Super SUPRTOOL Solutions

- A Kubler Consulting Presentation at HP World 2001, Chicago

What's Inside

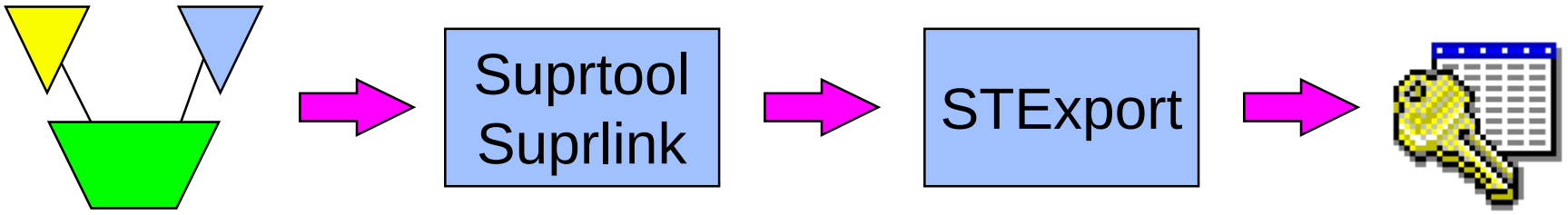
- Introduction
- Newer Features
  - STEXPORT – Webpage creation
  - CI Variables
- Updating tables using SUPRTOOL and DBEDIT
- Data Validation with SUPRTOOL
- Adding a record count and subtotaling

Introduction

- **SPEED Stories**
  - **SUPRTOOL replaces Query**
  - **SUPRTOOL2 interfaces with COBOL**
- **Great Uses**
  - **Archival application**
  - **HTML presentations of data**
  - **Data validation**
- **One of the most Prevalent tools**
  - AMISYS, SGA (now ECOMETRY), etc.

STEXPORT - Exporting IMAGE/SQL Data to other Applications

- Extract the IMAGE data using Suprtool and Suprlink
- Convert the files using STExport
- Transfer the file to the PC
- Import the delimited file



Data needs to be converted

- Image data has:
  - Fixed-width fields
  - binary storage formats (J2, K2, P28, etc)
  - Structure defined in Root File.

- PC Applications require:
  - variable-length fields
  - Ascii values for numerics
  - field delimiters
  - Field name declarations

STExport converts the data

- STExport reads self-describing files
- Outputs ascii files
- Allows you to specify:
  - field delimiters to use
  - date format
  - fieldnames in first record
  - numeric format
  - fixed or variable length
  - quotes on character fields
  - HTML - *table* or *preformatted*

3 ways to run STExport

- From the MPE prompt:
  :run stexport.pub.robelle

- From Suprtool:
  >export

STExport/iX/Copyright Robelle Consulting Ltd. 1995-1998 Type H for Help.
(Version 4.0.17 Pre-Release) MON, MAR 9, 1998, 1:50 PM
Licensee: Robelle Consulting Ltd. [0]

- From *inside* Suprtool:

```
>export input custsd
>export output custexp
>export exit
In=20. Out=20. CPU-Sec=1. Wall-Sec=1.
```

