Notices

Edition
Publication date: July 2008
Book number: UDT-720-UNQR-1
Product version: UniData 7.2

Copyright

Trademarks
The following trademarks appear in this publication:

<table>
<thead>
<tr>
<th>Trademark</th>
<th>Trademark Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rocket Software™</td>
<td>Rocket Software, Inc.</td>
</tr>
<tr>
<td>Dynamic Connect®</td>
<td>Rocket Software, Inc.</td>
</tr>
<tr>
<td>RedBack®</td>
<td>Rocket Software, Inc.</td>
</tr>
<tr>
<td>SystemBuilder™ TM</td>
<td>Rocket Software, Inc.</td>
</tr>
<tr>
<td>UniData®</td>
<td>Rocket Software, Inc.</td>
</tr>
<tr>
<td>UniVerse™ TM</td>
<td>Rocket Software, Inc.</td>
</tr>
<tr>
<td>U2™</td>
<td>Rocket Software, Inc.</td>
</tr>
<tr>
<td>U2.NET™</td>
<td>Rocket Software, Inc.</td>
</tr>
<tr>
<td>U2 Web Development Environment™</td>
<td>Rocket Software, Inc.</td>
</tr>
<tr>
<td>wIntegrate®</td>
<td>Rocket Software, Inc.</td>
</tr>
<tr>
<td>Microsoft® .NET</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>Microsoft® Office Excel®, Outlook®, Word</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>Windows®</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>Windows® 7</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>Windows Vista®</td>
<td>Microsoft Corporation</td>
</tr>
<tr>
<td>Java™ and all Java-based trademarks and logos</td>
<td>Sun Microsystems, Inc.</td>
</tr>
<tr>
<td>UNIX®</td>
<td>X/Open Company Limited</td>
</tr>
</tbody>
</table>
The above trademarks are property of the specified companies in the United States, other countries, or both. All other products or services mentioned in this document may be covered by the trademarks, service marks, or product names as designated by the companies who own or market them.

License agreement

This software and the associated documentation are proprietary and confidential to Rocket Software, Inc., are furnished under license, and may be used and copied only in accordance with the terms of such license and with the inclusion of the copyright notice. This software and any copies thereof may not be provided or otherwise made available to any other person. No title to or ownership of the software and associated documentation is hereby transferred. Any unauthorized use or reproduction of this software or documentation may be subject to civil or criminal liability. The information in the software and documentation is subject to change and should not be construed as a commitment by Rocket Software, Inc.

Restricted rights notice for license to the U.S. Government: Use, reproduction, or disclosure is subject to restrictions as stated in the "Rights in Technical Data-General" clause (alternate III), in FAR section 52.222-14. All title and ownership in this computer software remain with Rocket Software, Inc.

Note

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when exporting this product.

Please be aware: Any images or indications reflecting ownership or branding of the product(s) documented herein may or may not reflect the current legal ownership of the intellectual property rights associated with such product(s). All right and title to the product(s) documented herein belong solely to Rocket Software, Inc. and its subsidiaries, notwithstanding any notices (including screen captures) or any other indications to the contrary.

Contact information

Rocket Software
275 Grove Street Suite 3-410
Newton, MA 02466-2272
USA
Tel: (617) 614-4321 Fax: (617) 630-7100
Web Site: www.rocketsoftware.com
Chapter 1  UniQuery Commands Reference

Elements of Syntax Statements .................................................. 1-2
# .............................................................. 1-3
& .............................................................. 1-4
< .............................................................. 1-5
<= .............................................................. 1-6
= .............................................................. 1-7
<= .............................................................. 1-8
=> .............................................................. 1-9
> .............................................................. 1-10
>= .............................................................. 1-11
AFTER ......................................................... 1-12
ALL ............................................................ 1-13
AND ........................................................... 1-14
ASD ............................................................ 1-17
ASSOCIATED .................................................. 1-18
AVERAGE ..................................................... 1-21
AVG ........................................................... 1-24
BEFORE ....................................................... 1-25
BETWEEN ...................................................... 1-26
BREAK.ON ................................................... 1-28
BREAK.SUP .................................................. 1-33
BSELECT ...................................................... 1-35
BY ............................................................. 1-37
BY.DSND ..................................................... 1-39
BY.EXP ........................................................ 1-41
BY.EXP.DSND ............................................... 1-43
CALC .......................................................... 1-45
CALCULATE .................................................... 1-46
CLEARSELECT ................................................ 1-49
CNV ........................................................... 1-50
COL.HDG ....................................................... 1-51
<table>
<thead>
<tr>
<th>Колонка</th>
<th>Страница</th>
</tr>
</thead>
<tbody>
<tr>
<td>COL.HDR.SUPP</td>
<td>1-52</td>
</tr>
<tr>
<td>COL.SPACES.</td>
<td>1-53</td>
</tr>
<tr>
<td>COL.SPCS.</td>
<td>1-55</td>
</tr>
<tr>
<td>COL.SUP</td>
<td>1-56</td>
</tr>
<tr>
<td>COPY.LIST</td>
<td>1-57</td>
</tr>
<tr>
<td>COUNT</td>
<td>1-60</td>
</tr>
<tr>
<td>COUNT.SUP</td>
<td>1-61</td>
</tr>
<tr>
<td>DBL.SPC</td>
<td>1-63</td>
</tr>
<tr>
<td>DELETE.LIST</td>
<td>1-64</td>
</tr>
<tr>
<td>DET.SUP</td>
<td>1-66</td>
</tr>
<tr>
<td>DICT</td>
<td>1-68</td>
</tr>
<tr>
<td>EACH</td>
<td>1-69</td>
</tr>
<tr>
<td>EDIT.LIST.</td>
<td>1-70</td>
</tr>
<tr>
<td>EQ</td>
<td>1-71</td>
</tr>
<tr>
<td>EQUAL</td>
<td>1-72</td>
</tr>
<tr>
<td>ESEARCH.</td>
<td>1-73</td>
</tr>
<tr>
<td>EVAL</td>
<td>1-75</td>
</tr>
<tr>
<td>EVERY.</td>
<td>1-77</td>
</tr>
<tr>
<td>FMT.</td>
<td>1-78</td>
</tr>
<tr>
<td>FOOTING</td>
<td>1-80</td>
</tr>
<tr>
<td>FORM.LIST</td>
<td>1-83</td>
</tr>
<tr>
<td>GE</td>
<td>1-85</td>
</tr>
<tr>
<td>GET.LIST</td>
<td>1-86</td>
</tr>
<tr>
<td>GRAND.TOTAL</td>
<td>1-88</td>
</tr>
<tr>
<td>GREATER.</td>
<td>1-90</td>
</tr>
<tr>
<td>GT</td>
<td>1-91</td>
</tr>
<tr>
<td>HDR.SUPP</td>
<td>1-93</td>
</tr>
<tr>
<td>HEADER</td>
<td>1-94</td>
</tr>
<tr>
<td>HEADING.</td>
<td>1-95</td>
</tr>
<tr>
<td>ID.ONLY</td>
<td>1-98</td>
</tr>
<tr>
<td>ID.SUPP</td>
<td>1-99</td>
</tr>
<tr>
<td>INTERSECT</td>
<td>1-101</td>
</tr>
<tr>
<td>LE</td>
<td>1-102</td>
</tr>
<tr>
<td>LESS</td>
<td>1-104</td>
</tr>
<tr>
<td>LIKE</td>
<td>1-105</td>
</tr>
<tr>
<td>LIST.</td>
<td>1-109</td>
</tr>
<tr>
<td>LIST.ITEM</td>
<td>1-116</td>
</tr>
<tr>
<td>LIST.LABEL</td>
<td>1-118</td>
</tr>
<tr>
<td>LIST.XMLDATA</td>
<td>1-121</td>
</tr>
<tr>
<td>LPTR</td>
<td>1-122</td>
</tr>
<tr>
<td>LT</td>
<td>1-123</td>
</tr>
<tr>
<td>MARGIN</td>
<td>1-125</td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------</td>
<td>------</td>
</tr>
<tr>
<td>MATCH</td>
<td>1-126</td>
</tr>
<tr>
<td>MERGE.LIST</td>
<td>1-127</td>
</tr>
<tr>
<td>MODIFY</td>
<td>1-131</td>
</tr>
<tr>
<td>NE</td>
<td>1-135</td>
</tr>
<tr>
<td>N.SUP</td>
<td>1-136</td>
</tr>
<tr>
<td>NO</td>
<td>1-137</td>
</tr>
<tr>
<td>NO.INDEX</td>
<td>1-138</td>
</tr>
<tr>
<td>NO.NULLS</td>
<td>1-139</td>
</tr>
<tr>
<td>NO.PAGE</td>
<td>1-142</td>
</tr>
<tr>
<td>NO.SPLIT</td>
<td>1-144</td>
</tr>
<tr>
<td>NOT</td>
<td>1-146</td>
</tr>
<tr>
<td>NOT.MATCH</td>
<td>1-147</td>
</tr>
<tr>
<td>NSELECT</td>
<td>1-148</td>
</tr>
<tr>
<td>NULLVAL</td>
<td>1-150</td>
</tr>
<tr>
<td>ONLY</td>
<td>1-152</td>
</tr>
<tr>
<td>OR</td>
<td>1-154</td>
</tr>
<tr>
<td>PCT</td>
<td>1-156</td>
</tr>
<tr>
<td>PERCENT</td>
<td>1-157</td>
</tr>
<tr>
<td>QSELECT</td>
<td>1-159</td>
</tr>
<tr>
<td>REFORMAT</td>
<td>1-161</td>
</tr>
<tr>
<td>REQUIRE.INDEX.</td>
<td>1-163</td>
</tr>
<tr>
<td>REQUIRE.SELECT</td>
<td>1-164</td>
</tr>
<tr>
<td>SAMPLE</td>
<td>1-166</td>
</tr>
<tr>
<td>SAMPLED</td>
<td>1-167</td>
</tr>
<tr>
<td>SAVE.LIST</td>
<td>1-168</td>
</tr>
<tr>
<td>SAVING</td>
<td>1-172</td>
</tr>
<tr>
<td>SELECT</td>
<td>1-174</td>
</tr>
<tr>
<td>Using Select Lists</td>
<td>1-175</td>
</tr>
<tr>
<td>SELECT.ONLY.</td>
<td>1-178</td>
</tr>
<tr>
<td>SORT</td>
<td>1-179</td>
</tr>
<tr>
<td>SORT.ITEM.</td>
<td>1-182</td>
</tr>
<tr>
<td>SORT.LABEL</td>
<td>1-185</td>
</tr>
<tr>
<td>SORT.LIST</td>
<td>1-188</td>
</tr>
<tr>
<td>SREFORMAT</td>
<td>1-190</td>
</tr>
<tr>
<td>SSELECT</td>
<td>1-192</td>
</tr>
<tr>
<td>Using Select Lists</td>
<td>1-193</td>
</tr>
<tr>
<td>SUM</td>
<td>1-196</td>
</tr>
<tr>
<td>SUPP</td>
<td>1-198</td>
</tr>
<tr>
<td>TOTAL</td>
<td>1-199</td>
</tr>
<tr>
<td>UNLIKE</td>
<td>1-201</td>
</tr>
<tr>
<td>USING</td>
<td>1-204</td>
</tr>
<tr>
<td>VAL.OF</td>
<td>1-205</td>
</tr>
<tr>
<td>Term</td>
<td>Pages</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>VERT</td>
<td>1-207</td>
</tr>
<tr>
<td>VERTICAL</td>
<td>1-208</td>
</tr>
<tr>
<td>WHEN</td>
<td>1-209</td>
</tr>
<tr>
<td>WITH</td>
<td>1-212</td>
</tr>
<tr>
<td>WITHIN</td>
<td>1-215</td>
</tr>
</tbody>
</table>
UniQuery Commands Reference

This chapter contains a detailed alphabetic listing with descriptions and working examples of UniQuery commands and keywords.

Keywords provide selection criteria, sorting criteria, and formatting options. You always use keywords following the command and file name in a UniQuery statement.

Throwaway keywords are listed in Using UniQuery, and are not listed in this manual since they have no function.
Elements of Syntax Statements

This reference manual uses a common method for stating syntax for UniData commands. The syntax statement includes the command name, required arguments, and options that you can use with the command. Italics represents a variable that you can replace with any valid option. The following figure illustrates the elements of a syntax statement:

- **command names** appear in boldface
- **square brackets** indicate an optional argument
- **no brackets or braces** indicates a required argument
- **a vertical line** indicates that you may choose between the given arguments
- **braces** indicate that you must choose between the given arguments
- **an ellipsis** indicates that you may enter more than one argument
- **quotation marks** must enclose a literal string
#

# is a synonym for the NE keyword. For further information, see NE.

**Synonyms**

NE
&

& is a synonym for the AND keyword. For further information, see AND.

Synonym

AND
<

< is a synonym for the LT keyword. For further information, see LT.

**Synonyms**

BEFORE, LESS, LT
<=

<= is a synonym for the LE keyword. For further information, see LE.

Synonyms

=<=, LE
= is a synonym for the EQUAL keyword. For further information, see EQ.

**Synonyms**

EQ, EQUAL
<=

<= is a synonym for the LE keyword. For further information, see LE.

**Synonyms**

<=, LE
=>

=> is a synonym for the GE keyword. For further information, see GE.

**Synonyms**

>=, GE
> is a synonym for the GT keyword. For further information, see GT.

**Synonyms**

AFTER, GREATER, GT
>=

>= is a synonym for the GE keyword. For further information, see GE.

**Synonyms**

=>, GE
AFTER

AFTER is a synonym for the GT keyword. For further information, see GT.

Synonyms

>, GREATER, GT
**ALL**

**Syntax**

... filename ALL

**Description**

The UniQuery **ALL** keyword displays every data (D-type) attribute defined in the dictionary of *filename*. ALL does not display virtual (V-type) attributes.

**Example**

In the following example, the ALL keyword lists every D-type attribute in the INVENTORY file:

```
:LIST INVENTORY ALL
LIST INVENTORY ALL 10:25:34 Mar 25 1999 1
Product Number  15001
Inventory Date   08/20/1995
Inventory Time   01:00PM
Product Name     Modem
Features         14.4K Internal V34
Color        Quantity Price      Reorder
N/A           7486    $119.00      40

Product Number  35000
Inventory Date   07/09/1995
Inventory Time   10:00AM
Product Name     Speaker
Features         250W, Direct/reflecting
Color        Quantity Price      Reorder
Black           148    $198.93      50
Charcoal        125    $198.93      50
.            .               .          .
```
**AND**

**Syntax**

... *phrase1 AND phrase2*

**Synonym**

&

**Description**

The UniQuery **AND** keyword is the AND operator. AND is used in UniQuery to join selection criteria phrases and to join **WITH** and **WHEN** clauses. When you use AND, UniQuery returns only those records where both *phrase1* and *phrase2* are true.

The AND and **OR** operators have equal precedence in a UniQuery statement if you do not specify the hierarchy using parentheses. Without parentheses, UniQuery evaluates the statement from left to right.
Examples

In the following example, UniQuery retrieves records where both phrase1 and phrase2 are true, or phrase3 is true:

```
:LIST INVENTORY WITH PROD_NAME = "TV" AND INV_DATE LT "01/01/96"
OR WITH FEATURES LIKE "...Portable..." PROD_NAME INV_DATE FEATURES
LIST INVENTORY WITH PROD_NAME = "TV" AND INV_DATE LT "01/01/96" OR
WITH FEATURES LIKE "...Portable..." PROD_NAME INV_DATE FEATURES
11:09:33 Jun 25 2005 1

Product    Inventory
INVENTORY. Name...... Date...... Features......................

52070      Printer    01/23/1996 Portable Color, 3 ppm
11070      TV         06/21/1995 2" Color, Portable
11080      TV         07/13/1995 9" Color, Portable, Remote
          Control
12006      Printer    08/22/1995 Portable Color Ink Jet
11010      TV         06/20/1995 31" Color High Resolution
30000      CD System  06/01/1995 Portable Clock Radio

```

In the previous example, UniQuery evaluates the statements in the following way:

- Retrieves all records with PROD_NAME equal to TV AND INV_DATE less than 01/01/96. These records meet both the selection criteria in the phrase1 (PROD_NAME = "TV") and the selection criteria in phrase2 (INV_DATE LT "01/01/96").
- Retrieves all records that meet the selection criteria in phrase3 (FEATURES LIKE "...Portable..."), which is joined to phrase2 by the OR operator.

By using parentheses, you can alter the order in which UniQuery processes a statement that contains AND and OR. UniQuery still processes the statement from left to right, but evaluates conditions within parentheses as a group.
In the following example, UniQuery evaluates the same UniQuery statement used in the preceding example in a different way because parentheses are added to change the order of processing the statement:

```
:LIST INVENTORY WITH PROD_NAME = "TV" AND (INV_DATE LT "01/01/96"
OR FEATURES LIKE "...Portable...") PROD_NAME INV_DATE FEATURES
```

In the previous example, every record must first meet the selection criteria in phrase1 (PROD_NAME = "TV"). Then, UniQuery evaluates this subset of records and retrieves only those that meet the selection criteria in phrase2 (INV_DATE LT "01/01/96") OR that meet the selection criteria in phrase3 (FEATURES LIKE "...Portable..."). The records UniQuery selects differ from the previous example because phrase2 and phrase3 are evaluated as a group by adding parentheses, and only one of the conditions has to be true to satisfy the selection criteria.

**Related Command**

OR
ASD

ASD is a synonym for the ASSOCIATED keyword. For further information, see ASSOCIATED.

Synonym

ASSOCIATED
ASSOCIATED

Syntax

...WHEN [ASSOCIATED] [EVERY] selection_criteria [AND | OR] [EVERY] selection_criteria ...

Synonym

ASD

Description

The UniQuery ASSOCIATED keyword is used in conjunction with the WHEN keyword, and operates on multivalued and multi-subvalued attributes. The ASSOCIATED keyword returns only those records, and values within those records, that have values in the same position in the multivalued or multi-subvalued attribute matching both criteria you specify.

Attributes listed in an ASSOCIATED phrase must have an association defined in attribute 7 of the dictionary record. The association definitions must be the same for the associated attributes, and the association phrase record must be defined in the dictionary.

For information on creating associations, see Using UniData.

Note: UDT.OPTIONS 22 determines whether UniQuery WITH and WHEN comparisons use the numeric value or the string value of data. When this option is on, under certain conditions, the comparison uses the string value of the data. When this option is off, UniQuery uses standard comparisons.

UDT.OPTIONS 94 affects UniQuery statements that use a WHEN clause with two or more associated multivalued or multi-subvalued attributes. In this kind of statement, UDT.OPTIONS 94 on makes a WHEN clause the same as a WHEN ASSOCIATED clause.
Parameters

The following table lists the parameters of the ASSOCIATED syntax.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVERY</td>
<td>Returns only those records, and values within those records, that have values in the same position in the multivalued or multi-subvalued attribute matching both specified criteria.</td>
</tr>
<tr>
<td>selection_criteria</td>
<td>Conditions for retrieving or bypassing a record. UniQuery only retrieves records meeting the selection criteria.</td>
</tr>
</tbody>
</table>

Associated Parameters

Example

In the following example, UniQuery returns only those values from records in the ORDERS file that have a quantity greater than 10 and a price greater than $100.00, using WHEN ASSOCIATED:

```
:LIST ORDERS WHEN ASSOCIATED(QTY GT '10' AND PRICE GT "100.00")
QTY   PRICE
872   45   $129.87
877   45   $199.95
878   25   $179.99
971   15   $479.99
818   500  $119.95
       500  $1,200.00
       100  $395.00
       100  $500.00
       100  $1,200.00
       250  $199.95
       250  $695.00
       250  $199.95
789   100  $228.82
       50   $249.96
       50   $249.96
       850  $179.97
790   1000 $159.94
```
Related Commands

EVERY, WHEN
AVERAGE

Syntax

...AVERAGE attribute

Synonym

AVG

Description

The UniQuery AVERAGE keyword is used with the LIST or SORT command to calculate and print the average value for a numeric attribute at the end of the report or at defined breakpoints.

UniData does not include null (or any other nonnumeric) values when calculating average. Empty strings are included in average calculation unless you include the NO.NULLS keyword.

Note: UniQuery only supports the AVERAGE keyword in ECLTYPE U.
Examples

In the following example, the AVERAGE keyword returns the average quantity for each product in the INVENTORY file.

```
LIST INVENTORY BY PROD_NAME BREAK.ON PROD_NAME AVERAGE QTY
LIST INVENTORY BY PROD_NAME BREAK.ON PROD_NAME AVERAGE QTY
14:52:59 Jun 25 2005 1
Product
INVENTORY. Name...... Quantity

10007 Adapter         544
13001 Adapter         467
13002 Adapter         104
********** --------
Adapter         372

39400 CD Player       399
39400 CD Player       499
39500 CD Player       -551
********** --------
CD Player       116

30000 CD System       310
1
197
********** --------
CD System       254
1
```

The following two examples demonstrate how UniData processes null values and empty strings when calculating AVERAGE. (This processing is the same regardless of whether null value handling is turned on.)
In this first example, values in the PRICE attribute in records 55050 and 55020 of the INVENTORY file have been changed to the null value. As expected, these values are not included in the output.

```
LIST INVENTORY AVERAGE PRICE WHEN PRICE < 10
```

```
LIST INVENTORY AVERAGE PRICE WHEN PRICE < 10 11:12:33 Jun 16 2005
1
INVENTORY. Price.....

55090 $8.99
55060 $5.99
56080 $3.99
$3.99
$3.99
51070 $9.99
$9.99
$9.99
$9.99
55000 $7.95
55040 $9.99
55010 $8.99

=========
AVERAGE $7.82
7 records listed
```

Before executing the next example, we changed values in the PRICE attribute in records 55050 and 55020 of the INVENTORY file to empty strings. Because empty strings are included in calculation of averages, the number of records increases from 7 to 9, and the average decreases from $7.82 to $6.70.

```
LIST INVENTORY AVERAGE PRICE WHEN PRICE < 10
```

```
LIST INVENTORY AVERAGE PRICE WHEN PRICE < 10 11:13:18 Jun 16 2005
1
INVENTORY. Price.....

55020
55050
55090 $8.99
55060 $5.99
56080 $3.99
$3.99
$3.99
51070 $9.99
$9.99
$9.99
$9.99
55000 $7.95
55040 $9.99
55010 $8.99

=========
AVERAGE $6.70
9 records listed
```
AVG

AVG is a synonym for the AVERAGE keyword. For further information, see AVERAGE.

Synonym

AVERAGE
BEFORE

BEFORE is a synonym for the LT keyword. For further information, see LT.

Synonyms

<, LESS, LT
**BETWEEN**

Syntax

...[WITH | WHEN] attribute BETWEEN "string1" "string2"

Description

The UniQuery **BETWEEN** keyword returns records in *attribute* that are greater than *string1* and less than *string2*. *string2* must be greater than *string1*.

**Note**: In **ECLTYPE U**, UniQuery retrieves numeric attributes inclusive of *string1* and *string2* in the selection criteria. If *string1* and *string2* are alphanumeric, UniQuery includes *string1*, but excludes *string2*. You can enclose *string1* and *string2* in single or double quotation marks.

