ISBN: 978-1-326-26895-4

UniVerse Files

In a UniVerse database, data is held in files.

You can think of a UniVerse file as being very similar to a filing cabinet in an office. Inside the file, UniVerse stores any number of pieces of data in structures called records. Just like the paper forms in a filing cabinet, these records generally contain similar information: customer details, invoices, despatch notes or similar.

A single UniVerse file may hold hundreds of thousands or even millions of records. Just like any good filing system, the records are organized in such a way that the database can retrieve any particular record instantly.

In most database systems, all files or ‘tables’ are homogenous. That means that every record or row in the file has exactly the same structure and layout. A customer table holds the same information on every customer, and in the same order: a typical example might include the surname, forename, title, address lines, account code and credit limit.

UniVerse files are more like the physical filing cabinets after which they are modelled. A UniVerse file more often than not will contain homogenous records: a UniVerse customer file would be very similar to the customer files found on other databases (you will learn some of the differences later). But in the same way that a filing cabinet can heap in information from different sources, so a UniVerse file can be used to store information from different places and in different formats. Such files are generally no use for enquiring upon, but they are very useful to programmers who need to store things away.

Listing the Files in your Account

Because UniVerse holds all of its data in files, the first stage when looking for information is to discover the files that are visible in your account. The main command for doing this is the LISTFILES or LISTF command.

LISTF

The LISTFILES command displays a listing of all of the files that are directly accessible from your account, along with various other pieces of information that can be safely ignored for now. The listing will include any files that are created locally in that account, and certain files that may in fact be held in other accounts elsewhere in the UniVerse database.

```
FILES in your vocabulary
Filename.......................... Pathname..................... Type Modulo

     File - Contains all logical device names
DICT &DEVICE&                      C:\ROCKET\UV\D_&DEVICE&              2      1
DATA &DEVICE&                      C:\ROCKET\UV\&DEVICE&                2      3

     File - Data Encryption Key Store
DICT &KEYSTORE&                    C:\ROCKET\UV\D_&KEYSTORE&            3      1
DATA &KEYSTORE&                    C:\ROCKET\UV\&KEYSTORE&              2      3

     File - Used by MAKE.MAP.FILE
DICT &MAP&                         C:\ROCKET\UV\D_&MAP&                 2      1
DATA &MAP&                         C:\ROCKET\UV\&MAP&                   6      7

     File - Contains all parts of Distributed Files
DICT &PARTFILES&                   C:\ROCKET\UV\D_&PARTFILES&           2      1
DATA &PARTFILES&                   C:\ROCKET\UV\&PARTFILES&            18      7

DICT &SAVEDLISTS&                  D_&SAVEDLISTS&                    3      1
DATA &SAVEDLISTS&                  &SAVEDLISTS&                      1

Press any key to continue...
```

Listing the Content of Files

When you start looking through the list of files, the names that appear may or may not give a clue to the information that the files are holding. If the files are sensibly named, a file named CUSTOMERS may be expected to hold information on your customers, and so forth.

Assuming your system is using sensible names, the next step will be to start looking through some of the file content to work out just what each file is holding.

There are various means to look at the data held in a file, some of which can be generally more helpful than others. In the perfect world of a demonstration application, it should be possible for you to get a summary view of the information held in a file by using the command:

```
LIST filename
```

where filename is the name of the file to be listed. Do notice that file names in UniVerse are case sensitive.

Do This:

`LIST BOOK_TITLES`

```
LIST BOOK_TITLES
Key.. Title........................ Author Name..................

   10 Hancock a Comedy Genius (BBC   CAST
      Radio Collection)
   11 I'm Sorry I Haven't a Clue:    CAST
      Vol 8 (BBC Radio Collection)
   12 Friends, Lovers, Chocolate     ALEXANDER MCCALL SMITH
   13 The Legend of Spud Murphy      EOIN COLFER
   14 Farmer Giles of Ham and Other  J. R. R. TOLKIEN
      Stories
   15 The Lord of the Rings:         J. R. R. TOLKIEN
      Complete & Unabridged
```

Files and Fields

Each UniVerse record can contain a number of different pieces of information. In order to allow UniVerse to distinguish between each of these pieces of information, a record is divided into a set of entries known as **Fields**. A record may have no fields, one field or any number of fields. In the example of the customer file above, a customer record may have separate fields for the forename, surname, title and credit limit. This makes it possible to use these for enquiry purposes.