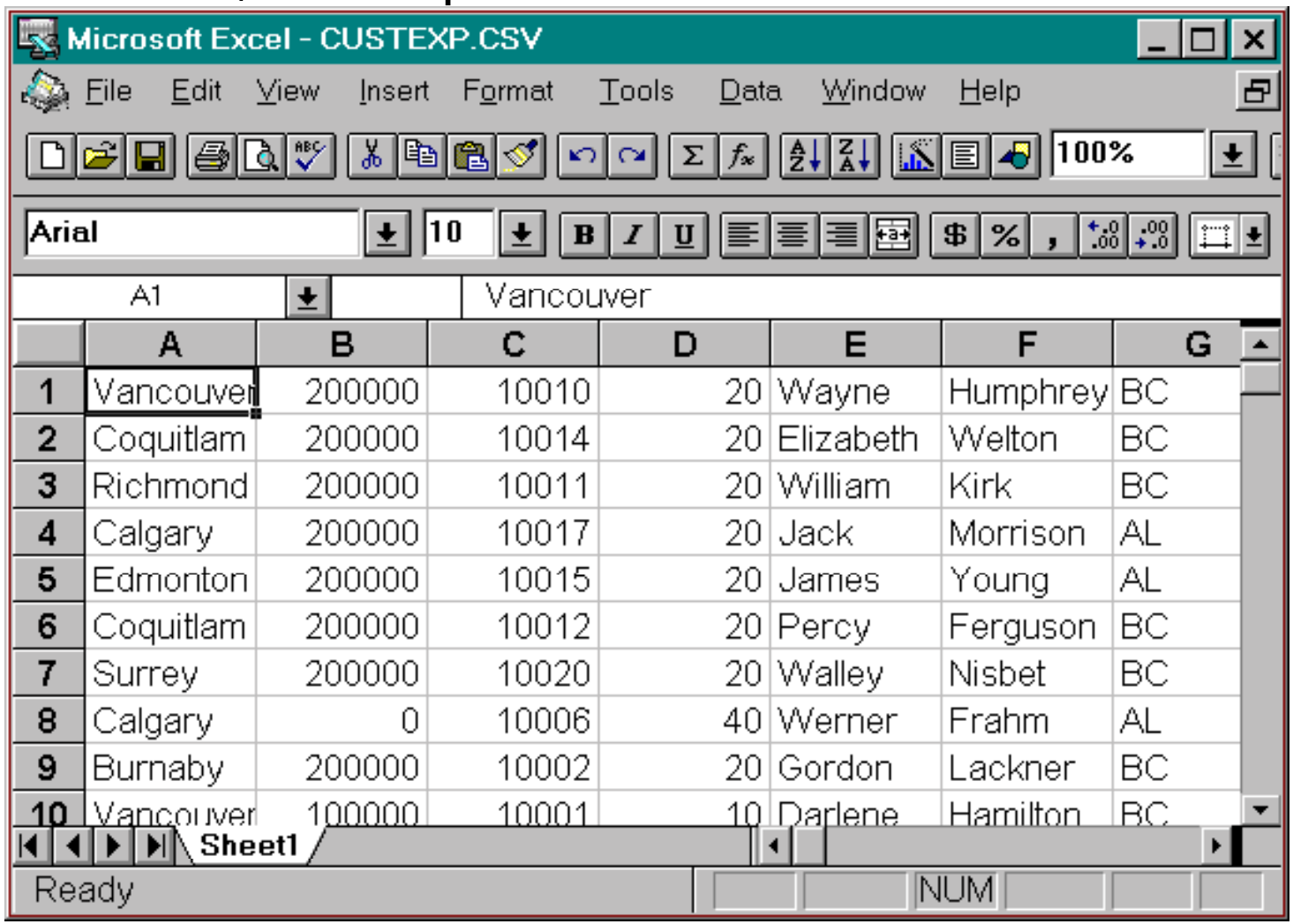
For example .....

```
>export
$in custsd
$out custexp
$xeq
In=19. Out=19. CPU-Sec=1. Wall-Sec=2.
$print custexp

"Vancouver",200000,10010,"20","Wayne","Humphreys","BC",....
"Coquitlam",200000,10014,"20","Elizabeth","Welton","BC",...
"Richmond",200000,10011,"20","William","Kirk","BC",.....
"Calgary",200000,10017,"20","Jack","Morrison","AL",.....
"Edmonton",200000,10015,"20","James","Young","AL",......
"Coquitlam",200000,10012,"20","Percy","Ferguson","BC",.....
"Surrey",200000,10020,"20","Walley","Nisbet","BC",.....
```

In MS Excel

- Transfer to PC, File/Open in Microsoft Excel:



	A	B	C	D	E	F	G
1	Vancouver	200000	10010	20	Wayne	Humphrey	BC
2	Coquitlam	200000	10014	20	Elizabeth	Welton	BC
3	Richmond	200000	10011	20	William	Kirk	BC
4	Calgary	200000	10017	20	Jack	Morrison	AL
5	Edmonton	200000	10015	20	James	Young	AL
6	Coquitlam	200000	10012	20	Percy	Ferguson	BC
7	Surrey	200000	10020	20	Walley	Nisbet	BC
8	Calgary	0	10006	40	Werner	Frahm	AL
9	Burnaby	200000	10002	20	Gordon	Lackner	BC
10	Vancouver	100000	10001	10	Darlene	Hamilton	BC