In **ECLTYPE P**, UniQuery retrieves numeric attributes exclusive of *string1* and *string2* in the selection criteria. If *string1* and *string2* are alphanumeric, UniQuery includes *string1*, but excludes *string2*. You must enclose both *string1* and *string2* in double quotation marks.
Example

In the following example, UniQuery displays records from the ORDERS file with an order date between 01/01/96 and 01/12/96:

```
:ECLTYPE U
:ELIST ORDERS WITH ORD_DATE BETWEEN "01/01/96" "01/12/96" BY
ORD_DATE ORD_DATE
LIST ORDERS WITH ORD_DATE BETWEEN "01/01/96" "01/12/96" BY
ORD_DATE ORD_DATE 11:37:54 Jun 26 2005 1
ORDERS.... Order Date

888      01/01/96
858      01/02/96
859      01/05/96
889      01/05/96
891      01/06/96
860      01/07/96
892      01/09/96
893      01/10/96
856      01/12/96
895      01/12/96
10 records listed
```
BREAK.ON

Syntax

...BY attribute BREAK.ON ["['option' ...][text]"] attribute

Synonym

BREAK-ON

Description

The UniQuery BREAK.ON keyword is used with the LIST and SORT commands to display attribute and create a break in a report when the value of attribute changes. UniQuery can display subtotals, averages, percentages, and other calculated values at each breakpoint.

When you use BREAK.ON, UniQuery, by default, displays a line of asterisks (***) equal to the length of attribute, following the last record before the breakpoint. The value of the last breakpoint prints below the asterisks. If you use the TOTAL, AVERAGE, PERCENT or CALCULATE keyword in conjunction with BREAK.ON, a row of dashes (---) displays on the same line as the asterisks to indicate a subtotal. You can substitute text of your choice for the line of asterisks by using the text option.
option Parameters

The following table describes the *option* parameters available with the BREAK.ON keyword.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Used in conjunction with the HEADING or FOOTING keywords. When you specify the &quot;B&quot; option in the heading or footing definition, UniQuery takes the value of the attribute name at the breakpoint and inserts it at the designated position in the heading or footing definition. If you specify more than one BREAK.ON &quot;B&quot; option, UniQuery only uses the first one it encounters in the statement. Each time the breakpoint value changes, UniQuery creates a new page.</td>
</tr>
<tr>
<td>D</td>
<td>Suppresses the breakpoint line if only one line of detail is present when the attribute value changes.</td>
</tr>
<tr>
<td>L</td>
<td>Suppresses the display of the line of asterisks at the breakpoint, but generates one line feed when an attribute value changes.</td>
</tr>
<tr>
<td>O</td>
<td>Only displays the breakpoint value once in the detail portion of the report.</td>
</tr>
<tr>
<td>P or ^</td>
<td>Creates a new page for each breakpoint value.</td>
</tr>
<tr>
<td>V</td>
<td>Inserts the value of the breakpoint attribute on the breakpoint line. The breakpoint value is printed in the breakline at the V position.</td>
</tr>
<tr>
<td>T</td>
<td>Displays a line of dashes (---) before the subtotal when you use more than one BREAK.ON with the TOTAL keyword and the DET.SUP keyword.</td>
</tr>
<tr>
<td>text</td>
<td>Specifies a text string to display at the breakpoint, replacing display of the default asterisk string.</td>
</tr>
</tbody>
</table>

**BREAK.ON Options**

*Note:* In ECLTYPE P, the BREAK.ON options must appear after the breakpoint attribute. In ECLTYPE U, the BREAK.ON options must appear before the breakpoint attribute.
When UDT.OPTIONS 25 is on, UniQuery overrides the 'L' option and prints the breakpoint text. If you turn the option on and use the DET.SUP keyword in the UniQuery statement, UniQuery suppresses the detail lines and break lines, only printing the breakpoint values.

UDT.OPTIONS 28 determines how a UniQuery report with a BREAK.ON clause and vertical output displays the breakpoint value. If this option is on, the breakpoint section displays only the column heading and the value producing the breakpoint. When this option is off, UniQuery displays all column headings and the value producing the breakpoint.

In a UniQuery report that prints vertically, UDT.OPTIONS 30 determines if UniQuery displays the breakpoint messages “start to break” and “finish breaking.” With this option on, UniQuery does not display the messages. With the option off, UniQuery displays both messages.

UDT.OPTIONS 43 determines the information that UniQuery prints at the breakpoint when DET.SUP is part of the UniQuery statement. With this option on, UniQuery displays the breakpoint value and the detail of the last value before the breakpoint. When this option is off, UniQuery does not display the last value before the breakpoint.

UDT.OPTIONS 47 determines how UniQuery calculates percentages at breakpoints. When UDT.OPTIONS 47 is on, UniQuery calculates percentages before rounding detail lines when the PERCENT keyword is used in a UniQuery statement. When this option is off, UniQuery calculates breakpoint percentages after rounding detail lines.

When UDT.OPTIONS 79 is on, break levels stay together on a page when you use the 'P' option. When UDT.OPTIONS 79 is off, break levels may be split across pages.

When UDT.OPTIONS 80 is on, UniQuery forces detail for multiple breakpoints to stay together on one page when the NO.SPLIT keyword is used. When this option is off, detail for one of the breakpoints could appear alone on a separate page.

When UDT.OPTIONS 98 is on, the breakpoint value does not print on the subtotal line, it only prints on the total line at each breakpoint.

When UDT.OPTIONS 108 is on, you can create a dictionary record to display the number of records at each breakpoint. If UDT.OPTIONS 108 is off, you cannot use this feature. For information about creating this type of virtual attribute, see Using UniQuery.
Examples

The following example illustrates using BREAK.ON with the 'B' option. Each “Enter <New line> to continue...” represents a new page. Notice that the breakpoint value appears in the heading:

```plaintext
:LIST ORDERS BY ORD_DATE BY PROD_NAME BREAK.ON "'B'" ORD_DATE
PROD_NAME HEADING "ORDERS RECEIVED ON 'B'
ORDERS RECEIVED ON 05/13/95
ORDERS.... Order Date Product Name

838 05/13/95 Memory
     Scanner
     Computer
     Telephone
     Telephone

**********
05/13/95

Enter <New line> to continue...

ORDERS RECEIVED ON 06/19/95
ORDERS.... Order Date Product Name

839 06/19/95 Computer
     Case

**********
06/19/95
```

In the next example, the 'D' option is used to suppress the breakpoint line when only one line of detail exists for the breakpoint.

```plaintext
:LIST ORDERS BY ORD_DATE BY PROD_NAME BREAK.ON "'D'" ORD_DATE
PROD_NAME
.
.
.

839 06/19/95 Computer
     Case

840 06/20/95 Video 16
     Memory
     Cable

841 07/23/95 Video 16
     Telephone

842 08/12/95 Printer
.
.
.
```
In the next example, the 'O' and 'L' options are combined in the same UniQuery statement. UniQuery prints the breakpoint attribute only once in the detail section of the report, and suppresses printing the line of asterisks at each breakpoint.

```
:LIST CLIENTS BY STATE BREAK.ON "'OL'" STATE CITY

10024      BC              Vancouver
10029      Vancouver
10033      Vancouver
10089      Vancouver
9988       Vancouver

9982       CA              Los Angeles
9986       Fresno

10047      CO              Portland
9987       Colo Spgs.
```

The next example illustrates the 'T' option. When you use the 'T' option with the DET.SUP keyword and the TOTAL keyword, UniQuery displays a line of dashes (---) before the subtotal at the breakpoint.

```
:LIST INVENTORY BY PROD_NAME BY INV_DATE BREAK.ON "'T'" PROD_NAME BREAK.ON "'T'" INV_DATE TOTAL QTY DET.SUP
LIST INVENTORY BY PROD_NAME BY INV_DATE BREAK.ON "'T'" PROD_NAME BREAK.ON "'T'" INV_DATE TOTAL QTY DET.SUP
13:50:42 Jun 28 2005 1

Product    Inventory
Name........ Date....... Quantity

06/07/1995      544
08/12/1995      467
08/18/1995      104
--------
Adapter                   1115
08/08/1995      898
08/30/1995     -551
--------
CD Player              347
```

**Related Command**

BREAK.SUP
BREAK.SUP

Syntax

...BY attribute BREAK.SUP "['option' ...][text]" attribute

Synonym

BREAK-SUP

Description

The UniQuery BREAK.SUP keyword is used with the LIST and SORT commands to create a break in a report when the value of attribute changes, but suppresses printing of the breakpoint value. Even though UniQuery does not display the breakpoint value, the output is separated by a blank line when a breakpoint occurs.

option Parameters

The following table describes the option parameters available with the BREAK.SUP keyword.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Used in conjunction with the HEADING or FOOTING keyword. When the &quot;B&quot; option is specified in the heading or footing definition, UniQuery takes the value of the attribute name at the breakpoint and inserts it at the position you designate in the heading or footing definition. If you specify more than one BREAK.ON &quot;B&quot; option, UniQuery only uses the first one it encounters in the statement. Each time the breakpoint value changes, UniQuery creates a new page.</td>
</tr>
<tr>
<td>P</td>
<td>Creates a new page for each breakpoint value.</td>
</tr>
<tr>
<td>T</td>
<td>Displays a line of dashes (---) before the subtotal when you use BREAK.SUP with the TOTAL keyword and the DET.SUP keyword.</td>
</tr>
</tbody>
</table>

BREAK.SUP Options
Example

In the following example, the BREAK.SUP keyword organizes the output by country, but does not display the country. The country is included in the heading of the report by use of the "'B'" option:

```
:LIST CLIENTS BY COUNTRY BREAK.SUP "'B'" COUNTRY LNAME CITY
HEADING "CLIENTS RESIDING IN 'B'"
CLIENTS RESIDING IN Australia
CLIENTS... Last Name...... City........
10007      Sloane          Kalgoorlie
10009      Wilbanks        Abbotsford
10021      Singh           Lindfield
10022      O'Connell       Strathfield
10023      Mirani          Canberra
10025      Kingston        Sydney
.
.
.
CLIENTS RESIDING IN Canada
CLIENTS... Last Name...... City........
10006      Kesić           Winnipeg
10011      Lomonosov       Fonthill
10015      di Grigorio     Regina
10017      Wu              Edmonton
10020      Andropolis      Calgary
10024      Guo             Vancouver
.
.
.
```

Related Command

BREAK.ON
BSELECT

Syntax

BSELECT filename ['record_IDs'] [selection_criteria] attribute [attribute...]

Description

The UniQuery BSELECT command retrieves data from a file into an active select list. Unlike the SELECT command, which retrieves only record IDs, BSELECT builds a list of the attributes you name in the UniQuery statement. You must name at least one attribute in the statement.

Note: UDT.OPTIONS 59 determines if a BSELECT statement creates a blank line in a select list if the designated attribute does not exist in a record. When UDT.OPTIONS 59 is on, UniQuery does not create a blank line for each nonexisting attribute. When UDT.OPTIONS 59 is off, a blank line is created for each nonexisting attribute.

Parameters

The following table describes the parameters of the BSELECT command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Name of file on which operations are to be performed. You may specify only one file name in a UniQuery statement.</td>
</tr>
</tbody>
</table>

BSELECT Parameters
Example

BSELECT is useful when selecting attributes from one file that are record IDs in another file. In the following example, BSELECT retrieves the client numbers from the ORDERS file, then a UniQuery statement lists the names and addresses of those clients from the CLIENTS file.

```
>BSELECT ORDERS CLIENT_NO

192 records selected to list 0.

>LIST CLIENTS NAME ADDRESS
LIST CLIENTS NAME ADDRESS 08:50:00 Jun 02 2005 1
CLIENTS... Name..........................
Address..................
9988       Dominic Warner                 7235 Laguna Blvd
            Suite 720
9987       Glen Asakawa                   220 Pearl
10002      Aude Grenelle                  Av. Bourgailh
9979       Andrea Herriot                 91, promenade Plage
9978       Mike Vidulich                  165 Market Street
.          .                              .
```

Related Commands

BSELECT, QSELECT, SELECT, SSELECT
BY

Syntax

...BY attribute

Description

The UniQuery BY keyword sorts attributes in ascending order with the LIST, SORT, and SELECT commands. attribute specifies the attribute to be sorted. You can use more than one BY sorting clause in a UniQuery statement. Since UniQuery evaluates a statement from left to right, the first BY clause defines the primary sort, the second BY clause defines the secondary sort, etc.

For multivalued and multi-subvalued attributes, the BY keyword treats the entire attribute as one unit, rather than individual values, in effect using only the first value for sorting purposes. To sort each value, use the BY.EXP keyword.

UniQuery uses the justification you define in the dictionary record of attribute to determine if attribute is left-justified or right-justified.

Example

In the following example, the BY keyword performs an ascending sort on the LNAME attribute in the CLIENTS file.

```
LIST CLIENTS BY LNAME LNAME
LIST CLIENTS BY LNAME LNAME 12:08:03 Jun 02 2005 1
CLIENTS... Last Name......

   9968 Adams
  10077 Albertson
  10054 Alps
  10034 Anderson
  10020 Andropolis
```
Note: UniQuery uses the attributes you specify with the BY clause for sorting purposes only; it does not display the attributes in the report. You must enter the attributes you want to display in the UniQuery statement. If you do not specify any attributes, UniQuery only displays the record IDs meeting the selection criteria, unless an @UQ phrase exists in the dictionary of the file.

Related Commands

BY.DSND, BY.EXP, BY.EXP.DSND
BY.DSND

Syntax

...BY.DSND attribute

Synonym

BY-DSND

Description

The UniQuery **BY.DSND** keyword sorts attributes in descending order with the **LIST**, **SORT**, and **SELECT** commands. *attribute* specifies the attribute to be sorted. You can use more than one BY.DSND sorting clause in a UniQuery statement. Since UniQuery evaluates a statement from left to right, the first BY.DSND clause defines the primary sort, the second BY.DSND clause defines the secondary sort, etc.

For multivalued and multi-subvalued attributes, the BY.DSND keyword treats the entire attribute as a one unit, rather than individual values, in effect using only the first value for sorting purposes. To sort each value, use the **BY.EXP.DSND** keyword.

UniQuery uses the justification you define in the dictionary record of *attribute* to determine if *attribute* is left-justified or right-justified.
Example

In the following example, the BY.DSND keyword performs an descending sort on the LNAME attribute in the CLIENTS file:

```plaintext
:LIST CLIENTS BY.DSND LNAME LNAME
LIST CLIENTS BY.DSND LNAME LNAME 14:35:25 Jun 02 2005 1
CLIENTS... Last Name......
  10079  Zhu
  9974  Zabeel
  10041  Yellowstone
  10017  Wu
  10033  Witherspoon
  9982  Willette
```

*Note:* UniQuery uses the attributes you specify with the BY.DSND clause for sorting purposes only; it does not display the attributes in the report. You must enter the attributes you want to display in the UniQuery statement. If you do not specify any attributes, UniQuery only displays the record IDs meeting the selection criteria, unless an @UQ phrase exists in the dictionary of the file.

Related Commands

BY, BY.EXP, BY.EXP.DSND
**BY.EXP**

**Syntax**

...BY.EXP attribute

**Synonym**

BY-EXP

**Description**

The UniQuery **BY.EXP** keyword sorts values in multivalued or multi-subvalued attributes in ascending order in a **LIST**, **SORT**, or **SELECT** statement. UniQuery “explodes” each value in a multivalued or multi-subvalued attribute so each value is treated as a singlevalued attribute. EXP stands for exploding.

You can use more than one **BY.EXP** sorting clause in a UniQuery statement. Since UniQuery evaluates a statement from left to right, the first **BY.EXP** clause defines the primary sort, the second **BY.EXP** clause defines the secondary sort, etc.

UniQuery uses the justification you define in the dictionary record of `attribute` to determine if `attribute` is left-justified or right-justified.

**Note:** **UDT.OPTIONS 76** determines how UniData treats a **BY.EXP** clause in a statement when a virtual attribute does not contain an association. When this option is on, UniQuery calculates the virtual attribute according to the raw data, then extracts the values and subvalues according to the **BY.EXP** clause. When this option is off, UniQuery calculates the virtual attribute after extracting the values and subvalues.
Example

In the following example, the BY.EXP keyword explodes the multivalued QTY attribute in the INVENTORY file and produces a report in ascending order. Record IDs appear multiple times because each multivalued attribute is treated as a single-valued attribute, containing the same record ID.

```
LIST INVENTORY BY.EXP QTY PROD_NAME QTY
LIST INVENTORY BY.EXP QTY PROD_NAME QTY 16:50:15 Jun 02 2005 1
Product
INVENTORY. Name...... Quantity
   11050 TV            -794
   39500 CD Player     -551
   11130 Video 12      -67
   38000 CD System     -49
   51020 Telephone     1
   50090 Computer      2
   54040 Hard Drive    2
   ...
   40010 Telephone     100
   40010 Telephone     100
   40014 Telephone     100
   40014 Telephone     100
   51040 Telephone     100
   51040 Telephone     100
   51040 Telephone     100
   ...
```

**Note:** UniQuery uses the attributes you specify with the BY.EXP clause for sorting purposes only; it does not display the attributes in the report. You must enter the attributes you want to display in the UniQuery statement. If you do not specify any attributes, UniQuery displays only the record IDs meeting the selection criteria, unless an @UQ phrase exists in the dictionary of the file.

Related Commands

BY, BY.DSND, BY.EXP.DSND
BY.EXP.DSND

Syntax

...BY.EXP.DSND attribute

Synonyms

ECLTYPE U
BY-EXP-DSND, BYEXPDSND

ECLTYPE P
BY-EXP-DSND

Description

The UniQuery BY.EXP.DSND keyword sorts values in multivalued or multi-subvalued attributes in descending order in a LIST, SORT, or SELECT statement. UniQuery “explodes” each value in a multivalued or multi-subvalued attribute so each value is treated as a singlevalued attribute. EXP stands for exploding.

You can use more than one BY.EXP.DSND sorting clause in a UniQuery statement. Since UniQuery evaluates a statement from left to right, the first BY.EXP.DSND clause defines the primary sort, the second BY.EXP.DSND clause defines the secondary sort, and so forth.

UniQuery uses the justification you define in the dictionary record of attribute to determine if attribute is left-justified or right-justified.
Example

In the following example, the BY.EXP.DSND keyword explodes the multivalued QTY attribute in the INVENTORY file and produces a report in descending order. Record IDs appear multiple times because each multivalued attribute is treated as a singlevalued attribute, containing the same record ID.

```
:LIST INVENTORY BY.EXP.DSND QTY PROD_NAME QTY
LIST INVENTORY BY.EXP.DSND QTY PROD_NAME QTY 17:00:33 Jun 02 2005

Product INVENTORY. Name...... Quantity

10150 Camera 14985
10140 Camera 12000
10050 Camera 9833
10130 Camera 9649
.
.
.
51070 Telephone 1000
50010 Computer 999
50000 Computer 988
50010 Computer 950
50010 Computer 908
52030 Printer 877
40001 Telephone 849
.
.
.
```

Note: UniQuery uses the attributes you specify with the BY.EXP.DSND clause for sorting purposes only; it does not display the attributes in the report. You must enter the attributes you want to display in the UniQuery statement. If you do not specify any attributes, UniQuery displays only the record IDs meeting the selection criteria, unless an @UQ phrase exists in the dictionary of the file.

Related Commands

BY, BY.DSND, BY.EXP
CALC

CALC is a synonym for the CALCULATE keyword. For further information, see CALCULATE.

Note: CALC and CALCULATE are not supported in ECLTYPE P.

Synonym

CALCULATE
CALCULATE

Syntax

...CALCULATE virtual_attribute

Synonym

CALC

Description

The UniQuery CALCULATE keyword performs total calculations using the LIST or SORT command. You use CALCULATE in a UniQuery statement in conjunction with a virtual attribute that contains the TOTAL function. Together, CALCULATE and virtual attributes can create reports with calculated subtotals, percentages, and averages. The intermediate values CALCULATE produces are printed at breakpoints and the final line of the report.

The TOTAL function is used in a virtual attribute to accumulate subtotals for numeric attributes. These subtotals can then be used to calculate totals and averages, and to determine percentages using the CALCULATE (or CALC) keyword. The calculated values are printed on breakpoint lines and the final line of the report.

*Note:* The TOTAL function is not a UniBasic function, and differs from the UniQuery TOTAL keyword.

*Note:* CALC and CALCULATE are not supported in ECLTYPE P.
Examples

Producing results with the CALCULATE keyword is a two-step process. First, create a virtual field using the TOTAL function for each field for which you want to accumulate subtotals. In the following example, the INV_COST virtual attribute created in the dictionary of the INVENTORY file multiplies PRICE times QTY when PRICE is greater than 0:

```
:AE DICT INVENTORY INV_COST
Top of "INV_COST" in "DICT INVENTORY", 6 lines, 78 characters.
  *--: P
  001: V
  002: IF TOTAL(PRICE) > 0 THEN TOTAL(PRICE)*TOTAL(QTY) ELSE TOTAL"
  003: MD2,
  004:
  005: 13R
  006: S
  Bottom.
  *--:
```

Now, use the CALCULATE keyword along with the virtual attribute just created to display totals. In the following example, records from the INVENTORY file are displayed with a breakpoint on PROD_NAME, totals are displayed for the QTY and PRICE attributes, and totals for INV_COST are calculated both at the breakpoints and as a grand total.

```
:LST INVENTORY BY PROD_NAME BREAK.ON PROD_NAME TOTAL QTY TOTAL
PRICE CALCULATE INV_COST
LIST INVENTORY BY PROD_NAME BREAK.ON PROD_NAME TOTAL QTY TOTAL
PRICE CALCULATE INV_COST 11:06:54 Jun 23 2005 1
Product
INVENTORY. Name...... Quantity Price......  ............
  14001 Memory 6131 $49.95 306,243.45
  ********** ----- --------- ------------
  Memory 6131 $49.95 306,243.45

  15001 Modem 7486 $119.00 890,834.00
  15002 Modem 3988 $199.99 797,560.12
  15003 Modem 4913 $259.99 1,277,330.87
  15004 Modem 146 $219.99 32,118.54
  ********** ----- --------- ------------
  Modem 16533 $798.97 13,209,371.01

  ========= =========== =============
  22664 $848.92 19,239,922.88

5 records listed
```
If you use the TOTAL keyword with INV_COST rather than CALCULATE, UniQuery displays the total of the breakpoint for the INV_COST column, rather than the calculated total of QTY * PRICE at the breakpoint.

```plaintext
>LIST INVENTORY BY PROD_NAME BREAK.ON PROD_NAME TOTAL QTY TOTAL PRICE TOTAL INV_COST
LIST INVENTORY BY PROD_NAME BREAK.ON PROD_NAME TOTAL QTY TOTAL PRICE TOTAL INV_COST 11:18:00 Jun 23 2005 1
Product
INVENTORY. Name...... Quantity Price..... ............
14001 Memory 6131 $49.95 306,243.45
********** -------- ---------- ------------- Memory 6131 $49.95 306,243.45
15001 Modem 7486 $119.00 890,834.00
15002 Modem 3988 $199.99 797,560.12
15003 Modem 4913 $259.99 1,277,330.87
15004 Modem 146 $219.99 32,118.54
********** -------- ---------- ------------- Modem 16533 $798.97 2,997,843.53
======== ========== =============
TOTAL 22664 $848.92 3,304,086.98
5 records listed
```

**Related Command**

TOTAL
CLEARSELECT

Syntax

CLEARSELECT [list_num]

Description

The UniQuery CLEARSELECT command clears an active select list. list_num is the number of the active select list you want to clear. If you do not indicate list_num, UniQuery clears select list 0. Use this command at the active select list prompt (>).

