SUPERDEX

Demonstration Manual

Version 3.1

All updates to or derivatives of the SUPERDEX™ computer software provided herein are copyrighted and may not be copied except for archival purposes, to replace a defective copy, or for program error verification by Licensee. Copyrighted material may not be copied onto any media (e.g. magnetic tape, paper tape, disc memory cartridges, read-only memory, etc.) for any other purposes. The authorization to duplicate copyrighted materials hereunder shall not be construed to grant the Licensee or Licensee's customer the right to use copyrighted SUPERDEX material in any manner other than which is provided in this agreement or otherwise approved in writing by Dr. Wolfgang Matt or Bradmark Technologies.

(c) **1988 Bradmark Technologies, Inc.**

Released March, 1992

IMAGE, TurboIMAGE, and TurboIMAGE/XL are trademarks of Hewlett-Packard Company

dBASE is a trademark of Ashton-Tate Corporation

SUPERDEX is a trademarked product names of Bradmark Technologies, Inc. for the SI-IMAGE package developed and implemented by Dr. Wolfgang Matt

About this manual

This manual, when used in conjunction with the demonstration database and programs supplied, will give you an introduction to SUPERDEX which will let you experience various SUPERDEX retrievals performed using search criteria which you provide.

No knowledge of the SUPERDEX package is assumed for this demonstration.

This manual is arranged in the following format:

Section 1 gives an *Introduction* of the demonstration package and explains how to set up the demo environment.

Section 2 describes SUPERDEX by leading you through the interactive *COBOL demonstration* programs provided, thus allowing you to experience first-hand SUPERDEX's powerful retrieval capabilities and amazing speed. Data values are suggested but you are free to choose any value(s) you want. An explanation which includes data structures, program operation and how the demo works is given for each demo.

Section 3 reviews the *SUPERDEX index structures* used in the demonstration database and explains how they are utilized throughout the demos. This is followed by a discussion on how to configure a new SUPERDEX access path.

Appendix A shows the OEDB *Demo database structure* utilized throughout the SUPERDEX demos.

Appendix B contains listings of the *COBOL source programs* used in the SUPERDEX demos in section 2.

Table of contents

Section 1: Introduction 1-1

Features 1-1
COBOL demonstration programs 1-1
Demonstration database 1-1
Database access 1-2
Loading the software 1-2
Running the demonstrations 1-2

Section 2: COBOL demo 2-3

Running the demonstrations 2-3
Simple key demo 2-4
About the demo 2-4
Running the demo 2-4
Further demonstration 2-7
How the demo works 2-7
Concatenated key demo 2-9
About the demo 2-9
Running the demo 2-10
Further demonstration 2-11
How the demo works 2-12
Keyworded key demo 2-13
About the demo 2-13
Running the demo 2-14
Further demonstration 2-15
How the demo works 2-15
Grouped key demo 2-16
About the demo 2-16
Running the demo 2-17
Further demonstration 2-18
How the demo works 2-19
Relational access demo - multiple criteria 2-20
About the demo 2-20
Running the demo 2-20
Further demonstration 2-22
How the demos work 2-22
Relational access demo - multiple datasets 2-23
About the demo 2-23
Running the demo 2-24
Further demonstration 2-25
How the demo works 2-26

Section 3: SUPERDEX index structures 3-27
SUPERDEX paths 3-27
Configuring SI-paths 3-28
Creating a new SI-path 3-28

Appendix A: Demo database structure A-32

Appendix B: COBOL source programs B-35
Simple Key Demo B-36
Concatenated Key Demo B-42
Keyworded Key Demo B-48
Grouped Key Demo B-52
Relational Access Demo - multiple datasets B-56

Section 1 Introduction

Features

This demonstration facility gives you the ability to interactively experience SUPERDEX's enhanced data retrieval capabilities which include:

- multiple keys in master and detail datasets
- concatenated keys containing multiple fields
- sorted sequential retrieval
- automatic keywording and keyword retrieval
- generic and partial-key retrieval
- grouping of functionally-equivalent fields
- multiple value lookup
- relational access across multiple fields, datasets, and databases

COBOL demonstration programs

These features are shown by use of five COBOL programs which call replacement IMAGE™ compatible SUPERDEX intrinsics. The replacement SUPERDEX intrinsics have the same names as, and are functionally equivalent to, the regular IMAGE intrinsics; they use the same methods that you would use in your programs.

Edited listings of the demonstration source programs are included in *Appendix B* of this manual, with complete sources contained in the DEMO.SUPERDEX files.

Demonstration database

A partial order entry database (called **OEDB**) is provided to facilitate the interactive demos. It contains only four datasets which are used as follows:

CUSTOMERS	Stand-alone manual master containing 1000 customer entries; IMAGE search item is CUSTOMER-NUMBER.
ORDER-HEADERS	Manual master containing 2620 order headers; IMAGE search item is ORDER-NUMBER.
ORDER-LINES	Detail dataset, related to ORDER-HEADERS, containing 10245 line items related to order headers; IMAGE search item is ORDER-NUMBER.
SI	Stand-alone detail dataset in which all SUPERDEX index structures are maintained. Contains only the item **SI**.

A complete database layout is contained in *Appendix A* of this manual.

Database access

Although the entries in this database can be accessed by their IMAGE search items, this demonstration utilizes SUPERDEX access techniques only.

Loading the software

First, load the SUPERDEX software from the installation tape, following the separate *SUPERDEX loading instructions*.

Then, logon:

```
:HELLO MGR.SUPERDEX,DEMO
```

Once you have done this, you are ready to run through the demonstrations.

Running the demonstrations

The demonstration programs utilize VPLUS forms, so you must use a terminal or be running a terminal emulator that supports VPLUS.

Remember to **TAB** between fields and use the **ENTER** key when you're done with a screen. If you want to clear a value entered in a field, type or SPACE over the old value, or press the CLEAR DISPLAY key.

Section 2 COBOL demo

Running the demonstrations

To run the COBOL demonstration programs, type

```
:COBOLDEMO
```

at the MPE colon prompt (do not type the :) to display the following menu:

```
                    Bradmark Technologies

                        S U P E R D E X

                         Demonstration


                    1. Simple Key Demo

                    2. Concatenated Key Demo

                    3. Keyworded Key Demo

                    4. Grouped Key Demo

                    5. Relational Access Demo


  Enter Selection _

SUPERDEX is a trademarked product name of Bradmark Technologies for the SI-IMAGE
package developed and implemented by Dr. Wolfgang Matt
```

Five separate demonstration programs which are described on the following pages may be run from this main menu.

Simple key demo

About the demo

A *simple* SUPERDEX key is very much like an IMAGE search item except that its capabilities are extended in various ways, such as:

- sorted sequential retrieval
- generic and partial key retrieval
- less-than, greater-than, and range retrieval

The Simple Key Demo illustrates how to use a *simple SI-key* (SUPERDEX key) to locate customer entries in the master dataset called CUSTOMERS.

Running the demo

Select option **1** from the Main Menu and press **ENTER** to proceed to the Simple Key Demo.

The following screen is displayed:

```
                    Simple Key Demo

Customer _____________________________________ Direction (F,B) _
================================================================================
Customer Name                  Customer #
```

The first input field is for the customer name to be searched for. The second field indicates whether entries should be returned in forward (ascending) or backward (descending) alphabetical order (**F** for forward or **B** for backward).

Type

UNITED AIRLINES

in the `Customer` field. Enter

F

in the `Direction` field. When you press **ENTER**, SUPERDEX returns the corresponding entry:

```
UNITED AIRLINES                0002112949
```

This is very much like performing an IMAGE DBFIND against a search item value. However, unlike IMAGE, SUPERDEX also supports partial key and generic retrievals. Change the value in the `Customer` field to

UNITED@

and press **ENTER**. All entries that start with "UNITED" are displayed:

```
UNITED AIRLINES                0002112949
UNITED ALLOYS & STEEL          0002100649
UNITED BUSINESS EQUIPMENT      0002100652
UNITED CEREBRAL PALSY ASSN     0002100400
UNITED CHURCH HOME             0002100304
UNITED FUND BUFF & ERIE        0002100401
UNITED IMPORT MOTORS INC       0002100700
UNITED PRESB CHURCH            0002100509
```

Similar to MPE's :LISTF command, the **@** character tells SUPERDEX to match zero or more characters in the position where the **@** is specified; the difference is that with SUPERDEX, characters following the **@** are ignored. If you specify a customer of just **@**, SUPERDEX will retrieve all 1000 entries in the dataset.

SUPERDEX automatically returned the entries in ascending sequential order because **F** is still in the `Direction` field.

To try a descending order retrieval using a new wildcard, type

UNI?E@

in the first field. Change the `Direction` field to

B

and press **ENTER**. The **?** matchcode is used as a place-holder and represents a single alphanumeric character (like in :LISTF). All entries that start with "UNI" and contain an "E" in the fifth position which is followed by alpha or numeric character(s) are displayed:

```
UNIVERSITY BOOKSTORE           0002100606
UNITED PRESB CHURCH            0002100509
UNITED IMPORT MOTORS INC       0002100700
UNITED FUND BUFF & ERIE        0002100401
UNITED CHURCH HOME             0002100304
UNITED CEREBRAL PALSY ASSN     0002100400
UNITED BUSINESS EQUIPMENT      0002100652
UNITED ALLOYS AND STEEL        0002100649
UNITED AIRLINES                0002112949
```

Note that entries are now returned in descending order.

In the `Customer` field, type

`>=UN@<=UNI@`

and press **ENTER**. This locates a range of entries starting with "UN" through "UNi," inclusive:

```
UNIVERSITY BOOKSTORE          0002100606
UNITED PRESB CHURCH           0002100509
UNITED IMPORT MOTORS INC      0002100700
UNITED FUND BUFF & ERIE       0002100401
UNITED CHURCH HOME            0002100304
UNITED CEREBRAL PALSY ASSN    0002100400
UNITED BUSINESS EQUIPMENT     0002100652
UNITED ALLOYS AND STEEL       0002100649
UNITED AIRLINES               0002112949
UNITARIAN CHURCH              0002100207
UNDERWRITERS SALVAGE CO       0002100347
```

Further demonstration

You are beginning to see the ease-of-use, flexibility and power of SUPERDEX SI-keys.

Try out additional values to further experiment with simple SI-keys. You may want to familiarize yourself with the following new operators by imbedding them in values for the `Customer` field:

>=*value*	greater-than or equal-to retrieval
<=*value*	less-than or equal-to retrieval
<>*value*	not-equal-to retrieval

Press the **f8** key when you are done to return to the Main Menu.

How the demo works

Although SUPERDEX offers amazingly fast and powerful retrievals, it is surprisingly easy to implement. SUPERDEX attempts to look and feel as much like IMAGE as possible so it is simple to learn and use.

The retrievals in this demonstration were accomplished by accessing SUPERDEX index structures contained in a special stand-alone detail dataset named **SI**. Each unique relationship is referred to as an *SI-path* and it is accessed in very much the same way as accessing an IMAGE path. In this demo, an SI-path exists for customer name.

The program uses SUPERDEX's DBFIND *mode* 1 followed by a DBGET *mode* 5 or 6 which specifies the SI-path in the *item* parameter of DBFIND. Doing a DBFIND on the manual master CUSTOMERS may seem odd -- IMAGE's DBFIND works only for details -- but SUPERDEX's replacement intrinsics also operate on master datasets because the dataset name is declared in the *dset* parameter of DBFIND.

SUPERDEX's DBFIND *mode* 1 accepts *arguments* that contain special operators, such as **@** and **?**. In this program, the customer you specify is passed as the *argument* for DBFIND and the number of qualifying entries is returned by SUPERDEX in words 5 and 6 of the *status* array. The entries are retrieved and displayed in sorted order with DBGET 5 or 6 and, as in IMAGE, return an end-of-chain or beginning-of-chain condition.

A complete copy of the source program appears in *Appendix B* and in the file SDEMOSK.DEMO.SUPERDEX.

Concatenated key demo

About the demo

A *concatenated* SI-key consists of the values of two or more fields concatenated together. This not only permits entries to be located by the combination of values for the various concatenated fields (thereby avoiding lengthy chained reads) but it also imposes extended sorting capabilities.

This demo shows

- concatenated keys containing multiple fields
- extended sorted sequential retrieval

The Concatenated Key Demo illustrates the use of a concatenated SI-key to locate order line items in the ORDER-LINES detail dataset.

Running the demo

Select option **2** from the Main Menu and press **ENTER** to proceed to the Concatenated Key Demo.

The following screen is displayed:

```
                    Concatenated Key Demo

Order Number _____________  Part Number _________________
================================================================================

Order #      Part Number      Part Description
```

The first input field is for the order number to search for and the second field is for the part number contained in each order line item. Entries must match on both fields in order to qualify.

Specify the `Order Number`

`701257`

in the first field and the `Part Number`

`SCM1511`

in the second field. When you press **ENTER**, SUPERDEX returns the corresponding entry:

```
0000701257  SCM1511          COPYSETS CANARY CA9B  1065
```

With the capability of specifying values for both fields, we were able to avoid a lengthy chained read of the order's chain.

As seen in the Simple Key Demo, SUPERDEX supports partial key retrievals by using **@**; however, the **@** is not required when doing a concatenated key retrieval (the reason is explained later under How The Demo Works). Change the value in the second field to

SCM

and press **ENTER**. All entries with the specified order number and part numbers starting with "SCM" are displayed:

```
0000701257  SCM1312          FOLDER MANILA LTR 1/  1120
0000701257  SCM1511          COPYSTES CANARY CA9B  1065
0000701257  SCM153-ST        PADS TELEPHONE MESSA  1250
0000701257  SCM835-ST        PAD SCRATCH 3X5 9120  1230
0000701257  SCM858-ST        PAD SCRATCH 5X8  912  1235
0000701257  SCM870           PAD STENO GREGG RULE  1240
0000701257  SCM8784          PAD STENO PITMAN RUL  1245
0000701257  SCM9014-ST       PAD LEGAL CANARY PER  1215
0000701257  SCM911-ST        PAD LETTER CANARY 8-  1210
0000701257  SCMA1312          FOLDER MANILA LGL 1/  1125
```

Note that entries are displayed in ascending alphabetical order by both order number and part number. This is because all values contained in a concatenated SI-key are used for sorting purposes; this permits extended sorting by multiple fields to be accomplished without the use of sorted chains. In fact, SUPERDEX concatenated SI-keys permit sorted chains to be eliminated entirely, thus permitting more flexible sorting while averting potential performance problems.