Dates and Decimals

- Use Suprtool's ITEM command to qualify the fields:

```
>get d-sales
>item deliv-date,date,YYYYMMDD
>item product-price,decimal,2
>out salesd,link
>x
IN=8, OUT=8. CPU-Sec=1. Wall-Sec=1.
>form salesd
File: SALESD.HANS.TRAINING     (SD Version B.00.00)
      Entry:                     Offset
         CUST-ACCOUNT          Z8      1
         DELIV-DATE            I2      9        <<YYYYMMDD>>
         PRODUCT-NO            Z8     13
         PRODUCT-PRICE         I2     21        << .2  >>
         PURCH-DATE            I2     25 ...etc
```

..... continued

- Specify date format in STEXPORT:

```
>export
$in salesd
$date DDMMYY "/"
$output *
$x
10020,04/10/97,50511501,98.31,19971000,2,2753,22415
10003,16/10/97,50511501,98.31,19971016,1,1376,11207
10003,16/10/97,50512501,145.62,19971016,1,2039,16600
10003,16/10/97,50513001,192.20,19971016,1,2691,21910
10016,20/10/97,50521001,24.59,19971020,3,1033,8411
10016,20/10/97,50532001,139.85,19971020,1,1958,15942
10020,28/10/97,50512501,146.60,19971028,1,2052,16713
10010,20/10/97,50533001,69.92,19971020,1,979,7970
In=8. Out=8. CPU-Sec=1. Wall-Sec=1.
```

Specifying field names

- Use HEADING command to add fieldnames in the first record:

```
$heading '"Description","Model"'
$heading add ',"Product Code"'
$output *
$xeq
   "Description","Model","Product Code"
   "Skil 3/8  Variable Speed Drill","#6523",50531501
   "B&D Router","#7613-04",50522001
   "Skil Var. Sp. Auto-Scroll Saw","#4560",50533001
   "Skil 8 1/2  Circular Saw","#5665",50532501
   .....etc.....
```

- HEADING FIELDNAMES uses Image field names.

Fixed-length output

- Force fixed-length with COLUMNS command

```
$input prodsd
$columns fixed
$out *
$x
"Description","Model","Product Code"
"Skil 3/8  Variable Speed Drill","#6523"     , 50531501
"B&D Router"                    ,"#7613-04"  , 50522001
"Skil Var. Sp. Auto-Scroll Saw" ,"#4560"     , 50533001
"Skil 8 1/2  Circular Saw"      ,"#5665"     , 50532501
"B&D Cordless Screwdriver"      ,"#9018-04"  , 50521001
```

- Also see SPACES and ZERO commands

Preparing Data For The Web

- STExport can create HTML files
- Data can be formatted in a table
  - HTML TABLE command
- Or it can be formatted like a List Standard listing
  - HTML PREFORMATTED command
- Formatting is applied by STExport
  - Numeric data is right justified, with decimal points
  - Alpha data is left justified
  - Dates are formatted as you specify