So to pull this journey together so far:

- UniVerse consists of a number of Accounts.
- Accounts contain a number of Files
- Files contain a number of Records.
- Records contain a number of Fields.

There are more levels, but that is enough for now.

The LIST command displays columns of information based on the content of the records held in that file. Each column represents one of the fields from the records. The LIST command interrogates each record in the file in turn, locating those fields and displaying them on the next line of the listing.

Exercise

List the major files of the demonstration database to get an idea of their contents.

These are the files

BOOK_TITLES, BOOK_AUTHORS, BOOK_SUPPLIES and BOOK_SALES.

Some of these listings will go on for many pages. Remember that you can type a "Q" in response to the *Press any key to Continue* prompt to stop the listing when you have a clear idea of the content.

The Default Listing

When you use the LIST command used to view the contents of a file, it does not in fact show you all of the information that is available - not even close. The LIST command only shows selected pieces of information that may be of interest: it is designed to act as a type of short cut to enable you to quickly grasp the most useful content in the file. Usually this list will show a small selection of the most commonly accessed fields that will fit easily across the width of a terminal display.

Specifically, the LIST command shows a set of fields termed the "**default listing**".

The choice of which fields should make up the default listing is made by the developer or administrator responsible for the file. UniVerse records typically hold a large amount of information and thus any listing showing all the available data might easily run to hundreds of columns - far too many to be easily readable!

Real World

In the real world, the extent to which the default listing shows anything useful depends entirely on the person responsible for looking after that file. Creating a default listing, whilst helpful, is not a UniVerse requirement, and some developers view it as just another chore to be avoided.

This means that many application files may not have a default listing at all. In these cases, only the record keys are displayed – which is generally not very useful. The BOOK_COMMENTS file, for example, has no default listing, as you can see if you list the file using the command:

LIST BOOK_COMMENTS

The default listing should give you some clues that you are looking at the right files, but if you want to find out what a file really holds, you need to look elsewhere. This introduces the final one of the key elements of the UniVerse database that you need to understand: the **File Dictionary**.

Files and Dictionaries

Each and every UniVerse file is generally made up of two separate sections:

- a Data section that holds the records.
- a Dictionary section that describes the content.

The File Dictionary holds the **Metadata** (data about data) that describes the information available in that file. This metadata holds the key to understanding the data held in a UniVerse database, and is the foundation upon which the enquiry languages are built.

The metadata describes:

- The structure of the data held in the file.
- The names for the various fields.
- The data types for the various fields.
- The display format used when presenting the fields.
- The default listing used with the LIST command.
- Common groupings of information.
- Calculated and virtual fields.
- Relations between files.

In short, the metadata held in the file dictionary tells you everything that you can know about the file if you want to enquire upon it.

Dictionaries and Fields

So just what does the dictionary tell you about the content of the file?

First off, the dictionary defines the fields in the data. A field holds one piece of information: for a customer file, that might be the customer forename, surname, address, date of birth or credit limit. For meaningful enquiries, each record in a file should be made up of the same fields so that the enquiry engine can extract them quickly (just like a set of paper forms).

In the training database the BOOK_TITLES file, for example, contains records defining each audio title in the bookstore catalogue. Each title record records the same information: the short title of the book, the author and reader, the ISBN, genre, department, price and stock level. You will have seen some of this information when you ran the default listing on the BOOK_TITLES file.

Whenever the LIST command encounters a file, it looks at the file dictionary to find the information for each of the fields it needs to display. When you LIST BOOK_TITLES to get the default listing, the only way that the LIST command knows the meaning of the fields that make up that default listing is because the dictionary of the BOOK_TITLES file defines the record layout and gives a name to each field in the record.

Tip:

Whenever you are looking for information, first identify the file likely to contain the information you require and then use the dictionary of that file to discover the actual content.