Further demonstration

You may try out additional order number and part number combinations. Because of the way the program is written, you must specify a full order number in the first field but you may specify partial part numbers of any length in the second field.

Clear the value in the second field so that only the order number **701257** is specified and press **ENTER**. Then, try varying the part number and see the results. Also try the order numbers **915066**, **711155**, and **929461** with various part numbers.

Press **f8** when you are done to return to the Main Menu.

How the demo works

This program accesses an SI-path that represents a concatenated SI-key which is comprised of the order number and part number.

The program performs a partial-key retrieval on part number without the use of an **@** in the *argument* as in the Simple Key Demonstration; the partial-key retrieval is accomplished using a special DBFIND *mode* that restricts the number of characters on which SUPERDEX matches.

In the demonstration database, order number is an I2 item and part number is an X14 item; their combined length is 18 bytes. For the retrieval using order number **701257** and part number **SCM**, it was only necessary to match on the first 7 bytes of the concatenated SI-key value (4 bytes for the I2 item and the first 3 bytes of the X14 item). Therefore, DBFIND was called with a *mode* of -107 and an *argument* of **701257SCM**. The *mode* reflects the base value of 100 plus the number of significant bytes (in this case 7). The *mode* is then made negative (if the *mode* were not negative, it would specify 7 *words* rather than *bytes*.)

The program is hard-coded to impose a DBFIND *mode* of at least -104 (the full length of the order number). It then determines the length in bytes of the part description specified and adds the two together. This permits retrievals using either the full order number, no part number, or any number of leading characters of the part number.

Note that the number of qualifying entries is not displayed in this demo program. This is because only DBFIND *mode* 1 returns the entry count in the *status* array. This program used *mode* -104 which is more efficient and provides additional functionality.

A complete copy of the source program appears in *Appendix B* and in the file SDEMOCAT.DEMO.SUPERDEX.

Keyworded key demo

About the demo

A *keyworded* SI-key is just like a simple SI-key except that every significant word contained in the key may be searched on. For example, the customer "BRADMARK TECHNOLOGIES" could be located by **BRADMARK** or **TECHNOLOGIES**.

This demo shows

- keyword retrieval
- generic and partial-keyword retrieval

The Keyworded Key Demo illustrates the use of a keyworded SI-key to locate customers stored in the CUSTOMERS master dataset. It is the same type of retrieval as in the Simple Key Demo using the same CUSTOMER-NAME field but this time it is configured as a keyworded SI-key.

Running the demo

Select option **3** from the Main Menu and press **ENTER** to proceed to the Keyworded Key Demo.

The following screen is displayed:

```
                        Keyworded Key Demo

Enter any word from a customer's name
      ____________________________________________________________
================================================================================
Customer Name                    Customer #
```

To do generic keyword retrieval, you may specify any word contained in any customer name. Type

FRANK

and press **ENTER**. All the customers that contain the word "FRANK" are displayed:

```
CIMINELLI FRANK CONST          0000300057
RIPPLE J FRANK                 0001800510
```

It does not matter where in the field the keyword occurs but it must be separated by spaces or special characters.

SUPERDEX also supports partial-keyword retrieval. Append an **@** to the specified value

FRANK@

and press ENTER. All entries that contain words that start with "FRANK" are displayed:

```
CIMINELLI FRANK CONST          0000300057
RIPPLE J FRANK                 0001800510
FRANKENSTEIN WM D              0000600628
```

As in the Simple Key Demonstration, you may use **@** and/or **?** to perform partial-keyword or generic keyword searches.

Further demonstration

Try using additional keyword values to further experiment with keyworded SI-keys. You may include the **@**, **?**, **>=**, **<=**, and **<>** operators described in the Simple Key Demo.

Note that you will not have any success using the values **ASSN**, **ASSOC**, **CO**, **COMPANY**, **CORP**, or **INC** -- these common words have been excluded from keywording (by entering them in a special file named **KWEXCLUD**) to conserve disk space and optimize retrieval speed.

If you specify just **@** in the input field, you will find that the program indicates that 2790 entries qualify -- even though there are only 1000 entries in the dataset! This is because each keyword occurrence is included in the entry count (returned in the *status* array) and the program is reporting this value. This count does not, however, include the excluded words "ASSN," "ASSOC," etc.

Press **f8** when you are done to return to the Main Menu.

How the demo works

This program is almost identical to the Simple Key Demo program. The main difference is that a keyworded SI-path is referenced and therefore all access against the SI-path is treated as keyworded.

In SUPERDEX, an SI-path may be configured as keyworded or not keyworded. This is strictly a configuration option specified when the SI-path is established; it does not impact any subsequent processing. Keywording is performed automatically when entries are DBPUT, DBUPDATEed, and DBDELETEed, or whenever DBFIND is used. There is no difference in handling a keyworded SI-path versus a non-keyworded SI-path.

A complete copy of the source program appears in *Appendix B* and in the file SDEMOKW.DEMO.SUPERDEX.

Grouped key demo

About the demo

A *grouped* SI-key permits multiple fields in a dataset to be handled as if they were a single field. For example, if three fields contain people's names and you need to locate a specific person, all three fields would be searched in a single simultaneous operation.

This demo shows

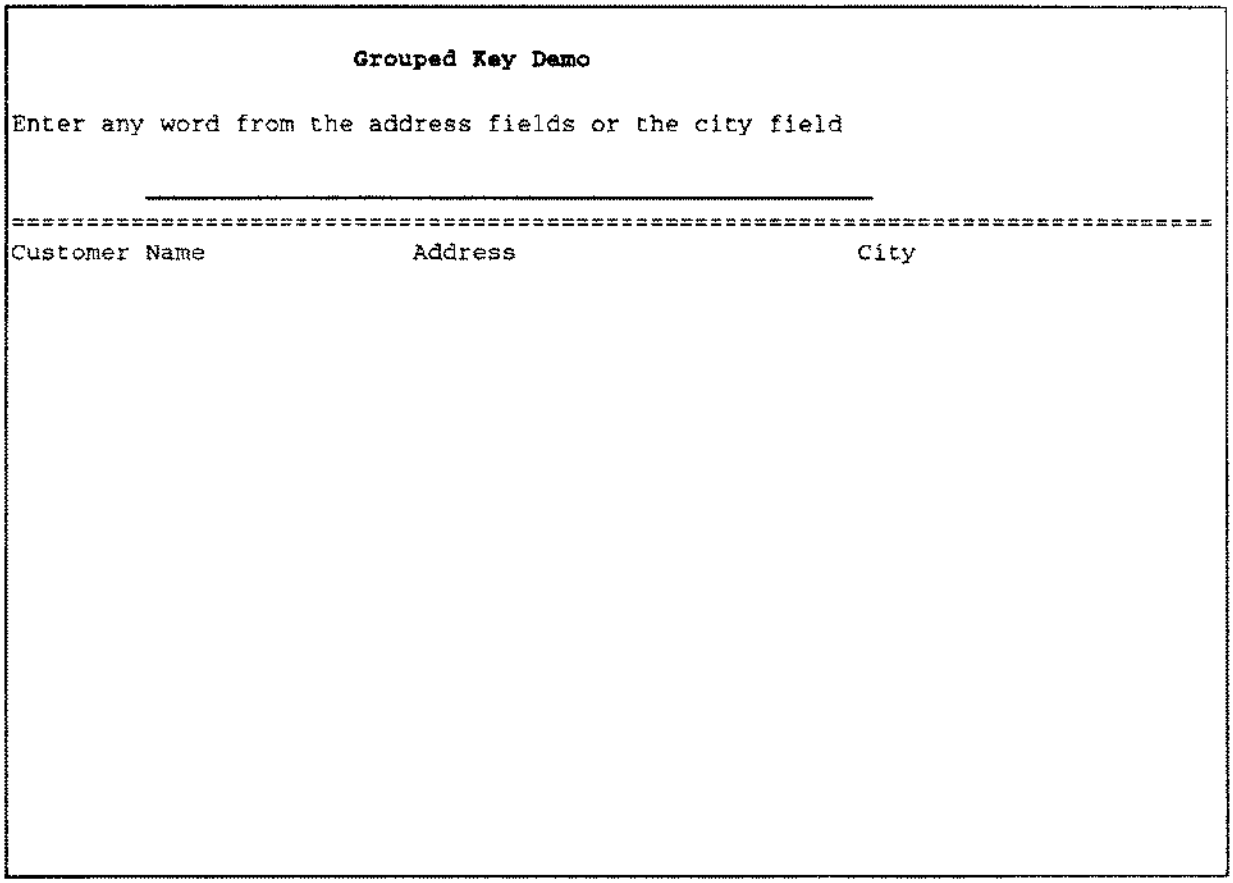
- grouping of functionally equivalent fields
- multiple keys in master and detail datasets
- generic and partial-key retrieval

The Grouped Key Demo illustrates the use of a grouped SI-key to locate customers stored in the CUSTOMERS master dataset by either address or city. These two fields are combined to form a group and the group is configured as keyworded to allow access to any word in either field.

Running the demo

Select option **4** from the Main Menu and press **ENTER** to proceed to the Grouped Key Demo.

The following screen is displayed:



You may specify any word contained in any address or city. Type

KENMORE

and press ENTER. All the customers that have an address on "KENMORE" Avenue or are in the city of "KENMORE" are displayed:

```
BARBER-COLMAN CO              1249 MILITARY RD         KENMORE
CASSETTA AGENCY CO INC        810 KENMORE AVE          BUFFALO
C B N                         3174 DELAWARE AVE        KENMORE
CEGLIA LAWRENCE               2070 SHERIDAN DR         KENMORE
C S F DESIGNS INC             61 GARDENWOOD LANE       KENMORE
CENTURY 21 GOLD JACKET        3411 DELAWARE AVE        KENMORE
CECOS ENVIRONMENTAL INC       2321 KENMORE AVENUE      BUFFALO
CHECKERCAR CLUB OF AMERICA    4693 TERMAINE AVE.       KENMORE
CHECKPOINT FOREIGN CAR        487 KENMORE AVE          BUFFALO
F B L ASSOCIATED AGENCIES     860 ENGLEWOOD AVE        KENMORE
FASO CHARLES P. AGENCY        860 ENGLEWOOD AVE        KENMORE
HOOD COMPANY INC              2225 KENMORE AVENUE      BUFFALO
IMMCO DIAGNOSTICS INC         963 KENMORE AVE          BUFFALO
KOCH RICHARD J CPA            1026 ENGLEWOOD AVE.      KENMORE
LAKELAND AUTOMOTIVE           536 NIAGARA FALLS BLVD   KENMORE
```

It does not matter where the specified keyword or partial-keyword occurs in either field so long as it occurs in one of them. Note that the customer name is displayed for information only -- it is not included in the group and therefore may not be searched on.

Further demonstration

You may try additional values to further experiment with grouping. Try including the **@**, **?**, **>=**, **<=**, and **<>** operators already described.

Try the values **AMHERST**, **NIAG@**, and **WILLIAM@** for interesting results.

If you specify just **@** in the input field, you will find that although the dataset contains only 1000 entries, the program indicates that 4414 entries qualify. This is because each keyword occurrence in both the address and city field is included in the entry count (returned in the *status* array) and the program is reporting this value.

Press **f8** when you are done to return to the Main Menu.

How the demo works

In SUPERDEX, an SI-path may be configured as grouped or not grouped. A grouped SI-path may be keyworded or not keyworded. In this example, the SI-path is configured as both grouped and keyworded; it is comprised of the address and city fields. Other fields, such as a second-line address, can also be included in the group, if desired.

Whether an SI-path is configured as grouped or not is completely transparent to programs. Grouping is performed automatically when entries are DBPUT, DBUPDATEed, DBDELETEed or whenever DBFIND is called. Keywording is also transparent so there is no difference when handling a grouped SI-path vs. a non-grouped SI-path.

A complete copy of the source program appears in *Appendix B* as well as in the file SDEMOGRP.DEMO.SUPERDEX.

Relational access demo - multiple criteria

About the demo

Before proceeding to the last demo program, we must introduce another very powerful concept which applies to the demo programs run thus far:

- *relational access* using multiple values for a field

We have shown how SUPERDEX permits both generic and partial-key retrievals by using the **@**, **?**, **>=**, **<=**, and **<>** operators. However, these capabilities may not always be sufficient to adequately qualify the entries you want. Therefore, you may sometimes find it useful to use a technique called *Relational Access* to further qualify entries.

Running the demo

To illustrate the concept of *Relational Access*, go back to the Simple Key Demo (option **1**) and type the following (including the trailing vertical bar)

```
~UNITED@ OR CENTRAL@;
```

in the `Customer` field. Type

```
F
```

in the `Direction` field and press **ENTER**. The following entries are displayed:

```
CENTRAL BFLO PROJECT CORP.     0000300209
CENTRAL PK UNITED METH         0000300236
CENTRAL AUTO WRECKING          0000300394
CENTRAL CITY RESTORATN         0000300427
CENTRAL ANESTHESIA SVCE        0000300527
CENTRAL ORGAN SERVICE          0000300559
UNITED CHURCH HOME             0002100304
UNITED CEREBRAL PALSY ASSN     0002100400
UNITED FUND BUFF & ERIE        0002100401
UNITED PRESB CHURCH            0002100509
UNITED ALLOYS & STEEL          0002100649
UNITED BUSINESS EQUIPMENT      0002100652
UNITED IMPORT MOTORS INC       0002100700
UNITED AIRLINES                0002112949
```

As illustrated, SUPERDEX selected all the entries that begin with either "CENTRAL" or "UNITED."