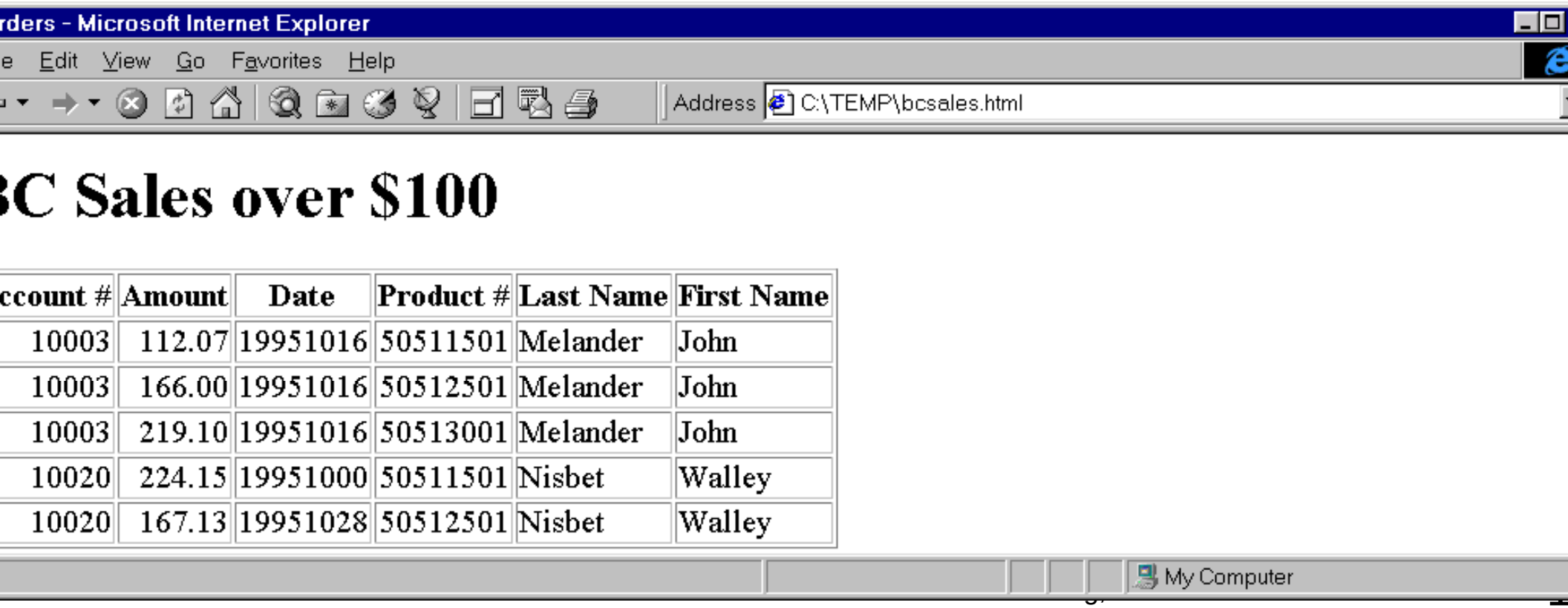
Preparing HTML Tables

- Use the HTML TABLE command

```
$input reptfile
$heading none
$heading column "Account #"
$heading column "Amount"
$heading column "Date"
$heading column "Product #"
$heading column "Last Name"
$heading column "First Name"
$html table title "Orders" heading "BC Sales over $100"
$output bcsales
$xeq
```