Listing a File Dictionary

The dictionary belonging to a file is always referenced using the name:

```
DICT filename
```

You can list the content of the dictionary in the same way that you listed the content of the data file: by using the LIST command

```
LIST DICT filename
```

Do this:

```
LIST DICT BOOK_TITLES
```

```
DICT BOOK_TITLES

Field......... Type & Field........ Conversion.. Column......... Output Depth
Name.......... Field. Definition... Code........ Heading........ Format
               Number

TITLE_ID       D    0                            Id              5R     S
@ID            D    0                            Key             5R     S
TITLE_NAME     D    1                            Title           50T    S
SHORT_TITLE    D    1                            Title           30T    S
AUTHOR_ID      D    2                            Author          5L     S
READER_ID      D    3                            Reader          5L     M
TYPE           D    4             MCU            Type            10L    S
UNITS          D    5             MD0            Units           5R     S
FORMAT         D    6             MCU            Format          5L     S
ISBN           D    8                            Isbn            10L    S
PRICE          D    9             MD2            Price           10R    S
DEPT           D   11             MCU            Dept            10L    S
GENRE          D   12             MCU            Genre           15L    S
PUBLISHER      D   13                            PUBLISHER       20L    S
PUB_YEAR       D   14                            PUB_YEAR        4R     S
ALLOCATED      D   15                            ALLOCATED       3R     S
```

At the start of each dictionary listing you will find the field definitions. These are identified by a 'D' (standing for Data) in the Type column of the list. This portion of the list shows the real fields that occupy set positions in each record: the position number of the field is given in the next column.

Conversion Codes

A conversion code is used to map data between different representations. UniVerse does not have data types in the sense of other databases, and so uses conversion codes to handle the way in which different types of information such as dates and times and numbers are displayed.

Imagine you have a field that holds a date value. Depending on your culture and the parts of the date that you need to see, you can choose the write the same date in a huge variety of different ways:

```
01 JAN 2007
Monday, 01 January 2007
01/01/07
2007-01-01
20070101
07001
```

Rather than storing dates in all these possible formats, UniVerse converts dates into a convenient internal format that it can use to quickly perform arithmetic and sorting operations.

In fact, all dates are stored as whole numbers (Universe treats times separately) representing the number of days between that date and 'day zero', which is set at 31 December, 1967. The 1st January 2007, for example, is stored as the number 14246. Any date falling before 31 December 1967 is simply held as a negative number. Older UniVerse programmers may talk jealously about younger colleagues with 'positive birth dates'.

Even the most ardent UniVerse geek doesn't set their calendar by the UniVerse internal date, and so you would naturally expect to see dates displayed in a human readable format like the ones above. All databases have some built in way of displaying dates and times. But which format does UniVerse choose?

This format is given by the conversion code: a short, often cryptic code that – like the display format and column header – is normally set as part of a field definition in the file dictionary but that can also be overridden as part of a LIST command.

There are many possible conversion codes and many, many combinations of codes, not all of which are useful and some of which have been rendered largely obsolete by new functionality in UniVerse. You do not need to learn all of these codes, but the main conversion codes are exceptionally helpful when working with character data, dates and times and with numbers.

Many training courses on RetrieVe devote huge sections of time and material to learning the intricacies of strange conversion codes that you never use ever again. This lesson will introduce you to the main families of conversion codes that are useful in daily reporting, and direct you where to look if you ever need more.

Conversion Codes for Character Data

All conversion codes begin with either a single character or pair of characters that introduce the type of conversion that is being applied. These introductory characters make up the various 'families' of conversion codes.

The main conversion code family for working with Character data is introduced with the letters: `MC` (Masked Character). The MC series of codes is used primarily for converting between cases, stripping out certain character types and for handling special characters.

Changing Character Case

UniVerse treats all data as case sensitive. If the data in the record is stored in upper case, it will be presented in upper case and all searches preformed upon that data will need to use upper case to find it. Similarly, if the data is stored in mixed or title case it will be presented and searched for in mixed case. This can be seen as a blessing or a curse, depending on how accurately the data has been provided in the first place.

There are three simple conversion codes in the MC family that can be used to present data converted into upper, lower and mixed case respectively, regardless of how the data is originally stored. These are:

- MCU Convert to Upper Case
- MCL Convert to Lower Case
- MCT Convert to Title Case

Remember that, just like all the other display formatting operations you have used in the previous lesson, this only affects the way that the data is formatted for enquiry and does not change the data that is actually stored on file.