This was accomplished by beginning the argument with a tilde (~) and ending it with a ;. When the argument is surrounded with these characters, the words AND, OR and NOT (the boolean operators) may be included in the argument itself.

To further illustrate the Relational Access concept, exit this demo and go to the Keyworded Key Demo (option **3**). Type

FRANK@

and press **ENTER**. The following entries are displayed:

```
CIMINELLI FRANK CONST          0000300057
RIPPLE J FRANK                 0001800510
FRANKENSTEIN WM D              0000600628
```

Now, change the value to

~FRANK NOT FRANKENSTEIN;

and press **ENTER**. This displays all the entries that contain a word starting with "FRANK" and not "FRANKENSTEIN."

```
CIMINELLI FRANK CONST          0000300057
RIPPLE J FRANK                 0001800510
```

To further demonstrate the power and flexibility of Relational Access within an SI-key, exit this demo and go to the Grouped Key Demo (option **4**). Specify

~KENMORE AND BUFFALO;

to display all the entries that contain both "KENMORE" and "BUFFALO" in either the address or city field. "KENMORE" appears only in the address field and "BUFFALO" appears only in the city field because there are no entries in the database for customers with "BUFFALO" in the address field or "KENMORE" in the city field. If there were, they would also qualify for selection.

```
CASSETTA AGENCY CO INC      810 KENMORE AVE          BUFFALO
CECOS ENVIRONMENTAL INC     2321 KENMORE AVENUE      BUFFALO
CHECKPOINT FOREIGN CAR      487 KENMORE AVE          BUFFALO
HOOD COMPANY INC            2225 KENMORE AVENUE      BUFFALO
IMMCO DIAGNOSTICS INC       963 KENMORE AVE          BUFFALO
LOEFFLER F.H. COMPANY INC   328 KENMORE AVE.         BUFFALO
```

Several values with corresponding boolean operators may be specified at one time or in multiple operations (using multiple successive DBFINDs). Type

```
~KENMORE;
```

and press **ENTER**. Note that 17 entries are displayed (the entry count is not shown because it is not returned by this program).

Now, replace the value in the field with

```
~ AND BUFFALO;
```

and press **ENTER**. SUPERDEX remembers the qualifying entries that were found previously and uses them for comparison in the next operation. Now only six entries qualify. Using this technique, you may use successive DBFINDs to refine the selected entries by additional criteria.

Further demonstration

Experiment with the Simple Key, Keyworded Key, and Grouped Key demo programs using boolean operations to get a greater understanding of Relational Access between values in an SI-key.

Several values may be specified with their corresponding boolean operators. For example, the combination

~(*value1 and value2)* OR *value3* NOT *value4;*

is interpreted as "all the entries that contain *value1* AND *value2* OR *value3* AND NOT *value4*."

Press **f8** when you are done to return to the Main Menu.

How the demos work

The three demo programs used to explain Relational Access were the very same programs that were run when illustrating indexed (non-relational) access; they accessed the same SI-paths as before. Whether the value specified is a single value or a multiple values, the value specified is transparent to the programs. Both types of retrievals are supported by the same SI-paths with the same code.

In writing programs for relational access, you may prefer to impose the tilde, ;, and/or boolean operators programmatically and instead present the user with an individual field for each value and function keys to specify the boolean operators. There are many methods for forming the complete value with the required delimiters and operators.

Regardless of how the delimited value is formed, it is passed as the *argument* for DBFIND *mode* 1, exactly as shown. SUPERDEX locates the corresponding entries and returns the qualifying number in words 5 and 6 of the *status* array, just as with non-relational access.

Other features are available for further managing the results of multiple DBFIND calls, including the ability to refine and undo the results of successive DBFINDs.

Relational access demo - multiple datasets

About the demo

As we've seen, *relational access* may be performed within a single field by specifying multiple values for the field and combining them by use of boolean operators.

Relational access can also be used to compare against multiple fields, datasets, and even multiple databases by using similar methods and boolean operators.

This demo shows

- relational access across multiple datasets

This example finds all the order line items that exist for a specified customer and contain a specified part number; this is not a trivial task since there is not a path between the CUSTOMERS master and ORDER-LINES detail. Therefore, a logical relationship must be formed via the ORDER-HEADERS master dataset. To add even greater flexibility, this program permits a partial-key or generic value to be specified for either field.

Running the demo

Select option **5** from the Main Menu and press **ENTER** to proceed to the Relational Access Demo.

The following screen is displayed:

```
                    Relational Access Demo

Enter a Customer Name and a Part Number

Customer Name _____________________  Part Number _____________
================================================================================
Order #    Part #      Part Description              Quan  Price
```

The first input field is for the customer name and the second field is for the part number contained in each order line item for the specified customer. Entries must match on both fields in order to qualify.

Type

UNITED CHURCH@

in the first field. Type

@

in the second field and press **ENTER**. This specifies that SUPERDEX should locate all the order line items for the customer whose name begins with "UNITED CHURCH."

A total of 65 entries are found, starting with:

```
0000701193 A626765N       BNDR,POST.11 X 17,GN         4 107.80
0000701193 Y4403CR        PUNCH,1 HOLE,1/4 DIA         1   1.69
0000701193 R9530609       TAPE,EMBOSS,1/2X144 RL,BK    6  16.50
0000701193 SRA            SR-B STAPLE REMOVER          1   0.68
0000701193 BCMRC21BE      REFILL,F/CLIC,MED,2PK,BE     2  23.52
0000701193 G27-12         COL SHEET                    1  29.61
0000701193 C15-BLK        DISPENSER                    1   4.22
0000701193 BCMRC21BK      REFILL,F/CLIC,MED,2PK,BK     2  23.52
0000701193 WES40290       90-CLASP 9X12 ENVELOPES      1   6.01
0000701193 710-01         JUST FOR COPIES              2   3.12
0000701193 482-2          #100080 MONGOL PENCIL        2   3.06
0000701193 332-01-RED-M   WRITE BROS                  12   1.08
0000701193 334-01-GRN-M   PEN                         12   1.08
0000701193 331-01-BLU-M   WRITE BROS                  24   2.16
0000701193 SCM1312        21-1/3 LTR  FILE FOLDERS     1   3.82
```

In order to narrow down the records selected, change the `Part Number` to

33@

and press **ENTER**. This specifies that only the line items whose part numbers begin with "33" for the customer whose name begins with "UNITED CHURCH" should be displayed. SUPERDEX now returns only the four following entries:

```
0000701193 332-01-RED-M   WRITE BROS                  12   1.08
0000701193 334-01-GRN-M   PEN                         12   1.08
0000701193 331-01-BLU-M   WRITE BROS                  24   2.16
0000928312 334-01-GRN-M   PEN,BALLPOINT,MED PT,GN     12   1.08
```

Further demonstration

You may try out additional customer name and part number combinations by using a full, generic, or partial key for each value.

Note that this demo program automatically encloses the values of both fields with a ~ and ; so you do not need to include the ~ and ; in the values specified. Doing so would cause an additional set of brackets to be imposed and, therefore, no entries would be found. Also, because this program disallows retrievals against more than one customer at a time, the `Customer Name` specified must qualify only one entry.

Press **f8** when you are done to return to the Main Menu.

How the demo works

The program must perform three distinct DBFINDs against three separate SI-paths to accomplish the retrieval.

First, SUPERDEX must locate the specified customer name in the CUSTOMERS master dataset and retain the corresponding CUSTOMER-NUMBER. This is done via the simple customer SI-path using a SUPERDEX DBFIND *mode* 1 against CUSTOMERS with the specified customer name, surrounded by a ~ and ;, as the *argument.*

Next, the retained customer number must be looked up in the ORDER-HEADERS master dataset to locate the corresponding order number(s). This is done via a special operation called a *projection,* which is accomplished simply by calling DBFIND *mode* 1 against ORDER-HEADERS and specifying an *argument* of ~ and ;.

The final DBFIND performs a boolean AND between the entries located in the ORDER-HEADERS dataset and the order line items in the ORDER-LINES dataset by using the common item ORDER-NUMBER in the *item* parameter and the part number, surrounded by ~ and ;, as the *argument.*

These same techniques may be used to perform relational retrievals against multiple databases by simply altering the value of the *base* parameter.

A complete copy of the source program appears in *Appendix B* and in the file SDEMOPRJ.DEMO.SUPERDEX.

Section 3 SUPERDEX index structures

SUPERDEX paths

Now that we've seen the quick and powerful retrievals that can be accomplished by SUPERDEX, let's take a look at the index structures that were used to facilitate them.

To do so, exit to MPE and type

SIMAINTLIST

and press **RETURN**. When prompted, enter the database name

OEDB

and **RETURN** to list the SUPERDEX structures:

```
RUN SIMAINT.PUB.SUPERDEX,LIST

SIMAINT.PRIV VERSION 3.1 (23JAN92) COPYRIGHT DR. MATT / IABG (1988,1991)

DATABASE >OEDB
THE FOLLOWING SI-PATHS AND ITEMS ARE DEFINED:
 DATASET     SI-PATH              ITEMS/LENGTHS

  10001      KWEXCLUDE                          4
CUSTOMERS
  10002      CUSTOMER-NAME        CUSTOMER-NAME    15
  10003      CUSTOMER-NAME-KW/K   CUSTOMER-NAME     8
  10004      ADDRESS1-CITY-KW/K   ADDRESS-1         4
  10004      ADDRESS1-CITY-KW/K   CITY              4
ORDER-LINES
  10005      ORDER-PART           ORDER-NUMBER     2       PART-NUMBER  7
  10006      PART-ORDER           PART-NUMBER      7       ORDER-NUMBER 2
ORDER-HEADERS
  10007      CUSTOMER-NUMBER      CUSTOMER-NUMBER  2

TOTAL TIME :                                     CPU 0:00:02.2  Elapsed 0:00:04
END OF PROGRAM
DEMO.SDX31:27> PSCREEN
```

Listed here are seven SI-paths which relate to eight SI-keys in the database. They are as follows:

KWEXCLUDE Special stand-alone SI-path used for excluding unneeded words from keywording, such as for excluding "CORP" and "INC" in the Keyworded Key Demos.

CUSTOMER-NAME Simple SI-path used for generic, partial-key, range, and other retrievals by CUSTOMER-NAME in the CUSTOMERS dataset. Used in the Simple Key Demos.

CUSTOMER-NAME-KW Same as CUSTOMER-NAME, but configured as keyworded (as noted by the **/K** following the SI-path name) with a keyword length of 8 words (16 characters). Used in the Keyworded Key Demos.

ADDRESS1-CITY-KW Grouped SI-path consisting of the ADDRESS-1 and CITY fields, shown as two separate entries above. Note the **/K** indicating that it is also configured as keyworded. Used in the Grouped Key Demos.

ORDER-PART Concatenated SI-path consisting of the ORDER-NUMBER and PART-NUMBER for each line item in the ORDER-LINES dataset. Used in the Concatenated Key Demos.

PART-ORDER Same as ORDER-PART, but order of items is reversed. Used in the dataset Relational Access demo.

CUSTOMER-NUMBER Simple SI-path related to the ORDER-HEADERS dataset, consisting of the CUSTOMER-NUMBER. Used in the Relational Access Demo using multiple datasets.

Configuring SI-paths

The SI-paths that have been used up to this point were created for you by using SUPERDEX's configuration program, **SIMAINT**. This program establishes the required index structures and creates the indices for the data entries which currently exist in the database; the indices are stored in the stand-alone detail dataset named **SI**.

The following section on creating new SI-paths demonstrates how the SIMAINT program works.

Creating a new SI-path

The CUSTOMERS dataset contains three fields for phone numbers:

PHONE-AREA-CODE	phone number area code (first three digits)
PHONE-PREFIX	phone number prefix (middle three digits)
PHONE-SUFFIX	phone number suffix (last four digits)

Creating a grouped SI-path which links PHONE-PREFIX and PHONE-SUFFIX together will permit a customer to be located by either value using a one prompt in a single operation (just like Address and City did in the Grouped Key Demo). It will also permit all the customers with a specified prefix to be identified.

Run the SIMAINT program by typing:

SIMAINT

and press **RETURN**. Then, specify the database name

OEDB

and press **RETURN**. SIMAINT lists the datasets that have related SI-paths and prompts for a dataset:

```
RUN SIMAINT.PUB.SUPERDEX

SIMAINT.PRIV VERSION 3.1 (23JAN92) COPYRIGHT DR. MATT / IABG (1988,1991)

DATABASE >OEDB
SI-PATHS EXIST FOR THE FOLLOWING SETS:

CUSTOMERS
ORDER-LINES
ORDER-HEADERS
ENTER NAME OF SET TO BE MODIFIED OR NEW NAME
DATASET >
```

At the dataset prompt, enter

CUSTOMERS

and press **RETURN**. Its related SI-paths are displayed and you are prompted for the name of an SI-path:

```
DATASET >CUSTOMERS
THE FOLLOWING SI-PATHS AND ITEMS ARE DEFINED:
CUSTOMER-NAME      CUSTOMER-NAME L =15
CUSTOMER-NAME-KW/K CUSTOMER-NAME L = 4
ADDRESS1-CITY-KW/K ADDRESS-1     L = 4
ADDRESS1-CITY-KW/K CITY          L = 4
ENTER SI-PATH WITH OPTION /D /R /G OR NEW NAME
SI-PATH >
```

Specify the new SI-path name

PHONE-PRFX-SUFX

and RETURN. Enter

?

and RETURN when prompted for an item name:

```
SI-PATH >PHONE-PRFX-SUFX
ITEM 1 >?
CUSTOMER-NUMBER CUSTOMER-ABBR CUSTOMER-NAME ADDRESS-1      ADDRESS-2
CITY            STATE         ZIP-CODE      PHONE-AREA-CODE   PHONE-PREFIX
PHONE-SUFFIX

ITEM 1 >
```

This causes SIMAINT to list the items in the dataset and re-prompt. Now, specify the first item

PHONE-PREFIX

to be included in the group and RETURN twice:

```
ITEM 1 >PHONE-PREFIX
ITEM 2 >RETURN
```

When prompted for the next SI-path, enter the same SI-path name as before but append **/G**:

PHONE-PRFX-SUFX/G

This indicates that you are configuring the SI-path as grouped:

```
SI-PATH >PHONE-PRFX-SUFX/G
ITEM 1 >
```

Now, specify the second item to be included in the group

PHONE-SUFFIX

as shown:

```
ITEM 1 >PHONE-SUFFIX
SI-PATH >
```

Press **RETURN** for the next two prompts and wait a few moments while the new SI-path is created:

```
SI-PATH >RETURN
DATASET >RETURN
PROCESSING SI-PATH PHONE-PRFX-SUFX  OF CUSTOMERS     # OF ENT: 1003
         INPUT:     1003 RECORDS 100%     CPU 0:00:03.2 ELAPSED 0:00:03
         SORT:      2006 INDICES          CPU 0:00:00.9 ELAPSED 0:00:01
         OUTPUT:    1700 INDICES 100%     CPU 0:00:01.9 ELAPSED 0:00:02
TOTAL TIME:                               CPU 0:00:09.8 ELAPSED 0:02:06

END OF PROGRAM
```

Appendix A Demo database structure

The following pages illustrate the dataset layouts for the **OEDB** demo database. Only the dataset **SI** and item **SI** were added to facilitate SUPERDEX access.