Examples

In the following example, CLEARSELECT clears the active select list (0) and returns to the ECL command line prompt (:):

:SELECT CLIENTS
131 records selected to list 0.

>CLEARSELECT

In the next example, a select list is selected and assigned to select list 4. Then CLEARSELECT clears select list 4.

:SELECT CLIENTS TO 4
131 records selected to list 4.

:CLEARSELECT 4

Related Commands

BSELECT, QSELECT, SELECT, SSELECT
CNV

Syntax

...attribute CNV "conversion_code"

Description

The UniQuery CNV keyword applies a conversion to an attribute or expression during the execution of the UniQuery statement only. When you use the CNV keyword, the conversion_code you specify overrides the conversion code defined in attribute 3 of the dictionary record for attribute. You must enclose conversion_code in single or double quotation marks.

If the conversion_code you specify is an empty string or invalid, UniQuery does not perform a conversion, even if one is present in the dictionary record for attribute. conversion_code can be any valid dictionary conversion code. For a list of valid conversion codes, see Using UniData.

Example

In the following example, the CNV keyword is used to display ORD_DATE from the ORDERS file in the D4- format. The ORD_DATE dictionary record has a conversion code of D2/.

```
:LIST ORDERS ORD_DATE CNV "D4-"
LIST ORDERS ORD_DATE CNV "D4-" 14:25:59 Jun 03 2005 1
ORDERS.... Order Date

  903  01-13-1996  
  965  01-15-1996  
   841  07-23-1995  
   872  01-22-1996  
   934  01-14-1996  
   810  09-01-1995  
   873  01-21-1996  
   966  01-15-1996  
```
COL.HDG

Syntax

...attribute COL.HDG "column_heading"

Description

The UniQuery COL.HDG keyword defines a column heading for attribute during the current execution of the UniQuery statement only. The column_heading you specify overrides the column heading defined in attribute 4 of the dictionary record for attribute. You must enclose column_heading in single or double quotation marks.

Example

In the following UniQuery statement, a column heading is defined for the CLIENT_NO attribute in the ORDERS file. The COL.HDG overrides the column heading defined in attribute 4 of CLIENT_NO, Client Number, with Customer Number. Customer Number appears on two lines because a right brace (}) is embedded in the column heading definition.

```
LIST ORDERS BY CLIENT_NO CLIENT_NO COL.HDG "Customer}Number"
PROD_NAME
LIST ORDERS BY CLIENT_NO CLIENT_NO COL.HDG "Customer}Number"
PROD_NAME 14:40:06 Jun 03 2005 1 Customer
ORDERS.... Number.... Product Name
862 9965 Keyboard
954 9965 Telephone
844 9966 VCR
    Video 16
    Speaker
    Cassette
    System
```
COL.HDR.SUPP

Syntax

...COL.HDR.SUPP

Synonyms

ECLTYPE U

COL-HDR-SUPP, COL.SUP, COL-SUP, COL.SUPP, COL-SUPP

ECLTYPE P

COL-HDR-SUPP

Description

The UniQuery COL.HDR.SUPP keyword suppresses the column headings defined in attribute 4 of the dictionary attributes you display in a UniQuery statement.

Example

In the following example, COL.HDR.SUPP suppresses the display of the column headings for all of the attributes in the report:

```plaintext
:LIST INVENTORY PROD_NAME COLOR QTY COL.HDR.SUPP
LIST INVENTORY PROD_NAME COLOR QTY COL.HDR.SUPP 09:38:01 Jun 04
2005 1
15001  Modem  N/A  7486
35000  Speaker Black  148
       Charcoal  125
15002  Modem  Gray  3988
54090  Disk Drive N/A  575
52070  Printer Black  4598
50050  Computer Black  15
```
COL.SPACES

Syntax

Synonyms

ECLTYPE U
COL-SPACES, COL-SPCS, COL.SPCS

ECLTYPE P
COL.SPCS

Description

The UniQuery COL.SPACES keyword defines the spacing between each column in a horizontal report when you use the LIST or SORT command.

The default number of spaces between columns is 1. If the width of the report is too wide for the screen or page, UniQuery automatically displays the report in vertical format.

COL.SPACES places $n$ number of spaces between all columns in the report.
Example

In the following example, COL.SPACES inserts 10 spaces between each column:

:LIST INVENTORY PROD_NAME INV_DATE COL.SPACES 10
LIST INVENTORY PROD_NAME INV_DATE COL.SPACES 10
15:03:44 Jun 03 2005 1

<table>
<thead>
<tr>
<th>Product</th>
<th>Inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVENTORY.</td>
<td>Name...... Date......</td>
</tr>
<tr>
<td>15001</td>
<td>Modem</td>
</tr>
<tr>
<td>35000</td>
<td>Speaker</td>
</tr>
<tr>
<td>15002</td>
<td>Modem</td>
</tr>
<tr>
<td>54090</td>
<td>Disk Drive</td>
</tr>
<tr>
<td>52070</td>
<td>Printer</td>
</tr>
<tr>
<td>50050</td>
<td>Computer</td>
</tr>
<tr>
<td>15003</td>
<td>Modem</td>
</tr>
</tbody>
</table>

.
COL.SPCS

COL.SPCS is a synonym for the COL.SPACES keyword. For further information, see COL.SPACES.

Synonyms

**ECLTYPE U**

COL.SPACES, COL-SPACES, COL-SPCS

**ECLTYPE P**

COL.SPACES
COL.SUP

COL.SUP is a synonym for the COL.HDR.SUPP keyword. For further information, see COL.HDR.SUPP.

Synonyms

**ECLTYPE U**

COL.HDR.SUPP, COL-HDR-SUPP, COL-SUP, COL.SUPP, COL-SUPP

**ECLTYPE P**

COL-HDR-SUPP
COPY.LIST

Syntax

Synonyms

COPY-LIST, COPYLIST

Description

The UniQuery COPY.LIST command copies an existing saved list in the SAVEDLISTS file to a new list within SAVEDLISTS, or to a new file within the same UniData account. If you do not specify new_list on the command line, UniQuery prompts you for the new list name. COPY.LIST can also store a list as a record in a UniData file, send the list to a printer, or display the list on the terminal screen.

Use the UniQuery SAVE.LIST command to create a saved list.

Parameters

The following table describes the parameters of the COPY.LIST command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>saved_list</td>
<td>Name of the existing list in the SAVEDLISTS file you are copying.</td>
</tr>
<tr>
<td>new_list</td>
<td>New list name to which UniData copies the contents of saved_list.</td>
</tr>
<tr>
<td>D</td>
<td>Deletes the original list after it has been copied. Cannot use with the P or T option.</td>
</tr>
<tr>
<td>N</td>
<td>Suppresses automatic paging. Valid only with the T option.</td>
</tr>
</tbody>
</table>

COPY.LIST Parameters
In the following example, the COPY.LIST command copies the existing saved list ALL.CLIENTS to a saved list called NEW.CLIENTS. COPY.LIST also deletes the ALL.CLIENTS saved list and overwrites the existing NEW.CLIENTS saved list by using the O and D options.

```
:COPY.LIST ALL.CLIENTS NEW.CLIENTS -OD
```

In the next example, COPY.LIST displays the contents of the NEW.CLIENTS saved list to the terminal screen using the T option:

```
:COPY.LIST NEW.CLIENTS -T
NEW.CLIENTS
001 9999
002 10034
003 9980
004 10015
005 10072
006 10053
007 10091
008 10092
009 10073
010 10054
011 10035
012 9981
013 10016
014 9982
015 10017
```

In the previous example, the numbers in the leftmost column represent a sequential number UniQuery assigns to each element in the saved list file.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Overwrites the contents of new_list if new_list already exists. Cannot use with the P or T option.</td>
</tr>
<tr>
<td>P</td>
<td>Sends contents of saved_list to a printer. Valid only with the T option.</td>
</tr>
<tr>
<td>T</td>
<td>Sends contents of saved_list to the terminal screen. Valid only with the N or P option.</td>
</tr>
</tbody>
</table>
Related Commands

DELETE.LIST, EDIT.LIST, FORM.LIST, GET.LIST, MERGE.LIST, SAVE.LIST, SORT.LIST
COUNT

Syntax

COUNT filename [selection_criteria]

Description

The UniQuery COUNT command counts the number of records in a file that meet the selection criteria you specify and displays the number of records counted. COUNT does not return an active select list or display attribute values.

Parameters

The following table lists the parameters of the COUNT command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Name of the file containing the records to be counted.</td>
</tr>
<tr>
<td>selection_criteria</td>
<td>Specifies conditions for retrieving or bypassing a particular record. If you do not specify any conditions, UniQuery counts the contents of the entire data file.</td>
</tr>
</tbody>
</table>

Example

In the following example, the COUNT command returns a count of the records in the CLIENTS file that contain the value “Canada” in the COUNTRY attribute:

:COUNT CLIENTS WITH COUNTRY = "Canada"

31 records counted

:
COUNT.SUP

Syntax

...COUNT.SUP

Synonyms

**ECLTYPE U**
COUNT-SUP, NI.SUP, NI-SUP, NI.SUPP, NI-SUPP

**ECLTYPE P**
COUNT-SUP, NI.SUPP, NI-SUPP

Description

The UniQuery **COUNT.SUP** keyword suppresses the number of items listed at the end of a report.
Example

In the following example, COUNT.SUP suppresses the number of items listed from the CLIENTS file:

```
:LIST CLIENTS NAME SAMPLE COUNT.SUP
LIST CLIENTS NAME SAMPLE COUNT.SUP 14:12:07 Jun 16 2005 1
CLIENTS... Name........................
9999     Paul Castiglione
10052    Paul O'Hare
10053    Gino Lee
10054    Gregory Alps
10055    Cathy Gross
10056    Samuel Morrison
10057    Subrina Iguano
10058    Antonette Larnelle
10059    Weiming Wade
.
.
.
:
```
DBL.SPC

Syntax

...DBL.SPC

Synonym

DBL-SPC

Description

The DBL.SPC keyword forces two lines between records in a report. By default, UniQuery skips a single line between records in a report.

Example

In the following example, the DBL.SPC keyword skips two lines between each record in the UniQuery report:

```plaintext
:LIST INVENTORY PROD_NAME COLOR DBL.SPC
LIST INVENTORY PROD_NAME COLOR DBL.SPC 12:25:03 Jun 04 2005 1
Product
INVENTORY. Name...... Color.....

15001      Modem      N/A
35000      Speaker    Black Charcoal
15002      Modem      Gray
54090      Disk Drive N/A
52070      Printer    Black
50050      Computer   Black
15003      Modem      Gray
..
.
```
DELETE.LIST

Syntax

DELETE.LIST list_name

Synonyms

DELETE-LIST, DELETELIST

Description

The UniQuery DELETE.LIST command deletes a saved list from the SAVEDLISTS file. If the list you specify does not exist, or you do not have permission to delete the list, UniQuery displays an error message. list_name is the name of the list you want to delete.

Warning: DELETE.LIST deletes the saved list you specify without prompting for confirmation that you want to delete the list.

Examples

In the following example, the DELETE.LIST command removes the NEWCLIENTS list from the SAVEDLISTS file:

:DELETE.LIST NEW.CLIENTS
'NEW.CLIENTS' deleted.
:

In the next example, the DELETE.LIST command returns an error message because the specified list does not exist in the SAVEDLISTS file:

:DELETE.LIST CALIFORNIA.CLIENTS
DELETE.LIST CALIFORNIA.CLIENTS failed.
:
Related Commands

COPY.LIST, EDIT.LIST, FORM.LIST, GET.LIST, MERGE.LIST, SAVE.LIST, SORT.LIST
DET.SUP

Syntax

...DET.SUP

Synonyms

_ECLTYPE U_

DET-SUP

_ECLTYPE P_

DET.SUPP, DET-SUPP

Description

The UniQuery DET.SUP keyword suppresses all detail lines in a report, displaying only breakpoint lines and total lines.

*Note:* UniQuery only supports DET.SUP and DET-SUP in **ECLTYPE U**. UniQuery only supports DET.SUPP and DET-SUPP in **ECLTYPE P**.

**UDT.OPTIONS 43** determines the information that UniQuery prints at the breakpoint when DET.SUP is part of the UniQuery statement. With this option on, UniQuery displays the breakpoint value and the detail of the last value before the breakpoint. When this option is off, UniQuery does not display the last value before the breakpoint.
Example

In the following example, UniQuery displays records from the INVENTORY file by product name, and totals the quantity and price for each product name. Only the product name and total appears because the UniQuery statement includes DET.SUP.

```
:LIST INVENTORY BY PROD_NAME BREAK.ON PROD_NAME TOTAL QTY TOTAL PRICE DET.SUP
LIST INVENTORY BY PROD_NAME BREAK.ON PROD_NAME TOTAL QTY TOTAL PRICE DET.SUP 11:35:28 Jun 08 2005 1
Product
Name...... Quantity Price.....
Disk Drive 1323 $259.90
CD Player 347  $359.71
CD System 507  $199.92
  1
CD System 399  $199.92
  10
CD System -49  $159.87
  11
CD System 397  $499.92
  12
CD System 700  $219.84
  13
.
.
.
```
DICT

Syntax

...DICT filename

Description

The UniQuery DICT keyword specifies the dictionary portion, rather than the data portion, of filename. DICT must appear immediately before filename.

Example

In the following example, UniQuery displays the dictionary of the CLIENTS file.

:LIST DICT CLIENTS
LIST DICT CLIENTS BY TYP BY @ID TYP LOC CONV NAME FORMAT SM ASSOC
11:45:57 Jun 08 2005 1
@ID............. TYP LOC........... CONV NAME........... FORMAT SM
ASSOC.....

@ID             D               0      CLIENTS         10R    S
FNAME           D               1      First Name      15T    S
CITY            D               5      City            15T    S
COMPANY         D               3      Company Name    30T    S
COUNTRY         D               8      Country         15T    S
ADDRESS         D               4      Address         25T    MV
ID              D               0      Client #        10R    S
LNAME           D               2      Last Name       15T    S
    .
    .

Note: You can also display the dictionary portion of a file by executing the LISTDICT paragraph. LISTDICT displays dictionary attributes in order of type, then attribute number. LIST DICT displays dictionary attributes in order of type, then alphabetically.
**EACH**

**Syntax**

...EACH selection_criteria

**Description**

The UniQuery EACH keyword retrieves only those records where every value exactly meets the selection_criteria. Without the EACH keyword, UniQuery returns all values from a record if one or more values meet the selection_criteria.

*Note: UniQuery only supports the EACH keyword in ECLTYPE P only. In ECLTYPE U, use the EVERY keyword.*

**Example**

The following example lists records in the INVENTORY file where every value of the COLORS attribute contains “Gray”:

```
:LIST INVENTORY WITH EACH COLOR = "Gray" PROD_NAME COLOR
LIST INVENTORY WITH EACH COLOR = "Gray" PROD_NAME COLOR 12:24:00
Jun 08 2005 1
Product
INVENTORY. Name...... Color.....

56060 Trackball   Gray
57030 Scanner     Gray
11001 Computer    Gray
12003 Printer     Gray
12006 Printer     Gray
12004 Printer     Gray
13004 Hard Drive  Gray
56040 Trackball   Gray
```
EDIT.LIST

Syntax

EDIT.LIST [savedlist_name]

Description

The UniQuery EDIT.LIST command enables you to edit a saved list that was previously saved to the SAVEDLISTS file. UniQuery opens the saved list to the system editor. If you do not specify savedlist_name, UniQuery opens the system editor anyway.

Example

In the following example, EDIT.LIST opens a saved list called PRODUCTS from the SAVEDLISTS file:

```
:EDIT.LIST PRODUCTS
15001
35000
15002
54090
52070
.
.
.
"SAVEDLISTS/PRODUCTS000" 175 lines, 1050 characters
```

Related Commands

COPY.LIST, DELETE.LIST, FORM.LIST, GET.LIST, MERGE.LIST, SAVE.LIST, SORT.LIST
EQ

EQ is a synonym for the EQUAL keyword. For further information, see EQ.

Synonyms

=, EQUAL
EQUAL

Syntax

...attribute1 EQUAL [value | attribute2]

Synonyms

=, EQ

Description

The UniQuery EQUAL keyword, also called an operator, is used in selection criteria to specify that an attribute value must be identical to a literal string or the value of another attribute. EQUAL is used in WITH and WHEN clauses.

Note: In ECLTYPE P, you must enclose value in double quotation marks.

When UDT.OPTIONS 1 is on, an empty string ("'") and zero are equal. When UDT.OPTIONS 1 is off, they are not equal.

Example

In the following example, the EQUAL keyword displays all clients that live in New York (NY):

```
:LIST CLIENTS WITH STATE EQUAL "NY" NAME STATE
LIST CLIENTS WITH STATE EQUAL "NY" NAME STATE 14:30:50 Jun 08 2005
1
CLIENTS... Name......................... State/Territory

10055      Cathy Gross                    NY
9965       Gary Phillips                  NY
2 records listed
```
ESEARCH

Syntax

ESEARCH filename [record_IDs] [selection_criteria] [(option]

Description

The UniQuery ESEARCH keyword enables you to search a file for specific values. The ESEARCH command creates an active select list of record IDs that satisfy the selection_criteria you define in a UniQuery statement, and also contain occurrences of the character strings you stipulate.

After you enter the ESEARCH statement, UniQuery prompts for a string. At the STRING prompt, enter the string you are searching for. You can specify more than one string. When you finish entering the strings, press ENTER at the STRING prompt. The total length of all the strings you specify cannot exceed 500 characters.

The active select list can be accessed by UniQuery statements or by a cataloged UniBasic program name you enter at the greater than (>) prompt. For more information about UniBasic, see Developing UniBasic Applications.

Parameters

The follow table describes the parameters of the ESEARCH command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Specifies the name of the file on which to perform operations. You can designate only one file name for each UniQuery command. The file name must follow the command name, except where the DICT keyword is used.</td>
</tr>
</tbody>
</table>

ESEARCH Parameters
The following table lists the ESEARCH options.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record_IDs</td>
<td>Specifies the record_IDs UniQuery tests against the selection criteria.</td>
</tr>
<tr>
<td>selection_criteria</td>
<td>States conditions for retrieving or bypassing a particular record.</td>
</tr>
<tr>
<td>option</td>
<td>Specifies additional functions to be performed or output conditions to be met during the primary operation. If you do not specify an option, UniQuery selects a record if it contains any of the specified string(s).</td>
</tr>
</tbody>
</table>

**ESEARCH Parameters (continued)**

The following table lists the ESEARCH options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>Selected records must contain all specified strings.</td>
</tr>
<tr>
<td>(I)</td>
<td>Display record IDs as they are selected.</td>
</tr>
<tr>
<td>(N)</td>
<td>Selected records must not contain any of the specified string(s).</td>
</tr>
<tr>
<td>(S)</td>
<td>Display only the record IDs of selected records; does not form the select list.</td>
</tr>
</tbody>
</table>

**Example**

In the following example, the ESEARCH command forms a select list of all records in the CLIENTS file that do not contain the string “Gross” in the record using the (N option. The (I option displays the record IDs as they are selected.

```plaintext
:ESEARCH CLIENTS WITH STATE = "NY" (NI
  STRING : Gross
  STRING :
  9965

  1 records selected to list 0.
>
```

*Note: You cannot use the LIKE keyword in conjunction with ESEARCH.*
EVAL

Syntax

EVAL "expression"

Description

The UniQuery EVAL keyword allows you to define a virtual attribute expression for the current execution of a UniQuery statement only. expression can be any expression valid in a virtual attribute. You must enclose expression in single or double quotation marks.

Note: UniQuery only supports the EVAL keyword in ECLTYPE U.
**Example**

In the following example, the EVAL keyword lists the city name for clients in the United States, and lists “Foreign” if the country does not equal USA.

```
:LIST CLIENTS NAME COUNTRY EVAL "IF COUNTRY = USA THEN CITY ELSE 'Foreign'"
LIST CLIENTS NAME COUNTRY EVAL "IF COUNTRY = USA THEN CITY ELSE 'Foreign'" 16:25:14 Jun 08 2005 1
COUNTRY =
USA THEN
CITY
CLIENTS... Name......................... Country....... ELSE
'Foreign'

9999      Paul Castiglione       France       Foreign
10052     Paul O'Hare           Australia     Foreign
10053     Gino Lee              Canada        Foreign
10054     Gregory Alps          France        Foreign
10055     Cathy Gross           USA           Lowell
10056     Samuel Morrison       Australia     Foreign
10057     Subrina Iguano        Canada        Foreign
10058     Antonette Larnelle    France        Foreign
10059     Weiming Wade          USA           Mount
Holly
10060     George Duncan         Australia     Foreign
```
EVERY

Syntax

...EVERY selection_criteria

Description

The UniQuery EVERY keyword retrieves only those records where every value exactly meets the selection_criteria. Without the EVERY keyword, UniQuery returns all values from a record if one or more values meet the selection_criteria.

Note: UniQuery supports the EVERY keyword in ECLTYPE U only. In ECLTYPE P, use the EACH keyword.

Example

The following example lists records in the INVENTORY file where every value of the COLORS attribute contains “Gray”:

```
:LIST INVENTORY WITH EVERY COLOR = 'Gray' PROD_NAME COLOR
LIST INVENTORY WITH EVERY COLOR = 'Gray' PROD_NAME COLOR 16:37:17
Jun 08 2005 1
        Product
        INVENTORY. Name...... Color.....
15002      Modem      Gray
15003      Modem      Gray
56010      Keyboard   Gray
55000      Cable      Gray
52080      Printer    Gray
57030      Scanner    Gray
56020      Mouse      Gray
52090      Printer    Gray
.          .
.          .
```
FMT

Syntax

...attribute FMT "format"

Description

The UniQuery FMT keyword formats attribute according to the format you specify. FMT overrides the format specified in attribute 5 of the dictionary record for attribute.

Options

The format options specify the column width and justification for text display. The following table lists the format options. \( n \) represents the column width you are assigning.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( nL )</td>
<td>Left-justified text in a column of ( n ) width. If the text exceeds the column width, text breaks after ( n ) characters.</td>
</tr>
<tr>
<td>( nR )</td>
<td>Right-justified text in a column of ( n ) width. If the text exceeds the column width, text breaks after ( n ) characters.</td>
</tr>
<tr>
<td>( nC )</td>
<td>Centered text in a column of ( n ) width. If the text exceeds the column width, text breaks after ( n ) characters.</td>
</tr>
<tr>
<td>( nT )</td>
<td>Text justification in a column of ( n ) width. If the text exceeds the column width, text breaks at a space between words.</td>
</tr>
</tbody>
</table>

format Options
Example

In the following example, the FMT keyword formats PROD_NAME in a left-justified column 20 characters in length. The dictionary record for PROD_NAME specifies a format of “10T”.

:LIST INVENTORY PROD_NAME FMT "20L"
LIST INVENTORY PROD_NAME FMT "20L" 11:31:37 Jun 09 2005 1
   Product
INVENTORY. Name.................

15001   Modem
35000   Speaker
15002   Modem
54090   Disk Drive
52070   Printer
50050   Computer
15003   Modem
53080   Photocopier Cartridge
   51060   Telephone
   .
   .
   .
FOOTING

Syntax

...FOOTING "[text] ['option...']"

