updated monthly. New and updated catalogue records from ABN are received once a month on diskette. These records are imported into a review file in the Datatrek cataloguing module where they are edited and then processed into the main catalogue database. Before being processed they are exported from the review file into a USMARC format file. This procedure takes about 30 minutes to process 150 or so records each month. The MARCvrt program processes these USMARC formatted records in about 30 seconds. The SQL*Plus procedure to insert these new/updated records into the existing WWW Oracle database takes about 20 minutes.

5. ORACLE DATABASE

The Oracle database on the AViiON database server consists of tables into which the converted data from the MARCvrt program are loaded. SQL*Loader© control files are used when a full export of data from the Datatrek database has been done and the existing WWW catalogue data is to be completely replaced. When an update of the WWW catalogue is to be done records are inserted using SQL scripts.

When a complete replacement of the WWW Oracle database is performed, the existing database tables and indexes are cleared before new data is added and reindexed; external access to the WWW catalogue is revoked during this processing.

An update of the WWW Oracle database involves the processing of the files produced by the MARCvrt program: an ASCII text file consisting of a series of SQL INSERT data statements, a "master" SQL script file to run each of the script files created, a "before" SQL script file which clears any data from

the Oracle temporary tables, and an "after" SQL script file to move data from the temporary tables into the main tables. External access to the WWW catalogue is not revoked during this processing.

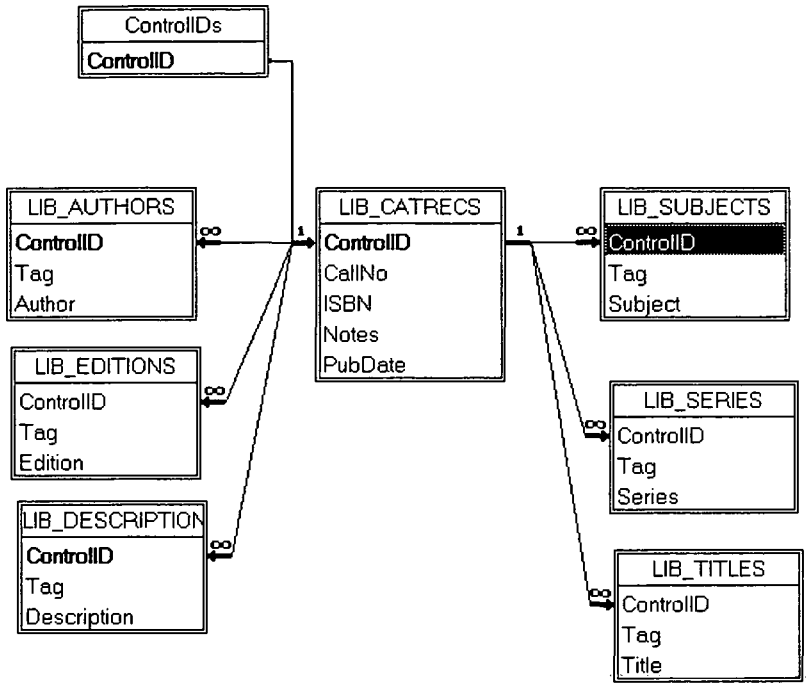

Figure 1. Library Catalogue Database Structure

5.1 Oracle database structure

The database consists of 7 temporary tables: CATRECS, AUTHORS, TITLES, SUBJECTS, SERIES, EDITIONS, and DESCRIPTION; and 7 corresponding active tables: LIB_CATRECS, LIB_AUTHORS, LIB_TITLES, LIB_SUBJECTS, LIB_SERIES, LIB_EDITIONS and LIB_DESCRIPTION. The table CONTROLIDS is used by the WWW interface to temporarily contain primary keys of records.

The structure of the database is represented in Fig. 1. The active tables, temporary tables and the corresponding USMARC fields from which data is extracted are listed in Table 1.

6. MARCvrt PROGRAM

6.1 Overview

The MARCvrt program, written in Microsoft Visual Basic, selectively converts the USMARC format data exported from the Datatrek catalogue into seven SQL*Loader files (in the case of a full catalogue replacement) or seven SQL script files (in the case of an update of the catalogue) suitable for loading into Oracle or most other database management systems. These text files consist of a series of SQL insert statements which add the data into each of the seven Oracle database tables.

The data exported from the Datatrek database in USMARC format is structured such that each record begins with a leader (containing information for the processing of the record), a directory (an index to the location of the variable fields within the record), and the variable fields (which contain a control number field, other control fields and data fields). A field terminator (ASCII character 1E) is used to terminate the directory and each variable field. A record terminator (ASCII character 1D) is used as the final character of the record.

The MARCvrt program separates selected fields from each USMARC record into text files consisting of SQL insert commands which put the data into the relevant table rows in the Oracle database.

The MARCvrt program is intended for use only with bibliographic data. It has not been thoroughly tested on materials other than books, maps and serials.

6.2 System Requirements

MARCvrt was written in Microsoft Visual Basic version 3, and requires Windows 3.1 or later to run. The program needs approximately 1MB of disk space for installation. Conversions of the full library catalogue may require as much as 35MB of available disk space (approximately 16MB for the USMARC format catalogue file, and up to a similar amount for the output, depending on selected options).

6.3 Installing MARCvrt

MARCvrt is distributed on a single diskette with its own setup program. To install MARCvrt insert the diskette into the diskette drive, select File | Run in Windows Program Manager and enter: ***a:\setup. exe***.

By default, the program files are installed in the \MARCVRT directory on the C: drive; however, the opportunity to specify an alternative installation drive and directory is provided. The Installation procedure also installs CMDIALOG.VBX and COMMDLG.DLL into the \WINDOWS\SYSTEM directory if up-to-date versions of these files are not already installed. The installation process installs the control file MARCVRT.INI into the \WINDOWS directory.

6.4 Using MARCvrt

Double-click on the MARCvrt icon. A simple Windows-style interface appears with a bar menu containing a File menu to open and convert a USMARC data file, a View option to browse the USMARC data, and a Help option.

6.4.1 Opening a USMARC file

The first step is to open a USMARC data file by selecting ***Open*** from the File menu. A File Open dialog box is displayed. By default the extension of the USMARC file is assumed to be ".mrc", so that initially the dialog box will show a list of all files with that file extension in the MARCvrt

Table No.	Main Table Name	Temporary Table Name	Field Name	USMARC Tag no. /Subfield code
1	LIB_CATRECS	CATRECS	ISBN CALLNO NOTES CONTROLID PUBDATE	020/a,023/a 984/cdef 500,501,502,515,520,525,538,546 035/a 008/4
2	LIB_AUTHORS	AUTHORS	AUTHOR CONTROLID	100/abcdq,110/abcdn,111/acdn,700/abcdq,710/abcdn,711/acdn 035/a
3	LIB_SUBJECTS	SUBJECTS	SUBJECT CONTROLID	600/abcdqtxyz,610/abcdntxyz,611/acdntxyz,630/atxyz,650/abxyz,651/axyz 035/a
4	LIB_TITLES	TITLES	TITLE CONTROLID	130/adfn,210/a,212/a,240/adf,243/adf,245/abnp,246/ab,730/adf,740/anp 035/a
5	LIB_EDITIONS	EDITIONS	EDITION CONTROLID	250,255,260/abc6 035/a
6	LIB_SERIES	SERIES	SERIES CONTROLID	400/abcdqtv,410/abcdntv,411/acdntv,440/av, 490/av,800/abcdqt,810/abcdntv, 811/acdntv, 830/adv 035/a
7	LIB_DESCRIPTION	DESCRIPTION	DESCRIPTION CONTROLID	300,310,362 035/a

Table 1

program directory. Select the drive and directory where the USMARC data file is located, highlight the file and click the OK button. The name of the currently open USMARC file is displayed in the title bar of the main MARCvrt window. Only one USMARC file can be open at a time. If a second file is opened the previously opened file will be automatically closed.

6.4.2 Converting the USMARC Data

To convert the USMARC data choose the ***Save As...*** option from the File menu. This opens the Convert MARC file dialog, as shown in Fig. 2.