DATA SET: CUSTOMERS

Items:	Fld No.	Itm No.	Srt Loc	End Loc	Itm Typ	Size/ Lngth	Array Size	Srch Item	Sort Item
CUSTOMER-NUMBER	1	1	1	2	I	2	1	X	
CUSTOMER-ABBR	2	11	3	4	X	4	1		
CUSTOMER-NAME	3	2	5	19	X	30	1		
ADDRESS-1	4	3	20	32	X	26	1		
ADDRESS-2	5	4	33	45	X	26	1		
CITY	6	5	46	53	X	16	1		
STATE	7	6	54	54	X	2	1		
ZIP-CODE	8	7	55	56	I	2	1		
PHONE-AREA-CODE	9	8	57	57	I	1	1		
PHONE-PREFIX	10	9	58	58	I	1	1		
PHONE-SUFFIX	11	10	59	59	I	1	1		

DATA SET: ORDER-HEADERS

Items:	Fld No.	Itm No.	Srt Loc	End Loc	Itm Typ	Size/ Lngth	Array Size	Srch Item	Sort Item
ORDER-NUMBER	1	12	1	2	I	2	1	X	
ORDER-TYPE	2	13	3	3	X	2	1		
ENTRY-DATE	3	14	4	4	I	1	1		
PO-NUMBER	4	15	5	11	X	14	1		
CUSTOMER-NUMBER	5	1	12	13	I	2	1		
SHIP-TO-NUMBER	6	42	14	14	K	1	1		
BRANCH-LOCATION	7	16	15	15	I	1	1		
NEXT-LINE-NUMBER	8	17	16	16	I	1	1		
PAYMENT-TERMS	9	18	17	17	I	1	1		
ATTENTION-CODE	10	19	18	18	I	1	1		
TAX-PAYABLE	11	20	19	19	I	1	1		
SALES-TAX-PCT	12	21	20	21	I	2	1		
BILLED-VALUE	13	22	22	23	I	2	1		
ENTRY-VALUE	14	23	24	25	I	2	1		
SHIPMENT-DATE	15	24	26	26	I	1	1		
ORDER-WEIGHT	16	25	27	27	I	1	1		
FREIGHT-CHARGE	17	26	28	29	I	2	1		
CARRIER-USED	18	27	30	30	I	1	1		
CARTON-QUANTITY	19	28	31	31	I	1	1		
PRICE-CODE	20	29	32	32	X	2	1		
CONFIRM-DATE	21	30	33	33	I	1	1		
LAST-INVOICE-DTE	22	31	34	35	I	2	1		
BACK-ORDER-CODE	23	32	36	36	X	2	1		
PICKING-CODE	24	33	37	37	I	1	1		
BILLING-CODE	25	34	38	38	I	1	1		
CONSOLIDATE-CODE	26	35	39	39	I	1	1		
SALES-REP-CODE	27	36	40	41	X	2	2		
BACKORDER-STATUS	28	37	42	42	I	1	1		
HOLD-CODE	29	38	43	43	I	1	1		
FREIGHT-TRUCK	30	39	44	45	I	2	1		
VALUE-CODE	31	40	46	46	X	2	1		
ORDER-STATUS	32	41	47	47	I	1	1		

DATA SET: ORDER-LINES

Items:	Fld No.	Itm No.	Srt Loc	End Loc	Itm Typ	Size/ Lngth	Array Size	Srch Item	Sort Item
ORDER-NUMBER	1	12	1	2	I	2	1	X	
INVOICE-LINE-NO	2	43	3	3	K	1	1		
PART-TYPE-CODE	3	44	4	4	X	2	1		
PART-NUMBER	4	45	5	11	X	14	1		
PART-DESCRIPTION	5	46	12	24	X	26	1		
PART-ENTRY-DATE	6	47	25	25	I	1	1		
QUANTITY-ORDERED	7	48	26	26	I	1	1		
UNIT-OF-MEASURE	8	49	27	27	X	2	1		
QTY-PER-PACKAGE	9	50	28	28	I	1	1		
LINE-ITEM-PRICE	10	51	29	30	I	2	1		
UNIT-PRICE	11	52	31	32	I	2	1		
UNIT-COST	12	53	33	34	I	2	1		
PRICE-DISCOUNT	13	54	35	35	I	1	1		
QUANTITY-SHIPPED	14	55	36	36	I	1	1		
BACK-ORDER-NEED	15	56	37	37	X	2	1		
SHIP-DATE	16	57	38	38	I	1	1		
PICKING-LIST	17	58	39	39	I	1	1		
BILL-CODE	18	59	40	40	I	1	1		
PREV-QTY-SHIPPED	19	60	41	41	I	1	1		
PART-HOLD-CODE	20	61	42	42	I	1	1		
INVOICE-REF-NO	21	62	43	44	I	2	1		
PRICE-DIFFERENTL	22	63	45	45	I	1	1		
STOCK-LOCATION	23	70	46	49	X	8	1		
BKORD-INDICATOR	24	64	50	50	I	1	1		
PART-PRICE-CODE	25	65	51	51	X	2	1		
COMMERCIAL-STAT	26	66	52	55	I	2	2		
INVOICE-CODE	27	67	56	56	I	1	1		
PACKAGE-WEIGHT	28	68	57	57	X	2	1		
LINE-ITEM-STATUS	29	69	58	58	I	1	1		

DATA SET: SI

Items:	Fld No.	Itm No.	Srt Loc	End Loc	Itm Typ	Size/ Lngth	Array Size	Srch Item	Sort Item
SI	1	71	1	508	X	254	4		

Appendix B COBOL source programs

The sources for the COBOL demonstration programs appear on the following pages with comments. These programs were written in COBOL85 and use VPLUS.

Simple Key Demo

```
$CONTROL SUBPROGRAM
 IDENTIFICATION DIVISION.
 PROGRAM-ID. KEY-DEMO.
 AUTHOR. BRADMARK TECHNOLOGIES

 ENVIRONMENT DIVISION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.

 01 SCREEN-BUFFER.
     02 SCREEN-KEY-VALUE              PIC X(31).
     02 SCREEN-DIRECTION              PIC X.
     02 DATA-LINES.
        05 SCREEN-LINE-ARRAY OCCURS 18 TIMES.
           10 SCREEN-LINE             PIC X(78).

 01 BUFFER-LENGTH                     PIC S9(4) COMP.

 01 ARRAY-INDEX                       PIC S9(4) COMP.

 01 IMAGE-BUFFER.
     02 IMAGE-CUSTOMER-NUMBER          PIC S9(9) COMP.
     02 IMAGE-CUSTOMER-NAME            PIC X(30).

 01 TEMP-LINE.
     02 TEMP-CUSTOMER-NAME             PIC X(30).
     02 FILLER                        PIC X VALUE SPACES.
     02 TEMP-CUSTOMER-NUMBER           PIC 9(10) USAGE DISPLAY.

 01 DONE                              PIC X.
 01 END-OF-SCREEN                     PIC X.
 01 NO-ENTRIES                        PIC X.

 01 FORM-KEYS                         PIC S9(4) COMP VALUE 1.
 01 NUMBER-OF-KEYS                    PIC S9(4) COMP VALUE 8.
 01 KEY-BUFFER                        PIC X(128).

 01 MESSAGE-BUFFER                    PIC X(72).
 01 MESSAGE-BUFFER-LENGTH             PIC S9(4) COMP.
 01 ACTUAL-LENGTH                     PIC S9(4) COMP.
```

```
01 QUALIFY-BUFFER.
    02 ENTRIES-FOUND                  PIC ZZ,ZZ9.
    02 FILLER                         PIC X(66) VALUE
       " Entries Qualified.  (More Entries Below)".

01 GET-MODE                           PIC S9(4) COMP.

LINKAGE SECTION.
 01 IMAGE.
     02 IMAGE-STATUS.
        05 CW                      PIC S9(4) COMP.
        05 IMAGE-ENTRY-LENGTH      PIC S9(4) COMP.
        05 IMAGE-RECORD-NUMBER     PIC S9(9) COMP.
        05 IMAGE-CHAIN-LENGTH      PIC S9(9) COMP.
        05 IMAGE-LAST-ON-CHAIN     PIC S9(9) COMP.
        05 IMAGE-FIRST-ON-CHAIN    PIC S9(9) COMP.

     02 ITEM.
        05 ITEM-VALUE              PIC X(16).

     02 IMAGE-SET.
        05 SET-VALUE               PIC X(16).

     02 PASSWORD.
        05 PASSWORD-VALUE          PIC X(16).

     02 BASE.
        05 BASE-ID                 PIC XX.
        05 BASE-VALUE              PIC X(32).

     02 LIST.
        05 LIST-VALUE              PIC X(200).

     02 MODES.
        05 MODE1                   PIC S9(4) COMP.
        05 MODE2                   PIC S9(4) COMP.
        05 MODE3                   PIC S9(4) COMP.
        05 MODE4                   PIC S9(4) COMP.
        05 MODE5                   PIC S9(4) COMP.
        05 MODE6                   PIC S9(4) COMP.
        05 MODE7                   PIC S9(4) COMP.
        05 MODE8                   PIC S9(4) COMP.

     02 DUMMY                      PIC S9(4) COMP.
```

```
 01 COMAREA.
     02 VSTATUS                  PIC S9(4) COMP.
     02 VLANGUAGE                PIC XX.
     02 COMAREA-LENGTH           PIC S9(4) COMP.
     02 FILLER                   PIC X(4).
     02 LAST-KEY                 PIC S9(4) COMP.
     02 NUMERRORS                PIC S9(4) COMP.
     02 WINDOWENH                PIC XX.
     02 FILLER                   PIC XX.
     02 LABELOPTION              PIC S9(4) COMP.
     02 FORM-NAME                PIC X(16).
     02 NEXT-FORM-NAME           PIC X(16).
     02 REPEATAPP                PIC S9(4) COMP.
     02 FREEZAPP                 PIC S9(4) COMP.
     02 FILLER                   PIC XX.
     02 VBUFFER-LENGTH           PIC S9(4) COMP.
     02 FILLER                   PIC X(64).

PROCEDURE DIVISION USING IMAGE,COMAREA.
BEGIN.
    MOVE "n" TO DONE.
    MOVE SPACES TO SCREEN-BUFFER.
    MOVE SPACES TO MESSAGE-BUFFER.
    MOVE 72 TO MESSAGE-BUFFER-LENGTH.

    MOVE "CUSTOMERS;" TO SET-VALUE.
    MOVE "SIMPLEKEY" TO NEXT-FORM-NAME.
    CALL "VGETNEXTFORM" USING COMAREA.
    CALL "VGETKEYLABELS" USING COMAREA,FORM-KEYS,NUMBER-OF-KEYS,
          KEY-BUFFER.
    CALL INTRINSIC ".LEN." USING SCREEN-BUFFER,GIVING,
          BUFFER-LENGTH.

    PERFORM UNTIL DONE IS EQUAL TO "y"
       MOVE "n" TO END-OF-SCREEN

       CALL "VPUTBUFFER" USING COMAREA,SCREEN-BUFFER
          BUFFER-LENGTH
       CALL "VSHOWFORM" USING COMAREA

       MOVE SPACES TO MESSAGE-BUFFER
       CALL "VPUTWINDOW" USING COMAREA,MESSAGE-BUFFER,
             MESSAGE-BUFFER-LENGTH

       CALL "VREADFIELDS" USING COMAREA
```

```
           IF LAST-KEY IS ZERO THEN
              CALL "VFIELDEDITS" USING COMAREA
              PERFORM UNTIL NUMERRORS IS ZERO
                 CALL "VERRMSG" USING COMAREA,MESSAGE-BUFFER,
                      MESSAGE-BUFFER-LENGTH,ACTUAL-LENGTH
                 CALL "VPUTWINDOW" USING COMAREA,MESSAGE-BUFFER,
                       MESSAGE-BUFFER-LENGTH
                 MOVE " G" TO WINDOWENH
                 CALL "VSHOWFORM" USING COMAREA
                 CALL "VREADFIELDS" USING COMAREA

                 MOVE " H" TO WINDOWENH
                 MOVE SPACES TO MESSAGE-BUFFER
                 CALL "VPUTWINDOW" USING COMAREA,MESSAGE-BUFFER,
                       MESSAGE-BUFFER-LENGTH
                 CALL "VFIELDEDITS" USING COMAREA

                 IF LAST-KEY IS EQUAL TO 8 THEN
                    MOVE ZERO TO NUMERRORS
                    MOVE "y" TO DONE
                 END-IF
              END-PERFORM

              IF LAST-KEY IS ZERO THEN
                 CALL "VGETBUFFER" USING COMAREA,
                       SCREEN-BUFFER,BUFFER-LENGTH

                 MOVE "n" TO NO-ENTRIES
      ******************************************************************
      * THE FOLLOWING MOVE STATEMENTS ARE USED TO INITIALIZE THE "ITEM" AND *
      *                                                                *
      * "LIST" VARIABLES FOR THE COORESPONDING SUPERDEX DBFIND AND DBGET.   *
      *                                                                *
      * THE ITEM VALUE REPRESENTS THE SI-PATH AS DEFINED DURING THE         *
      *                                                                *
      * CREATION OF THE INDEX. THE LIST VALUE REPRESENTS THE IMAGE ITEMS TO *
      *                                                                *
      * BE RETRIEVED BY THE THE RESULTING DBGET'S                          *
      *                                                                *
      ******************************************************************
                 MOVE "CUSTOMER-NAME;" TO ITEM-VALUE
                 MOVE "CUSTOMER-NUMBER,CUSTOMER-NAME;" TO LIST-VALUE
```

```
*************************************************************************
* THE FOLLOWING DBFIND IS USED BY SUPERDEX TO SCAN THE INDEX AS DEFINED  *
* BY THE ITEM VALUE PREVIOUSLY LOADED INTO THE ITEM PARAMETER. THE DBFIND *
* DETERMINES ALL CORRESPONDING ENTRIES WHICH QUALIFY TO THE REQUESTED    *
* SCREEN ENTRY VALUE AND HOLD THEM FOR THE FOLLOWING DBGETS              *
*************************************************************************
            CALL "DBFIND" USING BASE,IMAGE-SET,MODE1,
                  IMAGE-STATUS,ITEM,SCREEN-KEY-VALUE

            IF CW IS NOT ZERO THEN
               MOVE SPACES TO DATA-LINES
               MOVE "No Qualifying Entries Found" TO
                  MESSAGE-BUFFER
               CALL "VPUTWINDOW" USING COMAREA,MESSAGE-BUFFER,
                     MESSAGE-BUFFER-LENGTH
               MOVE "y" TO NO-ENTRIES
            ELSE
               MOVE IMAGE-CHAIN-LENGTH TO ENTRIES-FOUND
            END-IF
         END-IF
      ELSE
         IF LAST-KEY IS EQUAL TO 8 THEN
            MOVE "y" TO DONE
         END-IF
      END-IF

     IF (LAST-KEY IS EQUAL TO ZERO OR LAST-KEY IS EQUAL TO 1)
            AND NO-ENTRIES IS EQUAL TO "n"
         MOVE SPACES TO DATA-LINES
         MOVE 1 TO ARRAY-INDEX
*************************************************************************
* READ THE CHAIN FORWARD OR BACKWARD, DEPENDING ON USER REQUEST          *
*************************************************************************
         IF SCREEN-DIRECTION IS EQUAL TO "B" THEN
            MOVE 6 TO GET-MODE
         ELSE
            MOVE 5 TO GET-MODE
         END-IF

         PERFORM UNTIL END-OF-SCREEN IS EQUAL TO "y"
```

```
*************************************************************************
* THE FOLLOWING DBGET IS USED TO RETRIEVE INFORMATION FROM THE IMAGE    *
* DATASET WHICH CORRESPOND TO THE QUALIFYING ENTRIES RETRIEVED FROM     *
* THE PREVIOUS DBFIND.                                                  *
*************************************************************************
              CALL "DBGET" USING BASE,IMAGE-SET,GET-MODE,
                 IMAGE-STATUS,LIST,IMAGE-BUFFER,DUMMY

              IF CW IS NOT EQUAL TO ZERO THEN
                 MOVE "y" TO END-OF-SCREEN
                 MOVE "End of Current Entries" TO MESSAGE-BUFFER
                 CALL "VPUTWINDOW" USING COMAREA,MESSAGE-BUFFER,
                       MESSAGE-BUFFER-LENGTH
              ELSE
                 MOVE IMAGE-CUSTOMER-NUMBER TO TEMP-CUSTOMER-NUMBER
                 MOVE IMAGE-CUSTOMER-NAME TO TEMP-CUSTOMER-NAME
                 MOVE TEMP-LINE TO SCREEN-LINE(ARRAY-INDEX)
                 ADD 1 TO ARRAY-INDEX

                 IF ARRAY-INDEX IS GREATER THAN 16 THEN
                    MOVE "y" TO END-OF-SCREEN
                    MOVE QUALIFY-BUFFER TO MESSAGE-BUFFER
                    CALL "VPUTWINDOW" USING COMAREA,
                       MESSAGE-BUFFER,MESSAGE-BUFFER-LENGTH
                 END-IF
              END-IF
           END-PERFORM
        END-IF
     END-PERFORM.

     MOVE ZERO TO LAST-KEY

     EXIT PROGRAM.
```