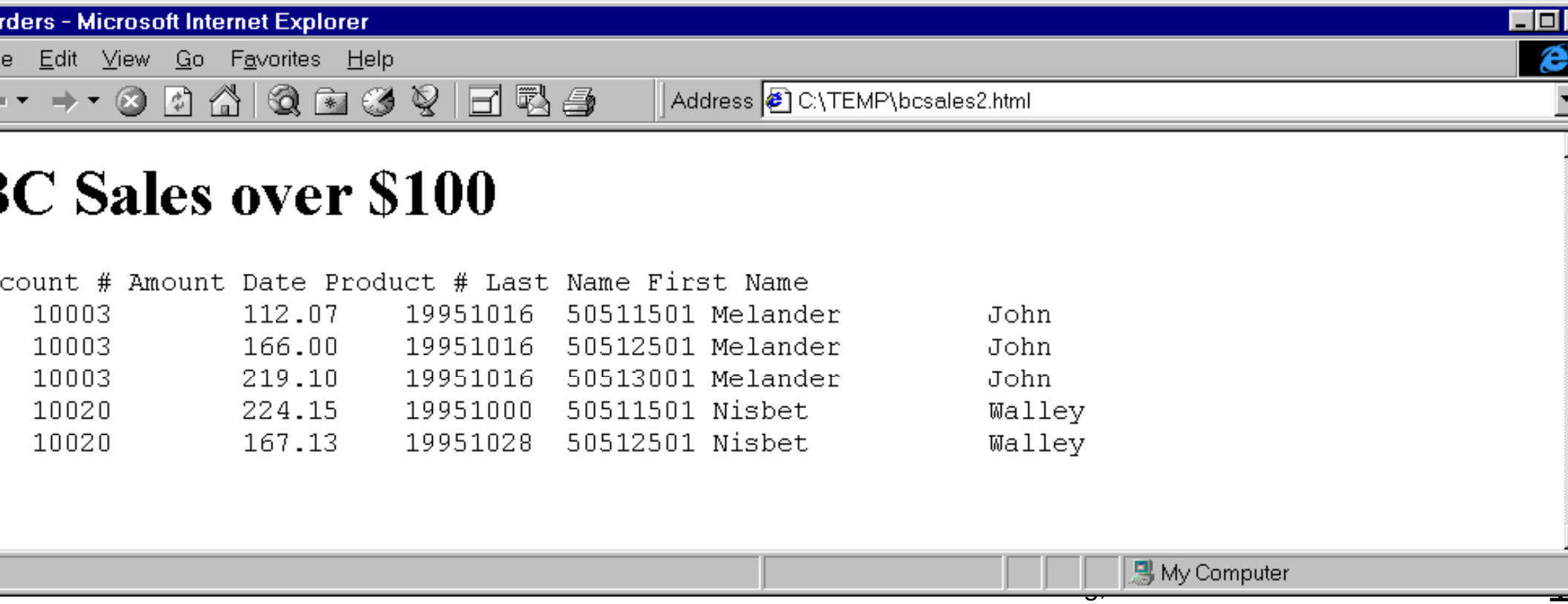
Table With Column Headings

- The table has one column per field, and one row per record



Listing-style Data

- Use the PREFORMATTED option instead of TABLE



Calling Command Interpreter Variables

- Command Interpreter Variables can be a very useful way of adding to the selection criteria for a SUPRTOOL task. CI Variables can also be nifty items to add into the headings of reports, etc.
  - However, SUPRTOOL could not replace these variables into a SUPRTOOL task until version 4.3 of SUPRTOOL.
  - Certain users did find a way to do this using the echo command as the following slide illustrates.

TIPS - USING THE ECHO COMMAND to FEED VARIABLES TO SUPRTOOL.

- !FILE MACORD=MACORD.MACSDATA
- !ECHO BASE MACORD,1,DOALL >> TEMP4040
- !ECHO GET FINANCIAL-ORDER >> TEMP4040
- !ECHO DEF A,BIG-STATUS:0,1 >> TEMP4040
- !ECHO DEF B,BIG-STATUS:1,1 >> TEMP4040
- !ECHO DEF C,BIG-STATUS:2,1 >> TEMP4040
- !ECHO DEF ORD,FULL-ORDER-NO[1],8 >> TEMP4040
- !ECHO IF (A = "N","P" AND B <> "V" AND C = "1","3") AND & >> TEMP4040
- !ECHO (DATE >= "!MONTHSTRT" AND DATE <= "!MONTHEND") AND & >> TEMP4040

TIPS - USING THE ECHO COMMAND to FEED VARIABLES TO SUPRTOOL.

- !ECHO (DIVISION = "01") >> TEMP4040
- !ECHO EXT ORD,'0000',FULL-ORDER-NO >> TEMP4040
- !ECHO OUT FO1,LINK >> TEMP4040
- !ECHO X >> TEMP4040
- !ECHO IN FO1 >> TEMP4040
- !ECHO DEF FON,1,12 >> TEMP4040
- !ECHO EXT FON,FULL-ORDER-NO >> TEMP4040
- !ECHO SORT FON >> TEMP4040
- !ECHO OUT FOTEMP,LINK >> TEMP4040
- !ECHO X >> TEMP4040
- !ECHO EXIT >> TEMP4040
- USE TEMP4040

Tip - inserting the current time

- There's no built-in Suprtool function for inserting the current time. However, you can use HP variables and command I/O redirection.

```
> define timestamp,1,8
> echo extract timestamp = "!HPTIMEF" > foo
> use foo
```

- This will insert an X8 field called "timestamp" into each output record. The timestamp contains the time the data was extracted.