To add a conversion code to a field as part of a LIST command you follow the name of the field with the `CONV` keyword and the new conversion code enclosed in double quotation marks:

```
fieldname CONV "code"
```

Do this:

```
LIST BOOK_TITLES SHORT_TITLE CONV "MCU"
```

```
LIST BOOK_TITLES SHORT_TITLE CONV "MCU"
Key.. Title........................

   10 HANCOCK A COMEDY GENIUS (BBC
      RADIO COLLECTION)
   11 I'M SORRY I HAVEN'T A CLUE:
      VOL 8 (BBC RADIO COLLECTION)
   12 FRIENDS, LOVERS, CHOCOLATE
   13 THE LEGEND OF SPUD MURPHY
   14 FARMER GILES OF HAM AND OTHER
      STORIES
   15 THE LORD OF THE RINGS:
      COMPLETE & UNABRIDGED
```

By now you should be familiar with the conventions of adding keywords to the field names. You can combine the CONV keyword with the FMT and COL.HDG keywords in any order, so long as they follow the field name to which they refer and always remembering to put the codes in double quotation marks.

Lab 1

The BOOK_SALES file stores sales orders.

Each sales order includes the name of the customer who placed the order.

List the customer forename and surname for each order, showing the name in upper, lower and title case in turn.

Sometimes, data can become 'dirty' if it is not correctly and carefully managed: users can accidentally type unprintable control characters into a field when updating a record and those characters can be written to the database causing errors when you come to select the data.

The `MCP` (Printable Character) conversion can be useful in this case: this displays a period (.) in place of any unprintable character when the data is displayed.

Extracting Text

In the previous lesson you learned how to change the display width of a field so that you could fit the columns across the screen even if the original display width was too wide. Changing the display width can force columns onto the report, but does not change that amount of data that is presented: the data presented simply wraps if it cannot fit into a single line of the column.

Sometimes you may wish to actually truncate the display to prevent the data from wrapping, or you may only be interested in showing a certain number of characters from a field. You might, for example, wish to display only an initial letter from a customer forename on a particular report.

This is the role of the Text Extraction (`T`) code.

Handling False Positives

Sometimes the use of multivalued data does force you to have to think. This time, consider the opposite situation – if you want to find multivalued fields that do **not** contain a particular value.

Consider the following situation: you wish to find the sales orders that don't have junior books.

You might well issue the following casual enquiry command that omits JUNIOR department sales orders:

Do This:

```
LIST BOOK_SALES WITH DEPT <> "JUNIOR" TITLE_NAME DEPT
```

This should select (you would assume) the sales orders that do not have junior books, but when you run the command you will see that the listing displays junior titles as well:

```
BOOK_SALES... Title.......................................... Dept......

13661*55800*1 Jingo                                           ADULT
              Harry Potter and the Goblet of Fire (Book 4 -   JUNIOR
              Unabridged 14 Audio Cassette Set)
              I'm Sorry I Haven't a Clue: Vol 8 (BBC Radio    ADULT
              Collection)
13660*37800*1 The Decline and Fall of the Roman Empire (Part 2): ADULT
              Audio CDs
              A Tale of Two Cities                            ADULT
              The Hostile Hospital: Complete & Unabridged     JUNIOR
              (Series of Unfortunate Events)
              Making Divorce Work: In 9 Easy Steps            ADULT
              Voices of History                               ADULT
              The Time Traveler's Wife                        ADULT
              Cirque Du Freak: Complete & Unabridged (Saga of JUNIOR
              Darren Shan S.)
              Making Divorce Work: In 9 Easy Steps            ADULT
              The Twits                                       JUNIOR
```

Why does this happen and how would you fix this?

Filtered Selections

The regular vectored selection and the exclusive selection both decide which records can be included in a listing. There is also a third possibility: that you wish to select the records that include a particular value but only wish to see the individual lines that match. In the vectored selection above, you were able to find a purchase order with a particular reference but the listing still showed you all of the lines on that purchase order, each with a different reference. A filtered selection removes those extra lines from the display.