Concatenated Key Demo

```
$CONTROL SUBPROGRAM
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CONCATENATE-DEMO.
 AUTHOR. BRADMARK TECHNOLOGIES.

 ENVIRONMENT DIVISION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.

 01 SCREEN-BUFFER.
     02 SCREEN-KEY-VALUE.
        05 SCREEN-ORDER-NUMBER        PIC 9(10) USAGE IS DISPLAY.
        05 SCREEN-PART-KEY            PIC X(14).
        05 PART-ARRAY REDEFINES SCREEN-PART-KEY.
           10 CHARACTER-ARRAY OCCURS 14 TIMES.
              15 FILLER               PIC X.
     02 DATA-LINES.
        05 SCREEN-LINE-ARRAY OCCURS 18 TIMES.
           10 SCREEN-LINE             PIC X(78).

 01 IMAGE-BUFFER.
     02 OMNUMB                        PIC S9(9) COMP.
     02 ITMPRT                        PIC X(14).
     02 ITMDES                        PIC X(26).

 01 DISPLAY-LINE.
     02 DISPLAY-OMNUMB                PIC 9(10) USAGE IS DISPLAY.
     02 FILLER                        PIC XX.
     02 DISPLAY-ITMPRT                PIC X(14).
     02 FILLER                        PIC XX.
     02 DISPLAY-ITMDES                PIC X(26).


01 KEY-VALUE.
     02 ORDER-NUMBER                  PIC S9(9) COMP.
     02 PART-KEY                      PIC X(14).

 01 FIND-MODE                         PIC S9(4) COMP.

 01 BUFFER-LENGTH                     PIC S9(4) COMP.
```

```
01 ARRAY-INDEX                          PIC S9(4) COMP.
01 DONE                                 PIC X.
01 END-OF-SCREEN                        PIC X.
01 NO-ENTRIES                           PIC X.

01 FORM-KEYS                            PIC S9(4) COMP VALUE 1.
01 NUMBER-OF-KEYS                       PIC S9(4) COMP VALUE 8.
01 KEY-BUFFER                           PIC X(128).

01 MESSAGE-BUFFER                       PIC X(72).
01 MESSAGE-BUFFER-LENGTH                PIC S9(4) COMP.
01 ACTUAL-LENGTH                        PIC S9(4) COMP.

01 I                                    PIC S9(4) COMP.

LINKAGE SECTION.
 01 IMAGE.
     02 IMAGE-STATUS.
        05 CW                      PIC S9(4) COMP.
        05 IMAGE-ENTRY-LENGTH      PIC S9(4) COMP.
        05 IMAGE-RECORD-NUMBER     PIC S9(9) COMP.
        05 IMAGE-CHAIN-LENGTH      PIC S9(9) COMP.
        05 IMAGE-LAST-ON-CHAIN     PIC S9(9) COMP.
        05 IMAGE-FIRST-ON-CHAIN    PIC S9(9) COMP.

     02 ITEM.
        05 ITEM-VALUE              PIC X(16).

     02 IMAGE-SET.
        05 SET-VALUE               PIC X(16).

     02 PASSWORD.
        05 PASSWORD-VALUE          PIC X(16).

     02 BASE.
        05 BASE-ID                 PIC XX.
        05 BASE-VALUE              PIC X(32).

     02 LIST.
        05 LIST-VALUE              PIC X(200).
     02 MODES.
        05 MODE1                   PIC S9(4) COMP.
        05 MODE2                   PIC S9(4) COMP.
        05 MODE3                   PIC S9(4) COMP.
        05 MODE4                   PIC S9(4) COMP.
        05 MODE5                   PIC S9(4) COMP.
        05 MODE6                   PIC S9(4) COMP.
        05 MODE7                   PIC S9(4) COMP.
        05 MODE8                   PIC S9(4) COMP.
     02 DUMMY                      PIC S9(4) COMP.
```

```
 01 COMAREA.
     02 VSTATUS                  PIC S9(4) COMP.
     02 VLANGUAGE                PIC XX.
     02 COMAREA-LENGTH           PIC S9(4) COMP.
     02 FILLER                   PIC X(4).
     02 LAST-KEY                 PIC S9(4) COMP.
     02 NUMERRORS                PIC S9(4) COMP.
     02 WINDOWENH                PIC XX.
     02 FILLER                   PIC XX.
     02 LABELOPTION              PIC S9(4) COMP.
     02 FORM-NAME                PIC X(16).
     02 NEXT-FORM-NAME           PIC X(16).
     02 REPEATAPP                PIC S9(4) COMP.
     02 FREEZAPP                 PIC S9(4) COMP.
     02 FILLER                   PIC XX.
     02 VBUFFER-LENGTH           PIC S9(4) COMP.
     02 FILLER                   PIC X(64).

PROCEDURE DIVISION USING IMAGE COMAREA.
BEGIN.
    MOVE 'n' TO DONE.
    MOVE SPACES TO SCREEN-BUFFER.
    MOVE SPACES TO MESSAGE-BUFFER.
    MOVE 72 TO MESSAGE-BUFFER-LENGTH.

    MOVE 'CONCATENATE' TO NEXT-FORM-NAME.
    CALL 'VGETNEXTFORM' USING COMAREA.
    CALL 'VGETKEYLABELS' USING COMAREA FORM-KEYS NUMBER-OF-KEYS
          KEY-BUFFER.
    CALL INTRINSIC '.LEN.' USING SCREEN-BUFFER GIVING
          BUFFER-LENGTH.

    PERFORM UNTIL DONE IS EQUAL TO 'y'
       MOVE 'n' TO END-OF-SCREEN

       CALL 'VPUTBUFFER' USING COMAREA SCREEN-BUFFER
          BUFFER-LENGTH
       CALL 'VSHOWFORM' USING COMAREA
       MOVE SPACES TO MESSAGE-BUFFER
       CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
             MESSAGE-BUFFER-LENGTH

       CALL 'VREADFIELDS' USING COMAREA

       IF LAST-KEY IS ZERO THEN
          CALL 'VFIELDEDITS' USING COMAREA
```

```
            PERFORM UNTIL NUMERRORS IS ZERO
               CALL 'VERRMSG' USING COMAREA MESSAGE-BUFFER
                    MESSAGE-BUFFER-LENGTH ACTUAL-LENGTH
               CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                     MESSAGE-BUFFER-LENGTH
               MOVE ' G' TO WINDOWENH
                     CALL 'VSHOWFORM' USING COMAREA
               CALL 'VREADFIELDS' USING COMAREA

               MOVE ' H' TO WINDOWENH
               MOVE SPACES TO MESSAGE-BUFFER
               CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                     MESSAGE-BUFFER-LENGTH
               CALL 'VFIELDEDITS' USING COMAREA

               IF LAST-KEY IS EQUAL TO 8 THEN
                  MOVE ZERO TO NUMERRORS
               END-IF
            END-PERFORM

            IF LAST-KEY IS NOT EQUAL TO 8 THEN
               CALL 'VFINISHFORM' USING COMAREA
               CALL 'VGETBUFFER' USING COMAREA SCREEN-BUFFER
                   BUFFER-LENGTH

               PERFORM FIND-LENGTH-OF-DESCRIPTION

      ******************************************************************
      * THE NEXT TWO MOVES CONCATENATES THE USER ENTERED VALUES TO BUILD THE *
      * ARGUMENT NEEDED IN THE DBFIND.                                 *
      ******************************************************************
               MOVE SCREEN-ORDER-NUMBER TO ORDER-NUMBER
               MOVE SCREEN-PART-KEY TO PART-KEY
      ******************************************************************
      * THE FIND MODE DEFAILTS TO A VALUE OF -104. THE DEFAULT VALUE   *
      * TELLS SUPERDEX THAT THERE ARE 4 BYTES IN THE KEY. FOR EACH     *
      * CHARACTER THAT THE USER ENTERS IN THE SECOND SCREEN FIELD THE  *
      * VALUE IN THE PARENTHESIS IS INCREMENTED                        *
      ******************************************************************
               COMPUTE FIND-MODE = 0 - (100 + 4 + I)

               MOVE 'n' TO NO-ENTRIES

               MOVE 'ORDER-LINES;' TO SET-VALUE
```

```
*************************************************************************
* THE FOLLOWING MOVE STATEMENTS ARE USED TO INITIALIZE THE "ITEM" AND   *
* "LIST" VARIABLES FOR THE CORRESPONDING SUPERDEX DBFIND AND DBGET.     *
* THE ITEM VALUE REPRESENTS THE SI-PATH AS DEFINED DURING THE           *
* CREATION OF THE INDEX. THE LIST VALUE REPRESENTS THE IMAGE ITEMS TO   *
* BE RETRIEVED BY THE THE RESULTING DBGET'S                             *
*************************************************************************

       MOVE 'ORDER-PART;' TO ITEM-VALUE
            MOVE 'ORDER-NUMBER,PART-NUMBER,PART-DESCRIPTION;'
                 TO LIST-VALUE
*************************************************************************
* THE FOLLOWING DBFIND IS USED BY SUPERDEX TO SCAN THE INDEX AS DEFINED *
* BY THE ITEM VALUE PREVIOUSLY LOADED INTO THE ITEM PARAMETER. THE DBFIND*
* DETERMINES ALL CORRESPONDING ENTRIES WHICH QUALIFY TO THE REQUESTED   *
* SCREEN ENTRY VALUE AND HOLD THEM FOR THE FOLLOWING DBGETS             *
*************************************************************************
            CALL 'DBFIND' USING BASE IMAGE-SET FIND-MODE
                IMAGE-STATUS ITEM KEY-VALUE
            IF CW IS NOT EQUAL TO 0 THEN
               MOVE 'No Qualifying Entries Found' TO
                   MESSAGE-BUFFER
               MOVE 'y' TO NO-ENTRIES
               MOVE SPACES TO DATA-LINES
               CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                     MESSAGE-BUFFER-LENGTH
            END-IF
         END-IF

        END-IF
        IF LAST-KEY IS EQUAL TO 8 THEN
           MOVE 'y' TO DONE
        END-IF

       IF (LAST-KEY IS EQUAL TO ZERO OR LAST-KEY IS EQUAL TO 1)
             AND NO-ENTRIES IS EQUAL TO 'n' THEN
          MOVE SPACES TO DATA-LINES
          MOVE 1 TO ARRAY-INDEX

          PERFORM UNTIL END-OF-SCREEN IS EQUAL TO 'y'
*************************************************************************
* THE FOLLOWING DBGET IS USED TO RETRIEVE INFORMATION FROM THE IMAGE    *
* DATASET WHICH CORRESPOND TO THE QUALIFYING ENTRIES RETRIEVED FROM     *
* THE PREVIOUS DBFIND.                                                  *
*************************************************************************
```

```
        CALL 'DBGET' USING BASE IMAGE-SET MODE5 IMAGE-STATUS
                    LIST IMAGE-BUFFER DUMMY
                 IF CW IS NOT EQUAL TO ZERO THEN
                    MOVE 'y' TO END-OF-SCREEN
                    MOVE 'End of Current Entries' TO MESSAGE-BUFFER
                    CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                         MESSAGE-BUFFER-LENGTH
                 ELSE
                    MOVE SPACES TO DISPLAY-LINE
                    MOVE OMNUMB TO DISPLAY-OMNUMB
                    MOVE ITMPRT TO DISPLAY-ITMPRT
                    MOVE ITMDES TO DISPLAY-ITMDES
                    MOVE DISPLAY-LINE TO SCREEN-LINE(ARRAY-INDEX)
                    ADD 1 TO ARRAY-INDEX
                    IF ARRAY-INDEX IS GREATER THAN 17 THEN
                       MOVE 'y' TO END-OF-SCREEN
                       MOVE 'More Entries Below' TO MESSAGE-BUFFER
                       CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                            MESSAGE-BUFFER-LENGTH
                    END-IF
                 END-IF
              END-PERFORM
           END-IF
        END-PERFORM.

        MOVE ZERO TO LAST-KEY

        EXIT PROGRAM.
*****************************************************************
*                                                               *
*    THIS ROUTINE WILL RETURN THE NUMBER OF CHARACTERS ENTERED  *
*    BY THE USER IN THE SECOND SCREEN FIELD.                    *
*                                                               *
*****************************************************************

 FIND-LENGTH-OF-DESCRIPTION.
        MOVE 14 TO I.
        PERFORM UNTIL (I IS EQUAL TO ZERO) OR
           (CHARACTER-ARRAY(I) IS NOT EQUAL TO SPACE)
              SUBTRACT 1 FROM I
        END-PERFORM.
 FIND-LENGTH-EXIT.
        EXIT.
```