CI VARIABLES

- A new set command has been added to turn on a Variable Substitution at the command line for Suprtool, Suprlink and STExport.
- The set command:
- >set VarSub On

- Turning this option on tells Suprtool, Suprlink and STExport to resolve any CI variables to be resolved on the command line.
  - NEW at 4.2.53

CI Variables

- After setting VARSUB ON
  - Get m-customer
  - Set varsub on
  - Ext “!hpsusan”
  - Ext cust-account
  - L s
  - X
- If statement can now call directly as in IF (A = "N","P" AND B <> "V" AND C = "1","3") AND & (DATE >= "!MONTHSTRT" AND DATE <= "!MONTHEND")

Updating a dataset from a table of values

- Lets say that you need to make mass changes and you would like to pull the data from an old dataset that is correct and update the second dataset.
- Or, you may need to make changes to a series of data that can be easily edited in and outside step and updated back into the database.
- SUPRTOOL does not currently allow for the use of table lookup followed by an update from the next piece of information in the table.
- However, using SUPRTOOL and a little used piece of the SUPRTOOL suite, called DBEDIT, you can do this type of updating.

Updating a dataset from a table of values

- The step of building a file containing a key item and the updated item can be another SUPRTOOL task, or any number of methods can be used to create the file.
- The primary key will be used for the update if you use the following:
  - `MODIFY dataset:item name`
- A secondary key can be used with the following syntax:
  - `MODIFY dataset:item name;key=keyfield`

Updating a dataset from a table of values

- Step 1 - Start by building a file and initializing it with the required Suprtool and Dbedit commands:
  - :build file1;rec=-60,,f,ascii;disc=1000
  - :file file1,old
  - :echo base store,1,writer >> *file1
  - :echo edit >> *file1
  - :echo modify m-customer:cust-status >> *file1
- After the Modify command place the name of the data set and the data item you wish to update.

Updating a dataset from a table of values

- Suprtool's List command can now be used to to load a file with the sequence of keyvalue and fieldvalue responses to Dbedit's prompts.
  - :file suprlist;disc=5000
  - :run suprtool.pub.robelle
  - > input custab {the "table file"}
  - > define acct,1,8
  - > define stat,9,2
  - > extract acct, stat
  - > list oneperline norec noname noskip dev disc
  - > xeq
  - IN=4,OUT=4.CPU-Sec=1.Wall-Sec=1.

Updating a dataset from a table of values

- Now append this data to the file that we constructed in the first step:
  - > in suprlist
  - > out file1,append
  - > xeq
  - Warning: >OUTPUT has different record size. IN=8, OUT=8. CPU-Sec=1. Wall-Sec=1.> exit

Updating a dataset from a table of values

- The file now contains the commands necessary to start Dbedit and to respond to the data prompts. We now need to end the prompt sequence with a blank line, then exit Dbedit and Suprtool:
  - :echo >> *file1 {a blank line}
  - :echo exit >> *file1 {exit Dbedit}
  - :echo exit >> *file1 {exit Suprtool}

Updating a dataset from a table of values

- Finally, run Suprtool using the file as the source of input commands:
  - :run suprtool.pub.robelle < file1
  - SUPRTOOL/iX/Copyright Robelle Solutions Technology Inc. 1981-1996.
  - > base store,1,writer
  - > edit ... .

Updating a dataset from a table of values

```
DBEDIT/iX/Copyright Robelle Consulting Ltd. 1984-2000.  Type H
  for help.
(Version 4.3)
Current: <default>
#modify m-customer:state-code
Modify within File: M-CUSTOMER
    CUST-ACCOUNT       >00010005_
Enter new values (or <return> to leave as is):
    CUST-ACCOUNT       =10005
      STATE-CODE       =AL
                        WA
CUST-ACCOUNT    = 10005          STATE-CODE      = WA
```

Data Validation using SUPRTOOL

- Another important capability SUPRTOOL provides is the ability to test and check data for proper form/type and range within the data.
- SUPRTOOL has a number of capabilities built into it that make it essential for any one concerned that their data is what it should be.
- Data validation capabilities within SUPRTOOL include ensuring that data is of numeric type, ensuring that a date is valid, ensuring that specific characters are in the record, subtotaling to ensure that totals are correct, etc.
- SUPRTOOL allows for pattern matching.
- SUPRTOOL can check dates to ensure they are valid.