Description

The UniQuery FOOTING keyword creates a footer for every page of a report when used with the LIST or SORT commands. You should always enclose the text in the footer in double quotation marks. You must enclose the options in single quotation marks. Footer text and options may be interspersed throughout the footer.

Options

The following table describes the valid FOOTING options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Takes the value of the attribute name at the breakpoint and inserts it at the designated position in the footing definition. If you specify more than one BREAK.ON &quot;B&quot; option, UniQuery only uses the first one it encounters in the statement. Each time the breakpoint value changes, UniQuery creates a new page.</td>
</tr>
<tr>
<td>C[n]</td>
<td>Centers footer text within a line. When you specify the n option, UniQuery centers the text according to this line length.</td>
</tr>
<tr>
<td>D</td>
<td>Inserts the current system date at the location of the &quot;D&quot; option in the footer.</td>
</tr>
<tr>
<td>F</td>
<td>Inserts the file name at the location of the &quot;F&quot; option in the footer.</td>
</tr>
<tr>
<td>G</td>
<td>Distributes the words in a footer evenly across the line length by inserting gaps.</td>
</tr>
<tr>
<td>L</td>
<td>Creates a line feed at the location of the &quot;L&quot; option. You can produce multiple footer lines using the &quot;L&quot; option.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>N</td>
<td>Suppresses the default “Enter new line to continue...” statement and causes the report to scroll without pausing until the end of the report.</td>
</tr>
<tr>
<td>P or ^</td>
<td>Inserts the current page number in the footer. The page number increments with each new page.</td>
</tr>
<tr>
<td>T</td>
<td>Inserts the current time and date at the location of the &quot;T&quot; option in the footer. The time and date are equal to their values at the time the UniQuery statement was executed.</td>
</tr>
</tbody>
</table>

**FOOTING Options (continued)**

*Note: UDT.OPTIONS 21 determines whether UniQuery executes a carriage return at the end of a report directed to the terminal screen. When the "L" option is present in a FOOTING statement and UDT.OPTIONS 21 is on, UniQuery pauses and waits for a carriage return before returning to the ECL command line prompt at the end of the report. If UDT.OPTIONS 21 is off, UniQuery automatically executes a carriage return and returns to the ECL command line prompt at the end of the report. If you use the "N" option with the "L" option, you automatically return to the ECL command line prompt regardless of the setting of UDT.OPTIONS 21.*

*UDT.OPTIONS 34 determines the format of the system date when using the D option in a FOOTING statement. When this option is on, UniQuery prints the system date in alphanumerics in the footer. When this option is off, UniQuery prints the system date in numeric format. If you issue the DATE FORMAT command, UniQuery prints the system date in European format either in alphanumerics format or numeric format, depending upon the setting of UDT.OPTIONS 34.*
Example

In the following example, the FOOTING keyword in conjunction with the C and D options prints centered footer text at the end of each page of the report, along with the current system date. Because of the "L" option, UniQuery breaks to a new line at its location in the footing.

```
LIST CLIENTS WITH COUNTRY = "USA" BY STATE NAME CITY BREAK.ON
STATE FOOTING "'C'Current Client Listing By State'LC'As of 'D'"
LIST CLIENTS WITH COUNTRY = "USA" BY STATE NAME CITY BREAK.ON
STATE FOOTING "'C'Current Client Listing By State'LC'As of 'D'"
14:12:12 Jun 09 2005 1
CLIENTS... Name.......................... City............
State/Territory

10094      Steve Barry                    Grand Forks     AR
**************

10000      Andre Halligan                 Phoenix         AZ
10086      Al Elliott                     St. Louis       AZ
**************

9982       Marc Willette                  Los Angeles     CA
9986       Sam Gunter                     Fresno          CA
**************

Current Client Listing by State
As of 08-09-98
```

Related Command

HEADING
FORM.LIST

Syntax

FORM.LIST filename record_ID

Synonyms

FORM-LIST, FORMLIST

Description

The UniQuery FORM.LIST command creates an active select list consisting of each attribute from a record. FORM.LIST reads the entire record of the record_ID you specify. You can only specify one record_ID with this command. FORM.LIST converts all special delimiters to attribute marks so that each value is on a separate line in the resulting select list. The specified record_ID is not saved in the list.

After you execute a statement with the FORM.LIST command, UniData displays the greater than prompt (>), indicating an active select list. You can execute the UniQuery SAVE.LIST command, a UniQuery statement, or execute a UniBasic program that contains READNEXT statements from this prompt.

Parameters

The following table describes the parameters of the FORM.LIST syntax.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Specifies the name of the file that contains the record ID FORM.LIST reads to form the select list. You can designate only one file name for each UniQuery command. The file name must follow the command name, except where the DICT keyword is used.</td>
</tr>
<tr>
<td>record_ID</td>
<td>Specifies the record_ID FORM.LIST reads to convert to an active select list.</td>
</tr>
</tbody>
</table>
Example

In the following example, FORM.LIST creates an active select list consisting of each attribute from a record in the INVENTORY file. Notice how each multivalue is saved to a separate line in the select list.

```
:FORM.LIST INVENTORY 51070
20 records formed to list 0.
>SAVE.LIST TEST
20 key(s) saved to 1 record(s).
:EDIT.LIST TEST
10209
46740
Telephone
Economy Trimline
Black
White
Green
Red
547
1000
33
289
999
999
999
999
50
50
50
50
~
~
~
"SAVEDLISTS/TEST000" 20 lines, 105 characters
```

Related Commands

COPY.LIST, DELETE.LIST, EDIT.LIST, GET.LIST, MERGE.LIST, SAVE.LIST, SORT.LIST
GE

Syntax

...attribute GE value

Synonyms

>=, =>

Description

The UniQuery GE keyword is a conditional operator that retrieves records containing values within attributes that are greater than or equal to value. value can be a numeric value, an alphanumeric string, or another attribute.

Note: When UDT.OPTIONS 1 is on, an empty string (""") and zero are equal. When UDT.OPTIONS 1 is off, they are not equal.

Example

In the following example, the GE keyword selects records from the ORDERS file where the order date is greater than or equal to January 1, 1996:

```
LIST ORDERS WITH ORD_DATE GE "01/01/96" BY ORD_DATE ORD_DATE
LIST ORDERS WITH ORD_DATE GE "01/01/96" BY ORD_DATE ORD_DATE
```

PROD_NAME

ORDERS.... Order Date Product Name

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>888</td>
<td>01/01/96</td>
<td>Photocopier Cartridge</td>
</tr>
<tr>
<td>858</td>
<td>01/02/96</td>
<td>Telephone Hard Drive Cable</td>
</tr>
<tr>
<td>859</td>
<td>01/05/96</td>
<td>Camcorder</td>
</tr>
<tr>
<td>889</td>
<td>01/05/96</td>
<td>Speaker</td>
</tr>
<tr>
<td>891</td>
<td>01/06/96</td>
<td>Telephone</td>
</tr>
<tr>
<td>860</td>
<td>01/07/96</td>
<td>Scanner</td>
</tr>
</tbody>
</table>

You must use double quotation marks around value in ECLTYPE P.
GET.LIST

Syntax

GET.LIST [account] [list_name] [TO list_num]

Synonyms

GET-LIST, GETLIST

Description

The UniQuery GET.LIST command retrieves a select list previously saved with the SAVE.LIST command. Any process that can be executed after a SELECT statement can also be executed after a GET.LIST statement.

Once you retrieve a list with the GET.LIST command, UniQuery assigns it the default list number of 0 and displays the active select list prompt (>), unless you assign the list a specific number differing from the default. Unlike UniBasic, UniQuery cannot directly process commands against numbered select lists. For information about using select list in UniBasic, see Developing UniBasic Applications.
Parameters

The following table describes the parameters of the GET.LIST syntax.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>account</td>
<td>Full path to the account where the SAVEDLISTS file resides if the list does not reside in the SAVEDLISTS file in the current account.</td>
</tr>
<tr>
<td>list_name</td>
<td>The name of the saved list to retrieve. If you do not specify list_name, UniQuery retrieves the default list 0.</td>
</tr>
<tr>
<td>TO [list_num]</td>
<td>List number (0 through 9) to which GET.LIST retrieves a saved list. If you do not specify TO list_num, UniQuery retrieves the saved list to the default list 0.</td>
</tr>
</tbody>
</table>

Example

In the following example, the GET.LIST keyword retrieves the list called TEST from the SAVEDLISTS file and stores it in the default list 0. Then the LIST command displays the names and addresses of the clients whose record IDs are contained in the list.

```
:GET.LIST TEST
4 records retrieved to list 0.
>LIST CLIENTS NAME ADDRESS
LIST CLIENTS NAME ADDRESS 13:38:15 Jun 10 2005 1
CLIENTS... Name.................. Address..................
9982       Marc Willette                  1800 Center Street
9986       Sam Gunter                     92 Barracks Road
9987       Glen Asakawa                   220 Pearl
10047      Ray Parker                     2021 Glenwood Boulevard
4 records listed
```

Related Commands

COPY.LIST, DELETE.LIST, EDIT.LIST, FORM.LIST, MERGE.LIST, SAVE.LIST, SORT.LIST
GRAND.TOTAL

Syntax

...GRAND.TOTAL "text"

Synonym

GRAND-TOTAL

Description

The UniQuery GRAND.TOTAL keyword inserts text on the grand total line in a UniQuery report. GRAND.TOTAL does not print text at breakpoint lines. text must be enclosed in double quotation marks.

Note: UniData only supports GRAND.TOTAL and GRAND-TOTAL in ECLTYPE P.
Example

In the following example, GRAND.TOTAL inserts the text “Total Modems in Stock” on the final total line:

```
LIST INVENTORY WITH PROD_NAME = "Modem" PROD_NAME TOTAL QTY
GRAND.TOTAL "Total Modems in Stock"
LIST INVENTORY WITH PROD_NAME = "Modem" PROD_NAME TOTAL QTY
GRAND.TOTAL "Total Modems in Stock" 14:12:14 Jun 10 2005 1
Product
INVENTORY. Name...... Quantity

15001      Modem          7486
15002      Modem          3988
15003      Modem          4913
15004      Modem          146
57040      Modem           293
57050      Modem           354
57060      Modem           292
57070      Modem           282
57080      Modem            55
57090      Modem           297

========
Total Modems in Stock    18106
10 records listed
```
GREATER

GREATER is a synonym for the GT keyword. For further information, see GT.

Synonyms

>, AFTER, GT
GT

Syntax

...attribute GT value

Synonyms

ECLTYPE U
> , GREATER

ECLTYPE P
> , AFTER, GREATER

Description

The UniQuery GT keyword is a conditional operator that retrieves records containing values within attribute that are greater than value. value can be a numeric value, an alphanumeric string, or another attribute.

Note: When UDT.OPTIONS 1 is on, an empty string (""") and zero are equal. When UDT.OPTIONS 1 is off, they are not equal.
Example

In the following example, the GT keyword select records from the ORDERS file where the order date is greater than January 1, 1996:

```
:LIST ORDERS WITH ORD_DATE GT "01/01/96" PROD_NAME ORD_DATE
LIST ORDERS WITH ORD_DATE GT "01/01/96" PROD_NAME ORD_DATE
14:40:08 Jun 10 2005 1
ORDERS.... Product Name Order Date

903 Cassette 01/13/96
965 Computer 01/15/96
872 TV 01/22/96
934 Adapter 01/14/96
873 Computer 01/21/96
```

Note: You must enclose value in double quotation marks in ECLTYPE P.
HDR.SUPP

Syntax

...HDR.SUPP

Synonyms

ECLTYPE U

HDR.SUP, HDR-SUP, HDR-SUPP

ECLTYPE P

HDR-SUPP, SUPP

Description

The UniQuery HDR.SUPP keyword suppresses the default heading in a UniQuery report. UniQuery continues to display the column headings.

Example

In the following example, HDR.SUPP suppresses the default heading:

```
:LIST CLIENTS NAME HDR.SUPP
CLIENTS... Name.........................

9999    Paul Castiglione
10052   Paul O'Hare
10053   Gino Lee
10054   Gregory Alps
.
.
.
```
**HEADER**

**HEADER** is a synonym for the **HEADING** keyword. For further information, see **HEADING**.

**Synonym**

**HEADING**
HEADING

Syntax

...HEADING "[text] [option...]

Synonym

HEADER

Description

The UniQuery HEADING keyword creates a header for every page of a report when used with the LIST or SORT commands. You should always enclose the text in the header should in double quotation marks. The options must be enclosed in single quotation marks. Header text and options may be interspersed throughout the header.

Options

The following table describes the valid HEADING options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Takes the value of the attribute name at the breakpoint and inserts it at the designated position in the heading definition when you specify BREAK.ON &quot;'B'&quot; in the UniQuery statement. If you specify more than one BREAK.ON &quot;'B'&quot; option, UniQuery only uses the first one it encounters in the statement. Each time the breakpoint value changes, UniQuery creates a new page.</td>
</tr>
<tr>
<td>C[n]</td>
<td>Centers header text within a line. When you specify the n option, UniQuery centers the text is according to this line length.</td>
</tr>
<tr>
<td>D</td>
<td>Inserts the current system date at the location of the &quot;'D'&quot; option in the header.</td>
</tr>
<tr>
<td>F</td>
<td>Inserts the file name at the location of the &quot;'F'&quot; option in the header.</td>
</tr>
</tbody>
</table>
Option | Description
--- | ---
G | Distributes the words in a header evenly across the line length by inserting gaps.
L | Creates a line feed at the location of the "L" option. You can produce multiple header lines using the "L" option.
N | Suppresses the default “Enter new line to continue...” statement and causes the report to scroll without pausing until the end of the report.
P or ^ | Inserts the current page number in the header. The page number increments with each new page.
T | Inserts the current time and date at the location of the "T" option in the header. The time and date are equal to their values at the time the UniQuery statement was executed.

**Note:** UDT.OPTIONS 34 determines the format of the system date when using the "D" option in a HEADING statement. When this option is on, UniQuery prints the system date in alphanumerics in the footer. When this option is off, UniQuery prints the system date in numeric format. If you issue the DATE FORMAT command, UniQuery prints the system date in European format either in alphanumerical format or numeric format, depending upon the setting of UDT.OPTIONS 34.
Example

In the following example, the HEADING keyword in conjunction with the C and D options prints centered header text at the start of each page of the report, along with the current system date. The L option breaks to a new line at its location in the heading.

```
:LIST CLIENTS WITH COUNTRY = "USA" BY STATE NAME CITY BREAK.ON STATE
HEADING "'C'Current Client Listing By State'LC'As of 'D'
Current Client Listing By State
As of 08-10-98
CLIENTS... Name......................... City......... State/Territory
10018      Mary Johnson                   Denver          CO
10063      Jan Elliott                    Youngstown      CT
10019      Sally Jackson                  Ft. Myers       FL

```

Related Command

FOOTING
**ID.ONLY**

**ID.ONLY** is a synonym for the ONLY keyword. For further information, see **ONLY**.

**Synonyms**

**ECLTYPE U**

ID-ONLY, ONLY
ID.SUPP

Syntax

...ID.SUPP

Synonyms

ECLTYPE U

ID.SUP, ID-SUP, ID-SUPP

ECLTYPE P

ID-SUPP

Description

The UniQuery ID.SUPP keyword suppresses the display of the record ID (@ID) in a UniQuery report. By default, the record IDs print in the first column of a report. If you want to display the record IDs in a column other than the first, you can use ID.SUPP, then specify @ID as a display attribute in the desired location in a UniQuery statement.
Example

In the following example, ID.SUPP suppresses the @ID in the UniQuery report:

LIST INVENTORY BY PROD_NAME PROD_NAME QTY ID.SUPP

Product Name...... Quantity

Adapter         544
Adapter         467
Adapter         104
CD Player       399

.
INTERSECT

Syntax

WITH attribute1 INTERSECT attribute2

Description

The UniQuery INTERSECT keyword selects records that have identical values in two different multivalued or multi-subvalued attributes. attribute1 and attribute2 do not have to be associated, but they must reside in the same file.

Example

In the following example, UniQuery lists records from the INVENTORY file that have the same values for quantity and reorder.

:LIST INVENTORY WITH QTY INTERSECT REORDER QTY REORDER
LIST INVENTORY WITH QTY INTERSECT REORDER QTY REORDER 11:45:54 Jun
30 2005 1
   Reorder
   INVENTORY. Quantity Point..
   57030    30    30
   51080    50    50
2 records listed
:
LE

Syntax

...attribute LE value

Synonyms

<=, =<

Description

The UniQuery LE keyword is a conditional operator that retrieves records containing values within attributes that are less than or equal to value. value can be a numeric value, an alphanumeric string, or another attribute.

Note: When UDT.OPTIONS 1 is on, an empty string (""") and zero are equal. When UDT.OPTIONS 1 is off, they are not equal.
Example

In the following example, the LE keyword selects records from the ORDERS file where the order date is less than or equal to January 1, 1996:

```
:LIST ORDERS WITH ORD_DATE LE "01/01/96" BY ORD_DATE ORD_DATE
PROD_NAME

LIST ORDERS WITH ORD_DATE LE "01/01/96" BY ORD_DATE ORD_DATE
PROD_NAME 10:59:58 Jun 10 2005 1
ORDERS.... Order Date Product Name

804 01/01/68 VCR
     Remote
     Control
     Video 12
     Video 14
     Camera

791 04/09/68 Computer

882 01/21/95 Telephone
     Remote
     Control

867 01/23/95 Memory
     Computer
     Modem

886 01/26/95 Telephone
.
.
.
```

Note: You must use double quotation marks around value in ECLTYPE P.
LESS

LESS is a synonym for the LT keyword. For further information, see LT.

Synonyms

<, BEFORE, LT
LIKE

Syntax

...attribute LIKE [string | pattern]

Synonyms

MATCH, MATCHES, MATCHING

Description

The UniQuery LIKE keyword is a conditional operator used with WITH and WHEN selection clauses to retrieve records containing a value in attribute that matches the specified string or pattern. string can be either a numeric or alphanumeric string. pattern can be any pattern described in the following table.

Ellipses (...) are used with the LIKE keyword to designate where in string to test for a match. If ellipses precede string, LIKE searches for the string at the end of the value. If ellipses follow string, LIKE searches for string at the beginning of the value. If ellipses precede and follow string, LIKE searches for string in the middle of the value.

Note: You cannot use an alternate index with a LIKE clause when the comparative attribute is numeric.

UniQuery only supports LIKE and its synonyms in ECLTYPE U.
Patterns

The following table lists the valid patterns you can use with the LIKE keyword.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0X or ...</td>
<td>Any number of characters, including no characters.</td>
</tr>
<tr>
<td>nX</td>
<td>(n) number of any character.</td>
</tr>
<tr>
<td>0A</td>
<td>Any number of alphabetic characters, including none.</td>
</tr>
<tr>
<td>nA</td>
<td>(n) number of alphabetic characters.</td>
</tr>
<tr>
<td>&quot;n-mA&quot;</td>
<td>(n) to (m) number of alphabetic characters.</td>
</tr>
<tr>
<td>&quot;n-mX&quot;</td>
<td>(n) to (m) number of any characters.</td>
</tr>
<tr>
<td>0N</td>
<td>Any number of numeric characters, including none.</td>
</tr>
<tr>
<td>nN</td>
<td>(n) number of numeric characters.</td>
</tr>
<tr>
<td>&quot;n-mN&quot;</td>
<td>(n) to (m) number of numeric characters.</td>
</tr>
<tr>
<td>text</td>
<td>Exact text. Text must be enclosed in quotation marks.</td>
</tr>
</tbody>
</table>

Supported Patterns

If you are combining pattern matching and literal strings in your LIKE statement, you should enclose the literal string in single quotations marks to avoid confusion. You should enclose the entire selection criteria in double quotation marks.
Examples

In the following example, the LIKE keyword returns records from the CLIENTS file where the name of the city begins with the letter P:

```
:LIST CLIENTS WITH CITY LIKE "P..." CITY
```

LIST CLIENTS WITH CITY LIKE "P..." CITY 09:50:11 Jun 14 2005 1
CLIENTS... City...........

9999   Paris
10052  Perth
10058  Paris
10067  Peekskill
10072  Perth
10074  Peoria
10079  Providence
10082  Pomona
10000  Phoenix
10013  Paris
9969   Philadelphia
9971   Paris
9975   Ponotoc
9978   Perth

In the next example, the LIKE keyword returns records from the CLIENTS file where the city contains 5 alphabetic characters:

```
:LIST CLIENTS WITH CITY LIKE "5A" CITY
```

LIST CLIENTS WITH CITY LIKE "5A" CITY 09:54:42 Apr 14 2005 1
CLIENTS... City...........

9999   Paris
10052  Perth
10058  Paris
10071  Flint
10072  Perth
10091  Omaha
10013  Paris
9971   Paris
9978   Perth
10032  Perth
10034  Paris
9983   Logan
10042  Paris
9991   Tulsa
9995   Paris
10050  Paris
9998   Boise
17 records listed

:
In the next example, UniQuery selects records from the CLIENTS file where the ZIP Code attribute begins with the literal string “R”, followed by any six characters. Since a literal string and pattern matching are both used, the literal string (in this case “R”) is enclosed within single quotation marks inside the entire selection criteria, which is enclosed in double quotation marks.

```plaintext
:D LIST CLIENTS WITH ZIP LIKE "'R'6X" ZIP
LIST CLIENTS WITH ZIP LIKE "'R'6X" ZIP 10:45:50 Jun 15 2005 1
CLIENTS... Postal Code

  10069   R3C 2V2
  10073   R3C 4J3
  10006   R3C 2V2

3 records listed
```
LIST

Syntax

LIST [DICT] filename [record_IDs] [selection_criteria] [sorting_criteria] [attributes | ALL] [format_options] [report_options] [TO [DELIM "char"] text_file] [TOXML [ELEMENTS] [WITHSCHEMA] [WITHDTD] [XMLMAPPING mapping_file]]

Description

The UniQuery LIST command selects, sorts, and displays data from the database to the terminal screen or a printer.

LIST always displays the record IDs of selected records in the report unless you suppress the record IDs with the ID.SUPP keyword. UniQuery lists attributes in the order in which you define them in the UniQuery statement. If you do not include any attributes in the UniQuery statement, the LIST keyword displays the attributes you defined in the @UQ record in the dictionary portion of the file, if one exists. If you include the LPTR keyword in the UniQuery statement to send the report to a printer, LIST displays the attributes you defined in the @LPTR phrase in the dictionary portion of the file, if one exists. If you do not define @UQ or @LPTR, UniQuery displays only the record_IDs. If you specify at least one attribute, UniQuery ignores @UQ and @LPTR.