Keyworded Key Demo

```
$CONTROL SUBPROGRAM
 IDENTIFICATION DIVISION.
 PROGRAM-ID. KEYWORD-DEMO.
 AUTHOR. BRADMARK TECHNOLOGIES.

 ENVIRONMENT DIVISION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.

 01 SCREEN-BUFFER.
     02 SCREEN-KEY-VALUE              PIC X(50).
     02 DATA-LINES.
        05 SCREEN-LINE-ARRAY OCCURS 18 TIMES.
           10 SCREEN-LINE             PIC X(78).

 01 BUFFER-LENGTH                     PIC S9(4) COMP.

 01 ARRAY-INDEX                       PIC S9(4) COMP.

 01 IMAGE-BUFFER.
     02 IMAGE-CUSTOMER-NUMBER          PIC S9(9) COMP.
     02 IMAGE-CUSTOMER-NAME            PIC X(30).

 01 TEMP-LINE.
     02 TEMP-CUSTOMER-NAME             PIC X(30).
     02 FILLER                        PIC X VALUE SPACES.
     02 TEMP-CUSTOMER-NUMBER           PIC 9(10) USAGE DISPLAY.

 01 DONE                              PIC X.
 01 END-OF-SCREEN                     PIC X.
 01 NO-ENTRIES                        PIC X.

 01 FORM-KEYS                         PIC S9(4) COMP VALUE 1.
 01 NUMBER-OF-KEYS                    PIC S9(4) COMP VALUE 8.
 01 KEY-BUFFER                        PIC X(128).

 01 MESSAGE-BUFFER                    PIC X(72).
 01 MESSAGE-BUFFER-LENGTH             PIC S9(4) COMP.
 01 QUALIFY-BUFFER.
     02 ENTRIES-FOUND                 PIC ZZ,ZZ9.
     02 FILLER                        PIC X(66) VALUE
       " Entries Found. (More Entries Below)".
```

```
LINKAGE SECTION.
 01 IMAGE.
     02 IMAGE-STATUS.
        05 CW                        PIC S9(4) COMP.
        05 IMAGE-ENTRY-LENGTH        PIC S9(4) COMP.
        05 IMAGE-RECORD-NUMBER       PIC S9(9) COMP.
        05 IMAGE-CHAIN-LENGTH        PIC S9(9) COMP.
        05 IMAGE-LAST-ON-CHAIN       PIC S9(9) COMP.
        05 IMAGE-FIRST-ON-CHAIN      PIC S9(9) COMP.

     02 ITEM.
        05 ITEM-VALUE                PIC X(16).

     02 IMAGE-SET.
        05 SET-VALUE                 PIC X(16).

     02 PASSWORD.
        05 PASSWORD-VALUE            PIC X(16).

     02 BASE.
        05 BASE-ID                   PIC XX.
        05 BASE-VALUE                PIC X(32).

     02 LIST.
        05 LIST-VALUE                PIC X(200).

     02 MODES.
        05 MODE1                     PIC S9(4) COMP.
        05 MODE2                     PIC S9(4) COMP.
        05 MODE3                     PIC S9(4) COMP.
        05 MODE4                     PIC S9(4) COMP.
        05 MODE5                     PIC S9(4) COMP.
        05 MODE6                     PIC S9(4) COMP.
        05 MODE7                     PIC S9(4) COMP.
        05 MODE8                     PIC S9(4) COMP.

     02 DUMMY                        PIC S9(4) COMP.
 01 COMAREA.
     02 VSTATUS                      PIC S9(4) COMP.
     02 VLANGUAGE                    PIC XX.
     02 COMAREA-LENGTH               PIC S9(4) COMP.
     02 FILLER                       PIC X(4).
     02 LAST-KEY                     PIC S9(4) COMP.
     02 NUMERRORS                    PIC S9(4) COMP.
     02 WINDOWENH                    PIC XX.
     02 FILLER                       PIC XX.
     02 LABELOPTION                  PIC S9(4) COMP.
     02 FORM-NAME                    PIC X(16).
     02 NEXT-FORM-NAME               PIC X(16).
     02 REPEATAPP                    PIC S9(4) COMP.
     02 FREEZAPP                     PIC S9(4) COMP.
     02 FILLER                       PIC XX.
     02 VBUFFER-LENGTH               PIC S9(4) COMP.
     02 FILLER                       PIC X(64).
```

```
        PROCEDURE DIVISION USING IMAGE COMAREA.
        BEGIN.
            MOVE 'n' TO DONE.
            MOVE SPACES TO SCREEN-BUFFER.
            MOVE SPACES TO MESSAGE-BUFFER.
            MOVE 72 TO MESSAGE-BUFFER-LENGTH.

            MOVE 'CUSTOMERS;' TO SET-VALUE.

            MOVE 'KEYWORD' TO NEXT-FORM-NAME.
            CALL 'VGETNEXTFORM' USING COMAREA.
            CALL 'VGETKEYLABELS' USING COMAREA FORM-KEYS NUMBER-OF-KEYS
                 KEY-BUFFER.
            CALL INTRINSIC '.LEN.' USING SCREEN-BUFFER GIVING
                 BUFFER-LENGTH.

            PERFORM UNTIL DONE IS EQUAL TO 'y'
               MOVE 'n' TO END-OF-SCREEN

               CALL 'VPUTBUFFER' USING COMAREA SCREEN-BUFFER
                  BUFFER-LENGTH
               CALL 'VSHOWFORM' USING COMAREA

               MOVE SPACES TO MESSAGE-BUFFER
               CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                    MESSAGE-BUFFER-LENGTH

               CALL 'VREADFIELDS' USING COMAREA
        IF LAST-KEY IS ZERO THEN
           CALL 'VGETBUFFER' USING COMAREA SCREEN-BUFFER
                BUFFER-LENGTH

           MOVE 'n' TO NO-ENTRIES
*************************************************************************
* THE FOLLOWING MOVE STATEMENTS ARE USED TO INITIALIZE THE "ITEM" AND   *
* "LIST" VARIABLES FOR THE CORRESPONDING SUPERDEX DBFIND AND DBGET.     *
* THE ITEM VALUE REPRESENTS THE SI-PATH AS DEFINED DURING THE           *
* CREATION OF THE INDEX. THE LIST VALUE REPRESENTS THE IMAGE ITEMS TO   *
* BE RETRIEVED BY THE THE RESULTING DBGET'S                             *
*************************************************************************
           MOVE 'CUSTOMER-NAME-KW;' TO ITEM-VALUE
           MOVE 'CUSTOMER-NUMBER,CUSTOMER-NAME;' TO LIST-VALUE
*************************************************************************
* THE FOLLOWING DBFIND IS USED BY SUPERDEX TO SCAN THE INDEX AS DEFINED *
* BY THE ITEM VALUE PREVIOUSLY LOADED INTO THE ITEM PARAMETER. THE DBFIND*
* DETERMINES ALL CORRESPONDING ENTRIES WHICH QUALIFY TO THE REQUESTED   *
* SCREEN ENTRY VALUE AND HOLD THEM FOR THE FOLLOWING DBGETS             *
*************************************************************************

           CALL 'DBFIND' USING BASE IMAGE-SET MODE1 IMAGE-STATUS
                ITEM SCREEN-KEY-VALUE
           IF CW IS NOT ZERO THEN
              MOVE 'No Qualifying Entries Found' TO
                 MESSAGE-BUFFER
```

```
               MOVE 'y' TO NO-ENTRIES
               MOVE SPACES TO DATA-LINES
               CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                        MESSAGE-BUFFER-LENGTH
            ELSE
               MOVE IMAGE-CHAIN-LENGTH TO ENTRIES-FOUND
            END-IF
         ELSE
            IF LAST-KEY IS EQUAL TO 8 THEN
               MOVE 'y' TO DONE
            END-IF
         END-IF

        IF (LAST-KEY IS EQUAL TO ZERO OR LAST-KEY IS EQUAL TO 1)
               AND NO-ENTRIES IS EQUAL TO 'n' THEN

            MOVE SPACES TO DATA-LINES
            MOVE 1 TO ARRAY-INDEX

            PERFORM UNTIL END-OF-SCREEN IS EQUAL TO 'y'
*************************************************************************
* THE FOLLOWING DBGET IS USED TO RETRIEVE INFORMATION FROM THE IMAGE    *
* DATASET WHICH CORRESPOND TO THE QUALIFYING ENTRIES RETRIEVED FROM     *
* THE PREVIOUS DBFIND.                                                  *
*************************************************************************
               CALL 'DBGET' USING BASE IMAGE-SET MODE5 IMAGE-STATUS
                  LIST IMAGE-BUFFER DUMMY

               IF CW IS NOT EQUAL TO ZERO THEN
                  MOVE 'y' TO END-OF-SCREEN
                  MOVE 'End of Current Entries' TO MESSAGE-BUFFER
                  CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                        MESSAGE-BUFFER-LENGTH
               ELSE
                  MOVE IMAGE-CUSTOMER-NUMBER TO TEMP-CUSTOMER-NUMBER
                  MOVE IMAGE-CUSTOMER-NAME TO TEMP-CUSTOMER-NAME
                  MOVE TEMP-LINE TO SCREEN-LINE(ARRAY-INDEX)
                  ADD 1 TO ARRAY-INDEX
                  IF ARRAY-INDEX IS GREATER THAN 15 THEN
                     MOVE 'y' TO END-OF-SCREEN
                     MOVE QUALIFY-BUFFER TO MESSAGE-BUFFER
                     CALL 'VPUTWINDOW' USING COMAREA
                          MESSAGE-BUFFER MESSAGE-BUFFER-LENGTH
                  END-IF
               END-IF
            END-PERFORM
         END-IF
      END-PERFORM.

      MOVE ZERO TO LAST-KEY

      EXIT PROGRAM.
```