Data Validation - Comparing fields

- You can compare one field to another

```
>if delivery-date = purchase-date
```

- You can compare a numeric field to a calculation

```
>if sales-total <> product-price * sales-qty
```

- You can compare a field to a constant

```
>if customer-status = "OK","DEAC"
```

Data Validation - Identifying a field as a date

- First use the ITEM command to identify a field as a date:

```
>item transaction-date,date,mmddyy
>item date-of-birth,date,phdate
>item disbursement-date,date,ccyymmdd
```

- Then use the IF command to select records:

```
>if transaction-date = $today and &
 date-of-birth < $date(1950/01/01) &
 and disbursement-date >= &
 $date(*+5/*/*)
```

Data Validation - Verify that dates are valid

- Use $INVALID to select records with invalid dates

```
>item purchase-date,date,yymmdd
>if $invalid(purchase-date)
>list standard title “Records with bad dates”
```

- Or use it to deselect invalid dates

```
>if not $invalid(purchase-date) and &
 purchase-date > $date(*/*-6/*)
```

Data Validation – Looking for duplicates

- Duplicates in the data can occur because of a number of human errors. Once they exist, how can you identify and get ride of them?
- Using SUPRTOOLS power DUP command duplicates can easily be identified and removed.
  - The command DUP NONE KEYS to remove duplicates.
    - A sort is a key to proper function.
  - The command DUP ONLY KEYS will identify only those records that are duplicates.

Data Validation – Checking the Pattern

- SUPRTOOL's powerful pattern matching capability to be of great help in checking data for the correct pattern of use. Is the supposed to be a numeric value in a specific position? Is there supposed to be a special character? These checks and many more are available using pattern matching.
- An example from ECOMETRY deals with the storage of email addresses:
  - Base macord,5,password
  - Get CUSTOMER-ADDL
  - Def EMAIL,ADDL-DATA[9],50
  - If email == "@&@@" (this gets you anything with an @ embedded)

Include a Record Count with Total

- List Standard Device XXX prints a simple report on device XXX
- Total $File $List prints the total of a field on the same List device
  - get dataset
  - ext id, zone, am ount
  - total am ount
  - total $file $list
  - list standard
  - device LP
  - xeq

Counting Records

- with two passes. The first pass adds a field with a value of 1, and the second pass totals that field, effectively providing a count of the number of records
  - get dataset
  - ext id, zone, amount
  - define count,1,4,int
  - ext count = 1
  - output tempfile,link
  - xeq

Counting Records

- input tempfile
- ext id\amount
- totalcount
- totalamount
- total$file $list
- list standard device LP
- xeq

Sub-Totaling with Suprtool

- >base store.demo,5,reader
- >get d-sales {open a dataset}
- >sort product-no {define a sort key}
- >duplicate none keys count totalsales-qty sales-total
- >out salessum ,link {Output to a link file}
- >xeq

The End

For more information on this presentation go to
www.kublerconsulting.com

Or

www.robelle.com

$DAYS FUNCTION

- `in   myfile`
- `def  cymd,1,8,display`
- `item cymd,date,ccyymmdd`
- `def  juldays,1,4,integer`
- `ext  cymd, juldays = $days(cymd)`
- `out  myfile2,link,temp`
- `xeq`
- `IN=3, OUT=3. CPU-Sec=1. Wall-Sec=1.`

$DAYS FUNCTION

- `in   myfile2`
- `def  askdate,1,2,integer`
- `ext  askdate = juldays - 2441255`
- `ext  cymd, juldays`
- `out  myfile3,link,temp`
- `x`
- `IN=3, OUT=3. CPU-Sec=1. Wall-Sec=1.`

$DAYS FUNCTION

```
in myfile3
item askdate,date,ask
def  newdate,1,8,display
ext  newdate=$stddate(askdate)
ext  cymd,juldays,askdate
list standard
x
```

$DAYS FUNCTION

```
NEWDATE    CYMD        JULDAYS      ASKDAT

19990729   19990729      2451389   10134
19730101   19730101      2441684     429
20000101   20000101      2451545   10290
IN=3, OUT=3. CPU-Sec=1. Wall-Sec=1.
```

Suprtool – Various Solutions

EXTRACTING Email addresses

- How to extract out e-mail addresses.

```
BASE MACORD,5,READALL (or 1,DOALL)
GET CUSTOMER-ADDL
DEFINE EMAIL,ADDL-DATA[9],50
IF EMAIL=="@&@@" (this part just gets you data with @ in
   it)
EXT EMAIL
OUTPUT EMAIL,ASCII (or whatever format you wish)
X
```