LIST displays selected records horizontally unless the width of the report exceeds the width of the page, normally 80 characters. If the width is exceeded, LIST displays the records vertically.
Parameters

The following table describes the parameters of the LIST command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DICT</td>
<td>Specifies the dictionary portion, rather than the data portion, of filename. DICT must immediately precede filename.</td>
</tr>
<tr>
<td>filename</td>
<td>Specifies the name of the file on which operations are to be performed. You may specify only one file name for each UniQuery command.</td>
</tr>
<tr>
<td>record_IDs</td>
<td>Specifies record_IDs to test against the selection criteria. If you specify more than one record_ID, separate them with spaces. You should enclose record_IDs in quotation marks so they will not be interpreted as keywords or field names.</td>
</tr>
<tr>
<td>selection_criteria</td>
<td>States conditions for bypassing or retrieving records.</td>
</tr>
<tr>
<td>sorting_criteria</td>
<td>Defines the order to display the records in the report.</td>
</tr>
<tr>
<td>attributes</td>
<td>Specifies which attributes within filename to include in the report. Each attribute must be defined in the dictionary of filename. Use ALL to display all D-type attributes in a record.</td>
</tr>
<tr>
<td>format_options</td>
<td>Specifies how to format the report, including page breaks, breakpoint values, headers, and footers.</td>
</tr>
<tr>
<td>report_options</td>
<td>Includes keywords for a variety of control and formatting options, including suppressing record_IDs and sending the report to a printer.</td>
</tr>
<tr>
<td>TO [DELIM &quot;char&quot;]</td>
<td>Lists records to a UNIX text file. If you use the DELIM keyword, UniData places char between each attribute in text_file. text_file cannot be strictly numeric.</td>
</tr>
<tr>
<td>TOXML</td>
<td>Outputs LIST results in XML format.</td>
</tr>
<tr>
<td>ELEMENTS</td>
<td>Outputs XML results in element-centric mode.</td>
</tr>
</tbody>
</table>
Predefined LIST Statements

Every UniData system includes a number of predefined LIST statements. These statements are convenient for reviewing the contents of a UniData account. The following table lists these predefined statements.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WITHSCHEMA</td>
<td>Creates an XML schema filename.xsd. By default, UniData writes this file to the <em>XML</em> directory.</td>
</tr>
<tr>
<td>WITHDTD</td>
<td>Output produces a DTD corresponding to the query.</td>
</tr>
<tr>
<td>XMLMAPPING</td>
<td>Specifies a mapping file containing transformation rules for display. This file must exist in the <em>XML</em> directory.</td>
</tr>
</tbody>
</table>

**LIST Parameters (continued)**

### Predefined LIST Statements

<table>
<thead>
<tr>
<th>VOC Item Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISTD</td>
<td>Lists all directory (DIR type) file names.</td>
</tr>
<tr>
<td>LISTDICT</td>
<td>Lists the dictionary for a designated file by TYPE, LOC, and @ID.</td>
</tr>
<tr>
<td>LISTDICTL</td>
<td>Lists the dictionary for a designated file by TYPE, LOC, and @ID, and sends the report to a printer.</td>
</tr>
<tr>
<td>LISTF</td>
<td>Lists all the VOC records of type F, LF, LD, or DIR, indicating a file pointer, by type, by name.</td>
</tr>
<tr>
<td>LISTFL</td>
<td>Lists all the files residing in the local account defined in the VOC, including F-type, DIR-type, LD-type and LF-type files by type, by name.</td>
</tr>
<tr>
<td>LISTFR</td>
<td>Lists all files residing in other accounts defined in the VOC, including F-type, DIR-type, LD-type and LF-type files by type, by name.</td>
</tr>
<tr>
<td>LISTM</td>
<td>Lists all menu records defined in the VOC file.</td>
</tr>
<tr>
<td>LISTO</td>
<td>Lists records in the VOC file that do not contain V, LF, LD, M, R, PA, or S in attribute 1 (for example, all others).</td>
</tr>
<tr>
<td>LISTPA</td>
<td>Lists all paragraphs (PA type records) in the VOC file.</td>
</tr>
</tbody>
</table>
Examples

In the following example, the LIST command displays the PROD_NAME, PRICE, QTY, and REORDER attributes from the INVENTORY file:

```
:LIST INVENTORY PROD_NAME PRICE QTY REORDER
LIST INVENTORY PROD_NAME PRICE QTY REORDER 11:20:41 Jun 14 2005
Product Reorder
INVENTORY. Name...... Price..... Quantity Point.. 
53050 Photocopier $369.95 785 50
56060 Trackball $98.99 494 70
57030 Scanner $2,995.95 42 30
31000 CD System $497.96 103 70
2
10140 Camera $129.97 12000 50
11001 Computer $1,400.00 196 40
10150 Camera $48.82 800 60
$48.82 14985 60
53040 Photocopier $699.95 458 40
56070 Mouse Pad $12.99 400 30
$12.99 500 30
$14.99 394 30
$12.99 399 30
```

Predefined LIST Statements (continued)

<table>
<thead>
<tr>
<th>VOC Item Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISTR</td>
<td>Lists all remote items (R type records) in the VOC file.</td>
</tr>
<tr>
<td>LISTS</td>
<td>Lists all sentences (S type records) in the VOC file.</td>
</tr>
<tr>
<td>LISTV</td>
<td>Lists all verbs (V type records) in the VOC file.</td>
</tr>
</tbody>
</table>
In the next example, selection criteria is added to the UniQuery statement to list only those records where PROD_NAME equals Telephone or Camera:

```
LIST INVENTORY WITH PROD_NAME = "Telephone" OR WITH PROD_NAME = "Camera"
```

```
PROD_NAME PRICE QTY REORDER

10140 Camera               $129.97  12000  50
10150 Camera               $48.82   800   60
10100 Camera               $199.97  4998   40
51090 Telephone            $39.95   95   60
10070 Camera               $34.97  8934   40
51080 Telephone            $159.95  54   50
10110 Camera               $149.97  5997   30
10050 Camera               $29.97  9833   50
40003 Telephone            $49.97  132   60
40009 Telephone            $88.64  125   70
```
In the next example, sorting criteria is added to the UniQuery statement to list records with PROD_NAME equal to Telephone or Camera alphabetically by price, breaking on PROD_NAME and totalling the PRICE.

```sql
:LIST INVENTORY WITH PROD_NAME = "Telephone" OR WITH PROD_NAME = "Camera" BY PROD_NAME BY PRICE BREAK.ON PROD_NAME TOTAL PRICE QTY
```

```
LIST INVENTORY WITH PROD_NAME = "Telephone" OR WITH PROD_NAME = "Camera" BY PROD_NAME BY PRICE BREAK.ON PROD_NAME TOTAL PRICE QTY
REORDER 11:33:55 Jun 14 2005 1
Product
Reorder
INVENTORY. Name...... Price..... Quantity Point..

10050 Camera         $29.97     9833      50
$29.97     4996      50
10130 Camera         $29.97     8000      50
$29.97     9649      50
$29.97     7008      50
10070 Camera         $34.97     8934      40
$34.97     6999      40

********** ----------
Camera $2,517.04

51070 Telephone       $9.99      547      50
$9.99     1000      50
$9.99      33      50
$9.99      289      50
51060 Telephone       $19.99      4      50
$19.99     34      50
40014 Telephone       $29.66     195      50
$29.66     100      50
$29.66     100      50
40001 Telephone       $29.92     684      60
$29.92     849      60
51020 Telephone       $29.95      23      40
$29.95      45      40
$29.95       1      40
$29.95      23      40
51040 Telephone       $29.95     100      50
$34.97     100      50
$29.95      96      50
$29.95     100      50

********** ----------
Telephone $4,035.58

==========
TOTAL $6,552.62
```
Related Commands

LIST.ITEM, LIST.LABEL, SELECT, SORT
LIST.ITEM

Syntax

LIST.ITEM [DICT] filename [record_IDs] [selection_criteria] [sorting_criteria] [report_options]

Synonym

LIST-ITEM

Description

The UniQuery LIST.ITEM command selects, sorts, and displays each D-type attribute of records you specify from filename. LIST.ITEM displays each attribute on one line. Multivalues and multi-subvalues are separated by spaces.

Parameters

The following table describes the parameters of the LIST.ITEM command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DICT</td>
<td>Specifies the dictionary portion, rather than the data portion, of filename.</td>
</tr>
<tr>
<td>filename</td>
<td>Specifies the file on which the operations are to be performed. You can only specify one filename for each UniQuery command.</td>
</tr>
<tr>
<td>record_IDs</td>
<td>Specifies record_IDs to test against the selection criteria. If you specify more than one record_ID, separate them with spaces. You must enclose record_IDs in quotation marks.</td>
</tr>
</tbody>
</table>

LIST.ITEM Parameters
Example

In the following example, LIST.ITEM lists each D-type attribute in record 31000 from the INVENTORY file:

```
:LIST.ITEM INVENTORY "31000"
  31000
  001 10021
  002 75600
  003 CD System 2
  004 Dual Cassette System, Remote Control
  005 Black
  006 103
  007 49796
  008 70

  1 record(s) listed.
 :
```

Related Commands

LIST, LIST.LABEL, SORT.ITEM
LIST.LABEL

Syntax

LIST.LABEL [DICT] filename [record_IDs] [selection_criteria] [sorting_criteria] [attributes | ALL] [format_options] [report_options]

Synonym

LIST-LABEL

Description

The UniQuery LIST.LABEL command selects, sorts, and prints data from a filename you specify in label format. LIST.LABEL prints each value in a multivalued or multi-subvalued attribute on a separate line.

Parameters

The following table describes the parameters of the LIST.LABEL command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DICT</td>
<td>Specifies the dictionary portion, rather than the data portion, of filename.</td>
</tr>
<tr>
<td>filename</td>
<td>Specifies the file on which the operations are to be performed. You can only specify one filename for each UniQuery command.</td>
</tr>
<tr>
<td>record_IDs</td>
<td>Specifies record_IDs to test against the selection criteria. If you specify more than one record_ID, separate them with spaces. You should enclose record_IDs in quotation marks so they will not be interpreted as keywords or attribute names.</td>
</tr>
<tr>
<td>selection_criteria</td>
<td>States conditions for bypassing or retrieving records.</td>
</tr>
<tr>
<td>sorting_criteria</td>
<td>Defines the order to display the records in the report.</td>
</tr>
</tbody>
</table>

LIST.LABEL Parameters
After you enter the LIST.LABEL statement, UniData prompts for the following information to format the labels.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Specifies which attributes within filename to include in the report. You must define each attribute in the dictionary of filename. Use ALL to display all D-type attributes in a record.</td>
</tr>
<tr>
<td>format_options</td>
<td>Specifies how to format the report, including page breaks, breakpoint values, headers, and footers.</td>
</tr>
<tr>
<td>report_options</td>
<td>Specifies keywords that control report output, including LPTR, NO.PAGE, SAMPLE, and SAMPLED.</td>
</tr>
</tbody>
</table>

**LIST.LABEL Parameters (continued)**

After you enter the LIST.LABEL statement, UniData prompts for the following information to format the labels.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COL</td>
<td>The number of columns across the page.</td>
</tr>
<tr>
<td>ROW</td>
<td>The number of lines included on each label. ROW should be equal to the number of attributes you specify, plus one if you do not specify ID-SUPP.</td>
</tr>
<tr>
<td>SKIP</td>
<td>The number of lines between each label.</td>
</tr>
<tr>
<td>INDNT</td>
<td>The number of spaces to indent on the left side of each label. If INDNT is greater than 0, you can specify a header at the beginning of each row.</td>
</tr>
<tr>
<td>SIZ</td>
<td>The width of each line on a label. If SIZ is greater than the width of the label, the line is truncated.</td>
</tr>
<tr>
<td>SPACE</td>
<td>The number of spaces between each column of labels.</td>
</tr>
<tr>
<td>C</td>
<td>Specifies to ignore empty strings. If the UniQuery statement does not contain C, a blank line is inserted for each empty string.</td>
</tr>
</tbody>
</table>

**LIST.LABEL Options**

If INDNT is greater than 0 and you do not suppress column headings, UniData prompts for a header for each row of the label. The text you enter appears adjacent to each row in the first column of the labels. If the setting of INDNT is not wide enough to accommodate the length of the header, the header for the row may overwrite the data in the label. If you do not want to print a header for each row, press ENTER at each header prompt.
Examples

In the following example, LIST.LABEL creates labels from data in the CLIENTS file:

```
LIST.LABEL CLIENTS WITH COUNTRY = "USA" NAME ADDRESS CSZ ID.SUPP
HDR.SUPP
COL,ROW,SKIP,INDNT,SIZ,SPACE(,C) :1,8,1,0,40,1,C
Cathy Gross
963 South A Boulevard
Lowell, NY 10572

Weiming Wade
2622 G Avenue
Mount Holly, SC 25092

Jan Elliott
1641 East 12th Street
Youngstown, CT 02021
```

Related Commands

LIST, LIST.ITEM, SORT.LABEL
LIST XMLDATA

Syntax

LIST XMLDATA xml_data “extraction_file” [fields]

Description

Use LIST XMLDATA to list the contents of an XML document.

When you list an XML document, UniQuery uses the dictionary you specify in the extraction file. The fields in the dictionary record must correspond to the position of the fields in the XML extraction file.

For more information about XML, see *Using UniQuery*.

Parameters

The following table describes each parameter of the syntax.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>extraction_file</td>
<td>The full path to the location of the extraction file. You must surround the path in quotation marks.</td>
</tr>
<tr>
<td>fields</td>
<td>The fields from the dictionary you specified in the extraction file that you want to display.</td>
</tr>
</tbody>
</table>

LIST XMLDATA Parameters
LPTR

Syntax

...LPTR [unit_num]

Description

The UniQuery LPTR keyword directs output from a UniQuery statement to a printer or _HOLD_ file based on parameters defined by the SETPTR command. *unit_num* is a value from 0 through 254. 0 is the default.

For more information about the SETPTR command, see the *UniData Commands Reference*.

Example

In the following example, the LPTR keyword sends a list of clients names to the printer.

```plaintext
:LIST CLIENTS NAME SAMPLE LPTR
request id is hpzone3-102 (1 file)
:
```
LT

Syntax

...attribute LT value

Synonyms

ECLTYPE U
<, LESS

ECLTYPE P
<, BEFORE, LESS

Description

The UniQuery LT keyword is a conditional operator that retrieves records containing values within an attribute that are less than value. value can be a numeric value, an alphanumeric string, or another attribute.

Note: When UDT.OPTIONS 1 is on, an empty string (""") and zero are equal. When UDT.OPTIONS 1 is off, they are not equal.
Example

In the following example, the LT keyword selects records from the ORDERS file where the order date is less than January 1, 1996:

```plaintext
:LIST ORDERS WITH ORD_DATE LT "01/01/96" BY ORD_DATE PROD_NAME
ORD_DATE
LIST ORDERS WITH ORD_DATE LT "01/01/96" BY ORD_DATE PROD_NAME
ORD_DATE 11:09:00 Jun 11 2005 1
ORDERS.... Product Name Order Date
804 VCR 01/01/68
Remote Control
Video 12
Video 14
Camera
791 Computer 04/09/68
882 Telephone 01/21/95
Remote Control
.
.
.
```

*Note:* You must enclose value in double quotation marks in ECLTYPE P.
MARGIN

Syntax

MARGIN width

Description

The UniQuery MARGIN keyword is used with the LIST and SORT commands to establish the width of the left margin in a report. MARGIN affects only the data display and column headings; it does not affect the default page headings, defined headings, or defined footers.

Note: UniQuery only supports MARGIN in ECTYPE U.

Example

In the following example, the MARGIN keyword indents the data 10 spaces from the left, but prints the footer flush with the left margin.

```
LIST INVENTORY PROD_NAME TOTAL QTY FOOTING "Inventory Report"
MARGIN 10
LIST INVENTORY PROD_NAME TOTAL QTY FOOTING "Inventory Report"
MARGIN 10 09:07:37 Jun 15 2005 1
Product
INVENTORY. Name...... Quantity
15001     Modem      7486
35000     Speaker     148
7486
15002     Modem      3988
54090     Disk Drive  575
52070     Printer     4598
50050     Computer   4598
15003     Modem      4913
55000     Cable      5709
```

Inventory Report
**MATCH**

**MATCH** is a synonym for the LIKE keyword. For further information, see LIKE.

**Synonyms**

LIKE, MATCHES, MATCHING
MERGE.LIST

Syntax

MERGE.LIST list1 {DIFF | INTERSECTION | UNION} list2 TO list3
[COUNT.SUP]

Synonym

MERGE-LIST

Description

The UniQuery MERGE.LIST command creates a select list from the difference, intersection, or union of two numbered select lists. To merge saved lists, use the GET.LIST command to assign a list number to each saved list.

Note: UniQuery only supports MERGE.LIST is in ECLTYPE U.

Parameters

The following table describes the parameters of the MERGE.LIST command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list1...list2</td>
<td>Specifies the two active lists to be merged. Must be a numbered list (0 through 9) rather than select list names. To merge named lists, use the GET.LIST command to assign a select list number to each saved list, then execute MERGE.LIST.</td>
</tr>
<tr>
<td>DIFF</td>
<td>Retrieves items that exist in list1 but not in list2.</td>
</tr>
<tr>
<td>INTERSECTION</td>
<td>Retrieves items the exist in both list1 and list2.</td>
</tr>
</tbody>
</table>
Note: The INTERSECTION and UNION operators build a list of unique items. If an item exists in list1 and list2, it is saved only once in list3.

Example

Complete the following steps when using the MERGE.LIST command:

1. Create the SELECT List

Create and save the select lists you want to merge. If you saved the lists to a list name, or if the lists already exist in the SAVEDLISTS file, use the GET.LIST command to retrieve the lists to a numbered list. In the following example, two lists are created and saved, then retrieved to a numbered list with the GET.LIST command:

```plaintext
:SELECT INVENTORY WITH COLOR = "Blue"
8 records selected to list 0.

>SAVE.LIST BLUE
8 key(s) saved to 1 record(s).
:
:SELECT INVENTORY WITH COLOR = "Green"
9 records selected to list 0.

>SAVE.LIST GREEN
Overwriting existing saved list.
9 key(s) saved to 1 record(s).
:
:GET.LIST BLUE TO 1
8 records retrieved to list 1.
:GET.LIST GREEN TO 2
9 records retrieved to list 2.
```


2. Merge the SELECT Lists

Execute the MERGE.LIST command to merge the two active select lists. In the following example, MERGE.LIST creates an active select list of the INTERSECTION between list 1 and list 2.

```plaintext
:MERGE.LIST 1 INTERSECTION 2
3 record(s) selected.

>SAVE.LIST DIFF
```

In the previous example, no list number was specified to save the MERGE.LIST results, so MERGE.LIST stores the results to list 0. From the > prompt, you can execute the SAVE.LIST command to name an active select list to save.

The next example illustrates how to specify a select list number to save the results of the MERGE.LIST command, then save the numbered list to a named list.

```plaintext
:GET.LIST BLUE TO 1
Overwriting existing select list.
8 records retrieved to list 1.

:GET.LIST GREEN TO 2
Overwriting existing select list.
9 records retrieved to list 2.

:MERGE.LIST 1 INTERSECTION 2 TO 3
3 record(s) selected.

:SAVE.LIST DIFF FROM 3
```

3. View the Merged Lists

You can view the contents of the list MERGE.LIST creates in one of the following ways:

- Use the COPY.LIST command with the -T option.
- Use the EDIT.LIST command.
- From the operating system prompt, change to the SAVEDLISTS directory and use a system editor.

In the following example, the COPY.LIST command displays the contents of the DIFF saved list.

```plaintext
:COPY.LIST DIFF -T

DIFF
001 10030
002 51020
003 56090
```

Related Commands

COPY.LIST, DELETE.LIST, EDIT.LIST, FORM.LIST, GET.LIST, SAVE.LIST, SORT.LIST
MODIFY

Syntax

MODIFY filename attribute operator value [record_IDs] [selection_criteria]

Description

The UniQuery MODIFY command changes the values of select attributes within a file. You may use MODIFY to change data for a single record, groups of records, records meeting the selection criteria you specify, or records contained in a select list.

You cannot use MODIFY to alter a particular value in a multivalued or multi-subvalued attribute. If UniQuery selects a record containing a multivalued or multi-subvalued attribute meeting the stated selection criteria, MODIFY will replace the entire attribute with the specified value. For this reason, you should use AE, ED, or UniEntry to modify a particular value in a multivalued or multi-subvalued attribute.

Parameters

The following table describes the parameters of the MODIFY command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Name of the file on which operations are to be performed. You may specify only one file name for each UniQuery command.</td>
</tr>
<tr>
<td>attribute</td>
<td>Attribute(s) within filename whose values will be altered.</td>
</tr>
<tr>
<td>operator</td>
<td>Operator that sets the new value. Arithmetic operations work only with constants on the right side of the operation equation.</td>
</tr>
<tr>
<td>value</td>
<td>Specifies any numeric, string constant, or virtual attribute to replace attribute.</td>
</tr>
<tr>
<td>selection_criteria</td>
<td>Specifies conditions to retrieve or bypass a particular record.</td>
</tr>
<tr>
<td>record_IDs</td>
<td>Specifies record IDs to test against the selection criteria.</td>
</tr>
</tbody>
</table>

MODIFY Parameters
The *operator* parameter may be any operator listed in the following table.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Action</th>
<th>Restrictions on value.exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equals</td>
<td>Used with virtual attributes, string constants, and numeric constants.</td>
</tr>
<tr>
<td>+=</td>
<td>Add</td>
<td>Used with a numeric constant.</td>
</tr>
<tr>
<td>-=</td>
<td>Subtract</td>
<td>Used with a numeric constant.</td>
</tr>
<tr>
<td>*=</td>
<td>Multiply</td>
<td>Used with a numeric constant.</td>
</tr>
<tr>
<td>/=</td>
<td>Divide</td>
<td>Used with a numeric constant.</td>
</tr>
</tbody>
</table>

*Note:* Before applying operations to the data, MODIFY “output converts” data according to the conversion code stored in attribute three of the associated dictionary record. Check the conversion code before using arithmetic assignment operators to ensure that the modification does not create a nonnumeric string.
Examples

In the first example, the simplest form of MODIFY changes all the records in the INVENTORY file that have a product name of “Camera” to “out of stock”:

:LIST INVENTORY PROD_NAME
LIST INVENTORY PROD_NAME 14:52:17 Jun 30 2005 1
Product
INVENTORY. Name......

  53050 Photocopier
  56060 Trackball
  57030 Scanner
  31000 CD System
  2
  10140 Camera
  11001 Computer
  10150 Camera
  53040 Photocopier
  56070 Mouse Pad
  10100 Camera
  55040 Cable
  51090 Telephone
  11020 TV
  10070 Camera
.
.

:MODIFY INVENTORY PROD_NAME = "Out of Stock" WITH PROD_NAME = "Camera"
11 record(s) modified.

:LIST INVENTORY PROD_NAME
LIST INVENTORY PROD_NAME 14:54:50 Jun 30 2005 1
Product
INVENTORY. Name......

  53050 Photocopier
  56060 Trackball
  57030 Scanner
  31000 CD System
  2
  10140 Out of Stock
  11001 Computer
  10150 Out of Stock
  53040 Photocopier
  56070 Mouse Pad
  10100 Out of Stock
.
Stock
55040 Cable
51090 Telephone
11020 TV
10070 Out of
   Stock

In the next example, MODIFY adds five days to the inventory date for record 53050:

```c
:LIST INVENTORY "53050" INV_DATE
LIST INVENTORY "53050" INV_DATE 15:01:59 Jun 30 2005 1
   Inventory
INVENTORY. Date....... 

   53050 01/14/1996
1 record listed
:MODIFY INVENTORY INV_DATE += 5 53050
INVENTORY=53050
1 record(s) modified.
:
:LIST INVENTORY "53050" INV_DATE
LIST INVENTORY "53050" INV_DATE 15:11:18 Jun 30 2005 1
   Inventory
INVENTORY. Date....... 

   53050 01/19/1996
1 record listed
:
```
NE

Syntax

...attribute1 {NE [value | attribute2]}

Synonym

#

Description

The UniQuery NE keyword, also called an operator, is used in selection criteria to specify that an attribute value is not equal to a literal string or the value of another attribute. NE is used in WITH and WHEN clauses.

