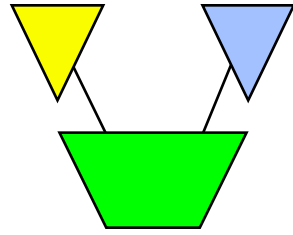


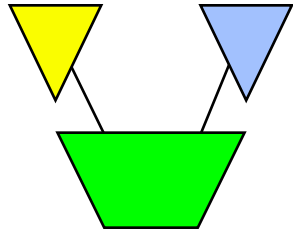
Inside Module 2



Working with Databases

Page

- Choosing input from databases 2
- Reading an entire dataset serially 6
- Determining fields in a dataset 8
- Reading specific data chains 10
- Choosing Get versus Chain 11
- Listing records 13
- Changing Output Field Structure 14
- Creating Basic Reports 16
- Changing data in datasets 22



Accessing data files

- These Suprtool commands access TurboIMAGE and Eloquence datasets:

BASE

GET

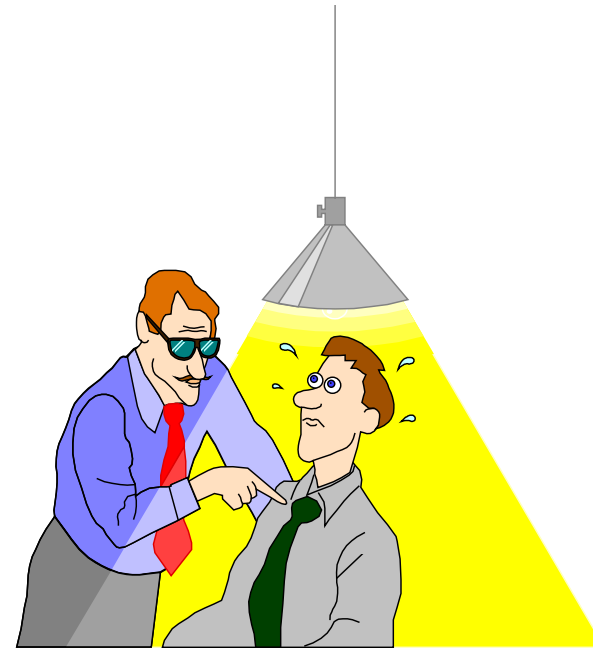
CHAIN

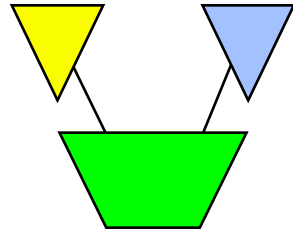
FORM

PUT

DELETE

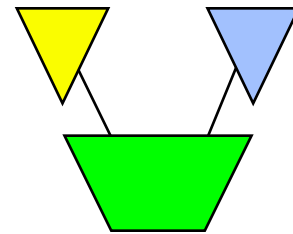
UPDATE





Opening and closing a database (Image)

- You can use the BASE command to open a database
`>base store,5,READER`
- The BASE command without parameters closes a database
- A database remains open until a BASE, RESET BASE or RESET ALL command is executed
- Eloquence Base command is slightly different; see module 3



How to find datasets in a database

- Use the FORM SETS command to display datasets

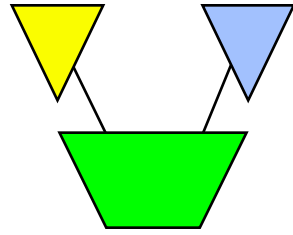
```
>base store.demo
```

```
Database password[;]?
```

```
>form sets
```

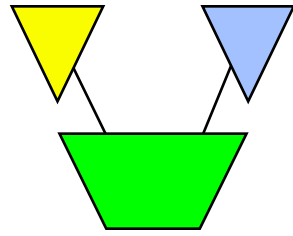
```
Database:STORE.DEMO.APPDEV TPI: SUPERDEX(15015d) 4.0.39
```

	Set	Item	Capa-	Entry	Load	Entry		
Sets:	Num	Type	Count	city	Count	Factor	Length	B/F
M-CUSTOMER	1	M	9	211	12	9 %	55	7
M-PRODUCT	2	M	2	307	13	4 %	24	12
M-SUPPLIER	3	M	6	211	3	1 %	49	8
D-INVENTORY	4	D	6	462	13	3 %	15	22
D-SALES	5	D	8	602	8	1 %	19	14