Grouped Key Demo

```
$CONTROL SUBPROGRAM
 IDENTIFICATION DIVISION.
 PROGRAM-ID. GROUP-DEMO.
 AUTHOR. BRADMARK TECHNOLOGIES.

 ENVIRONMENT DIVISION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.

 01 SCREEN-BUFFER.
     02 SCREEN-KEY-VALUE              PIC X(50).
     02 DATA-LINES.
        05 SCREEN-LINE-ARRAY OCCURS 18 TIMES.
           10 SCREEN-LINE             PIC X(78).

 01 BUFFER-LENGTH                     PIC S9(4) COMP.

 01 ARRAY-INDEX                       PIC S9(4) COMP.

 01 DONE                              PIC X.
 01 END-OF-SCREEN                     PIC X.
 01 NO-ENTRIES                        PIC X.

 01 IMAGE-BUFFER.
     02 IMAGE-CUSTOMER                PIC X(30).
     02 IMAGE-ADDRESS1                PIC X(26).
     02 IMAGE-CITY                    PIC X(16).

 01 TEMP-LINE.
     02 TEMP-CUSTOMER                 PIC X(30).
     02 FILLER                        PIC X(1) VALUE SPACES.
     02 TEMP-ADDRESS1                 PIC X(26).
     02 FILLER                        PIC X(1) VALUE SPACES.
     02 TEMP-CITY                     PIC X(26).

 01 FORM-KEYS                         PIC S9(4) COMP VALUE 1.
 01 NUMBER-OF-KEYS                    PIC S9(4) COMP VALUE 8.
 01 KEY-BUFFER                        PIC X(128).
 01 MESSAGE-BUFFER                    PIC X(72).
 01 MESSAGE-BUFFER-LENGTH             PIC S9(4) COMP.
 01 QUALIFY-BUFFER.
     02 ENTRIES-FOUND                 PIC ZZ,ZZ9.
     02 FILLER                        PIC X(66) VALUE
       " Entries Qualified.  (More Entries Below)".
```

```
LINKAGE SECTION.
 01 IMAGE.
     02 IMAGE-STATUS.
        05 CW                        PIC S9(4) COMP.
        05 IMAGE-ENTRY-LENGTH        PIC S9(4) COMP.
        05 IMAGE-RECORD-NUMBER       PIC S9(9) COMP.
        05 IMAGE-CHAIN-LENGTH        PIC S9(9) COMP.
        05 IMAGE-LAST-ON-CHAIN       PIC S9(9) COMP.
        05 IMAGE-FIRST-ON-CHAIN      PIC S9(9) COMP.

     02 ITEM.
        05 ITEM-VALUE                PIC X(16).

     02 IMAGE-SET.
        05 SET-VALUE                 PIC X(16).

     02 PASSWORD.
        05 PASSWORD-VALUE            PIC X(16).

     02 BASE.
        05 BASE-ID                   PIC XX.
        05 BASE-VALUE                PIC X(32).

     02 LIST.
        05 LIST-VALUE                PIC X(200).

     02 MODES.
        05 MODE1                     PIC S9(4) COMP.
        05 MODE2                     PIC S9(4) COMP.
        05 MODE3                     PIC S9(4) COMP.
        05 MODE4                     PIC S9(4) COMP.
        05 MODE5                     PIC S9(4) COMP.
        05 MODE6                     PIC S9(4) COMP.
        05 MODE7                     PIC S9(4) COMP.
        05 MODE8                     PIC S9(4) COMP.

     02 DUMMY                        PIC S9(4) COMP.

 01 COMAREA.
     02 VSTATUS                      PIC S9(4) COMP.
     02 VLANGUAGE                    PIC XX.
     02 COMAREA-LENGTH               PIC S9(4) COMP.
     02 FILLER                       PIC X(4).
     02 LAST-KEY                     PIC S9(4) COMP.
     02 NUMERRORS                    PIC S9(4) COMP.
     02 WINDOWENH                    PIC XX.
     02 FILLER                       PIC XX.
     02 LABELOPTION                  PIC S9(4) COMP.
     02 FORM-NAME                    PIC X(16).
     02 NEXT-FORM-NAME               PIC X(16).
     02 REPEATAPP                    PIC S9(4) COMP.
     02 FREEZAPP                     PIC S9(4) COMP.
     02 FILLER                       PIC XX.
     02 VBUFFER-LENGTH               PIC S9(4) COMP.
     02 FILLER                       PIC X(64).
```

```
        PROCEDURE DIVISION USING IMAGE COMAREA.
        BEGIN.
            MOVE 'n' TO DONE.
            MOVE SPACES TO SCREEN-BUFFER.
            MOVE SPACES TO MESSAGE-BUFFER.
            MOVE 72 TO MESSAGE-BUFFER-LENGTH.

            MOVE 'CUSTOMERS;' TO SET-VALUE.

            MOVE 'GROUPKEY' TO NEXT-FORM-NAME.
            CALL 'VGETNEXTFORM' USING COMAREA.
            CALL 'VGETKEYLABELS' USING COMAREA FORM-KEYS NUMBER-OF-KEYS
                 KEY-BUFFER.
            CALL INTRINSIC '.LEN.' USING SCREEN-BUFFER GIVING
                 BUFFER-LENGTH.

            PERFORM UNTIL DONE IS EQUAL TO 'y'
               MOVE 'n' TO END-OF-SCREEN

               CALL 'VPUTBUFFER' USING COMAREA SCREEN-BUFFER
                  BUFFER-LENGTH
               CALL 'VSHOWFORM' USING COMAREA

               MOVE SPACES TO MESSAGE-BUFFER
               CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                    MESSAGE-BUFFER-LENGTH
        CALL 'VREADFIELDS' USING COMAREA

        IF LAST-KEY IS ZERO THEN
           CALL 'VGETBUFFER' USING COMAREA SCREEN-BUFFER
                BUFFER-LENGTH

           MOVE 'n' TO NO-ENTRIES
      ******************************************************************
      * THE FOLLOWING MOVE STATEMENTS ARE USED TO INITIALIZE THE "ITEM" AND    *
      * "LIST" VARIABLES FOR THE CORRESPONDING SUPERDEX DBFIND AND DBGET.      *
      * THE ITEM VALUE REPRESENTS THE SI-PATH AS DEFINED DURING THE            *
      * CREATION OF THE INDEX. THE LIST VALUE REPRESENTS THE IMAGE ITEMS TO    *
      * BE RETRIEVED BY THE THE RESULTING DBGET'S                              *
      ******************************************************************
           MOVE 'ADDRESS1-CITY-KW;' TO ITEM-VALUE
           MOVE 'CUSTOMER-NAME,ADDRESS-1,CITY;' TO LIST-VALUE
      ******************************************************************
      * THE FOLLOWING DBFIND IS USED BY SUPERDEX TO SCAN THE INDEX AS DEFINED  *
      * BY THE ITEM VALUE PREVIOUSLY LOADED INTO THE ITEM PARAMETER. THE DBFIND*
      * DETERMINES ALL CORRESPONDING ENTRIES WHICH QUALIFY TO THE REQUESTED    *
      * SCREEN ENTRY VALUE AND HOLD THEM FOR THE FOLLOWING DBGETS              *
      ******************************************************************
```

```
           CALL 'DBFIND' USING BASE IMAGE-SET MODE1 IMAGE-STATUS
                 ITEM SCREEN-KEY-VALUE
            IF CW IS NOT ZERO THEN
               MOVE SPACES TO DATA-LINES
               MOVE 'y' TO NO-ENTRIES
               MOVE 'No Qualifying Entries Found' TO
                 .MESSAGE-BUFFER
                  CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                        MESSAGE-BUFFER-LENGTH
               ELSE
                  MOVE IMAGE-CHAIN-LENGTH TO ENTRIES-FOUND
               END-IF
            ELSE
               IF LAST-KEY IS EQUAL TO 8 THEN
                  MOVE 'y' TO DONE
               END-IF
            END-IF

           IF (LAST-KEY IS EQUAL TO ZERO OR LAST-KEY IS EQUAL TO 1)
                  AND NO-ENTRIES IS EQUAL TO 'n' THEN
               MOVE SPACES TO DATA-LINES
               MOVE 1 TO ARRAY-INDEX

               PERFORM UNTIL END-OF-SCREEN IS EQUAL TO 'y'
*************************************************************************
* THE FOLLOWING DBGET IS USED TO RETRIEVE INFORMATION FROM THE IMAGE    *
* DATASET WHICH CORRESPOND TO THE QUALIFYING ENTRIES RETRIEVED FROM     *
* THE PREVIOUS DBFIND.                                                  *
*************************************************************************
                  CALL 'DBGET' USING BASE IMAGE-SET MODE5 IMAGE-STATUS
                     LIST IMAGE-BUFFER DUMMY
                  IF CW IS NOT EQUAL TO ZERO THEN
                     MOVE 'y' TO END-OF-SCREEN
                     MOVE 'End of Current Entries' TO MESSAGE-BUFFER
                     CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                           MESSAGE-BUFFER-LENGTH
                  ELSE
                     MOVE IMAGE-CUSTOMER TO TEMP-CUSTOMER
                     MOVE IMAGE-ADDRESS1 TO TEMP-ADDRESS1
                     MOVE IMAGE-CITY TO TEMP-CITY
                     MOVE TEMP-LINE TO SCREEN-LINE(ARRAY-INDEX)

                     ADD 1 TO ARRAY-INDEX
                     IF ARRAY-INDEX IS GREATER THAN 15 THEN
                        MOVE 'y' TO END-OF-SCREEN
                        MOVE QUALIFY-BUFFER TO MESSAGE-BUFFER
                        CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                           MESSAGE-BUFFER-LENGTH
                     END-IF
                  END-IF
               END-PERFORM
            END-IF
         END-PERFORM.

         MOVE ZERO TO LAST-KEY

         EXIT PROGRAM.
```