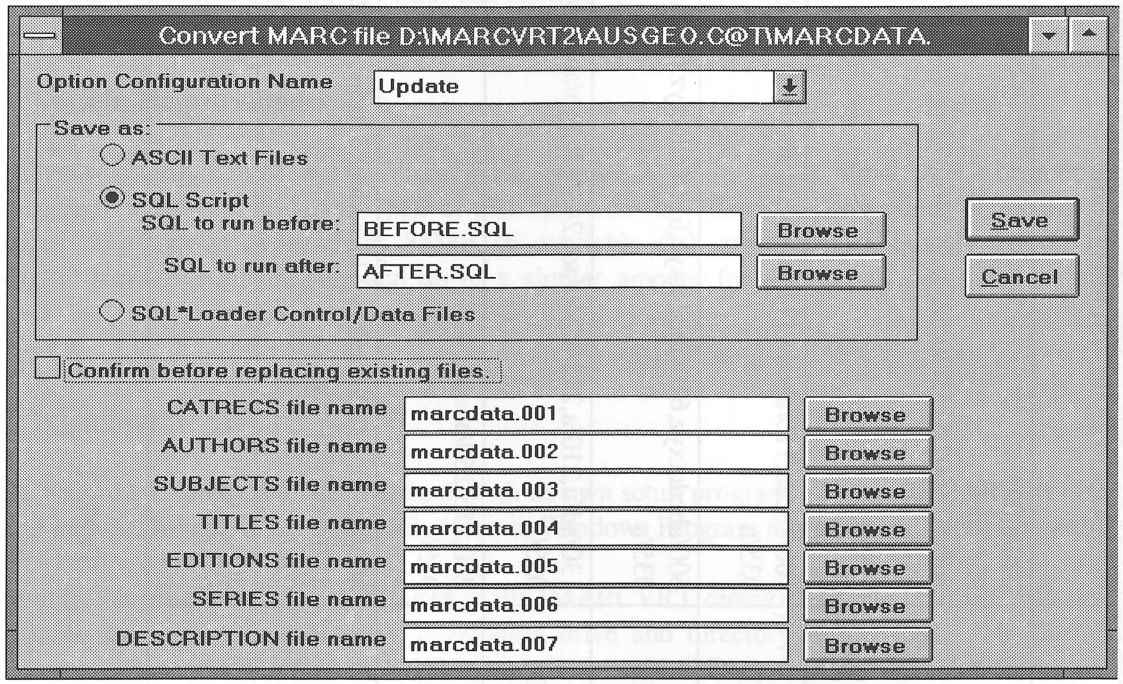

Figure 2: USMARC Conversion dialog box.

In this window specify the format of the output by choosing an Option Configuration Name. The format of the converted data can be an ASCII Text file, a SQL Script file, or a SQL*Loader Control/Data file. One file is created for each of the currently defined tables. By default each of the output files is saved in the same directory as the source USMARC data file; the filename is the same as that of the source data file and the file extension a three digit number corresponding to the table number. The path and filenames can be changed as required by directly editing the filename, or by clicking the Browse button adjacent to the filename. MARCvrt does not overwrite files of the same filename without confirmation unless the ***Confirm before replacing existing files*** checkbox is unchecked.

Files saved as SQL script files contain a series of SQL INSERT statements. MARCvrt also creates a "master" script to run each of the script files created. The master script is named MARCVRT.SQL and is saved in the same directory as the converted files. The script files can be run before and/or after the scripts that insert the data. For example, the "before" script can contain commands that set the SQL environment and open a log file. The "after" script could perform batch updates or validation.

Once the appropriate conversion settings have been chosen, clicking the ***Save*** button starts the conversion process. During processing a dialog box shows the number of records processed. Any errors detected during processing are saved in an error log file, MARCVRT.ERR in the same directory as the USMARC data file. The number of errors and the location of the error log file is displayed when the processing is completed.

6.4.3 Viewing a USMARC file

An open USMARC file can be viewed by choosing the ***View*** menu option. This activates the USMARC viewer which displays the USMARC data, record by record. The viewer window is shown in Fig. 3. This is not a comprehensive viewer; it is intended for the user to be able to confirm that the data in the open USMARC file is as expected. The only function that can be performed in the viewer is to move from record to record. The user needs to have an understanding of the USMARC format in order to interpret the tags and codes.

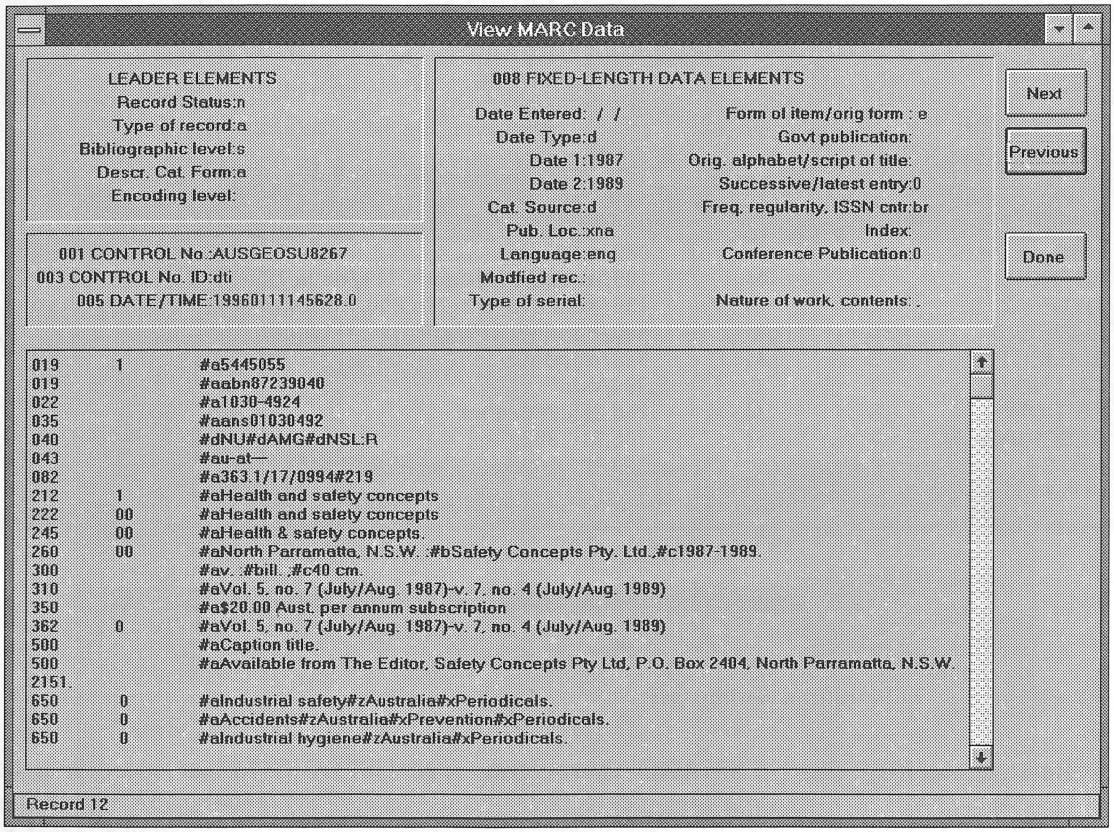

Figure 3: USMARC Data Viewer

6.4.4 Conversion options

Although the output from the MARCvrt program is essentially an ASCII text file a variety of conversion configurations is available to make the output more useful. To add or change a conversion configuration choose ***Options...*** from the File menu. Fig. 4 shows the Conversion Options dialog which is displayed.

Configuration Name: As many different conversion configurations as necessary can be maintained. A drop-down list shows the available configurations.