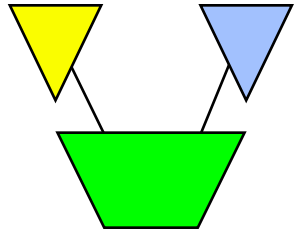
More about the Form command

- The FORM command without parameters first defaults to the current input dataset. If no input has been specified, then it defaults to FORM SETS.
- All output is written to the Formout file, which can be redirected to a line printer or a disc file. Currently the Formout file is not available on HP-UX.



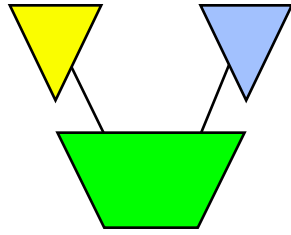
Datasets as input sources

- The GET command reads a dataset in one of several ways
 - It can read the entire dataset serially
 - It can read a subset of dataset records serially
 - It can read records at a specified interval (e.g., every 5th record). This kind of sampling is useful for test purposes.
- A database must be open before you can use the GET command



Warnings using Get

- Suprtool checks the dataset entry count before and after processing, and warns you if it has changed.
- Suprtool permits concurrent changes, but warns you when this happens. If you need exclusive access, open the database in mode-4.
- If you repeatedly receive warnings of new entries, use the SET EOFREAD ON command to read to end-of-file. (Must be specified before the GET command!)



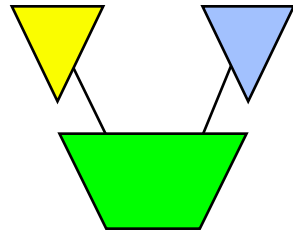
Determining fields in a dataset

- Use FORM *setname* to display the fields in a dataset

>form m-customer

```
M-CUSTOMER      MASTER      SET 1
Entry:          Offset
  CITY          X12          1
  CREDIT-RATING J2           13
  CUST-ACCOUNT  Z8           17  <<SearchField>>
  CUST-STATUS   X2           25
  NAME-FIRST    X10          27
  NAME-LAST     X16          37
  STATE-CODE    X2           53
  STREET-ADDRESS 2X25         55
  ZIP-CODE      X6           105
```

```
Capacity:211 (7)  Entries:20  Bytes:110
```

Defining New Fields

- Create new field definitions:

```
> define D-STATUS, 25, 1, CHAR
```

- ABSOLUTE definition:

```
define field, byteposition, length [, type]
```

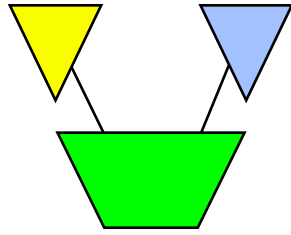
```
e.g. > define ord-total, 20, 4, integer
```

- RELATIVE definition:

```
define field, fieldname [(subscript)] [[offset]], length [, type]
```

```
e.g. > define branch-no, cust-code[1], 2
```

Relative defines are associated with a record item, so will stay correct if the field sequence changes.



Reading specific data chains

- If you know the key value(s), use the CHAIN command to search a dataset and select records with the specified key

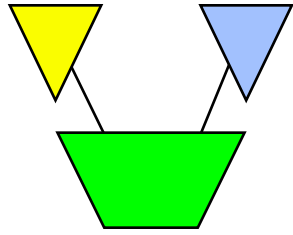
```
>chain d-sales, customer = "123456"
```

```
>chain dtrans, partnum = "A123", "B654", "G999"
```

```
>chain d-sales, customer = slist {use a table}
```

- Even when you know the key values, the GET command may select the same records faster than CHAIN can

```
>get d-sales; if $lookup(slist, customer)
```



Get versus Chain command

GET

Serial access

Any dataset

All records

Physical order

MR NOBUF reads

Selection by any data fields

CHAIN

Keyed access

Only keyed datasets

Only records with key values

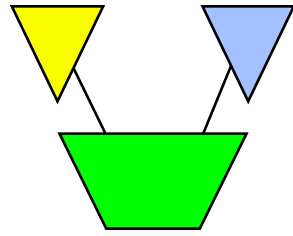
Forward chain pointers

DBFIND and DBGET mode-5 and -7

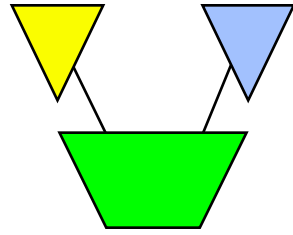
Selection by key field

Exercise 1

Get versus Chain: quick, choose one!