*Note:* In ECLTYPE P, you must enclose value in double quotation marks.

Example

In the following example, the NE keyword displays all clients that do not live in New York (NY):

```
:LIST CLIENTS WITH STATE NE "NY" NAME STATE
LIST CLIENTS WITH STATE NE "NY" NAME STATE 13:42:58 Jun 16 2005 1
CLIENTS... Name.......................... State/Territory
9999   Paul Castiglione
10052  Paul O'Hare
10053  Gino Lee             Ontario
10054  Gregory Alps
10056  Samuel Morrison      NSW
10057  Subrina Iguano       Newfoundland
10058  Antonette Larnelle
10059  Weiming Wade         SC
10060  George Duncan        NSW
10061  Kelly Donalley       Saskatchewan
```
NI.SUP

NI.SUP is a synonym for the COUNT.SUP keyword. For further information, see COUNT.SUP.

Synonyms

ECLTYPE U
COUNT.SUP, COUNT-SUP, NI-SUP, NI.SUPP, NI-SUPP

ECLTYPE P
COUNT-SUP, NI.SUPP, NI-SUPP

Note: UniQuery does not support NI.SUP and NI-SUP in ECLTYPE P.
NO

Syntax

...NO attribute

Synonym

NOT

Description

The UniQuery NO keyword returns records where attribute is an empty string.

Example

In the following example, the NO keyword lists records from the CLIENTS file where the STATE attribute is an empty string.

:[LIST CLIENTS WITH NO STATE CITY STATE
LIST CLIENTS WITH NO STATE CITY STATE 14:51:11 Jun 16 2005 1
CLIENTS... City......... State/Territory

9999 Paris
10052 Perth
10054 Orleans
10058 Paris
10062 Nice
10072 Perth
10092 Alice Spgs
10002 Bordeaux
10008 Orleans
10010 Lyon
10013 Paris
9971 Paris
9972 Lyon
10027 Lyon
.
.
.

NO.INDEX

Syntax

...NO.INDEX WITH attribute

Description

The UniQuery NO.INDEX keyword suppresses the use of an alternate key index for attribute in a UniQuery statement. You can specify NO.INDEX only once in a UniQuery statement.

Note: UniQuery only supports NO.INDEX in ECLTYPE U.

Example

In the following example, the NO.INDEX keyword specifies not to use the alternate key index for the STATE attribute:

:LIST CLIENTS NO.INDEX WITH STATE = "CA" NAME STATE
LIST CLIENTS NO.INDEX WITH STATE = "CA" NAME STATE 10:10:27 Jun 18 2005 1
CLIENTS... Name......................... State/Territory

9982       Marc Willette                  CA
9986       Sam Gunter                     CA
2 records listed

:
NO.NULLS

Syntax

...SAVING [UNIQUE] attribute [NO.NULLS]

...AVERAGE attribute [NO.NULLS]

Description

The UniQuery NO.NULLS keyword suppresses the use of empty strings when using the SAVING or AVERAGE keywords. When you specify NO.NULLS with the SAVING keyword, UniQuery does not save attributes that contain an empty string. When you specify NO.NULLS with the AVERAGE keyword, UniQuery does not include attributes containing an empty string in the average calculation.

*Note:* When you use NO.NULLS with the SAVING keyword, UniQuery includes the null value in the resulting select list. When you use NO.NULLS with the AVERAGE keyword, the null value is ignored.
Examples

In the following example, the NO.NULLS keyword ignores the attribute containing an empty string when performing an AVERAGE.

:GET.LIST AVG.SAMPLE
10 records retrieved to list 0.

>LIST INVENTORY TOTAL QTY AVERAGE QTY NO.NULLS
LIST INVENTORY TOTAL QTY AVERAGE QTY NO.NULLS
10:05:59 Jun 21 2005 1
INVENTORY. Quantity Quantity

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15001</td>
<td>7486</td>
<td>7486</td>
</tr>
<tr>
<td>35000</td>
<td>148</td>
<td>148</td>
</tr>
<tr>
<td>15002</td>
<td>3988</td>
<td>3988</td>
</tr>
<tr>
<td>54090</td>
<td>575</td>
<td>575</td>
</tr>
<tr>
<td>52070</td>
<td>4598</td>
<td>4598</td>
</tr>
<tr>
<td>50050</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>15003</td>
<td>4913</td>
<td>4913</td>
</tr>
<tr>
<td>53080</td>
<td>3965</td>
<td>3965</td>
</tr>
<tr>
<td>51060</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15004</td>
<td>146</td>
<td>146</td>
</tr>
</tbody>
</table>

======== ========
25834 2870.44

10 records listed
In the next example, the NO.NULLS keyword does not save the attribute containing an empty string when used with the SAVING keyword:

```
:GET.LIST NO.STATE
8 records retrieved to list 0.

>LIST CLIENTS STATE
LIST CLIENTS STATE 10:10:58 Jun 21 2005 1
CLIENTS... State/Territory

10055       NY
10056       NSW
10057       Newfoundland
10058
10059       SC
10060       NSW
10061       Saskatchewan
10062
8 records listed
:
:GET.LIST NO.STATE
8 records retrieved to list 0.

>SELECT CLIENTS SAVING STATE NO.NULLS
6 records selected to list 0.

>SAVE.LIST TEST
6 key(s) saved to 1 record(s).

:COPY.LIST TEST -T
TEST
001 NY
002 NSW
003 Newfoundland
004 SC
005 NSW
006 Saskatchewan
:
```
NO.PAGE

Syntax

...NO.PAGE

Synonyms

ECLTYPE U

NOPAGE, NO-PAGE

Description

The **NO.PAGE** keyword suppresses pausing for page breaks when printing a report to the terminal screen. When using the **LIST** or **SORT** command, UniQuery does not display the “Enter <New line> to continue...” prompt.
Example

In the following example, the NOPAGE keyword prints a report to the terminal screen without pausing for page breaks:

```
:LIST CLIENTS NAME NOPAGE
LIST CLIENTS NAME 10:59:44 Jun 21 2005 1
CLIENTS... Name.........................

9999       Paul Castiglione
10052      Paul O'Hare
10053      Gino Lee
10054      Gregory Alps
10055      Cathy Gross
10056      Samuel Morrison
10057      Subrina Iguano
10058      Antonette Larnelle
.
.
9993       Kathleen Donohue
10047      Ray Parker
9994      Edouard Nielsen
10048      Thomas Montero
9995      Omar Saulnier
10049      Jennifer Vaughn
9996      Wei Chin
10050      David Silvers
9997      Carol Haig
10051      Tim Knoblauch
9998      Brian Douglass
131 records listed
:
```
NO.SPLIT

Syntax

...NO.SPLIT

Synonym

ECLTYPE U

NO-SPLIT

Description

The NO.SPLIT keyword prevents records from being split across page boundaries in a UniQuery report. If an entire record does not fit on the remaining lines of a page, UniQuery prints the record on the following page.

NO.SPLIT also prevents breakpoints and totals from printing at the beginning of a page. UniQuery prints at least one associated record with the breakpoint or total on the same page.
Example

In the following example, the NO.SPLIT keyword forces UniQuery to keep all values for a record on the same page. The “Enter <New line> to continue” prompt indicates the end of a page.

```
:LIST INVENTORY PROD_NAME QTY NO.SPLIT
LIST INVENTORY PROD_NAME QTY NO.SPLIT 11:15:25 Jun 21 2005 1
  Product
INVENTORY. Name...... Quantity

15001  Modem....... 7486
15000  Speaker...... 148
15002  Modem....... 3988
54090  Disk Drive... 575
52070  Printer...... 4598
50050  Computer.... 15
15003  Modem....... 4913
53080  Photocopier.. 3965
  Cartridge
51060  Telephone... 146
15004  Modem....... 473
56010  Keyboard.... 487
58030  Monitor..... 5709
55000  Cable....... 3356
10060  Camera...... 3795
11070  TV.......... 685
11070  TV........ 559
Enter <New line> to continue...
  Product
INVENTORY. Name...... Quantity

14001  Memory...... 6131
34000  Speaker..... 197
14002  Memory...... 6415
50060  Computer... 325
  .
  .
  .
```
NOT is a synonym for the NO keyword. For further information, see NO.

Synonym

NOT
NOT.MATCH

NOT.MATCH is a synonym for the UNLIKE keyword. For further information, see UNLIKE.

Synonyms

NOT.MATCHING, UNLIKE

Note: UniQuery only supports NOT.MATCH and its synonyms in ECLTYPE U.
NSELECT

Syntax

NSELECT filename [FROM list1] [TO list2]

Description

The UniQuery NSELECT command compares the record IDs in filename to the record IDs in list1. UniQuery creates list2, containing record IDs that occur in list1 but not in filename.

Parameters

The following table describes the parameters of the NSELECT command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Name of the file containing record IDs to compare to list1. You can specify only one file name per UniQuery command.</td>
</tr>
<tr>
<td>FROM list1</td>
<td>Specifies the active numbered select list containing record IDs to compare to filename. If list1 is a named list, use the GET.LIST command to retrieve it to a numbered list.</td>
</tr>
<tr>
<td>TO list2</td>
<td>Specifies the numbered list UniQuery creates containing record IDs from list1 that do not exist in filename. If list2 already exists, UniQuery overwrites it without warning.</td>
</tr>
</tbody>
</table>

NSELECT Parameters
Example

In the following example, the GET.LIST command retrieves a previously saved list to a numbered list, then the NSELECT command compares the record IDs in the numbered list to record IDs in the ORDERS file. Three record IDs that exist in list2 do not exist in the ORDERS file.

:get.list june.orders to 2
Overwriting existing select list.
54 records retrieved to list 2.
:nselect orders from 2 to 3
3 record(s) selected.
:

Note: In ECTYPE P, you must execute the GET.LIST command in lower case when creating a numbered list.
**NULLVAL**

**Syntax**

...{WITH | WHEN} attribute {IS NOT | IS} NULLVAL

**Description**

When null value handling is on, use the UniQuery IS [NOT] NULLVAL keywords to test for the null value.

**Examples**

Before executing the next examples, we added a value to the PRICE attribute to record 50050 in the demo database INVENTORY file. The multivalued attribute now contains two values: 0 and 99999. In record 50020, we also added a value. This attribute now contains 1349999, ^129 (the ASCII character that represents the null value for the English language group), 1349999, and 1349999.

*Note: The NVLMARK parameter in the udtconfig file is set to “X”. For more information about NVLMARK, see Administering UniData.*

In the first example, UniQuery lists records containing the null value using the IS NULLVAL keywords:

```plaintext
:LIST INVENTORY WHEN PRICE IS NULLVAL PRICE
LIST INVENTORY WHEN PRICE IS NULLVAL PRICE 10:51:49 Jun 26 2005 1 INVENTORY. Price.....

50020    X
1 record listed
```
The next example lists values in two records that do not contain the null value, using the IS NOT NULLVAL keywords:

```
:LIST INVENTORY “50020” “50050” WHEN PRICE IS NOT NULLVAL PRICE
LIST INVENTORY “50020” “50050” WHEN PRICE IS NOT NULLVAL PRICE
10:58:17 Jun 26 2005 1
INVENTORY. Price.....

  50020  $13,499.99
    $13,499.99
    $13,499.99
  50050   $0.00
    $999.99
2 records listed
```

For more information about using the null value in UniQuery, see *Using UniQuery*. 
ONLY

Syntax

...ONLY

Synonyms

ECLTYPE U

ID.ONLY, ID-ONLY

Description

The UniQuery **ONLY** keyword suppresses the use of the @UQ and @LPTR phrases, if they exist.

If you do not include any attributes in the UniQuery statement, the LIST and SORT commands display the attributes you defined in the @UQ record in the dictionary portion of the file, if one exists. If you include the LPTR keyword in the UniQuery statement to send the report to a printer, LIST displays the attributes you defined in the @LPTR phrase in the dictionary portion of the file, if one exists. If you do not define @UQ or @LPTR, UniQuery displays only the record IDs. If you specify at least one display_attribute, @UQ and @LPTR are ignored.
Example

The following example displays the @UQ dictionary record in INVENTORY file. UniQuery displays the attributes defined in the @UQ dictionary record when no other attributes are defined in the UniQuery statement.

```
:AE DICT INVENTORY @UQ
Top of "@UQ" in "DICT INVENTORY", 7 lines, 55 characters.
001: PH
002: INV_DATE INV_TIME PROD_NAME FEATURES LINE_ITEMS
003:

:LIST INVENTORY
LIST INVENTORY INV_DATE INV_TIME PROD_NAME FEATURES LINE_ITEMS
16:43:03 Apr 21 1997 1
Inventory Inventory Product
INVENTORY. Date...... Time..... Name......
Features......................
15001 08/20/1995 01:00PM Modem 14.4K Internal V34
35000 07/09/1995 10:00AM Speaker 250W, Direct/reflecting
15002 08/12/1995 07:00AM Modem 14.4K External V34
54090 01/03/1995 10:00AM Disk Drive 5.25" Floppy
52070 01/23/1996 02:50PM Printer Portable Color, 3 ppm
50050 01/24/1995 09:39AM Computer 486SL133 CPU, 4MB,
250MB
15003 08/15/1995 06:00PM Modem 28.8K External V34
...
```

In the next example, the ONLY keyword suppresses the use of @UQ:

```
:LIST ONLY INVENTORY
LIST ONLY INVENTORY 16:44:43 Jun 21 2005 1
INVENTORY.
15001
35000
15002
54090
52070
50050
15003
53080
51060
15004
56010
...
```
OR

Syntax

... phrase1 OR phrase2

Description

The UniQuery OR keyword is the OR operator. OR is used in selection criteria to join selection criteria phrases and to join WITH and WHEN clauses. When you use OR, UniQuery returns records where either phrase1 or phrase2 is true.

You do not have to implicitly specify OR when a UniQuery statement contains more than one selection clause. If you specify more than one value in a conditional expression, the OR keyword is implied.

The AND and OR operators have equal precedence in a UniQuery statement if you do not specify the hierarchy using parentheses. Without parentheses, UniQuery evaluates the statement from left to right. The following UniQuery statement returns records where both phrase1 and phrase2 are valid, or phrase3 is valid:

phrase1 AND phrase2 OR phrase3

In the following example, UniQuery retrieves records where both phrase1 and phrase2 are true, or phrase3 is true:

LIST CLIENTS WITH CITY = "Flagstaff" AND STATE = 'OH' OR COUNTRY = 'Canada' CITY STATE COUNTRY
LIST CLIENTS WITH CITY = "Flagstaff" AND STATE = 'OH' OR COUNTRY = 'Canada' CITY STATE COUNTRY 09:53:57 Jun 08 2005 1
CLIENTS... City......... State/Territory Country........
10053      Fonthill   Ontario       Canada
10057      St. John's Newfoundland  Canada
10061      Regina      Saskatchewan Canada
10065      Toronto     Ontario       Canada
10069      Winnipeg    Manitoba      Canada
10073      Larksburg   Manitoba      Canada
10075      Flagstaff   OH            USA
10077      Calgary     Alberta       Canada
10081      Montreal    Quebec        Canada
.          .                    .          .
In the next example, UniQuery alters the order of processing because parentheses are placed around phrase2 and phrase3 in the OR clause. In this case, the city must equal Flagstaff, and the state must be Ohio or the country must be Canada.

```
LIST CLIENTS WITH CITY = "Flagstaff" AND (STATE = 'OH' OR COUNTRY = 'Canada') CITY STATE COUNTRY
LIST CLIENTS WITH CITY = "Flagstaff" AND (STATE = 'OH' OR COUNTRY = 'Canada') CITY STATE COUNTRY 10:02:57 Jun 08 2005 1
CLIENTS... City............ State/Territory Country........
10061 Flagstaff Saskatchewan Canada
10075 Flagstaff OH USA
2 records listed
```

Parentheses are useful to specify complex conditional criteria. There is no limit to the number of conditional expressions in a UniQuery statement, but the statement cannot exceed 9,247 characters.

**Related Command**

`AND`
PCT

PCT is a synonym for the PERCENT keyword. For further information, see PERCENT.

Synonyms

PERCENT, PERCENTAGE
PERCENT

Syntax

...PERCENT attribute

Synonyms

PERCENTAGE, PCT

Description

The UniQuery **PERCENT** keyword is used with the LIST and SORT commands to calculate and display the percentage each record holds of the total of a numeric attribute you specify. PERCENT first calculates the total of the attribute you specify, then calculates and displays each record’s percentage of the total.

PERCENT is often used in conjunction with the **TOTAL** keyword to display the total for the attribute as well as the total percent.

**Note**: UniQuery only supports **PERCENT** and its synonyms in ECLTYPE U.

**UDT.OPTIONS 47** determines how UniQuery calculates percentages at breakpoints. When **UDT.OPTIONS 47** is on, UniQuery calculates percentages before rounding detail lines when the **PERCENT** keyword is used in a UniQuery statement. When this option is off, UniQuery calculates breakpoint percentages after rounding detail lines.
Example

In the following example, the TOTAL keyword displays the total price for five sample records, and the percentage each record’s price represents of the total.

:LIST ORDERS TOTAL PRICE PERCENT PRICE SAMPLE 5
LIST ORDERS TOTAL PRICE PERCENT PRICE SAMPLE 5 10:25:25 Jun 08 2005 1
ORDERS.... Price..... Price.....

903 $228.82 9.07
965 $1,799.99 71.35
     $139.99 5.55
841 $99.97 3.96
     $29.99 1.19
872 $129.87 5.15
934 $94.00 3.73

========== =========
$2,522.63 100.00
5 records listed

:
QSELECT

Syntax

QSELECT filename [record IDs... | *] [(n]

Description

The UniQuery QSELECT command extracts data from records you specify in a file and creates an active select list.

Parameters

The following table describes the parameters of the QSELECT command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Specifies the name of the file from which UniQuery creates the select list. You can only specify one file name in a UniQuery statement.</td>
</tr>
<tr>
<td>record IDs</td>
<td>Specifies the records from which UniData extracts data.</td>
</tr>
<tr>
<td>*</td>
<td>Specifies to extract the defined attributes from all records in the file. If you do not specify any attributes, extracts all attributes from all records.</td>
</tr>
<tr>
<td>n</td>
<td>Stores the $n$th attribute in the active select list. Must be the location of the attribute in the data file, not the name of the attribute.</td>
</tr>
</tbody>
</table>

QSELECT Parameters
Example

QSELECT is useful when selecting attributes from one file that are record IDs in another file. In the following example, QSELECT retrieves the client numbers, which are located in attribute 3, from the ORDERS file, then a UniQuery statement lists the names and addresses of those clients from the CLIENTS file.

```
:QSELECT ORDERS * (3

192 records selected to list 0.

>LIST CLIENTS NAME ADDRESS
LIST CLIENTS NAME ADDRESS 08:50:00 Jun 02 2005 1
CLIENTS... Name.........................
Address.........................

9988       Dominic Warner                 7235 Laguna Blvd
           Suite 720
9987       Glen Asakawa                   220 Pearl
10002      Aude Grenelle                  Av. Bourgailh
9979       Andrea Herriot                 91, promenade Plage
9978       Mike Vidulich                  165 Market Street
.
.
```

Related Commands

BSELECT, SELECT, SSELECT
REFORMAT

Syntax

REFORMAT filename attributes [selection_criteria]

Description

The UniQuery REFORMAT command copies record attributes you specify from one data file to another data file. The destination file must already exist. REFORMAT uses the first attribute named in the UniQuery statement as the record ID in the destination file. The remaining attributes in the UniQuery statement become record attributes in the destination file. UniQuery prompts for the name of the destination file after you enter the REFORMAT command.

Parameters

The following table describes the parameters of the REFORMAT command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Name of the file from which record attributes are selected. You may only specify one file name in a UniQuery statement.</td>
</tr>
<tr>
<td>attributes</td>
<td>Specifies the record attributes to construct the new file. The first attribute you specify becomes the record ID of the new record.</td>
</tr>
<tr>
<td>selection_criteria</td>
<td>Specifies conditions for selecting or bypassing a record. UniQuery only selects records meeting the selection criteria.</td>
</tr>
</tbody>
</table>

REFORMAT Parameters
Example

In the following example, the REFORMAT command copies the name, city, state and phone number attributes from the CLIENTS file to a new file called COLORADO_CLIENTS.

:REFORMAT CLIENTS NAME CITY STATE PHONE_NUM WITH STATE = "CO"
File name: COLORADO_CLIENTS:

To list the contents of the new file, you must create dictionary records for each attribute if they do not already exist. In the following example, UniQuery lists each attribute in the new file using dictionary items previously created.

:LIST COLORADO_CLIENTS CITY STATE PHONE
LIST COLORADO_CLIENTS CITY STATE PHONE 13:52:44 Jun 20 2005 1
COLORADO_CLIENTS ................... . ..........
Ray Parker       Portland             CO  4087695340
 4087698834
Glen Asakawa     Colo Spgs.           CO  7198569584
 7195868554
2 records listed:

For information about creating dictionary records, see Using UniData.

Related Command

SREFORMAT
REQUIRE.INDEX

Syntax

...REQUIRE.INDEX WITH attribute

Synonym

ECLTYPE U

REQUIRE-INDEX

Description

The UniQuery REQUIRE.INDEX keyword forces UniData to use the alternate key index for attribute. If no alternate key index exists for attribute, UniQuery displays an error message.

Note: UniQuery only supports REQUIRE.INDEX in ECLTYPE U.

Example

In the following example, the REQUIRE.INDEX keyword forces UniQuery to use the alternate key index for the PROD_NAME attribute of the ORDERS file.

```
:LIST ORDERS BY PROD_NAME REQUIRE.INDEX WITH PROD_NAME = "Camera"
PROD_NAME
LIST ORDERS BY PROD_NAME REQUIRE.INDEX WITH PROD_NAME = "Camera"
PROD_NAME 15:09:49 Jun 08 2005 1
ORDERS.... Product Name

817        Camera
864        Camera
885        Camera
900        Camera
972        Camera
816        Camera
...
:```
REQUIRE.SELECT

Syntax

...REQUIRE.SELECT

Synonyms

ECLTYPE U

REQUIRE-SELECT, SELECT.ONLY, SELECT-ONLY

Description

The UniQuery REQUIRE.SELECT keyword specifies that a select list must be active before the query statement can be processed.

Note: UniQuery only supports REQUIRE.SELECT and its synonyms in ECLTYPE U.

Examples

In the following example, the UniQuery statement is terminated because there is not an active select list.

:LIST ORDERS COLOR PRICE REQUIRE.SELECT
LIST ORDERS COLOR PRICE REQUIRE.SELECT 16:13:30 Jun 08 2005 1
ORDERS..... Color..... Price.....

No active select list. Processing terminated.
:

1-164
In the next example, a select list is active and the UniQuery statement is processed normally.

```
:GET.LIST ORDERS
5 records retrieved to list 0.
>LIST ORDERS COLOR PRICE REQUIRE.SELECT
LIST ORDERS COLOR PRICE REQUIRE.SELECT 16:16:02 Jun 08 2005 1
ORDERS.... Color..... Price.....