Buttons on the right allow for:

- the Save button saves changes to the current configuration
- the New button clears the current options so that a new set of options can be entered
- the Copy button clears only the name so that a new configuration can be created, starting from a pre-existing set of options
- the Delete button removes a configuration

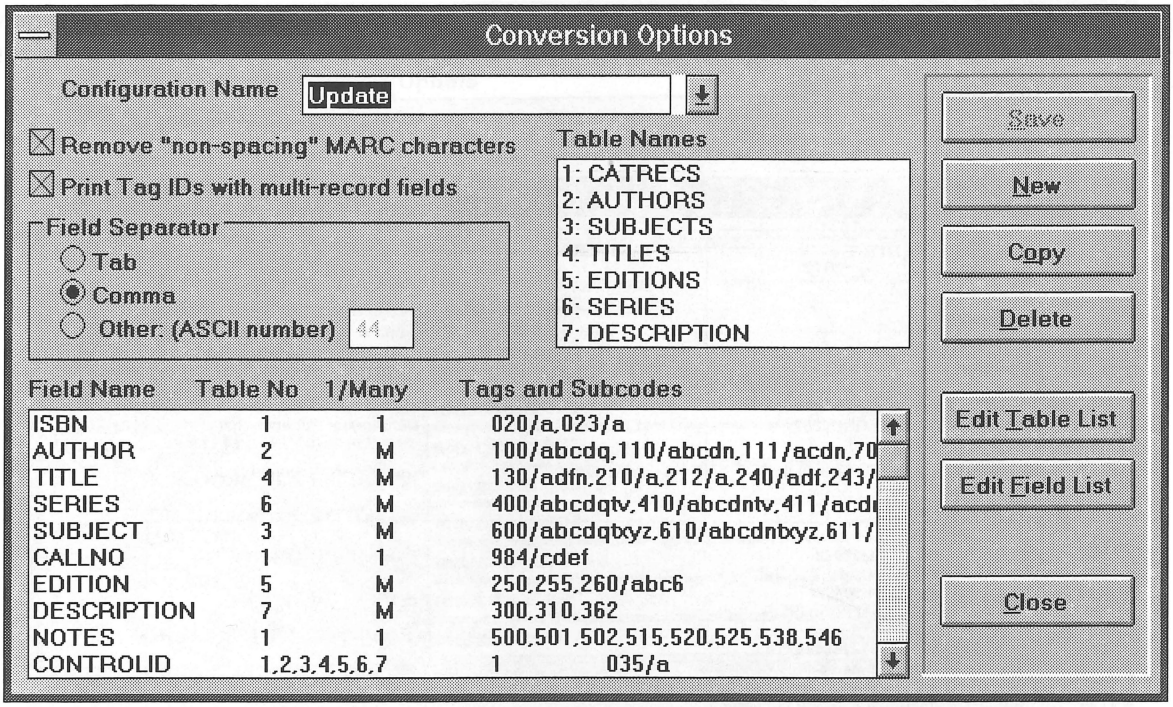

Figure 4: Conversion Options dialog box.

The check box and radio button options are:-

Remove "non-spacing" USMARC characters: The USMARC format specifies a device-independent way of recording accented characters such as cedilla, grave, acute, circumflex and umlaut and implies overprinting. Because most of these characters cannot be handled by the usual search algorithms in database systems (whether they are represented by one character or two "non-spacing" characters) they can be ignored during the conversion. For example, with the option set to remove "non-spacing" characters, the pair of characters that would produce "ç" will be saved as "c".

Print tag IDs with multi-record fields: The structure of the USMARC data is usually such that the same type of field may be repeated a number of times in a single record. For example, a single record may have several Titles recorded in fields tagged with "130", "210", "212", "240", "243" or "245" IDs. If the Title field is identified as being a multi-record field, one record will be written to the saved file for each of the Title fields encountered. The database may require that

not all of these tags be saved but that all should be searchable. To accomplish this, choose to save the tag ID for each field so that it is possible to retain the source of the particular field.

Field Separator: Each field of a record is enclosed in double quotes " ... " and separated by a user-defined character; usually this character is a comma (the default) or Tab character, but any other ASCII character could be used. Select the Tab or Other: (ASCII Number) radio button to use a character other than a comma and enter the decimal equivalent of the character in the adjacent text box.

Table Name list: This is a list of all defined database tables that are saved by the MARCvrt program. Each of the tables corresponds to a saved file. The number beside each table name corresponds to the table number in the field name list, and to the number appended to the saved file. If the output is to be formatted as a SQL script or SQL*Loader input file, the table name is used in the corresponding script. Therefore the table names should correspond to the database names of the tables into which the data will be loaded. Choose the Edit Table List button to modify the table name list.

Field Name list: This is a list of all defined fields that are saved by the MARCvrt program. The four columns list the field name, the number corresponding to the table or tables to which the field is inserted, whether or not to concatenate all occurrences of the field into one saved field, and the field tag numbers and subfield codes in the USMARC data file. If the format of the saved file is specified as a SQL script or SQL*Loader input file, the field name is used in the corresponding script. Therefore the field names should correspond to the database names of the fields into which the data will be loaded. To modify the field name list, choose the Edit Field List button.

These configuration options are maintained by MARCvrt in the MARCVRT.INI program control file which is located in the \WINDOWS directory.

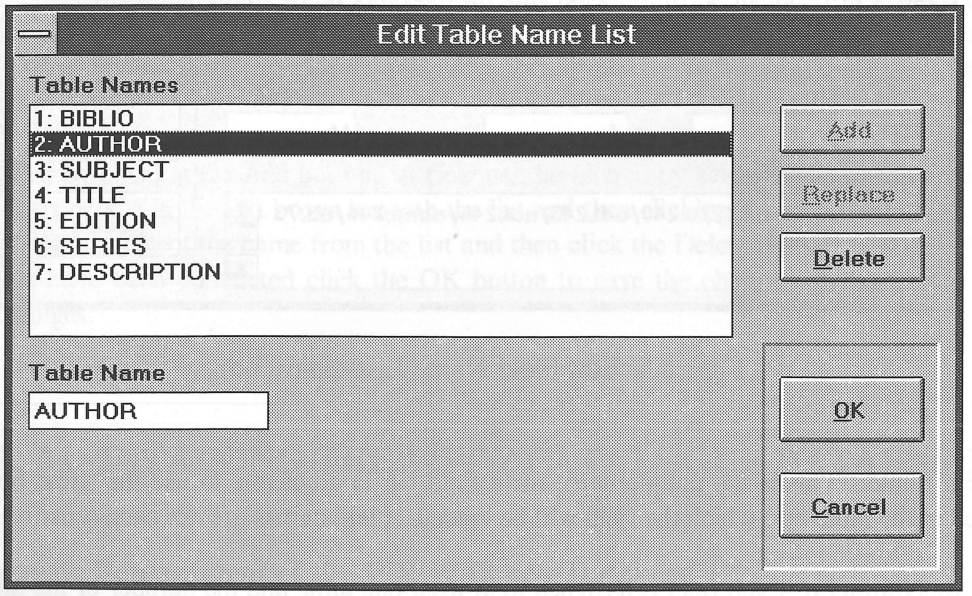

Figure 5: Edit Table Name List dialog box.

Editing the Table List

To modify the list of table names click the Edit Table List button in the Conversion Options dialog. This opens the Edit Table Name List dialog as shown in Fig. 5.

The list box at the top of the dialog contains the currently defined table names and their corresponding table numbers. A new table name can be added by typing the name in the text box beneath the list box and then clicking the Add button. A number is allocated automatically to the new table name. A table name can be altered by selecting the name in the list box, altering it in the text box below, and then clicking the Replace button. Duplicate table names cannot be entered. To delete a table name from the list, select the name in the list then click the Delete button. The remaining table numbers are renumbered and the field name list is updated to reflect the new table numbering whenever you delete a table name. When modifications to the table name list have been completed click the OK button to save the changes, or the Cancel button to ignore all changes.

Editing the Field Name List

The list box at the top of the dialog displays the currently defined field names and the conversion options as described above. Beneath the list box are text boxes corresponding to each of the columns:

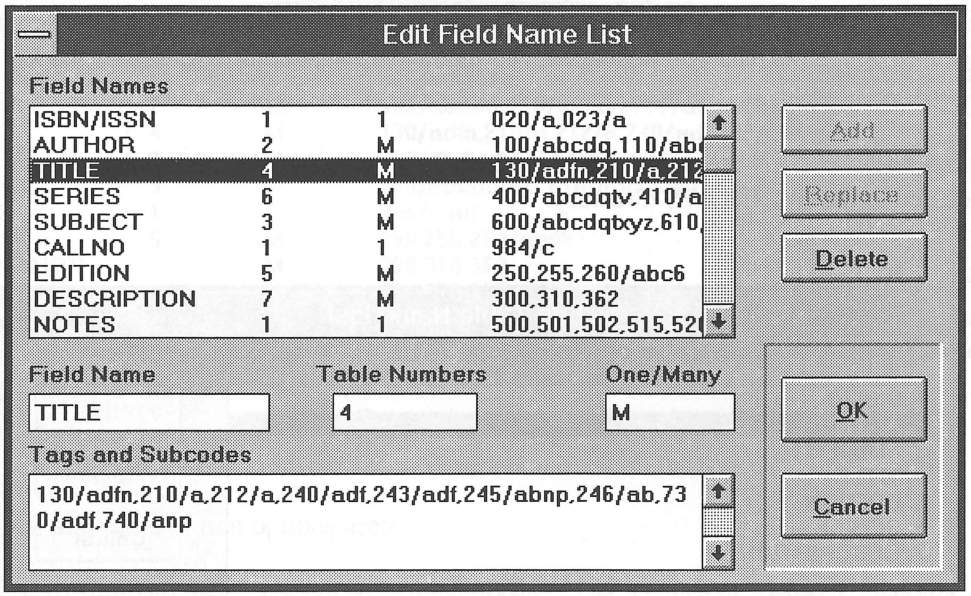

Figure 6: Edit Field Name List dialog box.