- Your task is to retrieve records from the infamous ord-line detail dataset which contains 2.3 million records of 308 bytes each. The key values to be selected are in a file called Ordfile. These 162,000 ord-num field values will select 261,000 records from the dataset.
- Your mission, Jim, should you decide to accept it, will be to access the records as quickly as possible, using either the GET command or the CHAIN command. The final results must be sorted in ord-num sequence.
- As always, should you fail, the Secretary will disavow all knowledge of your actions.



Listing data from datasets

- Use the LIST command without parameters to list records whose format is known

```
>get m-customer
```

```
>list
```

```
>xeq
```

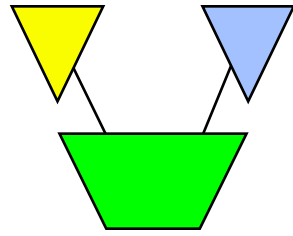
```
>GET M-CUSTOMER (1) >OUT $NULL (0)
```

```
CITY          = Edmonton  CREDIT-RATING = 240000
```

```
CUST-ACCT     = 10005      CUST-STATUS   = 30
```

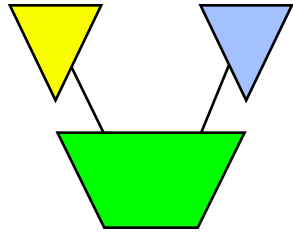
```
NAME-FIRST    = Terry      NAME-LAST     = Coyle
```

```
STATE-CODE    = AL
```



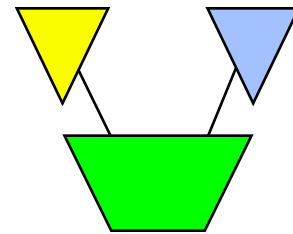
Changing Field Structure of Output

- By default **all** fields in the input record are copied to the output record.
- The EXTRACT command overrides this default.
`extract field [(subscript)] [=value][,.....]`
`extract field1 \ field2`
- Can have multiple EXTRACT commands
- Up to 255 extracted fields
- Can specify fieldnames, constants, strings
- Output record will be assembled with fields in the same sequence as the EXTRACT commands.



Extract example

```
>get m-customer
>extract name-first, name-last
>extract " City: "
>extract city
>output *
>xeq
Wayne      Humphreys      City: Vancouver
Elizabeth  Welton         City: Coquitlam
William    Kirk           City: Richmond
Jack       Morrison       City: Calgary
James     Young          City: Edmonton
Percy     Ferguson       City: Coquitlam
Walley    Nisbet         City: Surrey
.....
```



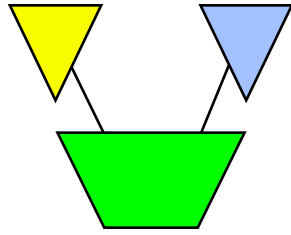
A quick way to produce basic reports

- Use the LIST STANDARD command to produce a report with a predefined format

Feb 03, 1996 Base STORE.DEMO Set M-CUSTOMER Page 1

CUST-ACCO	CITY	NAME-FIRST	NAME-LAST
10004	Edmonton	Arthur	Rogers
10005	Edmonton	Terry	Coyle
10015	Edmonton	James	Young
10016	Edmonton	Tara	Bamford

IN=4, OUT=4. CPU-SEC=1. WALL-SEC=1.

