The WINDOW Statement

1. Statement Construct

The WINDOW statement introduces the window construct and specifies its target record type. It is coded:

```
WINDOW id [USING rt1 | rtidx [rta rtb...rtn] [DEPENDENT ON rt2|(fielda, ...fieldn [TO|MATCH|RULE key])]]
```

where *id* is a unique two-character window-id by which the window is identified, *rt1* is the name of the target record, *rta* to *rtd* are master records to be fetched with the target record, *rt1*. *rt2* is the record-id of the record type upon which displays are dependent unless a field list, *fielda*, ... *fieldn* is defined. The TO and MATCH option are available for a field list and *key* is either the key block specifying the end of range for the TO option or the match key block for the MATCH option. A default index, *rtidx* may be specified as the default index for use in enquiries by coding an index name instead of *rt1*.

The skeleton structure of a window is as below:

```
WINDOW id [USING ... [DEPENDENT ON .... ]]
[window-options]
window-body
[ROUTINES SECTION]

ENDWINDOW
```

2. Window-id

The window-id is a two-character alphanumeric name which must start with an alphabetic character, and is used to reference the window during processing, such as when using the ENTER verb. It must therefore be unique amongst the windows declared within the frame, and should not be the same as any record-id referenced by the ACCESS statement.

3. USING Clause

The optional USING clause specifies that the record-id of the window's target record type is *rt1*, access to which must previously have been declared using the ACCESS statement. When used instead of *rt1*, the index name *rtidx* simultaneously defines the record-id and default current index by which records are to be retrieved using the <PGE> and <BPG> operations. If the record-id is coded, the first index defined for that record in the database dictionary will be used as the default index.

Up to four master records which are to be retrieved with the target record are specified by *rta* to *rtd*. Whenever the window manager fetches the target record, these master records will also be retrieved, in much the same way as the READ verb. This facility is used to display fields that are not stored on the target record (e.g. when displaying the stock description from the stock record within order line items). Note that records retrieved using this facility **will be unlocked**.

This facility can only be used for records that are directly linked to the target record type. If this is not the case, the same effect can be achieved by coding a fetch routine within the routines section. This routine must then perform the necessary fetch or get operations to cause the required records to be retrieved.

Omitting the USING clause indicates that the window is not associated with records residing on the database. The window will therefore operate in ADD and EDT modes only, and any necessary I/O operations will have to be explicitly coded within the frame.

4. DEPENDENT ON Clause

The optional DEPENDENT ON clause has two different uses. If coded in conjunction with record type *rt2*, it restricts access to the group of records belonging to record *rt2*, which must be established before the window is executed. For example, it may be desirable to process only those invoices that belong to a given customer. This is achieved by first reading, or creating that customer record, and then invoking the invoice window.

When the DEPENDENT ON clause is used with record-id *rt2*, only indexes that begin with the primary index of *rt2* will be used to retrieve the target record type. In other words, an index must start with the same fields as the primary index of the master record. Returning to the example, enquiries can use only those indexes that start with the customer number. This allows the window manager to get straight to the invoices belonging to the customer without having to scan through the entire file. If no such index exists this clause may not be used. If index *rtidx* is specified instead of *rt1*, it must also meet these criteria.

The DEPENDENT ON clause may instead be followed by a list of fields containing the significant portion of the index key upon which the operation of the window will be dependent. For example, coding:

```
WINDOW W1 USING IN DEPENDENT ON (Z-CUST)
```

makes the operation of this window dependent on the value contained in field Z-CUST, which must be set up before the window is entered.

When compiling this statement, the compiler searches for an index which starts with index segment named INCUST, which must have the same picture clause as the field Z-CUST. The important point here is that the last four characters of the field name serve to identify the corresponding index key segment as stored on the target record.

Where a list of fields is coded to define dependent operation, each field must be defined in the same order as at least one index supported by the target record type. The picture clause of each field must match the picture clause of the corresponding index key segments, and the specified fields must be **located in contiguous memory locations**. For example, let us create an order line item window which shows only the order lines for a given order. Given that the order line record contains an index composed of:

Customer Number	OLCUST X(4)
Order Number	OLORDN X(5)
Stock Number	OLSTCK X(8)

we must specify a window which is dependant on **both** the customer and order number fields. This should be done by coding the following in the Data Division:

```
01    Z-KEY                     * Dependency key for order lines
  03  Z-CUST      PIC X(4)      * - Customer number
  03  Z-ORDN      PIC X(5)      * - Order number
```

The Window Statement would then be coded:

```
WINDOW W1 USING OL DEPENDENT ON (Z-CUST, Z-ORDN)
```

During compilation, the compiler identifies the appropriate index from the names of the specified field segments, and checks that the picture clauses are identical. At run time, it is then necessary to move the required customer and order numbers into Z-CUST and Z-ORDN respectively, prior to entering the window.