Field name: This is the unique name for a field. It is advisable to use the same field name as used in the table into which the field will be loaded. Field names are not case-sensitive.

Table Numbers: Each field is assigned to at least one table and the number of the table is in the adjacent text box. If a field occurs in more than one table (i.e. it is a primary or foreign key), then enter all the table numbers as a comma separated list. Each number entered must be in the range of defined table numbers.

One/Many: Enter "1" if all occurrences of the field in a USMARC record are to be saved to the same record by concatenating them into one field. Enter "M" if each occurrence of the field is to be saved to a separate record.

Tags and subcodes: This field is a list of all the USMARC numbered field tags and their subfield codes in the data file that will make up the field. Each tag is taken as a new occurrence of the field, whilst all nominated subcodes for a tag are concatenated to make up the field. Separate tags from subcodes with a forward slash and separate tag/subcode combinations with a comma. Refer to the USMARC format specifications to determine which tags and subcodes should be grouped together. In addition to the normal tags and subcodes, MARCvrt allows for extracting the "008" control codes by adding values as shown in the following table:

To output "008" field:	Use
Date entered	1
Date Type	2
Date 1	4
Date 2	8
Catalogue Source	16
Publication Location	32
Language	64

Table 2

For example: to output Language and Date 1 add 64 and 4 to make 68; thus, entering 008/68 would output those two fields.

A tag must consist of three digits to be valid. If all subcodes are to be included in a field, there is no need to list them; alternatively, list only those subcodes that should be included in a field. A tag can be used for only one field; MARCvrt cannot put some of a tag's subcodes in one field and others into a different field.

Add new fields by clicking the Add button. A field can be altered by selecting the field name in the list box, altering its details in the text boxes beneath the list and then clicking the Replace button. To delete a name from the list, select the name from the list and then click the Delete button. When all field name modifications have been completed click the OK button to save the changes or the Cancel button to ignore all changes.

7. MAINTAINING THE WWW CATALOGUE DATABASE

7.1 Complete Replacement of the WWW Oracle Database

A complete replacement of the WWW Oracle database requires a PC with suitable TCP/IP software (such as TNVT220 or Host Presenter in Novell's LAN Workplace) to gain access to the database server from the PC, a user ID which allows access to the server's /SCRATCH directory, and a user ID

(LIBRARY) and password for SQL*Plus. The 7 files created by the MARCvrt program must be copied from the PC to the /SCRATCH/LIBRARY.LOAD directory and then processed in SQL*Plus.

The steps involved in processing the MARCvrt files are:

- Using a program such as Rapidfiler (in Novell's Lan Workplace), login to the database server and copy the 7 files created by the MARCvrt program plus the files preload.sql and postload.sql (which can be found in the C:\MARCVRT directory).
- Log into the database server from a PC using TNVT220 program.
- Change to the ***/scratch/library.load*** directory.
- Type ***setoraprod*** to set the Oracle environment to production.
- Create a backup file of the current Oracle WWW database. This file may be necessary if it is found that the database has been corrupted.
- Login to SQL*Plus and start the ***preload.sql*** script. This script prevents WWW users from accessing the Library's WWW catalogue and deletes all indexes and data records from the Oracle database. WWW users will see the following message if they try to search the WWW catalogue: "Sorry... the library catalogue is currently unavailable. Please try again later".
- Enter the following sqlload command to process the data from the MARCvrt program into the Oracle tables: ***sqlload control=dt961025.001 log=dt961025.001.log*** (where dt961025.001 is the name of the first of the 7 MARCvrt export files; the .log files, one for each of the 7 data files, record the SQL*Load process and the number of errors in each of the data files. Any records that cannot be loaded are recorded in a "bad" file in the /SCRATCH/LIBRARY.LOAD directory and named dt9610255.xxx.bad - a separate file for each of the 7 data files, but only if there are "bad" records; there will be no bad files if the data files have "good" data in them.
- Repeat the step above for each of the remaining files: dt961025.002 to dt961025.007.
- Login to SQL*Plus and start the ***postload.sql*** script. Starting this file firstly creates indexes for the data in each of the Oracle table; it then restores WWW users' access to the WWW catalogue database.

7.2 Updating the WWW Oracle database

An update of the WWW Oracle database can also be performed from a PC with the above setup. However, if the PC has the following Personal Oracle 7 for Windows modules installed, the SQL*Plus processing can be done from a Windows window and, in addition, Oracle forms can be used to edit the WWW Oracle database directly, if necessary.

- Oracle Developer/2000 Forms 4.5, run-time only (double-click on Forms 4.5 to find the run-time)
- Oracle TCP/IP adaptor
- SQL*Net TCP/IP version 2
- SQL*Plus version 3

The steps involved in processing the MARCvrt files on the database server are:

- In the MARCvrt program, choose the Update option from the Option Configuration Name selection box. In the boxes adjacent to SQL Script, the "before" and "after" SQL scripts should be listed. When the Save button is clicked and the MARCvrt program closed, the 7 converted files and a SQL control file (marcvrt.sql) are saved.

- Open the Windows SQL*Plus program, log in with the Host String set to ***oraprod***. Then start the ***marcvrt.sql*** script. This processing should take about 10-15 minutes for files containing up to about 150 records. A log file named CATRECS.LST (in the directory: C:\ORAWIN\BIN) records any errors which might have occurred during processing.

- Deletions can be performed directly on the active database using Oracle forms which have triggers to perform cascading deletes.

8. WWW SEARCH AND DISPLAY

The WWW "search form" allows for searching one or more of the Title, Author, Subject, and Series fields in the Oracle database (Fig. 7). The following provisos must be adhered to or noted when entering search terms:

- because searching by keyword within each field is not supported, the user must enter search terms in the correct order and include articles (a, an, the and so on), if applicable.
- what is entered is not case sensitive - except for the Author field where, in the interests of faster searching, capitalising the first letter of surnames or given names is required.
- except for the author search field, automatic left and right truncation is set as the default.
- truncation can be controlled to some extent by using an asterisk (*) which can represent any number of characters separating one word from another in the same field. For example, entering "*manual*mineral*" (without the quotes) in the title box will retrieve *Manual of mineralogy, A manual of new mineral names*, and *Manual of optical mineralogy*. Note that right truncation is still operative.
- a question mark (?) represents one character (but not the absence of a character). For example, typing "*minerali?ation*" will retrieve *mineralisation* and *mineralization*; "*pal?eoecology*" will retrieve *palaeoecology*, but not *paleoecology*.