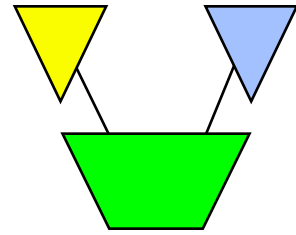


Suprtool lets you customize reports

You can modify reports to improve their appearance or functionality by doing the following:

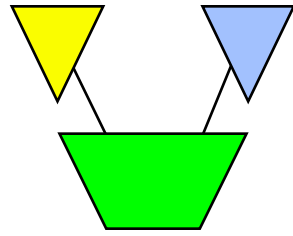
- changing the report title
- changing heading names
- changing the sort key to make the report contents more meaningful

Customizing a report title and column headings



- It is easy to change your report title or column headings

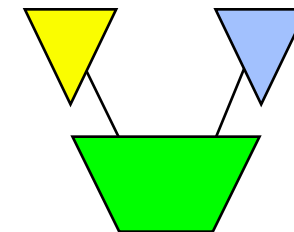
```
>get m-customer
>if city = "Edmonton"
>sort name-last
>list standard,title "Customers in Edmonton",&
>> heading "Customer Name           ",&
>>           "City                 ",&
>>           "Account"
>ext name-last,name-first,city,cust-account
>xeg
```



MPE/iX third-party indexing

- Requires Omnidex or Superdex indexing software or HP B-tree support (not currently supported in Suprtool/UX)
- CHAIN command can access third-party or IMAGE indexes

```
>chain m-customer,name-last = "A@"
```
- FORM command marks IMAGE fields with third-party indexing as "<<TPI>>", and B-trees as "<<Indexed>>"
- VERIFY BASE command displays name and version of indexing software



Form command shows third-party indexes

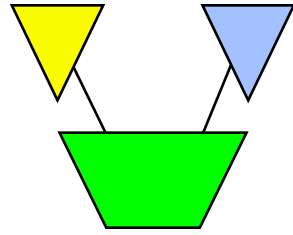
>form m-customer

M-CUSTOMER	Master	Set#1
Entry:	Offset	
CITY	X12	1 <<TPI>>
CREDIT-RATING	J2	13
CUST-ACCOUNT	Z8	17 <<SearchField>> <<TPI>>
CUST-STATUS	X2	25
NAME-FIRST	X10	27 <<TPI>>
NAME-LAST	X16	37 <<TPI>>
STATE-CODE	X2	53 <<TPI>>
STREET-ADDRESS	2X25	55
POSTAL-CODE	X6	105

Capacity: 211 Entries:20 Entry Length:55 Blocking:7

Exercise 2

Create a listing of the Alberta customers



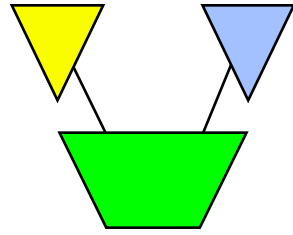
- Create the following report from the STORE database:

Mar 20, 1996 20:32

Alberta Customers

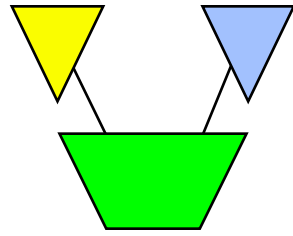
Page 1

Account#	Name	City
10004	Rogers	Edmonton
10005	Coyle	Edmonton
10006	Frahm	Calgary
10007	Tiernan	Calgary
10015	Young	Edmonton
10016	Bamford	Edmonton
10017	Morrison	Calgary
10018	Johnston	Calgary



Changing data in datasets

- The Put, Delete and Update commands make changes to the contents of a dataset
- You must open the database in mode 1, 2, 3, or 4
- You can disable the Put, Delete, and Update functions via the Set Limits ReadOnly command
`>set limits readonly on`

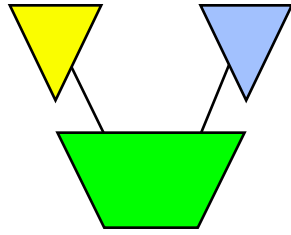


Moving data into datasets

- We recommend this set of commands to perform a major load of a dataset from a file

```
>input loadfile
>set dumponerror on           {default}
>set defer on
>set ignore on
>put m-cust,store.pub,3
>xeq
```

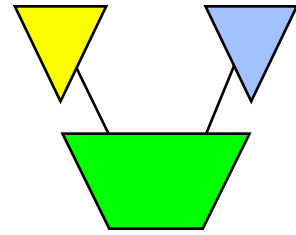
- Input file record structure must match the destination dataset structure exactly!



What if the data doesn't match exactly?