Filtered selections are introduced using a WHEN clause. This replaces the WITH clause you have been using throughout this chapter.

```
LIST filename WHEN field operator value
```

These two listings below can illustrate the difference:

Do This:

```
LIST BOOK_SUPPLIES WITH DELIV_DATE = "01 AUG 2005"
```

```
LIST BOOK_SUPPLIES WITH DELIV_DATE = "01 AUG 2005"     1

BOOK_SUPPLIES. A/00020
SUPPLIER...... OGG
Catalog....... OGG/0K8
TITLE_ID......   238
Title......... The Austere Academy (Series of
                Unfortunate Events)
                                                          Delivery....
Order Date.. Delivery Time Quantity Order Ref. Price..... Date........
 30 JUL 2005         15:00        6 13240           8.79  02 AUG 2005
 31 JUL 2005         15:00        6 20337           8.79  01 AUG 2005
 01 AUG 2005         12:00        3 15255           8.79  02 AUG 2005

BOOK_SUPPLIES. A/00022
SUPPLIER...... OGG
Catalog....... OGG/QOO1
TITLE_ID......   250
Title......... Eric
                                                          Delivery....
Order Date.. Delivery Time Quantity Order Ref. Price..... Date........
 28 JUL 2005         09:00        3 19714          11.24  30 JUL 2005
 29 JUL 2005         10:00        4 26255          11.24  30 JUL 2005
 30 JUL 2005         09:00        4 14231          11.24  31 JUL 2005
 31 JUL 2005         12:00        5 16128          11.24  01 AUG 2005
```

Do This:

`LIST BOOK_SUPPLIES WHEN DELIV_DATE = "01 AUG 2005"`

```
LIST BOOK_SUPPLIES WHEN DELIV_DATE = "01 AUG 2005"

BOOK_SUPPLIES. A/00020
SUPPLIER...... OGG
Catalog....... OGG/0K8
TITLE_ID......   238
Title......... The Austere Academy (Series of
                Unfortunate Events)
                                                          Delivery....
Order Date.. Delivery Time Quantity Order Ref. Price..... Date........
 31 JUL 2005         15:00        6 20337            8.79  01 AUG 2005

BOOK_SUPPLIES. A/00022
SUPPLIER...... OGG
Catalog....... OGG/QOO1
TITLE_ID......   250
Title......... Eric
                                                          Delivery....
Order Date.. Delivery Time Quantity Order Ref. Price..... Date........
 31 JUL 2005         12:00        5 16128           11.24  01 AUG 2005
```

The same records have been selected, but in the second case using the WHEN clause has filtered out all of the other deliveries that do not match the date criteria from the display.

Embedding Break Text Values

The break text can also include special instructions. These take the form of letters that are embedded into the text using the peculiar convention that you first met right back at the start of Chapter 3 when creating multiple line column headers.

The embedded instructions are inserted into the break text enclosed in single quotation marks. This differentiates the instructions from the rest of the text. One such instruction is the `P` instruction:

Do This:

```
SORT BOOK_SALES BY SALE_DATE BREAK.ON SALE_DATE "End of
Group'P'" SURNAME
```

A `'P'` instruction within the break text is a page throw instruction. This causes RetrieVe to throw a page after the break line, so that the next break set will begin on a fresh page.

```
13434*58500*1 11 OCT 2004 BRANDON
13434*62100*1 11 OCT 2004 PARHAM
13434*62100*3 11 OCT 2004 CROOT
13434*63000*1 11 OCT 2004 PECKHAM
13434*63000*3 11 OCT 2004 BRENCHLEY

             End of Group

Press any key to continue...
```

Another useful instruction is the `V` instruction. This places the break value into the break point line and will be essential later on when building summary reports.
You can combine several instructions in the same text by enclosing them together in the same set of single quotation marks:

Do This:

SORT BOOK_SALES BY SALE_DATE BREAK.ON SALE_DATE "Total' VP'" SURNAME

```
13434*58500*1 11 OCT 2004 BRANDON
13434*62100*1 11 OCT 2004 PARHAM
13434*62100*3 11 OCT 2004 CROOT
13434*63000*1 11 OCT 2004 PECKHAM
13434*63000*3 11 OCT 2004 BRENCHLEY

         Total 11 OCT 2004

Press any key to continue...
```

Lab 6

Display a listing of the BOOK_TITLES grouped by DEPT_GENRE. Each grouping should begin on a new page, and the current value of the DEPT_GENRE should appear in the break line.

The following table lists the embedded codes:

B	Includes the break value in a HEADING or FOOTING statement. The header or footer text must have a matching B code to receive it.
D	Suppress break line if there is only a single row in the break set, but keeps a blank line between the break sets.
L	Suppress break line but still keeps a blank line between break sets.
N	Resets page number to 1 for each break set.
O	Outputs each break value only once.
P	Adds a page break for each break set.
V	Inserts the break value.