The "search form" is a bordered table inside an HTML FORM. Code for the HTML form follows:

```
<form method="get"
action="/bin/htsqr"> <input type=hidden name="file"
value="/information/structure/isd/library/catalogue/catrecs.htsqr">
<table border>
<tr><td>
<strong>Title: </strong></td>
<td><input type="text" name="Title" size="50" maxlength=2000></td></tr>
<tr><td>
<strong>Author: </strong></td>
<td><input type="text" name="Author" size="50" maxlength=255></td></tr>
<tr><td>
<strong>Subject: </strong></td>
<td><input type="text" name="Subject" size="50" maxlength=255></td></tr>
<tr><td>
<strong>Series: </strong></td>
<td><input type="text" name="Series" size="50" maxlength=255></td></tr>
<tr><th colspan=2></th></tr>
<tr><th colspan=2>
```

```
<input type="submit" value="Start search"><input type="reset" value="Clear">
</th></tr>
<tr><th colspan=2></th></tr>
</table>
<input name="ControlID" size="80" type="hidden" value="">
<input name="ShowDetails" size = "80" type="hidden" value="F">
<input name="MaxRows" type="hidden" value="200" size="3" maxlength=2><p>
</form>
```

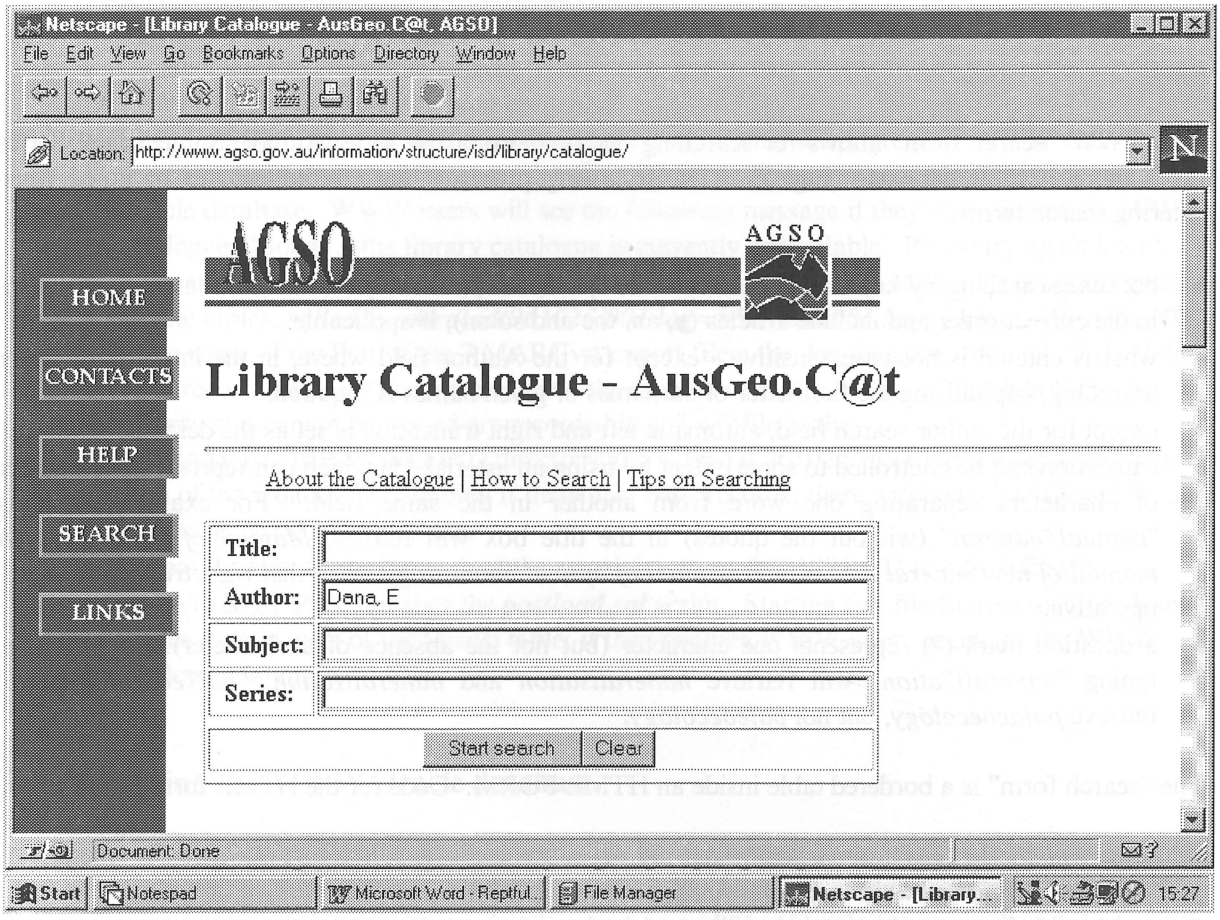

Figure 7. WWW "search form".

When the *Start Search* button is clicked the FORM "action" command first runs catrecs.htsqr which subsequently passes control to SQL reporting script catrecs.sqr. This script contains SQL commands to query the Oracle database and Structured Query Report Writer commands to manage the query results, including print statements containing HTML coding to format the results for the Web browser. There are three hidden inputs in the form:

- ControlID is always null initially; it is passed on as a number if a full catalogue record is requested, as described below
- ShowDetails determines the type of record displayed: brief or full; it is initially F which produces a brief catalogue record display

- MaxRows is the maximum number of entries which can be returned as the result of a catalogue search. A message asking the user to "Please be more specific" is displayed if the number is too high. The maximum was originally set to 100 and users were given the option to reduce the number of "hits", but not increase it. The maximum has since been increased to 200 and this figure can no longer be altered by the user.

Unless there is only one record matching the search criteria, the results of a search are displayed in a brief format which shows the call number, author, and title for each catalogue record in the database, as shown in Fig. 8.

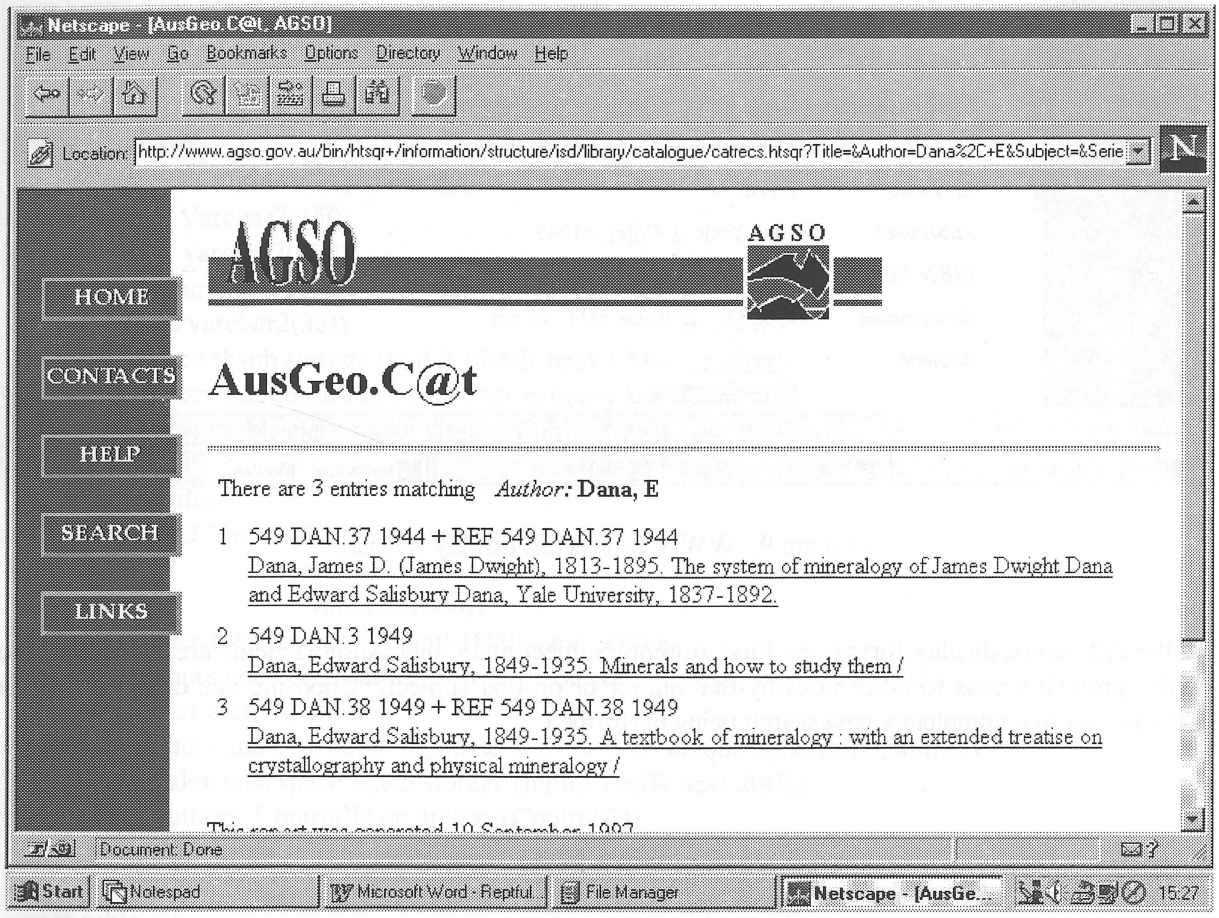

Figure 8. WWW brief record display format.

