

# IBM UniVerse Enquiry Self Paced Training

Brian Leach

## Files and Dictionaries

---

Each 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?

As you have seen with the Default Listing, in order to make sense of a record UniVerse first has to divide up a record into its constituent fields.

Each 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 presented in the same order, so that the enquiry engine can extract them quickly (just like a set of paper forms).

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 number, 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.

How does UniVerse know that the BOOK\_TITLES file holds this information?

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. The only way that the LIST command knows the meaning of the fields that make up the default listing of the BOOK\_TITLES, is because the dictionary of the BOOK\_TITLES file defines the record layout and gives the 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**

Field Name	Type	Field Number	Field Definition	Conversion Code	Column Heading	Output Format	Depth
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 lists the real fields that occupy set positions in each record: the position number is given in the next column.

You can see from the listing that the records held in the BOOK\_TITLES file contain the following fields:

- Title Name
- Author Id
- Reader Id
- Media type
- Units in Stock
- Format
- ISBN
- Department
- Genre
- Publisher
- Publication Year
- Number Allocated

Armed with this information, you can begin to perform some meaningful enquiries.

## Lesson 2: Conversion Codes

---

In the previous lesson you learned to change the way in which a field is displayed on a listing by adjusting the display format and the column headers. You can also change the way data is presented in a more fundamental way. This feature is provided through the medium of **conversion codes**.

A conversion code is used to map data between different representations.

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 to 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 numbers representing the number of days between that date and 'day zero', which is set at 31 December, 1967. The 1<sup>st</sup> January 2007 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. But which format does UniVerse choose?

Dates, times, numbers, and even character data can be displayed in a variety of different formats depending on circumstance and on how you wish to lay out a report. So UniVerse needs a way of defining which format to use in any particular situation.

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 dictionary but that can also be overridden as part of a LIST command.

There are many possible conversion 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, times and with numbers.

Many training courses on RetrieveVe 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 point you at where to look if you need more information.

## Conversion Codes for Character Data

---

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

The main 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 data is case sensitive. If the data in the record is stored in upper case, it will be presented in upper case and all searches performed 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.

**Locating Dirty Data**

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 is a useful assistant in this case: this displays a period (.) in place of any unprintable character when the data is displayed.

## Handling False Positives

Sometimes the use of multivalued 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 - Unabridged 14 Audio Cassette Set)	JUNIOR
	I'm Sorry I Haven't a Clue: Vol 8 (BBC Radio Collection)	ADULT
13660*37800*1	The Decline and Fall of the Roman Empire (Part 2): Audio CDs	ADULT
	A Tale of Two Cities	ADULT
	The Hostile Hospital: Complete & Unabridged (Series of Unfortunate Events)	JUNIOR
	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 Darren Shan S.)	JUNIOR
	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/OK8
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/Q001
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
 01 AUG 2005      15:00         7 32091          11.24 03 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/OK8
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/Q001
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.

## Grouping Reports

Once you have sorted a listing, you will naturally see groups of rows where the same values are repeated. A listing that is sorted by date will display each of records for each date within the selection. This provides opportunities for grouping and for analysis.

A grouped report is one in which the report is broken into sections based on the sort order. A listing of sales orders by date might easily be broken down into sections, with one section per day. This makes for an easier presentation of the information, and it is also the first stage in generating subtotals and in summarising reports.

A report grouping is signalled by the `BREAK.ON` clause:

```
SORT filename BY field BREAK.ON field
```

The `BREAK.ON` field is almost always a sort field, though it need not always be the outermost sort field. The immediate effect of the `BREAK.ON` is to simply add lines to the display that visually distinguish the break points.

The `BREAK.ON` clause has one other effect: the break field, unlike a sort field, is added to the list of fields that is displayed, and therefore overrides the default listing:

### Do this:

```
SORT BOOK_SALES BY SALE_DATE BREAK.ON SALE_DATE SURNAME
```

```
SORT BOOK_SALES BY SALE_DATE BREAK.ON SALE_DATE SURNAME
BOOK_SALES... Sale Date..  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

          ***

13435*34200*1 12 OCT 2004  WETHERELL
13435*36900*1 12 OCT 2004  HILTON
13435*40500*1 12 OCT 2004  ALBERT
13435*40500*3 12 OCT 2004  HOMEWOOD
13435*42300*1 12 OCT 2004  GRATTON
13435*44100*1 12 OCT 2004  SHEFFIELD
```

The `BREAK .ON` clause has added the blank lines and the asterisks to mark the point in the listing where the `SALE_DATE` value changes.

#### Lab 4

The dictionary item `DEPT_GENRE` in the `BOOK_TITLES` file combines the department and genre into a single field. Sort and group the `BOOK_TITLES` based on the values in the `DEPT_GENRE` field.

## Adding Break Text

The break point is signalled by two blank lines and a series of asterisks. This is also a convenient point into which you may choose to place some text.

The `BREAK.ON` keyword optionally allows you to specify some text after the name of the field on which the break is occurring:

```
SORT filename BY field BREAK.ON field "text"
```

The break text must be enclosed in double quotation marks.

#### Do this:

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

```
SORT BOOK_SALES BY SALE_DATE BREAK.ON SALE_DATE "End of Group" SURNAME
BOOK_SALES... Sale Date.. 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

                End of Group

13435*34200*1 12 OCT 2004 WETHERELL
13435*36900*1 12 OCT 2004 HILTON
13435*40500*1 12 OCT 2004 ALBERT
13435*40500*3 12 OCT 2004 HOMEWOOD
13435*42300*1 12 OCT 2004 GRATTON
13435*44100*1 12 OCT 2004 SHEFFIELD
13435*45000*1 12 OCT 2004 FENNEY
```

#### Lab 5

Sort and group the `BOOK_TITLES` based on the values in the `DEPT_GENRE` field. In the line between each break set, add the text: End of Series.

## 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 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.

## Adding Totals and Sub-Totals

Retrieve makes it easy to add totals and subtotals to a report.

If you want to add a total or sub-total for a field, you can instruct Retrieve to add this by placing the keyword TOTAL before the field name:

```
LIST filename TOTAL field
```

Adding a TOTAL keyword at this level produces a final total at the very end of the listing covering the whole set of records selected. All of the individual lines of details are still included in the listing:

**Do this:**

```
LIST BOOK_TITLES SHORT_TITLE TOTAL UNITS
```

```
LIST BOOK_TITLES SHORT_TITLE TOTAL UNITS
Key.. Title..... Units
  245 The Other Side of the Story      3
  246 Forty Years on: AND A Woman of   6
      No Importance
  247 The Ersatz Elevator (Series of   18
      Unfortunate Events)
  248 Little Women: Abridged (Puffin   1
      Classics)
  249 7 Steps to Fearless Speaking    18
  250 Eric                             4
***                                     3336
250 records listed.
```