Relational Access Demo - multiple datasets

```
$CONTROL SUBPROGRAM
 IDENTIFICATION DIVISION.
 PROGRAM-ID. PROJECTION-DEMO.
 AUTHOR. BRADMARK TECHNOLOGIES.

 ENVIRONMENT DIVISION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.

 01 SCREEN-BUFFER.
     02 SCREEN-COMPANY                PIC X(30).
     02 SCREEN-ITEM                   PIC X(14).
     02 DATA-LINES.
        05 SCREEN-LINE-ARRAY OCCURS 15 TIMES.
           10 SCREEN-LINE             PIC X(78).

 01 BUFFER-LENGTH                     PIC S9(4) COMP.

 01 COUNT-ITEM.
     02 COUNT-ITEM-VALUE              PIC XX VALUE '@@'.

 01 FIND-ITEM.
     02 FILLER                        PIC X VALUE '['.
     02 FIND-ITEM-VALUE               PIC X(15).
     02 FILLER REDEFINES FIND-ITEM-VALUE.
        05 ITEM-CHARACTER-ARRAY OCCURS 15 TIMES.
           10 FILLER                  PIC X.

 01 FIND-CUST.
     02 FILLER                        PIC X VALUE '['.
     02 FIND-CUSTOMER-VALUE           PIC X(31).
     02 FILLER REDEFINES FIND-CUSTOMER-VALUE.
        05 CUST-CHARACTER-ARRAY OCCURS 31 TIMES.
           10 FILLER                  PIC X.

 01 ARRAY-INDEX                       PIC S9(4) COMP.

 01 DONE                              PIC X.
 01 END-OF-SCREEN                     PIC X.
 01 NO-ENTRIES                        PIC X.
 01 FORM-KEYS                         PIC S9(4) COMP VALUE 1.
 01 NUMBER-OF-KEYS                    PIC S9(4) COMP VALUE 8.
 01 KEY-BUFFER                        PIC X(128).
```

```
01 MESSAGE-BUFFER                       PIC X(72).
01 MESSAGE-BUFFER-LENGTH                PIC S9(4) COMP.

01 QUALIFY-BUFFER.
    02 ENTRIES-FOUND                    PIC ZZ,ZZ9.
    02 FILLER                           PIC X(66) VALUE
       " Entries Qualified.  (More Entries Below)".

01 CUSTOMER-NUMBER                       PIC S9(9) COMP.

01 IMAGE-BUFFER.
    02 IMAGE-ORDER-NUMBER                PIC S9(9) COMP.
    02 IMAGE-ITEM-KEY                    PIC X(14).
    02 IMAGE-ITEM-DESCRIPTION            PIC X(26).
    02 IMAGE-QUANTITY-ORDERD             PIC S9(4) COMP.
    02 IMAGE-LIST-PRICE                  PIC S9(9) COMP.

01 LIST-PRICE                           PIC 9(5)V99 COMP.

01 TEMP-LINE.
    02 TEMP-ORDER-NUMBER                 PIC 9(10) USAGE IS DISPLAY.
    02 FILLER                            PIC XX VALUE SPACES.
    02 TEMP-ITEM-KEY                     PIC X(14).
    02 FILLER                            PIC XX VALUE SPACES.
    02 TEMP-ITEM-DESCRIPTION             PIC X(26).
    02 FILLER                            PIC XX VALUE SPACES.
    02 TEMP-QUANTITY-ORDERD              PIC Z,ZZZ.
    02 FILLER                            PIC XX.
    02 TEMP-LIST-PRICE                   PIC Z,ZZ9.99.

01 PROJECTION-ARG                        PIC X(4) VALUE '[*];'.
01 I                                     PIC S9(5) COMP.

LINKAGE SECTION.
 01 IMAGE.
     02 IMAGE-STATUS.
        05 CW                      PIC S9(4) COMP.
        05 IMAGE-ENTRY-LENGTH      PIC S9(4) COMP.
        05 IMAGE-RECORD-NUMBER     PIC S9(9) COMP.
        05 IMAGE-CHAIN-LENGTH      PIC S9(9) COMP.
        05 IMAGE-LAST-ON-CHAIN     PIC S9(9) COMP.
        05 IMAGE-FIRST-ON-CHAIN PIC S9(9) COMP.
           02 ITEM.
        05 ITEM-VALUE              PIC X(16).
     02 IMAGE-SET.
        05 SET-VALUE               PIC X(16).

     02 PASSWORD.
        05 PASSWORD-VALUE          PIC X(16).
```

```
       02 BASE.
           05 BASE-ID                PIC XX.
           05 BASE-VALUE             PIC X(32).

        02 LIST.
           05 LIST-VALUE             PIC X(200).

        02 MODES.
           05 MODE1                  PIC S9(4) COMP.
           05 MODE2                  PIC S9(4) COMP.
           05 MODE3                  PIC S9(4) COMP.
           05 MODE4                  PIC S9(4) COMP.
           05 MODE5                  PIC S9(4) COMP.
           05 MODE6                  PIC S9(4) COMP.
           05 MODE7                  PIC S9(4) COMP.
           05 MODE8                  PIC S9(4) COMP.

        02 DUMMY                     PIC S9(4) COMP.

    01 COMAREA.
        02 VSTATUS                   PIC S9(4) COMP.
        02 VLANGUAGE                 PIC XX.
        02 COMAREA-LENGTH            PIC S9(4) COMP.
        02 FILLER                    PIC X(4).
        02 LAST-KEY                  PIC S9(4) COMP.
        02 NUMERRORS                 PIC S9(4) COMP.
        02 WINDOWENH                 PIC XX.
        02 FILLER                    PIC XX.
        02 LABELOPTION               PIC S9(4) COMP.
        02 FORM-NAME                 PIC X(16).
        02 NEXT-FORM-NAME            PIC X(16).
        02 REPEATAPP                 PIC S9(4) COMP.
        02 FREEZAPP                  PIC S9(4) COMP.
        02 FILLER                    PIC XX.
        02 VBUFFER-LENGTH            PIC S9(4) COMP.
        02 FILLER                    PIC X(64).
    PROCEDURE DIVISION USING IMAGE COMAREA.
    BEGIN.
        MOVE 'n' TO DONE.
        MOVE SPACES TO SCREEN-BUFFER.
        MOVE SPACES TO MESSAGE-BUFFER.
        MOVE 72 TO MESSAGE-BUFFER-LENGTH.
        MOVE 'PROJECTION' TO NEXT-FORM-NAME.
        CALL 'VGETNEXTFORM' USING COMAREA.
        CALL 'VGETKEYLABELS' USING COMAREA FORM-KEYS NUMBER-OF-KEYS
              KEY-BUFFER.
        CALL INTRINSIC '.LEN.' USING SCREEN-BUFFER GIVING
              BUFFER-LENGTH.

        PERFORM UNTIL DONE IS EQUAL TO 'y'
           MOVE 'n' TO END-OF-SCREEN
```

```
           CALL 'VPUTBUFFER' USING COMAREA SCREEN-BUFFER
              BUFFER-LENGTH
           CALL 'VSHOWFORM' USING COMAREA

           MOVE SPACES TO MESSAGE-BUFFER
           CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                 MESSAGE-BUFFER-LENGTH

           CALL 'VREADFIELDS' USING COMAREA

           IF LAST-KEY IS ZERO THEN
              CALL 'VGETBUFFER' USING COMAREA SCREEN-BUFFER
                     BUFFER-LENGTH

      ************************************************************************
      * INSERT THE SUPERDEX RELATIONAL OPERATORS INTO THE CUSTOMER NAME      *
      ************************************************************************
           PERFORM MAKE-FIND-CUST

           MOVE 'CUSTOMERS;' TO SET-VALUE
      ************************************************************************
      * THE FOLLOWING MOVE STATEMENT IS  USED TO INITIALIZE THE "ITEM"       *
      * VARIABLE  FOR THE FIRST SUPERDEX DBFIND.                             *
      * THE ITEM VALUE REPRESENTS THE SI-PATH AS DEFINED DURING THE          *
      * CREATION OF THE INDEX. THE LIST VALUE REPRESENTS THE IMAGE ITEMS TO  *
      * BE RETRIEVED BY THE THE RESULTING DBGET'S                            *
      ************************************************************************
           MOVE 'CUSTOMER-NAME;' TO ITEM-VALUE
           MOVE 'n' TO NO-ENTRIES

      ************************************************************************
      * THE FOLLOWING DBFIND IS USED BY SUPERDEX TO SCAN THE INDEX AS DEFINED *
      * BY THE ITEM VALUE PREVIOUSLY LOADED INTO THE ITEM PARAMETER. THE DBFIND*
      * DETERMINES ALL CORRESPONDING ENTRIES WHICH QUALIFY TO THE REQUESTED   *
      * SCREEN ENTRY VALUE.                                                  *
      ************************************************************************
           CALL 'DBFIND' USING BASE IMAGE-SET MODE1 IMAGE-STATUS
                 ITEM FIND-CUST
           IF CW IS NOT ZERO THEN
              MOVE 'y' TO NO-ENTRIES
              MOVE 'No Qualifying Entries Found' TO
                 MESSAGE-BUFFER
                 MOVE SPACES TO DATA-LINES
                 CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                       MESSAGE-BUFFER-LENGTH
```

```
           ELSE
            IF IMAGE-CHAIN-LENGTH IS NOT EQUAL TO 1 THEN
               MOVE 'y' TO NO-ENTRIES
               MOVE 'More than one Entry Qualified' TO
                  MESSAGE-BUFFER
                  MOVE SPACES TO DATA-LINES
                  CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                        MESSAGE-BUFFER-LENGTH
            ELSE

*************************************************************************
* HERE IS WHERE THE PROJECTION FROM THE ORDER-HEADERS DATASET           *
* IS PERFORMED. FIRST THE NAME OF THE IMAGE DATASET THAT THE PROJECTION *
* WILL BE PERFORMED AGAINST IS MOVED INTO THE SET PARAMETER.            *
*************************************************************************
               MOVE 'ORDER-HEADERS;' TO SET-VALUE
*************************************************************************
* SECONDLY THE SI-PATH NAME OF THE PROJECTION IS MOVED TO THE ITEM      *
* PARAMETER                                                             *
*************************************************************************
               MOVE 'CUSTOMER-NUMBER;' TO ITEM-VALUE
*************************************************************************
* THE PROJECTION IS PERFORMED BY USING A DBFIND WITH THE PROJECTION     *
* ARGUMENT (*[*];*).                                                    *
*************************************************************************
               CALL 'DBFIND' USING BASE IMAGE-SET MODE1
                  IMAGE-STATUS ITEM PROJECTION-ARG

               IF CW IS NOT ZERO THEN
                  MOVE 'y' TO NO-ENTRIES
                  MOVE 'No Orders Found for the Customer' TO
                     MESSAGE-BUFFER
                  MOVE SPACES TO DATA-LINES
                  CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                        MESSAGE-BUFFER-LENGTH
               ELSE
*************************************************************************
* HERE IS WHERE THE ORDER-LINES ARE QUALIFIED BY A BOOLEAN "AND"        *
* OPERATION BETWEEN THE ALREADY QUALIFIED ORDER-HEADERS ENTRIES         *
* AND THE ORDER-LINES DATA SET.                                         *
*************************************************************************

*************************************************************************
* INSERT THE SUPERDEX RELATIONAL OPERATORS INTO THE PART NUMBER         *
*************************************************************************
                  PERFORM MAKE-FIND-ITEM
                  MOVE 'ORDER-LINES;' TO SET-VALUE
*************************************************************************
* THE SI-PATH THAT CONTAINS THE PART NUMBER AND THE ORDER NUMBER,       *
* IN THAT ORDER, IS MOVED TO THE ITEM ARGUMENT OF THE DBFIND.           *
*************************************************************************
```

```
                MOVE 'PART-ORDER;' TO ITEM-VALUE
*************************************************************************
* THE DBFIND WILL PERFORM THE BOOLEAN "AND" BETWEEN THE TWO SETS.       *
* THE "AND" OPERATOR ("&") WAS MOVED INTO THE ARGUMENT PARAMETER        *
* BY THE MAKE-FIND-ITEM PROCEDURE.                                      *
*************************************************************************
                CALL 'DBFIND' USING BASE IMAGE-SET MODE1
                    IMAGE-STATUS ITEM FIND-ITEM

                IF CW NOT EQUAL ZERO THEN
                   MOVE 'y' TO NO-ENTRIES
                   MOVE 'No Items Found for the Customer'
                      TO MESSAGE-BUFFER
                   MOVE SPACES TO DATA-LINES
                   CALL 'VPUTWINDOW' USING COMAREA
                        MESSAGE-BUFFER MESSAGE-BUFFER-LENGTH
                ELSE
*************************************************************************
* THE NEXT DBFIND IS NEEDED TO DETERMINE THE NUMBER OF QUALIFYING       *
* ENTRIES IN THE ITEM DATASET. THE CHAIN LENGTH VALUE OF THE IMAGE      *
* STATUS ARRAY CONTAINED THE TOTAL NUMBER OF FOUND ENTRIES BY THE       *
* THREE DBFIND'S. THE NULL ITEM INSTRUCTS SUPERDEX TO COUNT THE         *
* QUALIFYING ENTRIES OF THE DATASET SPECIFIED BY THE SET PARAMETER      *
*************************************************************************
                   MOVE ';' TO ITEM-VALUE

                   CALL 'DBFIND' USING BASE IMAGE-SET MODE1
                      IMAGE-STATUS ITEM COUNT-ITEM

                   MOVE IMAGE-CHAIN-LENGTH TO
                        ENTRIES-FOUND
                END-IF
             END-IF
          END-IF
          END-IF        ELSE
          IF LAST-KEY IS EQUAL TO 8 THEN
             MOVE 'y' TO DONE
          END-IF
       END-IF

      IF (LAST-KEY IS EQUAL TO ZERO OR LAST-KEY IS EQUAL TO 1)
               AND NO-ENTRIES IS EQUAL TO 'n' THEN
          MOVE SPACES TO DATA-LINES
          MOVE 1 TO ARRAY-INDEX
          MOVE
           'ORDER-NUMBER,PART-NUMBER,PART-DESCRIPTION,QUANTITY-OR
-           'DERED,UNIT-PRICE;' TO LIST-VALUE

          PERFORM UNTIL END-OF-SCREEN IS EQUAL TO 'y'
```

```
*************************************************************************
* THE FOLLOWING DBGET IS USED TO RETRIEVE INFORMATION FROM THE IMAGE    *
* DATASET WHICH CORRESPONDS TO THE QUALIFYING ENTRIES RETRIEVED FROM    *
* THE PREVIOUS DBFIND.                                                  *
*************************************************************************
              CALL 'DBGET' USING BASE IMAGE-SET MODE5 IMAGE-STATUS
                 LIST IMAGE-BUFFER DUMMY
              IF CW IS NOT EQUAL TO ZERO THEN
                 MOVE 'y' TO END-OF-SCREEN
                 MOVE 'End of Current Entries' TO MESSAGE-BUFFER
                 CALL 'VPUTWINDOW' USING COMAREA MESSAGE-BUFFER
                       MESSAGE-BUFFER-LENGTH
              ELSE
                 MOVE IMAGE-ORDER-NUMBER
                    TO TEMP-ORDER-NUMBER
                 MOVE IMAGE-ITEM-KEY
                    TO TEMP-ITEM-KEY
                 MOVE IMAGE-ITEM-DESCRIPTION
                    TO TEMP-ITEM-DESCRIPTION
                 MOVE IMAGE-QUANTITY-ORDERD
                    TO TEMP-QUANTITY-ORDERD
                 COMPUTE LIST-PRICE = IMAGE-LIST-PRICE / 100 *
                     IMAGE-QUANTITY-ORDERD
                 MOVE LIST-PRICE TO TEMP-LIST-PRICE

                 MOVE TEMP-LINE TO
                    SCREEN-LINE(ARRAY-INDEX)
                 ADD 1 TO ARRAY-INDEX
                 IF ARRAY-INDEX IS GREATER THAN 15 THEN
                    MOVE 'y' TO END-OF-SCREEN
                    MOVE QUALIFY-BUFFER TO MESSAGE-BUFFER
                    CALL 'VPUTWINDOW' USING COMAREA
                         MESSAGE-BUFFER MESSAGE-BUFFER-LENGTH
                 END-IF
              END-IF
           END-PERFORM
        END-IF
     END-PERFORM.

     MOVE ZERO TO LAST-KEY

     EXIT PROGRAM.
```

```
*************************************************************************
* THIS ROUTINE BUILDS THE ARGUMENT FOR THE DBFIND ON THE CUSTOMER*
* DATASET. THE ARGUMENT IS PRECEDED BY A '[' AND IS TERMINATED BY*
* A ']'. THE SQUARE BRACKETS ARE THE OPERATORS FOR THE RELATIONAL*
* SUBSYSTEM OF SUPERDEX.                                          *
*************************************************************************
 MAKE-FIND-CUST.
     MOVE SCREEN-CUSTOMER TO FIND-CUSTOMER-VALUE.
     MOVE 30 TO I.
     PERFORM UNTIL (I IS EQUAL TO ZERO) OR
        (CUST-CHARACTER-ARRAY(I) IS NOT EQUAL TO SPACE)
           SUBTRACT 1 FROM I
     END-PERFORM.
     ADD 1 TO I.
     MOVE ']' TO CUST-CHARACTER-ARRAY(I).

 MAKE-FIND-CUST-EXIT.
     EXIT.
*************************************************************************
* THIS ROUTINE BUILDS THE ARGUMENT FOR THE DBFIND ON THE ORDER-NUMBER   *
* DATASET. THE VALUE OF THE ENTRY IS PRECEDED BY A '[' AND IS           *
* FOLLOWED BY A ']'. AFTER THE ']', A '&' IS APPENDED TO THE STRING.THE *
* '&' IS SUPERDEX'S OPERATOR FOR A LOGICAL AND.                         *
*************************************************************************
 MAKE-FIND-ITEM.
     MOVE SCREEN-ITEM TO FIND-ITEM-VALUE.
     MOVE 14 TO I.
     PERFORM UNTIL (I IS EQUAL TO ZERO) OR
        (ITEM-CHARACTER-ARRAY(I) IS NOT EQUAL TO SPACE)
           SUBTRACT 1 FROM I
     END-PERFORM.
     ADD 1 TO I.
     MOVE ']' TO ITEM-CHARACTER-ARRAY(I).
     ADD 1 TO I.
     MOVE '&' TO ITEM-CHARACTER-ARRAY(I).

 MAKE-FIND-ITEM-EXIT.
     EXIT.
```