Each brief format entry is a hypertext link which, when selected, results in the complete catalogue record being displayed (Fig. 9). The ControlIDs for each of the brief entries displayed are stored in a temporary Oracle database table. The ControlID number from this table together with a change of value (from F to T which gives a full record display) for ShowDetails within hypertext link provide the search criteria for an entirely new search. For example:

*http://www.agso.gov.au/bin/htsqr?file=/information/structure/isd/library/catalogue/catrecs.htsqr?Title=&Author=&Subject=&Series=**&ControlID=+++44008346&ShowDetails=T**&MaxRows=999*

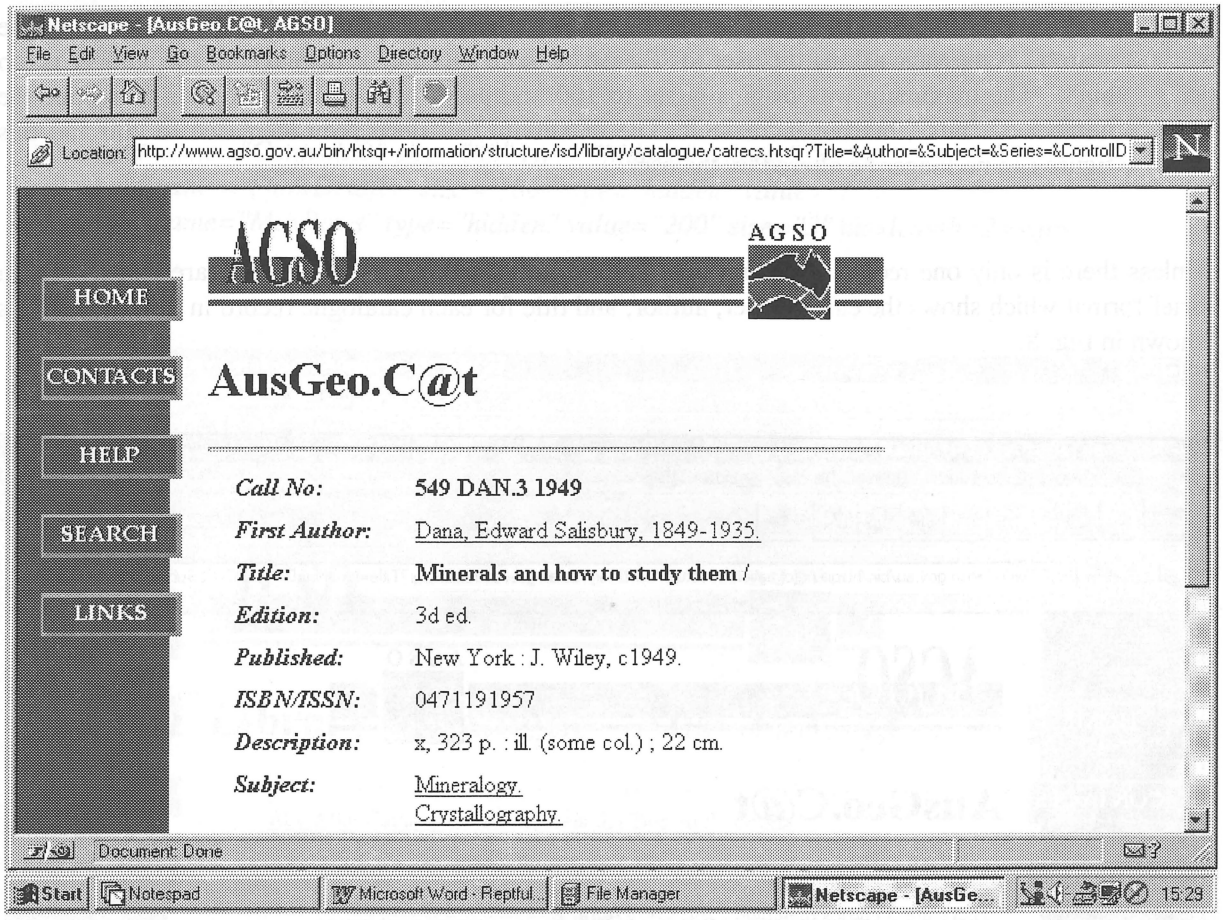

Figure 9. WWW full record display format.

In the full record display format the First Author, Subject and Other Authors fields are hypertext links which provide access to other titles by that author or on that subject. Choosing one or other of these links results in a completely new search being performed.

9. ACKNOWLEDGMENTS

This Record has benefited greatly from reviews by Anne Franklin and Jonathon Root.

10. REFERENCES

Library of Congress Network Development and MARC Standards Office, 1994. *USMARC specifications for record structure, character sets, and exchange media*. Library of Congress Cataloging Distribution Service, Washington.