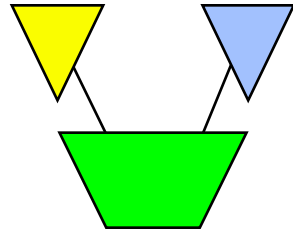
- Use EXTRACT commands to construct the output record
- Use DEFINE and EXTRACT to change storage formats:
 - > `define amount,1,8,display {...in input file}`
 - > `define new-amount,1,4,integer {new field}`
 - > `extract new-amount = amount`
- Field will have attributes as defined, and value from input record, so the output record will contain the 4-byte integer value of the 8-byte display field in the input record.

Deleting selected records from the input dataset



- Open the database in mode-1, -3, or -4
- Access the dataset using GET or CHAIN
- Select records to be deleted with IF command
- Delete the selected records using DELETE
- Optional step: copy the deleted records somewhere else (e.g., OUTPUT file, LIST file, PUT to another dataset)

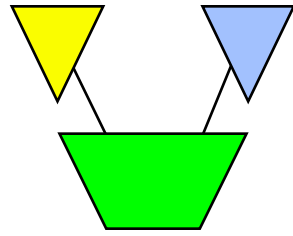
```
>get d-sales
>item purch-date,date,yymmdd
>if purch-date < $date(*-
1/*/*)
>delete
>output oldsales.data,append
>xeq
```



Using two passes guarantees safety

```
>get d-sales  
>item purch-date,date,yymmdd  
>if purch-date < $date(*-1/*/*)  
>output oldsales.data,append  
>xeq
```

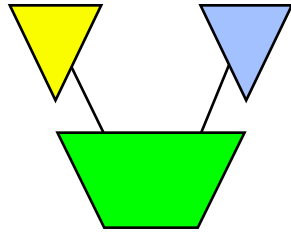
```
>get d-sales  
>if purch-date < $date(*-1/*/*)  
>delete  
>output $null  
>xeq
```



Update selected records with new values

- Open the database in mode-1, -2, -3, or -4
- Access the dataset using GET or CHAIN
- Select records to be updated using IF
- Enable updating using UPDATE command; use CIUPDATE parameter to update critical fields
- Specify fields and new values using EXTRACT commands

```
>get d-sales  
>item purch-date,date,yymmdd  
>if purch-date < $date(*-1/*/*)  
>update  
>extract purch-status = "OLD"  
>xeq
```



Assigning Calculated Values

```
>get d-sales
```

```
>update
```

```
>extract sales-total = &
```

```
    (product-price * sales-qty) + sales-tax
```

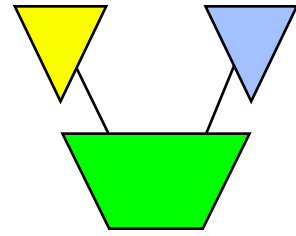
```
>xeq
```

```
Update all records from the D-SALES dataset [no]: yes
```

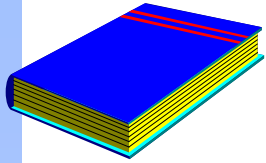
```
Warning: Using DBGET for the input records
```

```
IN=8, OUT=8. CPU-Sec=1. Wall-Sec=1.
```

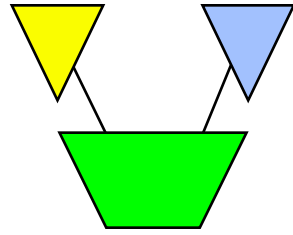
Set Lock to control concurrent dataset access



- SET LOCK 1
 - Lock the dataset and unlock it again around every DELETE, PUT, and UPDATE
 - Least contention with other processes, but slowest option for Suprtool
- SET LOCK 0
 - Lock the dataset at the beginning of the task and unlock it only at the end
 - Best performance for Suprtool, but locks out other processes for duration of Suprtool run
- SET LOCK n
 - Lock dataset on n DELETE, PUT, or UPDATE transactions, then unlock
 - Compromise between SET LOCK 0 and SET LOCK 1



Summary



- Display datasets
- Field names and formats
- Data chains
- List datasets
- Reports (e.g., standard, customized)
- Third-party indexing
- Adding, deleting, and modifying records
- Changing data formats
- Locking options