<table>
<thead>
<tr>
<th></th>
<th>Color</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>903</td>
<td>Black</td>
<td>$228.82</td>
</tr>
<tr>
<td>965</td>
<td>Black</td>
<td>$1,799.99</td>
</tr>
<tr>
<td>841</td>
<td>Black</td>
<td>$139.99</td>
</tr>
<tr>
<td>872</td>
<td>Black</td>
<td>$99.97</td>
</tr>
<tr>
<td>934</td>
<td>N/A</td>
<td>$29.99</td>
</tr>
</tbody>
</table>

5 records listed
```

**Tip**: REQUIRE.SELECT is useful in a paragraph when a LIST or SORT statement is preceded by a SELECT statement. If a SELECT statement does not return any records, the LIST or SORT statement will terminate. If REQUIRE.SELECT is not present in the LIST or SORT statement, UniQuery processes the query against the entire data file.

**Related Commands**

GET.LIST, SELECT
SAMPLE

Syntax

...SAMPLE n

Description

The UniQuery SAMPLE keyword returns n records from a file name you specify. If you do not specify n, UniQuery defaults to 10 records. SAMPLE has a limit of 1 GB.

Example

In the following example, UniQuery lists the name for five records from the CLIENTS file.

```plaintext
:LIST CLIENTS NAME SAMPLE 5
LIST CLIENTS NAME SAMPLE 5 10:32:36 Jun 09 2005 1
CLIENTS... Name..........................
  9999       Paul Castiglione
  10052      Paul O'Hare
  10055      Cathy Gross
  10053      Gino Lee
  10054      Gregory Alps
5 records listed
:
```

Related Command

SAMPLED
SAMPLED

Syntax

...SAMPLED n

Description

The UniQuery SAMPLED keyword returns every nth record of the file name you specify. If you do not specify n, UniQuery returns every tenth record.

Example

In the following example, the SAMPLED keyword lists the name for every fifth record from the CLIENTS file.

```plaintext
:LIST CLIENTS NAME SAMPLED 5
LIST CLIENTS NAME SAMPLED 5 10:47:10 Jun 09 2005 1
CLIENTS... Name..........................
10054      Gregory Alps
10059      Weiming Wade
10064      Carlos Martinez
10069      Dean Bronson
10075      Larry Harrell
10080      Sie Ming
10085      Odin Gregors
10090      Cyndi Ryan
10094      Steve Barry
10093      Adam Monterey
10088      Sophia Anitpoli
...
...
```

Related Commands

SAMPLE
SAVE.LIST

Syntax

SAVE.LIST [list.name] [FROM list.num]

Synonyms

SAVE-LIST, SAVELIST

Description

The UniQuery SAVE.LIST command saves an active select list created by a UniQuery SELECT, SSELECT, BSELECT, QSELECT, ESEARCH, FORM.LIST, or GET.LIST command to a default list or a list name you specify. If you do not specify a list name, SAVE.LIST names the list by using the process ID or the name set in the UDT_SAVELIST environment variable. For information on setting environment variables, see Administering UniData.

If you create a list using one of the above commands and want to save the list for later processing, or if you want to use the selected records in multiple processes, save the list with the SAVE.LIST command. You can retrieve the list repeatedly using the GET.LIST command. The list remains available for use until you delete it with the DELETE.LIST command, or remove it with the appropriate system-level command from the SAVEDLISTS directory.

Saved lists are stored in the SAVEDLISTS directory. A saved list that exceeds approximately 34,810 characters on UniData for UNIX or 29,408 on UniData for Windows NT is saved in multiple parts. Each part has an extension to the specified saved list name, beginning at 000 and incrementing sequentially (001, 002, and so forth).
The following example illustrates selecting a large number of records, saving those records to a list, and the resulting records in the SAVEDLIST file.

```
:SELECT FAMILY_FILE1

40000 records selected to list 0.

>SAVE.LIST FAMILY.RECORDS
40000 key(s) saved to 7 record(s).

:LIST SAVEDLISTS WITH @ID LIKE "...FAMILY..."
LIST SAVEDLISTS WITH @ID LIKE "...FAMILY..." 16:05:49 Jun 09 2005
1
SAVEDLISTS

FAMILY.RECORDS000
FAMILY.RECORDS001
FAMILY.RECORDS002
FAMILY.RECORDS003
FAMILY.RECORDS004
FAMILY.RECORDS005
FAMILY.RECORDS006
7 records listed
```
Although you do not have to specify each part of the SAVEDLIST record ID when you execute the GET.LIST command, you must specify each part if you use the COPY command or the AE command. The EDIT.LIST command will display the first part of the SAVEDLIST record, then prompt you to display the subsequent parts, as shown in the following example:

```
:EDIT.LIST FAMILY.RECORDS
SAVEDLISTS

FAMILY.REC
ORDS000
FAMILY.REC
9230
1158
10523
11532
12541
2167
.
.
.
q
Do you want to continue with FAMILY.RECORDS001 ? (y/n/q) ?n
Do you want to continue with FAMILY.RECORDS002 ? (y/n/q) ?n
Do you want to continue with FAMILY.RECORDS002 ? (y/n/q) ?q
```

## Parameters

The following table describes the parameters of the **SAVE.LIST** command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list.name</td>
<td>Specifies name of record to which the active select list is be saved. UniQuery overwrites an existing saved list of the same name without warning.</td>
</tr>
<tr>
<td>FROM list.num</td>
<td>If more than one saved list is active, specifies which list to save. If you do not specify list.num, UniQuery saves the default list 0.</td>
</tr>
</tbody>
</table>

**SAVE.LIST Parameters**
Examples

In the following example, the SELECT command retrieves records from the ORDERS file. The SAVE.LIST command then saves the records to a list named ALL.ORDERS.

```
:SELECT ORDERS
192 records selected to list 0.

>SAVE.LIST ALL.ORDERS
192 key(s) saved to 1 record(s).
```

In the next example, the SELECT command retrieves records from the ORDERS file to list 1. The second select list then creates another active list to the default active select list 0. Then, even though the default select list 0 is active, the SAVE.LIST command with the FROM option saves list 1 to a saved list called ORD.LIST.

```
:SELECT ORDERS TO 1
192 records selected to list 1.

:SELECT ORDERS WITH ORD_DATE GT "01/01/96"
135 records selected to list 0.

>SAVE.LIST ORD.LIST FROM 1
192 key(s) saved to 1 record(s).
```

Related Commands

COPY.LIST, DELETE.LIST, EDIT.LIST, FORM.LIST, GET.LIST, MERGE.LIST, SORT.LIST
SAVING

Syntax

...SAVING [UNIQUE] attribute [NO.NULLS]

Description

The UniQuery SAVING keyword used with the SELECT or SSELECT commands saves attribute to an active select list. The UNIQUE option eliminates duplicate values in the resulting select list. The NO.NULLS option prevents empty strings from being saved in the resulting select list.

Note: The NO.NULLS option does not prevent the null value from being saved to the select list.
Example

The SAVING keyword is useful if you want to save a value in one file that is the record ID of another file. In the following example, the SAVING keyword with the UNIQUE option saves the unique client numbers from the ORDERS file. From the resulting select list, information is then listed from the CLIENTS file.

:SELECT ORDERS SAVING UNIQUE CLIENT_NO

68 records selected to list 0.

>LIST CLIENTS NAME ADDRESS CITY STATE
LIST CLIENTS NAME ADDRESS CITY STATE 09:25:08 Jun 10 2005 1
CLIENTS 9965
Name Gary Phillips
Address 8899 S. Taylor St.
City New York
State/Territory NY

CLIENTS 9966
Name Phil Becker
Address P.O. Box 212
City Hawthorn
State/Territory Vict.

CLIENTS 9967
Name Tamara Vincent
Address 6825 N. Filmore Blvd.
City Charlotte Town
State/Territory PEI

.
SELECT

Syntax

SELECT filename [record_IDs] [selection_criteria] [sorting_criteria] [SAVING [UNIQUE] attribute [NO.NULLS]] [TO list_num]

Description

When you issue a UniQuery statement without defining specific record IDs or selection criteria, UniQuery executes the statements against the entire data file. Normally, you probably do not want to view all the records in a file, but rather a subset of data. For instance, you might want to create a report for clients that live in certain states, for inventory items of a certain color, for orders that were placed on a certain date, etc. You can create reports on subsets of data by using selection criteria.

The UniQuery SELECT command creates up to 10 active select lists (0-9) or record IDs (or keys) that may be used by other UniQuery commands or other UniData processes.

If you do not specify the TO option with list_num, UniQuery saves the select list to the default list number of 0. The greater than (>) prompt indicates that a select list is active.

UniQuery accepts attribute names in selection and sorting criteria, but you cannot specify attributes for display purposes in a SELECT statement. SELECT statements do not produce output, and therefore do not accept any format specifications.
Parameters

The following table describes the parameters of the SELECT syntax.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Name of the file from which to retrieve data. You may specify only one file name for each UniQuery statement.</td>
</tr>
<tr>
<td>record_IDs</td>
<td>Specifies record IDs to test against the selection criteria. You can specify up to 128 record IDs in a SELECT statement.</td>
</tr>
<tr>
<td>selection_criteria</td>
<td>Specifies conditions for bypassing or retrieving a particular record.</td>
</tr>
<tr>
<td>sorting_criteria</td>
<td>Specifies how to sort the retrieved attributes.</td>
</tr>
<tr>
<td>SAVING [UNIQUE] attribute [NO.NULLS]</td>
<td>Saves specified attribute to an active select list, rather than the primary key. The UNIQUE option prevents UniQuery from saving duplicate values. The NO.NULLS option prevents UniQuery from saving empty strings, but does not prevent UniQuery from saving the null value.</td>
</tr>
<tr>
<td>TO list_num</td>
<td>Specifies the active select list number to which records are to be saved. The list can be numbered from 0 through 9.</td>
</tr>
</tbody>
</table>

**SELECT Parameters**

**Using Select Lists**

When you execute a UniQuery statement, UniQuery evaluates each record in the file to see if it meets the defined selection criteria. If you are working with a large file, this searching can degrade performance. By using the SELECT and SAVE.LIST commands, you can retrieve and store a list of record IDs for use in multiple processes.

SELECT creates a list of records IDs called an active select list. Any UniQuery command that uses an active list can then be executed subsequent to the creation of an active list, and use that list for processing. Programs written in UniBasic can use the READNEXT function against an active select list to read successive record IDs.

You can store the lists you create with the SELECT command in the SAVEDLISTS file using the SAVE.LIST command. To activate the list, use the GET.LIST command. UniQuery displays the greater than prompt (>) when a select list is active. To deactivate the list, use the CLEARSELECT command.
Examples

In the following example, the SELECT command retrieves records from the ORDERS file with an order date greater than 01/01/96. The greater than (>) indicates that the default select list of 0 is active. A LIST statement is then executed against the active select list:

```
:SELECT ORDERS WITH ORD_DATE GT "01/01/96"
```

135 records selected to list 0.

```
>LIST ORDERS COLOR QTY
LIST ORDERS COLOR QTY 16:40:24 Jun 10 2005 1
ORDERS.... Color..... Qty...
903    Black    3
965    Black    1
872    Black    2
934    N/A    25
```

In the next example, UniQuery executes the same SELECT statement, but the active select list is saved to a record called ORDERS.1996 in the SAVEDLISTS file:

```
:SELECT ORDERS WITH ORD_DATE GT "01/01/96"
```

135 records selected to list 0.

```
>SAVE.LIST ORDERS.1996
135 key(s) saved to 1 record(s).
```

In the next example, the GET.LIST keyword retrieves the saved list and processes a report:

```
:GET.LIST ORDERS.1996
135 records retrieved to list 0.
```

```
>LIST ORDERS ORD_DATE QTY PRICE
LIST ORDERS ORD_DATE QTY PRICE 09:20:48 Jun 11 2005 1
ORDERS.... Order Date Qty... Price.....
903    01/13/96    3    $228.82
965    01/15/96    1    $1,799.99
872    01/22/96    45    $129.87
934    01/14/96    25    $94.00
```
In the next example, UniQuery clears the active select list:

```
SELECT ORDERS WITH ORD_DATE GT "01/01/96"
```

135 records selected to list 0.

```
CLEARSELECT
```

**Related Commands**

BSELECT, CLEARSELECT, LIST, NSELECT, QSELECT, SSELECT
SELECT.ONLY

SELECT.ONLY is a synonym for the REQUIRE.SELECT keyword. For further information, see REQUIRE.SELECT.

Synonyms

REQUIRE.SELECT, SELECT-ONLY
SORT

Syntax

SORT [DICT] filename [record_IDs] [selection_criteria] [sorting_criteria] [attributes | ALL] [format_options] [report_options] [TO text_file | list_num]

Description

The UniQuery SORT command selects, sorts, and displays data from the database by record ID, if you do not specify any other sorting criteria. If you define sorting criteria, UniQuery sorts the records accordingly, with the final sort level the record ID.

SORT always displays the record IDs of selected records in the report unless you suppress the record IDs with the ID.SUPP keyword. UniQuery lists attributes in the order you define them in the UniQuery statement. If you do not include any attributes in the UniQuery statement, the SORT keyword displays the attributes you defined in the @UQ record in the dictionary portion of the file, if one exists. If you include the LPTR keyword in the UniQuery statement to send the report to a printer, SORT displays the attributes you defined in the @LPTR phrase in the dictionary portion of the file, if one exists. If you do not define @UQ or @LPTR, UniQuery displays only the record IDs. If you specify at least one attribute, @UQ and @LPTR are ignored.

SORT displays selected records horizontally unless the width of the report exceeds the width of the page, which is normally 80 characters. If the width is exceeded, SORT displays the records vertically.
## Parameters

The following table describes the parameters of the SORT command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DICT</td>
<td>Specifies the dictionary portion, rather than the data portion, of <code>filename</code>. DICT must immediately precede <code>filename</code>.</td>
</tr>
<tr>
<td><code>filename</code></td>
<td>Specifies the name of the file on which operations are to be performed. You may specify only <code>filename</code> for each UniQuery command.</td>
</tr>
<tr>
<td><code>record_IDs</code></td>
<td>Specifies <code>record_IDs</code> to test against the selection criteria. If you specify more than one <code>record_ID</code>, separate them with spaces. You should enclose <code>record_IDs</code> in quotation marks so UniQuery will not be interpret them as keywords or attribute names.</td>
</tr>
<tr>
<td><code>selection_criteria</code></td>
<td>States conditions for bypassing or retrieving records.</td>
</tr>
<tr>
<td><code>sorting_criteria</code></td>
<td>Defines the order to display the records in the report.</td>
</tr>
<tr>
<td>`attributes</td>
<td>ALL`</td>
</tr>
<tr>
<td><code>format_options</code></td>
<td>Specifies how to format the report, including page breaks, break-point values, headers, and footers.</td>
</tr>
<tr>
<td><code>report_options</code></td>
<td>Includes keywords for a variety of control and formatting options, including suppressing <code>record_IDs</code> and sending the report to a printer.</td>
</tr>
<tr>
<td>TO `text_file</td>
<td>list_num`</td>
</tr>
</tbody>
</table>
Examples

In the following example, the SORT command lists records from the INVENTORY file. Since no sorting criteria is specified, UniQuery displays the records in record ID order.

```
:SORT INVENTORY PROD_NAME PRICE
SORT INVENTORY PROD_NAME PRICE 11:47:52 Jun 11 2005 1
Product
INVENTORY. Name...... Price.....

10001 Computer    $1,995.00
10002 Computer    $1,200.00
10003 Computer    $3,200.00
10004 Computer    $2,500.00
```

In the next example, the SORT command lists the records in product name order because sorting criteria is specified. Within product name, the records are sorted by record ID:

```
:SORT INVENTORY BY PROD_NAME PROD_NAME PRICE
SORT INVENTORY BY PROD_NAME PROD_NAME PRICE 11:50:40 Jun 11 2005 1
Product
INVENTORY. Name...... Price.....

10007 Adapter       $129.95
13001 Adapter        $94.00
13002 Adapter       $150.00
39400 CD Player      $89.87
39401 CD Player      $89.87
39500 CD Player     $179.97
```

Related Commands

LIST, SORT.ITEM, SORT.LABEL, SSELECT
SORT.ITEM

Syntax

SORT.ITEM [DICT] filename [record_IDs] [selection_criteria] [sorting_criteria] [format_options] [control_options]

Synonym

SORT-ITEM

Description

The UniQuery SORT.ITEM command sorts records within a file by record ID, and displays each D-type attribute on a line-by-line basis. If an attribute is multivalued or multi-subvalued, UniQuery displays all values on one line.

SORT.ITEM outputs attributes without any conversion or formatting.

Parameters

The following table lists the parameters of the SORT.ITEM command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DICT</td>
<td>Specifies the dictionary portion, rather than the data portion, of filename. DICT must immediately precede filename.</td>
</tr>
<tr>
<td>filename</td>
<td>Specifies the name of the file on which operations are to be performed. You may specify only filename for each UniQuery command.</td>
</tr>
<tr>
<td>record_IDs</td>
<td>Specifies record_IDs to test against the selection criteria. If you specify more than one record_ID, separate them with spaces. You should enclose record_IDs in quotation marks so UniQuery will not be interpret them as keywords or field names.</td>
</tr>
</tbody>
</table>

SORT.ITEM Parameters
Example

In the following example, the SORT.ITEM command lists two records from the CLIENTS file:

```
:SORT.ITEM CLIENTS "10068" "10055"
  10055
  001 Cathy
  002 Gross
  003 Goddard Royale
  004 963 South A Boulevard
  005 Lowell
  006 NY
  007 10572
  008 USA
  009 6096460030 6096469570
  010 Work Fax

  10068
  001 Mitchell
  002 Benson
  003 Garrison Auto Co.
  004 13 S. Bolton St.
  005 Sydney
  006 NSW
  007 0211
  008 Australia
  009 8658183832 8658183511
  010 Work Fax
  2 record(s) listed.
:  
```
Related Commands

LIST.ITEM, SORT, SORT.LABEL
SORT.LABEL

Syntax

SORT.LABEL [DICT] filename [record_IDs] [selection_criteria] [sorting_criteria] [attributes | ALL] [format_options] [report_options]

Synonym

SORT-LABEL

Description

The UniQuery SORT.LABEL command selects, sorts, and prints data from a file name you specified \textit{filename} in label format by record ID, if no other sorting criteria is defined. If you define sorting criteria, UniQuery sorts the records accordingly, so the record ID is the final sort level. SORT.LABEL prints each value in a multivalued or multi-subvalued attribute on a separate line.

Parameters

The following table describes the parameters of the SORT.LABEL command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DICT</td>
<td>Specifies the dictionary portion, rather than the data portion, of \textit{filename}.</td>
</tr>
<tr>
<td>\textit{filename}</td>
<td>Specifies the file on which the operations are to be performed. You can only specify one \textit{filename} for each UniQuery command.</td>
</tr>
<tr>
<td>\textit{record_IDs}</td>
<td>Specifies record IDs to test against the selection criteria. If you specify more than one \textit{record_ID}, separate them with spaces. You should enclose \textit{record_IDs} in quotation marks so they will not be interpreted as keywords or field names.</td>
</tr>
<tr>
<td>\textit{selection_criteria}</td>
<td>States conditions for bypassing or retrieving records.</td>
</tr>
</tbody>
</table>

SORT.LABEL Parameters
After you enter the SORT.LABEL statement, UniData prompts for the following information to format the labels.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sorting_criteria</td>
<td>Defines the order to display the records in the report.</td>
</tr>
<tr>
<td>attributes</td>
<td>ALL</td>
</tr>
<tr>
<td>format_options</td>
<td>Specifies how to format the report, including page breaks, breakpoint values, headers, and footers.</td>
</tr>
<tr>
<td>report_options</td>
<td>Specifies keywords that control report output, including LPTR, NO.PAGE, SAMPLE, and SAMPLED.</td>
</tr>
</tbody>
</table>

**SORT.LABEL Parameters (continued)**

After you enter the SORT.LABEL statement, UniData prompts for the following information to format the labels.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COL</td>
<td>The number of columns across the page.</td>
</tr>
<tr>
<td>ROW</td>
<td>The number of lines included on each label. ROW should be equal to the number of attributes specified, plus one if ID_SUPP is not specified.</td>
</tr>
<tr>
<td>SKIP</td>
<td>The number of lines between each label.</td>
</tr>
<tr>
<td>INDNT</td>
<td>The number of spaces to indent on the left side of each label. If INDNT is greater than 0, you can specify a header at the beginning of each row.</td>
</tr>
<tr>
<td>SIZ</td>
<td>The width of each line on a label. If SIZ is greater than the width of the label, the line is truncated.</td>
</tr>
<tr>
<td>SPACE</td>
<td>The number of spaces between each column of labels.</td>
</tr>
<tr>
<td>C</td>
<td>Specifies to ignore empty strings. If the UniQuery statement does not contain C, a blank line is inserted for each empty string.</td>
</tr>
</tbody>
</table>

**SORT.LABEL Options**

If INDNT is greater than 0 and you do not suppress column headings, UniData prompts for a header for each row of the label. The text you enter appears adjacent to each row in the first column of the labels. If the setting of INDNT is not wide enough to accommodate the length of the header, the header for the row may overwrite the data in the label. If you do not want to print a header for each row, press RETURN at each header prompt.
Examples

In the following example, SORT.LABEL creates labels from data in the CLIENTS file.

```
:SORT.LABEL CLIENTS NAME ADDRESS CSZ ID.SUPP
COL,ROW,SKIP,INDNT,SIZ,SPACE(,C) :1,8,1,0,40,1,C
Andre Halligan
854 Ivy St.
Suite 4200
Phoenix, AZ  80598

Patricia Halberg
62 LaSalle Ave.
South Bend, IN  20687

Aude Grenelle
Av. Bourgailh
Bordeaux,   75001
```

Related Commands

LIST.LABEL, SORT, SORT.ITEM
SORT.LIST

Syntax

SORT.LIST list_name [-R | -N]

Synonyms

SORTLIST, SORT-LIST

Description

The UniQuery SORT.LIST keyword sorts an existing list saved in the SAVEDLISTS file by record ID.

Parameters

The following table describes the parameters of the SORT.LIST command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list_name</td>
<td>Specifies the existing list to be sorted.</td>
</tr>
<tr>
<td>-R</td>
<td>Sorts each attribute in the saved list in descending order. If you do not specify the R option, UniQuery sorts the list in ascending order.</td>
</tr>
<tr>
<td>-N</td>
<td>Sorts each attribute in the saved list according to their numeric values. If you use the N option with the NULL_FLAG off, UniQuery treats empty strings and the null value as zero. If the NULL_FLAG is on, the null value is treated as the smallest value in the list, followed by the empty string.</td>
</tr>
</tbody>
</table>
Examples

In the following example, the EDIT.LIST command display a select list named TEST.SORT. Notice that the record IDs are in random order.

```
:EDIT.LIST TEST.SORT
Photocopier
Trackball
Scanner
CD System 2
Camera
Computer
Camera
Photocopier
Mouse Pad
Adapter
```

In the next example, the SORT.LIST command sorts the saved list in ascending order.