APPENDIX A

Oracle Database Schema

```
rem Schema for Library Catalogue replicated database in ORACLE
rem This database is continuously available to the WWW; all
rem data mods should be done via temporary tables
rem CATRECS, AUTHORS, TITLES, SUBJECTS, SERIES, EDITIONS
rem DESCRIPTION which have referential constraints applied,
rem and thence to the active databases
rem Script files are available to do the updates and inserts.
rem Deletions may be performed via forms directly on the active database,
rem which have triggers to perform cascading deletes.

create table catrecs
      (ControlID Varchar2(150) not null,
       CallNo  Varchar2(150),
        ISBN   varchar2(600),
       Notes   varchar2(1000),
       PubDate varchar2(32))
        tablespace tbspb storage (initial 1500K next 100k);
alter table catrecs add constraint pk_catrecs primary key (ControlID)
       using index tablespace indxb storage (initial 1000k next 200k);

create table authors
      (ControlID Varchar2(150) not null,
        Tag          varchar2(3),
        Author              varchar2(600))
       tablespace tbspb storage (initial 1500k next 200k);
alter table authors add constraint fk_authors_ControlID foreign key (ControlID)
       references catrecs(ControlID);
alter table authors add constraint uk_authors_ControlID unique (ControlID,Author)
       using index tablespace indxb storage (initial 1500k next 200k);
create index authors_ControlID on authors(ControlID)
       tablespace indxb storage (initial 1000k next 200k);

create table subjects
      (ControlID Varchar2(150) not null,
        Tag          varchar2(3),
        Subject      varchar2(255))
       tablespace tbspb storage (initial 1500k next 200k);
alter table subjects add constraint fk_subjects_ControlID foreign key (ControlID)
       references catrecs(ControlID);
alter table subjects add constraint uk_subjects_ControlID
       unique (ControlID,subject)
       using index tablespace indxb storage (initial 1000k next 200k);
create index subjects_ControlID on subjects(ControlID)
       tablespace indxb storage (initial 1000k next 200k);
```

```
create table titles
      (ControlID Varchar2(150) not null,
        Tag          varchar2(3),
        Title  varchar2(500))
       tablespace tbspb storage (initial 1500k next 200k);
alter table Titles add constraint fk_Titles_ControlID foreign key (ControlID)
       references catrecs(ControlID);
alter table Titles add constraint uk_Titles_ControlID
       unique (ControlID,Title)
       using index tablespace indxb storage (initial 1000k next 200k);
create index Titles_ControlID on Titles(ControlID)
       tablespace indxb storage (initial 1000k next 200k);

create table editions
      (ControlID Varchar2(150) not null,
        Tag          varchar2(3),
        edition      varchar2(600))
       tablespace tbspb storage (initial 1500k next 200k);
alter table editions add constraint fk_editions_ControlID foreign key (ControlID)
       references catrecs(ControlID);
alter table editions add constraint uk_editions_ControlID
       unique (ControlID,edition)
       using index tablespace indxb storage (initial 1000k next 200k);
create index editions_ControlID on editions(ControlID)
       tablespace indxb storage (initial 1000k next 200k);

create table series
      (ControlID Varchar2(150) not null,
        Tag          varchar2(3),
        series       varchar2(255))
       tablespace tbspb storage (initial 1500k next 200k);
alter table series add constraint fk_series_ControlID foreign key (ControlID)
       references catrecs(ControlID);
alter table series add constraint uk_series_ControlID
       unique (ControlID,series)
       using index tablespace indxb storage (initial 1000k next 200k);
create index Series_ControlID on Series(ControlID)
       tablespace indxb storage (initial 1000k next 200k);

create table Description
      (ControlID Varchar2(150) not null,
        Tag          varchar2(3),
      Description varchar2(600))
       tablespace tbspb storage (initial 1500k next 200k);
alter table description add constraint fk_Description_ControlID foreign key (ControlID)
       references catrecs(ControlID);
alter table Description add constraint uk_Description_ControlID
       unique (ControlID,Description)
       using index tablespace indxb storage (initial 1000k next 200k);
create index Description_ControlID on Description(ControlID)
       tablespace indxb storage (initial 1000k next 200k);
```

```
REM  Master tables for WWW access
Rem    ControlIDs is a temporary table for query control
rem    it should never have data in it, as locks may prevent
rem    users from deleting from it.
create table ControlIDS (ControlID varchar2(100))
       tablespace tbspb storage (initial 100K next 100k);
grant all on ControlIDs to external;

create table lib_catrecs
      (ControlID  Varchar2(150) not null,
       CallNo  Varchar2(150),
        ISBN   varchar2(600),
       Notes   varchar2(1000),
       PubDate varchar2(32))
       tablespace tbspb storage (initial 1500K next 100k);
alter table lib_catrecs add constraint pk_lib_catrecs primary key (ControlID)
       using index tablespace indxb storage (initial 1000k next 200k);

create table lib_authors
      (ControlID Varchar2(150) not null,
        Tag          varchar2(3),
        Author              varchar2(600))
       tablespace tbspb storage (initial 1500k next 200k);
create index lib_authors_Author on lib_authors(Author)
       tablespace indxb storage (initial 1000k next 200k);
create index lib_authors_ControlID on lib_authors(ControlID)
       tablespace indxb storage (initial 1000k next 200k);

create table lib_subjects
      (ControlID Varchar2(150) not null,
        Tag          varchar2(3),
        Subject      varchar2(255))
       tablespace tbspb storage (initial 1500k next 200k);
create index lib_subjects_Subject on lib_subjects(Subject)
       tablespace indxb storage (initial 1000k next 200k);
create index lib_subjects_ControlID on lib_subjects(ControlID)
       tablespace indxb storage (initial 1000k next 200k);

create table lib_titles
      (ControlID Varchar2(150) not null,
        Tag          varchar2(3),
        Title  varchar2(500))
       tablespace tbspb storage (initial 1500k next 200k);
create index lib_Titles_Title on lib_Titles(Title)
       tablespace indxb storage (initial 1000k next 200k);
create index lib_Titles_ControlID on lib_Titles(ControlID)
       tablespace indxb storage (initial 1000k next 200k);
```

```
create table lib_editions
      (ControlID Varchar2(150) not null,
        Tag          varchar2(3),
        edition      varchar2(600))
       tablespace tbspb storage (initial 1500k next 200k);
create index lib_editions_ControlID on lib_editions(ControlID)
       tablespace indxb storage (initial 1000k next 200k);

create table lib_series
      (ControlID Varchar2(150) not null,
        Tag          varchar2(3),
        series       varchar2(255))
       tablespace tbspb storage (initial 1500k next 200k);
create index lib_Series_Series on lib_Series(Series)
       tablespace indxb storage (initial 1000k next 200k);
create index lib_Series_ControlID on lib_Series(ControlID)
       tablespace indxb storage (initial 1000k next 200k);

create table lib_Description
      (ControlID Varchar2(150) not null,
        Tag          varchar2(3),
      Description varchar2(600))
       tablespace tbspb storage (initial 1500k next 200k);
create index lib_Description_ControlID on lib_Description(ControlID)
       tablespace indxb storage (initial 1000k next 200k);
```

APPENDIX B

Structured Query Report (SQR) script

The following extracts from the SQR script show how the Oracle database is queried and the result of the query being returned to the WWW browser in HTML table format.

The initial part of the script collects the input from the WWW search form, strips leading and trailing blanks from the input of each field and replaces "" and "?" wildcards with corresponding SQL wildcards.*

```
! NAME
!     main
!
! DESCRIPTION
!     Parse user-supplied search parameters, produce detailed report
!     listing retrieved library catalogue records.
!
! ARGUMENTS
!     $Title          - search criteria for Title field
!     $Author               - search criteria for Author field
!     $Subject        - search criteria for Subject field
!     $Series         - search criteria for Series field
!      $ControlID     - search criteria for ControlID field (used in hypertext links only)
!     $ShowDetails          - display brief (="F") or details (="T")
!     $MaxRows    - Maximum number of returned records
!
begin-procedure main
 !
 ! read the arguments
 !
 input $Title_In
 input $Author_In
 input $Subject_In
 input $Series_In
 input $ControlID_In
 input $ShowDetails_In
 input $MaxRows_Str

 let #maxrows = to_number($MaxRows_str)
 if #maxrows < 1
  let #MaxRows = 200
 end-if

!   Global error flag
 let $DBCancelled = 'False'

 !     Strip leading and trialing blanks, and
 !     replace "*" and "?" wildcards with corresponding SQL wildcards
```

```
 begin-select on-error = Database_error
upper(replace(replace(rtrim(ltrim( $Title_In )),'*','%'),'?','_')) &FindTitle,
replace(replace(rtrim(ltrim( $Author_In )),'*','%'),'?','_') &FindAuthor,
upper(replace(replace(rtrim(ltrim( $Subject_In )),'*','%'),'?','_')) &FindSubject
upper(replace(replace(rtrim(ltrim( $Series_In )),'*','%'),'?','_')) &FindSeries
upper(substr(rtrim(ltrim( $ShowDetails_In )),1,1)) &ShowDetails
    let $Title = &FindTitle
    let $Author = &FindAuthor
    let $Subject = &FindSubject
    let $Series = &FindSeries
    let $ControlID = $ControlID_in
    let $ShowDetails = &ShowDetails
  from dual
 end-select

 If $DBCancelled <> 'True'
  if substr($Title,length($Title),1) <> '%'
   let $Title = $Title ||'%'
  end-if
  if substr($Title,1,1) <> '%'
   let $Title = '%' ||$Title
  end-if
  if $Title = '%'
     let $Title = ''
  end-if
  if substr($Author,length($Author),1) <> '%'
   let $author = $author||'%'
  end-if
  if substr($Author,1,1) <> '%'
   let $author = '%' || $author
  end-if
  if $Author = '%'
     let $Author = ''
  end-if
  if substr($Subject,length($Subject),1) <> '%'
   let $Subject = $Subject||'%'
  end-if
  if substr($Subject,1,1) <> '%'
   let $Subject = '%' || $Subject
  end-if
  if $Subject = '%'
     let $Subject = ''
  end-if
  if substr($Series,length($Series),1) <> '%'
   let $Series = $Series||'%'
  end-if
  if substr($Series,1,1) <> '%'
   let $Series = '%' || $Series
  end-if
  if $Series = '%'
     let $Series = ''
  end-if
```

The script then performs a count of the number of records satisfying the query and inserts that value, the name(s) of the field(s) that were chosen to be searched and the search terms entered in an HTML coded statement at the top of the page displaying the records retrieved from the database:

```
!       Determine number of records satisfying this query
   do count-matching-records
   print '<h2>Australian Geological Survey Library</h2><hr>'        (+1,1)
   if $DBCancelled = 'False'
      if $ControlID = ''
      print '<table cellpadding=3><tr><td valign=top>There ' (+1,1)
      if #rec_count = 1
         print 'is '        ()
      else
         print 'are '       ()
      end-if
      if #rec_count = 0
         print 'no'         ()
      else
         print #rec_count () edit '99999'
      end-if
      if #rec_count = 1
         print  ' entry '    ()
      else
         print  ' entries '  ()
      end-if
      print 'matching ... </td><td>'  ()
      if #criteria <> 1
         let $prefix = '<em>'
         let $suffix = '</b><br>'
      else
         let $prefix = '<em>'
         let $suffix = '</b>'
      end-if
      if $Title_In <> ''
       if $prefix <> ''
          print $prefix   ()
       end-if
       print 'Title: </em><b>'  ()
       print $title_in (+1,1) wrap 70 30
       print $suffix ()
      end-if
      if $Subject_In <> ''
       if $prefix <> ''
          print $prefix   ()
       end-if
       print 'Subject: </em><b>' ()
       print $subject_In (+1,1) wrap 70 30
       print $suffix ()
      end-if
```

```
      if $Author_In <> ''
       if $prefix <> ''
          print $prefix  ()
       end-if
       print 'Author: </em><b>' ()
       print $author_In (+1,1) wrap 70 30
       print $suffix ()
      end-if
      if $Series_In <> ''
       if $prefix <> ''
          print $prefix  ()
       end-if
       print 'Series: </em><b>' ()
       print $Series_In (+1,1) wrap 70 30
       print $suffix ()
      end-if
      print '</td>' (+1,1)
      print '</table>' (+1,1)
      end-if
      if #rec_count = 0
       print '<h3>Please try again.</h3>' (+1,1)
      else
       if #MaxRows <= 200
          if #rec_count > #MaxRows
             print '<h3>This is more than the limit of ' (+1,1)
             print #MaxRows () edit '999'
             print ' names.  Please be more specific.</h3>' ()
           else
             do CatRecs
          end-if
       else
          do CatRecs
       end-if
      end-if
   end-if
  end-if
end-procedure
```