It is also possible to supply a range of key values for example coding:

```
WINDOW W1 USING IN DEPENDENT ON (Z-CUST, Z-ORDN TO Z-EKEY)
```

will give you all the customers in the range of values from Z-CUST, Z-ORDN to Z-EKEY. The structure of the end key block must be the same as the key block so for this example the key block in the DATA DIVISION should be coded:

```
01    Z-KEY                     * Dependency key for order lines
  03  Z-CUST      PIC X(4)      * - Customer number
  03  Z-ORDN      PIC X(5)      * - Order number
```

The end key block should also be coded as:

```
01    Z-EKEY                    * Dependency key for order lines
  03  Z-ECUST     PIC X(4)      * - Customer number
  03  Z-EORDN     PIC X(5)      * - Order number
```

At run time, it is then necessary to move the required start key for the customer and order number into Z-CUST and Z-ORDN respectively and the required end key in Z-ECUST and Z-EORDN, prior to entering the window.

If the key range is not sufficient to define the list of records it is possible to define a key table containing the specific keys for the records required by for example coding:

```
WINDOW W1 USING IN DEPENDENT ON (Z-CUST, Z-ORDN MATCH Z-KTAB)
```

This will give you all the customers that match the values of customers/order combination of the key entries in the key table Z-KTAB. The values held in the fields Z-CUST and Z-ORDN are not considered but they must be present in the DATA DIVISION as above.

The key table must contain a start marker of the length of the key set to HIGH-VALUES, followed by the list of keys to match against in order of key value, and an end marker of the length of the set to HIGH-VALUES. For this example the key table coded in the DATA DIVISION may be coded as:

```
01     Z-KTB                     * Dependency key table for order lines
 02    FILLER      PIC X(9)      * Start marker
                   VALUE HIGH-VALUES
 02    Z-K OCCURS 3              * Key 1
  03   Z-KCUST     PIC X(4)
  03   Z-KORDN     PIC X(5)
 02    FILLER      PIC X(9)            * End marker
                   VALUE HIGH-VALUES
```

At run time, it is then necessary to move the required three key values for the customer and order number combinations Z-KCUST, Z-KORDN respectively prior to entering the window.

You may also restrict the records shown in the window to a multiple set of ranges by using a RULE table for example coding:

```
WINDOW W1 USING IN DEPENDENT ON (Z-CUST, Z-ORDN RULE Z-RTAB)
```

This will give you all the customers that are in the ranges of values of customers/order combinations defined in the rule table. The values held in the fields Z-CUST and Z-ORDN are not considered but they must be present in the DATA DIVISION as above.

The rule table must contain the start and end key values (inclusively) of the allowed ranges, which must be held in key order. For this example the key table coded in the DATA DIVISION may be coded as:

```
01     Z-RTB                     * Rule table
 02    Z-RTV       PIC 9(4) C    * Table version must be 1
                   VALUE 1
 02    Z-RTM       PIC 9(4) C    * Table entry mode
                                 * = 1 - Entries of start and end range
 02    Z-RFIL      PIC X(50)     * work space for Speedbase must be set to
                                 * LOW-VALUES by application before window
                                 * is first displayed/entered
 02    Z-RNO       PIC 9(4)      * Number of table pair entries
 02    Z-RKYT  OCCURS Z-RNO      * MUST BE IN KEY ORDER
  03   Z-RST                     * Start from key
   04  Z-SCUST     PIC X(4)
   04  Z-SORDN     PIC X(5)
  03   Z-REN                     * End key
   04  Z-ECUST     PIC X(4)
   04  Z-EORDN     PIC X(5)
```

At run time, it is necessary to move the required key values for the customer and order number ranges required prior to entering the window. The Z-RFIL field must be set to LOW-VALUES before the window is **first** displayed. If the window is cleared, Z-RFIL must be set once more before the window is redisplayed.

It should be noted that for DBX databases, even though an index may contain up to 16 segments, only the first 8 segments can be used in the enquiry key on a window.

Also note that the window order can also be changes by using the SCROLL REVERSE and REVSUBKEY window options. (See WINDOW options)

5. See Also

ENTER WINDOW Statement
DISPLAY WINDOW Statement
CLEAR WINDOW Statement
WINDOW Options
WINDOW Body
ROUTINES SECTION