```
:SORT.LIST TEST.SORT
:EDIT.LIST TEST.SORT

Adapter
CD System 2
Camera
Camera
Computer
Mouse Pad
Photocopier
Photocopier
Scanner
Trackball
```

Related Commands

COPY.LIST, DELETE.LIST, EDIT.LIST, FORM.LIST, GET.LIST, MERGE.LIST, SAVE.LIST
SREFORMAT

Syntax

SREFORMAT filename attributes [selection_criteria]

Description

The UniQuery SREFORMAT command sorts specified records by their record IDs and copies the record attributes from one file to another file. SREFORMAT uses the first attribute named in the UniQuery statement as the record ID in the destination file. The remaining attributes in the UniQuery statement become record attributes in the destination file. UniQuery prompts for the name of the destination file after you enter the SREFORMAT command.

Parameters

The following table describes the parameters of the SREFORMAT command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Name of the file from which record attributes are selected. You may only specify one file name in a UniQuery statement.</td>
</tr>
<tr>
<td>attributes</td>
<td>Specifies the record attributes to construct the new file. The first attribute you specify becomes the record ID of the new record.</td>
</tr>
<tr>
<td>selection_criteria</td>
<td>Specifies conditions for selecting or bypassing a record. UniQuery only selects records meeting the selection criteria.</td>
</tr>
</tbody>
</table>

SREFORMAT Parameters
Example

In the following example, the SREFORMAT command copies the name, city, state and phone number attributes from the CLIENTS file to a new file called COLORADO_CLIENTS.

```
:SREFORMAT CLIENTS NAME CITY STATE PHONE_NUM WITH STATE = "CO"
File name : COLORADO_CLIENTS
```

Related Command

REFORMAT
SSELECT

Syntax

SSELECT filename [record_IDs] [selection_criteria] [sorting_criteria] [SAVING [UNIQUE] attribute [NO.NULLS]] [TO list_num]

Description

When you issue a UniQuery statement without defining specific record IDs or selection criteria, UniQuery executes the statements against the entire data file. Normally, you probably do not want to view all the records in a file, but rather a subset of data. For instance, you might want to create a report for clients that live in certain states, for inventory items of a certain color, for orders that were placed on a certain date, etc. You can create reports on subsets of data by using selection criteria.

The UniQuery SSELECT command creates up to 10 active select lists (0-9) or record IDs (or keys) in record ID order that may be used by other UniQuery commands or other UniData processes.

If you do not specify the TO option with list_num, the select list is saved to the default list number of 0. The greater than (> ) prompt indicates that a select list is active.

Attribute names are accepted in selection and sorting criteria, but you cannot specify attributes for display purposes in a SSELECT statement. SSELECT statements do not produce output, and therefore do not accept any format specifications.
Parameters

The following table describes each parameter of the syntax.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Name of the file from which to retrieve data. You can specify only one file name for each UniQuery statement.</td>
</tr>
<tr>
<td>record_IDs</td>
<td>Specifies record IDs to test against the selection criteria. You can specify up to 128 record IDs in a SELECT statement.</td>
</tr>
<tr>
<td>selection_criteria</td>
<td>Specifies conditions for bypassing or retrieving a particular record.</td>
</tr>
<tr>
<td>sorting_criteria</td>
<td>Specifies how to sort the retrieved records. If you specify sorting criteria, the final sort is by record ID when you use the SSELECT command.</td>
</tr>
<tr>
<td>SAVING [UNIQUE]</td>
<td>Saves specified attribute to an active select list, rather than the primary key. The UNIQUE option prevents duplicate values from being saved. The NO.NULLS option prevents empty strings from being saved, but does not prevent the null value from being saved.</td>
</tr>
<tr>
<td>attribute [NO.NULLS]</td>
<td></td>
</tr>
<tr>
<td>TO list_num</td>
<td>Specifies the active select list number to which records are to be saved. The list can be numbered from 0 through 9.</td>
</tr>
</tbody>
</table>

SSELECT Parameters

Using Select Lists

When you execute a UniQuery statement, UniQuery evaluates each record in the file to see if it meets the defined selection criteria. If you are working with a large file, this searching can degrade performance. By using the SSELECT and SAVE.LIST commands, you can retrieve and store a list of record IDs for use in multiple processes.

SSELECT creates a list of records IDs in ascending order, called an active select list. Any UniQuery command that uses an active list can then be executed subsequent to the creation of an active list, and use that list for processing. Programs written in UniBasic can use the READNEXT function against an active select list to read successive record IDs.
The lists you create with the SSELECT command can be stored in a the SAVEDLISTS file using the SAVE.LIST command. To activate the list, use the GET.LIST command. UniQuery displays the greater than (>) prompt when a select list is active. To deactivate the list, use the CLEARSELECT command.

Examples

In the following example, the SSELECT command retrieves records from the CLIENTS file with the country equal to Canada. The greater than (>) indicates that the default select list of 0 is active. A LIST statement is then executed against the active select list. Notice that the records are in order by record ID.

:SELECT CLIENTS WITH COUNTRY = "Canada"

31 records selected to list 0.

>LIST CLIENTS NAME CITY COUNTRY

CLIENTS... Name.................... City.........
Country........

10006  Bob Kesic Winnipeg Canada
10011  Natalia Lomonosov Ponthill Canada
10015  Cal di Grigorio Regina Canada
10017  Karen Wu Edmonton Canada
10020  Tony Andropolis Calgary Canada
10024  Sharon Guo Vancouver Canada
10026  Samantha Greenwald Halifax Canada
10029  Alfred Cunningham Vancouver Canada
10033  Benson Witherspoon Vancouver Canada

In the next example, UniQuery executes the same SSELECT statement, but the active select list is saved to a record called CANADA.CLIENTS in the SAVEDLISTS file.

:SELECT CLIENTS WITH COUNTRY = "Canada"

31 records selected to list 0.

>SAVE.LIST CANADA.CLIENTS

31 key(s) saved to 1 record(s).
In the next example, the **GET.LIST** keyword retrieves the saved list and processes a report.

```
:GET.LIST CANADA.CLIENTS
31 records retrieved to list 0.
<List CLIENTS NAME CITY PHONE_NUM
LIST CLIENTS NAME CITY PHONE_NUM 14:55:26 Jun 21 2005 1
CLIENTS... Name.......................... City........... Phone Number..
10006   Bob Kesic                      Winnipeg            2048752698
                    2048752698
10011   Natalia Lomonosov              Fonthill            9058756984
                    9058756984
10015   Cal di Grigorio                Regina              3067881245
                    3065487569
10017   Karen Wu                       Edmonton            4034958564
                    4034958875
10020   Tony Andropolis                Calgary             4032928847
                    4032928854
10024   Sharon Guo                     Vancouver
```

In the next example, UniQuery clears the active select list.

```
:SSELECT CLIENTS WITH COUNTRY = "Canada"
31 records selected to list 0.

>CLEARSELECT
```

**Related Commands**

**SELECT, SORT**
SUM

Syntax

SUM filename attribute [record_IDs] [selection_criteria]

Description

The UniQuery SUM command adds numeric attributes within a file. SUM produces a total for the attributes added, as well as a total count of the number of records included in the calculation. Only a single result is produced, and no detail is displayed. To display detail, use the UniQuery TOTAL keyword.

Parameters

The following table describes the parameters of the SUM command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Name of the file from which to retrieve data. You may specify only one file name for each UniQuery statement.</td>
</tr>
<tr>
<td>attribute</td>
<td>Specifies the record attributes to be added. At least one attribute name is required. The attributes to be summed should only contain numeric values.</td>
</tr>
<tr>
<td>record_IDs</td>
<td>Specifies which record IDs to sum and/or test against the selection criteria.</td>
</tr>
<tr>
<td>selection_criteria</td>
<td>Specifies conditions for bypassing or retrieving a particular record.</td>
</tr>
</tbody>
</table>
Example

In the following example, the SUM command adds the PRICE attribute in the INVENTORY file.

```
:SUM INVENTORY PRICE
Total PRICE = $198,494.05
175 records summed
:
```
**SUPP**

SUPP is a synonym for the HDR.SUPP keyword. For further information, see HDR.SUPP.

**Synonyms**

**ECLTYPE U**

HDR.SUP, HDR-SUP, HDR.SUPP, HDR-SUPP

**ECLTYPE P**

HDR.SUPP, HDR-SUPP
TOTAL

Syntax

...TOTAL attribute

Description

The UniQuery TOTAL keyword is used with the LIST and SORT commands to total a numeric attribute. If a UniQuery statement contains breakpoints, the total displays at each breakpoint and at the end of the report. If no breakpoints are contained in the UniQuery statement, the total displays only at the end of the report.

Note: The TOTAL keyword is not the same as the TOTAL function used in virtual attribute definitions.
Example

In the following example, records are listed from the INVENTORY file, with a breakpoint on the product name. The TOTAL keyword totals the QTY and PRICE attributes, displaying totals at the breakpoint and at the end of the report.

```
LIST INVENTORY BY PROD_NAME BREAK.ON PROD_NAME TOTAL QTY TOTAL PRICE
LIST INVENTORY BY PROD_NAME BREAK.ON PROD_NAME TOTAL QTY TOTAL PRICE 13:38:45 Jun 22 2005 1
Product
INVENTORY. Name...... Quantity Price.....

10007 Adapter 544 $129.95
13001 Adapter 467 $94.00
13002 Adapter 104 $150.00
********** -------- ----------
Adapter 1115 $373.95

39400 CD Player 399 $89.87
499 $89.87
39500 CD Player -551 $179.97
********** -------- ----------
CD Player 347 $359.71
.
.
.
56090 Wrist Rest 500 $12.99
500 $12.99
500 $12.99
500 $12.99
499 $12.99
********** -------- ----------
Wrist Rest 2499 $64.95

======== =========
TOTAL 343332 $198,494.05

175 records listed
```
UNLIKE

...attribute UNLIKE [string | pattern]

Synonyms

NOT.MATCH, NOT.MATCHING

Description

The UniQuery UNLIKE keyword is a conditional operator used with WITH and WHEN selection clauses to retrieve records containing a value in attribute that does not match the specified string or pattern. string can be either a numeric or alphanumeric string. pattern can be any pattern described in the following table.

Ellipses (...) are used with the UNLIKE keyword to designate where in string to test for a match. If ellipses precede string, UNLIKE searches for the string at the end of the value. If ellipses follow string, UNLIKE searches for string at the beginning of the value. If ellipses precede and follow string, UNLIKE searches for string in the middle of the value.

Note: You cannot use an alternate index with an UNLIKE clause when the comparative attribute is numeric.

UNLIKE and its synonyms are only supported in ECLTYPE U.

Patterns

The following table lists the valid patterns you can use with the UNLIKE keyword.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0X or ...</td>
<td>Any number of characters, including no characters.</td>
</tr>
<tr>
<td>nX</td>
<td>n number of any character.</td>
</tr>
<tr>
<td>0A</td>
<td>Any number of alphabetic characters, including none.</td>
</tr>
</tbody>
</table>

Supported Patterns
In the following example, the UNLIKE keyword returns records from the CLIENTS file where the name of the city does not begin with the letter P.

```plaintext
LIST CLIENTS WITH CITY UNLIKE "P..." CITY
```

```
LIST CLIENTS WITH CITY UNLIKE "P..." CITY 16:01:06 Jun 22 2005 1
CLIENTS... City...........
10053      Fonthill
10054      Orleans
10055      Lowell
10056      Sydney
10057      St. John's
10059      Mount Holly
10060      Bondi Junction
10061      Regina
```

A number of alphabetic characters.

"n-mA"   n to m number of alphabetic characters.

"n-mX"   n to m number of any characters.

0N       Any number of numeric characters, including none.

nN       n number of numeric characters.

"n-mN"   n to m number of numeric characters.

`text`   Exact text. Text must be enclosed in quotation marks.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nA</td>
<td>n number of alphabetic characters.</td>
</tr>
<tr>
<td>&quot;n-mA&quot;</td>
<td>n to m number of alphabetic characters.</td>
</tr>
<tr>
<td>&quot;n-mX&quot;</td>
<td>n to m number of any characters.</td>
</tr>
<tr>
<td>0N</td>
<td>Any number of numeric characters, including none.</td>
</tr>
<tr>
<td>nN</td>
<td>n number of numeric characters.</td>
</tr>
<tr>
<td>&quot;n-mN&quot;</td>
<td>n to m number of numeric characters.</td>
</tr>
<tr>
<td><code>text</code></td>
<td>Exact text. Text must be enclosed in quotation marks.</td>
</tr>
</tbody>
</table>

**Supported Patterns (continued)**

**Examples**

In the following example, the UNLIKE keyword returns records from the CLIENTS file where the name of the city does not begin with the letter P.
In the next example, the UNLIKE keyword returns records from the CLIENTS file where the city is not five alphabetic characters in length.

```
:LIST CLIENTS WITH CITY UNLIKE "5A" CITY
LIST CLIENTS WITH CITY UNLIKE "5A" CITY 16:04:20 Jun 22 2005 1
CLIENTS... City.........

10053      Fonthill
10054      Orleans
10055      Lowell
10056      Sydney
10057      St. John's
10059      Mount Holly
10060      Bondi Junction
10061      Regina
10062      Nice
10063      Youngstown
```

.
USING

Syntax

...USING DICT filename

Description

The UniQuery USING keyword specifies the use of a dictionary from a file other than the current file to display or retrieve data.

Example

In the following example, the USING keyword specifies the dictionary of the VOC file to be used to list information from the CLIENTS file.

```
LIST CLIENTS F1 F5 USING DICT VOC
9999    Paul     Paris
10052   Paul     Perth
10053   Gino     Fonthill
10054   Gregory  Orleans
10055   Cathy    Lowell
10056   Samuel   Sydney
10057   Subrina  St. John's
10058   Antonette Paris
10059   Weiming  Mount Holly
```
VAL.OF

Syntax

...WHEN VAL.OF ms_attribute selection_criteria

Description

The UniQuery VAL.OF keyword is used with the WHEN keyword to return all of the multi-subvalues of an attribute associated with a multivalued attribute when at least one of the subvalues matches selection_criteria.

Examples

In the following example, the WHEN keyword returns the product name of records in the ORDERS file that are green. PROD_NAME is defined as a multivalued attribute, associated with COLOR, which is a multi-subvalued attribute.

```
:LIST ORDERS WHEN COLOR = 'Green' PROD_NAME COLOR
LIST ORDERS WHEN COLOR = 'Green' PROD_NAME COLOR 15:06:43 Jun 31
2005 1
ORDERS.... Product Name Color.....

848    Telephone    Green
950    Telephone    Green
828    Camcorder    Green
Bag
802    Telephone    Green
Telephone    Green
806    Camcorder    Green
Bag
5 records listed
```
WHEN is a restrictive keyword, returning only those values of the multi-subvalued field, COLOR, that exactly match “Green”. By adding the VAL.OF keyword to the same UniQuery statement, all colors are returned when at least one of the multi-subvalues matches “Green”.

```
:LIST ORDERS WHEN VAL.OF COLOR = 'Green' PROD_NAME COLOR
LIST ORDERS WHEN VAL.OF COLOR = 'Green' PROD_NAME COLOR 15:11:11
Jun 31 2005 1
ORDERS.... Product Name Color.....
848     Telephone   Green
950     Telephone   Red
         Green
828     Camcorder   Green
         Bag
802     Telephone   Green
806     Camcorder   Green
         Bag
5 records listed
```
VERT

VERT is a synonym for the VERTICAL keyword. For further information, see VERTICAL.

Synonyms

VERTICAL, VERTICALLY
VERTICAL

Syntax

...VERTICAL

Synonyms

VERT, VERTICALLY

Description

The UniQuery VERTICAL keyword is used with the LIST and SORT commands to print each attribute of a report vertically. UniQuery prints reports horizontally by default. If the information is too wide to fix across the page or screen, UniQuery prints reports vertically.

Example

In the following example, UniQuery prints the names and address from records in the CLIENTS file vertically.

```
:LIST CLIENTS NAME ADDRESS CSZ VERTICAL
LIST CLIENTS NAME ADDRESS CSZ VERTICAL 09:51:40 Jun 23 2005 1
CLIENTS 9999
Name     Paul Castiglione
Address  45, reu de Rivoli
         Paris,    75008

CLIENTS 10052
Name     Paul O'Hare
Address  918 W. Alta St.
         Perth,   8569

CLIENTS 10053
Name     Gino Lee
Address  483 E. Silverman St.
         Fonthill, Ontario  L0S1E5
```
WHEN

Syntax

...WHEN [ASSOCIATED] [EVERY] condition [AND | OR] [EVERY] condition ...

Description

The UniQuery WHEN keyword is used with the LIST or SORT commands to return only those values in a multivalued attribute that satisfy selection criteria. WHEN limits the values that a UniQuery statements displays, it does not limit the records selected. Therefore, you cannot use the WHEN keyword with the SELECT command.

A UniQuery statement can contain multiple WHEN statements, but no more than 15 WHEN statements can appear in one UniQuery statement. If two WHEN statements are joined with the AND operator, both conditions must be true. If the statements are joined with the OR operator, either condition must be true.

Note: The WHEN keyword is only valid in ECLTYPE U.

Using WHEN ASSOCIATED

The UniQuery ASSOCIATED keyword in used in conjunction with the WHEN keyword, and operates on multivalued and multi-subvalued attributes where an association is defined in the dictionary record of the attribute. The ASSOCIATED keyword returns only those records, and values within those records, that have values in the same position in the multivalued or multi-subvalued attribute matching both specified criteria.

Attributes listed in an ASSOCIATED phrase must have an association defined in attribute 7 of the dictionary record. The association definitions must be the same for the associated attributes, and the association phrase record must be defined in the dictionary.

For information on creating associations, see Using UniData.
Note: UDT.OPTIONS 22 determines whether UniQuery WITH and WHEN comparisons use the numeric value or the string value of data. When this option is on, under certain conditions, the comparison uses the string value of the data. When this option is off, UniQuery uses standard comparisons.

UDT.OPTIONS 94 affects UniQuery statements that use a WHEN clause with two or more associated multivalued or multi-subvalued attributes. In this kind of statement, UDT.OPTIONS 94 ON makes a WHEN clause the same as a WHEN ASSOCIATED clause.

Example

In the following example, UniQuery returns only those values from records in the ORDERS file that have a quantity greater than 10, and a price greater than $100.00. Since the dictionary items for QTY and PRICE are associated, WHEN ASSOCIATED is used in the query statement.

```
LIST ORDERS WHEN ASSOCIATED(QTY GT '10' AND PRICE GT "100.00")
QTY  PRICE
LIST ORDERS WHEN ASSOCIATED(QTY GT '10' AND PRICE GT "100.00") QTY  PRICE 14:38:34 Jun 24 2005 1
ORDERS.... Qty... Price.....
872        45    $129.87
877        45    $199.95
878        25    $179.99
971        15    $479.99
818        500   $119.95
818        500   $1,200.00
100        $395.00
100        $500.00
100        $1,200.00
250        $199.95
250        $695.00
250        $199.95
789        100   $228.82
789        50    $249.96
789        50    $249.96
789        850   $179.97
790        1000  $159.94
```
Related Commands

ASSOCIATED, WITH
WITH

Syntax

...WITH [EVERY] condition [AND | OR] [EVERY] condition ...

Description

The UniQuery WITH keyword is used to select or display records that meet defined conditions. WITH displays all values of a multivalued attribute, even if only one value meets the selection criteria. To limit the display of values, use the WHEN keyword.

A UniQuery statement can contain multiple WITH statements, but no more than 120 WITH statements can appear in one UniQuery statement. If two WITH statements are joined with the AND operator, both conditions must be true. If the statements are joined with the OR operator, either condition must be true. If a UniQuery statement contains multiple WITH statements not joined with AND or OR, UniQuery assumes the AND operator.

Note: UDT.OPTIONS 22 determines whether UniQuery WITH and WHEN comparisons use the numeric value or the string value of data. When this option is on, the comparison uses the string value of the data under certain conditions. When this option is off, UniQuery uses standard comparisons.
Examples

In the following example, the WITH keyword is used to select records from the ORDERS file with the ORD_DATE equal to 01/14/96, 01/15/96, or 01/21/96.

```
LIST ORDERS WITH ORD_DATE = "01/14/96" OR WITH ORD_DATE = "01/15/96" OR WITH ORD_DATE = "01/21/96"
```

```
LIST ORDERS WITH ORD_DATE = "01/14/96" OR WITH ORD_DATE = "01/15/96" OR WITH ORD_DATE = "01/21/96" ORD_DATE
```

```
15:14:26 Mar 18 2005
ORDERS.... Order Date
965 01/15/96
934 01/14/96
873 01/21/96
966 01/15/96
935 01/14/96
.
.
.
```

If more than one value is specified using WITH statements, the values may be separated by spaces if OR is the desired operator, as shown in the following example.

```
LIST ORDERS WITH ORD_DATE = "01/14/96" "01/15/96" "01/21/96"
```

```
LIST ORDERS WITH ORD_DATE = "01/14/96" "01/15/96" "01/21/96" ORD_DATE 15:46:18 Mar 18 2005
ORDERS.... Order Date
965 01/15/96
934 01/14/96
873 01/21/96
966 01/15/96
935 01/14/96
.
.
.
```

Notice that both previous examples return the same results. This is because OR is implied when you enter multiple values; you do not have to reenter “OR WITH”.
If a UniQuery statement contains multiple WITH statements not joined with AND or OR, UniQuery assumes the AND operator. In the following example, UniQuery returns all records from the ORDERS file for client number 9968, with an order date prior to 02/15/96.

```
LIST ORDERS WITH CLIENT_NO = '9968' WITH ORD_DATE < '01/15/96'
ORD_DATE
LIST ORDERS WITH CLIENT_NO = '9968' WITH ORD_DATE < '01/15/96'
ORD_DATE 16:03:19 Mar 18 2005 1
ORDERS.... Order Date
  901       01/13/96
  840       06/20/95
2 records listed
```

The previous UniQuery statement returns the records where both conditions are true. If you use the OR operator, UniQuery returns all records where either condition is true.

**Related Commands**

EVERY, WHEN
WITHIN

Syntax

`command filename record_IDs WITHIN attribute`

Description

UniData has the capability to support the “sublist” concept, where a multivalued or multi-subvalued attribute consists of record IDs in the same file to represent hierarchical relationships. If your data uses this sublist concept, the UniQuery WITHIN keyword displays the hierarchical relationship of `attribute`.

A company could have an employee file consisting of all employees. One employee could supervise several employees who, in turn, supervise other employees. You can represent this type of relationship in a UniData file, and display the relationship using the WITHIN keyword.
Examples

In the following example, the NAME and SUPERVISE attributes from an EMPLOYEES file are displayed. The SUPERVISE attribute consists of record IDs within the EMPLOYEES file, representing the record IDs of other employees the person supervises.

```
:LIST EMPLOYEES NAME SUPERVISE
LIST EMPLOYEES NAME SUPERVISE 15:47:51 Aug 01 2005 1
EMPLOYEES. ....................... .......

1          Jonathan Smith                   4
           9
           6
2          Abigail Martin                  11
           47
3          Doug Rather
4          Michael Gibb
5          Jan Emery
6          Carol Applegate
7          Samuel Jergen                   8
           96
           41
           75
8          Heather Jennison
9          Barry Crompton
10         Abigail Parks
.
.
.
```

In the previous example, the data shows that Jonathan Smith supervises Michael Gibb (employee 4), Barry Crompton (employee 9), and Carol Applegate (employee 6). You can use the WITHIN keyword to clearly display this relationship, as shown in the following example.

```
:LIST EMPLOYEES "1" WITHIN SUPERVISE NAME SUPERVISE
LIST EMPLOYEES "1" WITHIN SUPERVISE NAME SUPERVISE 15:56:09 Jun 01 2005 1
LEVEL EMPLOYEES. ....................... .......

1     1          Jonathan Smith                   4
           9
           6
2     4          Michael Gibb
2     9          Barry Crompton
2     6          Carol Applegate
4 records listed
```