The next procedure prepares a temporary table containing the primary keys of the records satisfying the search criteria and then counts the number of records:

```
!
! NAME
!     count-matching-records
!
! DESCRIPTION
!     This procedure prepares a temporary table containing the primary keys
!      of the catalogue records to be returned, then counts the number of records
!
!
```

```
! ARGUMENTS
!     $Title          global variable
!     $Author               global variable
!     $Subject        global variable
!     $Series         global variable
!     $LimitDate      global variable
!
! RETURNS
!       #rec_count    global variable
!
begin-procedure count-matching-records
 let #rec_count = -1

 begin-sql on-error=DataBase_Error
    delete from library.controlids
 end-sql
 let $Inserted = 'F'

!       Query by either one ControlID or combination of other criteria
 if $ControlID <> ''
  begin-sql on-error=Database_Error
    insert into Library.ControlIDs (controlid) values ($ControlID)
  end-sql
  let #rec_count = 1
 else

  if $Author <> ''
   begin-sql on-error=DataBase_Error
    insert into library.controlids (select distinct controlid from library.lib_authors
                                    where author like $Author )
   end-sql
   let #rec_count = #sql-count
   Let $Inserted = 'T'
  end-if

  if $DBCancelled = 'False'
   if $Title <> ''
    if $inserted = 'T'
     begin-sql on-error=DataBase_Error
      delete from library.controlids where controlid not in
              (select controlid from library.lib_titles
                          where upper(Title) like $Title and
                              controlid in
                             (select * from library.controlids))
     end-sql
     let #rec_count = #rec_count - #sql-count
    else
     begin-sql on-error=DataBase_Error
      insert into library.controlids (select distinct controlid from library.lib_titles
                          where upper(Title) like $Title)
     end-sql
     let #rec_count = #sql-count
```

```
      Let $Inserted = 'T'
     end-if
    end-if
   end-if

   if $DBCancelled = 'False'
    if $Subject <> ''
     if $inserted = 'T'
      begin-sql on-error=DataBase_Error
       delete from library.controlids where controlid not in
              (select controlid from library.lib_subjects
                     where upper(Subject) like $Subject and
                        controlid in
                       (select * from library.controlids))
      end-sql
      let #rec_count = #rec_count - #sql-count
     else
      begin-sql on-error=DataBase_Error
       insert into library.controlids (select distinct controlid from library.lib_subjects
                     where upper(Subject) like $Subject
                        )
      end-sql
      let #rec_count = #sql-count
      Let $Inserted = 'T'
     end-if
    end-if
   end-if

   if $DBCancelled = 'False'
    if $Series <> ''
     if $inserted = 'T'
      begin-sql on-error=DataBase_Error
       delete from library.controlids where controlid not in
              (select controlid from library.lib_series
                     where upper(Series) like $Series and
                        controlid in
                       (select * from library.controlids))
      end-sql
      let #rec_count = #rec_count - #sql-count
     else
      begin-sql on-error=DataBase_Error
       insert into library.controlids (select distinct controlid from library.lib_series
                     where upper(Series) like $Series
                      )
      end-sql
      let #rec_count = #sql-count
      Let $Inserted = 'T'
     end-if
    end-if
   end-if
  end-if
```

```
  if #rec_count < 0
    let #rec_count = 0
  end-if
  if #rec_count > 0
    do CheckCount
  end-if
end-procedure
```

The main printing procedure prints each of the non-empty fields in the database. An if $ShowDetails = 'T' *statement is inserted for those fields which are only displayed in full record format.*

```
!
! NAME
!      CatRecs-print
!
! DESCRIPTION
!      This procedure prints a row of the Library.Catrecs table.
!       The various columns are looked up and printed in text form.
!
! ARGUMENTS
!       Global columns:  ISBN, Notes, ControlID
!
begin-procedure CatRecs-print

  if $DBCancelled <> 'True'
    ! If Call Number is not blank, print it
    if &CallNo <> ''
      if $ShowDetails = 'T'
        print '<tr><td valign=top align=left><strong><em>Call No:
</em></strong></td><td><strong>' (+1,1)
        print &CallNo (+1,1) wrap 70 30
        print '</strong></td>' (+1,1)
      else
        print '<tr><td align=left valign=top>'     (+1,1)
        print &Rownum    ()
        print '.</td><td>' (+1,1)
        print &CallNo (+1,1) wrap 70 30
        print '<br><a href="catrecs.htsqr?Title=&Author=&Subject=&Series=&ControlID=' ()
        print &ThisEntry ()
        print '&ShowDetails=T&MaxRows=999"> ' ()
      end-if
    end-if
  end-if

  !     print principal author
  do AuthorPrint
   !    print Title
   if $DBCancelled <> 'True'
     do TitlePrint
   end-if

   if $ShowDetails = 'T'
```

```
!     print Edition
    if $DBCancelled <> 'True'
     do EditionPrint
    end-if

!     print Published date
    if $DBCancelled <> 'True'
     do PubPrint
    end-if
  end-if

  if $ShowDetails = 'T'
   if $DBCancelled <> 'True'
    ! Print ISBN if not blank
     if &ISBN > ''
      print '<tr><td valign=top align=left><strong><em>ISBN/ISSN: </em></strong></td><td>'
(+1,1)
      print &ISBN (+1,1) wrap 70 30
      print '</td>' (+1,1)
     end-if

  !
  ! The "other" details:
  ! Description
  !
  !     print Description, separated by commas, unordered.
     do DescPrint
    end-if

  !     print extent, separated by commas, unordered.
    if $DBCancelled <> 'True'
     do ExtentPrint
    end-if

  !     print Subjects, separated by commas, unordered.
    if $DBCancelled <> 'True'
     do SubjectPrint
    end-if

  !     print Series, separated by commas, unordered.
    if $DBCancelled <> 'True'
     do SeriesPrint
    end-if

  !     print other authors
    if $DBCancelled <> 'True'
     do OthersPrint
    end-if
```

```
  !
  ! print the Notes, if defined
  !
   if $DBCancelled <> 'True'
      if &Notes > ''
        print '<tr><td valign=top align=left><strong><em>Notes: </em></strong></td><td>' (+1,1)
        print &Notes (+1,1) wrap 70 30
      end-if
    end-if
    print '</td>' ()
   end-if
end-procedure
```

Most of the fields have a procedure similar to the one below in which the field data is displayed as a cell of an HTML table. The Title, Author, and Call Number fields print procedures contain if $ShowDetails = 'T' *statements which determine the format of the display for full records.*

```
!
! NAME
!      TitlePrint
!
! DESCRIPTION
!      This procedure prints the title of the Catrecs record.
!
! ARGUMENTS
!       Global columns:  ControlID
!
begin-procedure TitlePrint
    begin-select on-error=Database_Error
Title
      if $DBCancelled = 'True'
        exit-select
      end-if
      if $ShowDetails = 'T'
        print '<tr><td valign=top align=left><strong><em>Title:</em></strong></td><td><strong>'
(+1,1)
      end-if
      print &Title (+1,1) wrap 70 30
      if $ShowDetails = 'T'
        print '</strong></td>' ()
      else
        print '</a></td>' ()
      end-if
      from library.lib_Titles
      where Tag = '245' and ControlID = &ControlID
    end-select
end-procedure
```