

Transforming TurboIMAGE Data for Eloquence, Oracle, and More

By Bob Green, Robelle
bgreen@robelle.com

Transforming Your TurboIMAGE Data - paper 1003

Practical tips on how to transform TurboIMAGE data so that it can be used in other databases:

- Image to Eloquence
- Cleaning Your Data
- Image to Oracle on HP-UX
- Image to SQL Server
- Image to mySQL
- Image to PostgreSQL

Special Cases:

- Combining Image Note Fields into Large SQL Text Fields

Presenter:

Bob Green is the founder of Robelle, an HP vendor since 1977.

Email: bgreen@robelle.com

Web: www.robelle.com/library/papers/transform.pdf

Transform to Eloquence/UX

- Migrate your data without change
- Use -T option on Schema Processor
- Eloquence handles all IMAGE datatypes, except j (COBOL integer) which is mapped to i
- Suprtool supports Eloquence, other MPE tools to follow

Eloquence is an IMAGE-like database that runs on HP-UX, LINUX and Windows NT/2K. Eloquence was created in 1989 to migrate HP 250/260s.

Eloquence home page: www.hp-eloquence.com

Marxmeier Software AG: www.marxmeier.com

Email contact at Marxmeier: info@marxmeier.com

Eloquence reseller/support: www.eloquence3000.com

Eloquence 7 maps datatype J to I because J is not supported by the database kernel. It doesn't really matter, as J is only a COBOL documentation thing. You would need to do all database edits with COBOL programs with PIC 9999 without sign to enforce this, since IMAGE doesn't enforce the J edits. R is mapped to E because HP-UX has only IEEE floating point, not the old floating-point format of the original 1970 HP 3000, but hopefully you are not using that anymore anyway! If you find that you are, then Suprtool can convert from Oldreal to IEEE real.

Robelle converted Suprtool for HP-UX to support Eloquence and completed the project in March 2002. Other vendors have announced plans to support Eloquence and HP has blessed it as a migration option for small to medium-sized databases. Besides being much cheaper than an Oracle solution, the performance is better than an IMAGE wrapper for Oracle.

TurboIMAGE to Eloquence

- Install the DBEXPORT utility on HP3000
- Export the database
- Create and import the database on Eloquence

Eloquence allows you migrate data without any conversion issues. See:

<http://hp-eloquence.com/hp3k/migration.html>

Eloquence provides the dbexport and dbimport utilities to unload a database to one or multiple files or respectively load a file into the database. The export files are text files which follow a simple syntax. An equivalent DBEXPORT utility is available for the HP3000 and can be used to unload your database. It can be downloaded from the Eloquence server.

Usage: DBEXPORT [-p password] database [set [...]] Specify the -help command line option to get information on usage.

When running from the MPE shell (CI) you need to enclose the arguments in quotes. : DBEXPORT "-p SECRET -v SAMPLEDB"

Transfer your schema files and the export files to the Eloquence system. When using ftp please make sure to use **binary** mode to transfer the export files. On Eloquence, run the schema and dbcreate utility and then use dbimport to fill the database: \$ dbimport database

On the Windows and Linux platform you should use the **-z roman8** commandline option of dbimport to make sure any national characters ("Umlaute") are converted properly. Please note: Unless you import a trivial volume of data you may want to consider to set the SyncMode = 0 in your eloqdb6.cfg configuration file during the dbimport.

Cleaning Your Data

- Imports can fail due to invisible characters
- Common: Linefeed, Tab, Escape
- Use Suprtool's Clean command

```
set CleanChar "."  
clean "^9", "^10"
```

It is very frustrating to have a data migration fail because your HP 3000 data fields contain characters, usually invisible, that make the database import fail on your target machine. Sometimes un-printable or extraneous characters get stored in files or databases that have no business being there. This may be some tab characters in an address field or perhaps an embedded carriage return or line-feed. What you need is a way to "clean" your data before moving it to a new home.

Suprtool has just such a feature, called the Clean Command. There are three things that Suprtool needs to know in order to "clean" a field: which characters to clean, what character to change the "bad" characters to, and also what field to clean.

The Clean command tells Suprtool what characters to look for in a given byte type field. For example:

```
clean "^9", "^10"
```

will tell Suprtool to replace the Tab character (Decimal 9) and the Line Feed (Decimal 10) to whatever the Clean character is set to. The Clean command takes both, decimal notation and the character itself, however, it is probably most convenient to use the Decimal notation for the characters that you wish to clean. The Decimal notation is indicated by the "^" character.

Setting the Clean Character - By default, Suprtool replaces any of the characters specified in the Clean command with a space. You can specify what character to use to replace any of the characters that qualify with the Set CleanChar command:

Example of Cleaning

```
>base mydb,1,; get customer
>clean "^9","^10","^0","^7"
>set cleanchar " "
>update
>ext address(1) = $clean(address(1))
>ext address(2) = $clean(address(2))
>ext address(3) = $clean(address(3))
>xeq
```

Cleaning a Field - In the If and Extract commands, you call the Clean function the same way you normally use other functions. For example:

```
ext address1=$clean(address1)
```

This shows how to clean the field address1. The target field does not have to be the same as the source field.

```
def new-address,1,30
ext new-address=$clean(address1)
```

An example of cleaning your data – the example on the slide looks at the three instances of address and replaces the tab, linefeed, null and bell with a space.

STExport can also clean, but can also automatically clean **all** the byte type fields for a given SD file.

```
$in mysdfile {the default CleanChar is " "}
$clean "^9","^10","^0","^7"
$out myexport
$xeq
```

Advanced Clean Syntax – specify the decimal value of the character and specify a range of characters:

```
Clean "^0:^31","^240:^255"
```

TurboIMAGE Datatypes

- *Subitemcount Datatype Subitemlength*
- X, U - character data
- Z - zoned numeric bytes, overpunch
- I, J, K - integers
- E - floating point (R is deprecated)
- P - packed decimal

A typical data item definition is J2, where J is the Datatype and 2 is the SubItemLength, or 5X8 where 5 is the Subitemcount and 8 is the SubitemLength. The type designators are E, I, J, K, P, R, U, X, and Z.

E = ieee floating point. sub-item length is in halfwords (R is old floating point, you should not be using that).

I = signed integer, sub-item length is in halfwords

J = signed integer, but conforms to COBOL standards (i.e. S9999 should not have any values greater than 9999). sub-item length is in halfwords

K = unsigned integer, no negative value. 1 halfword = 0-65K, 2 halfwords= 0-2 Billion, sub-item length is in halfwords

P = packed decimal, sub-item length is in nibbles, 2 to 28, with one digit used for the sign (note: TurboIMAGE will let you create a P48 or even larger, but COBOL will not process it).

U = uppercase ASCII chars, **IMAGE does not enforce uppercase!**

X = any ASCII characters, sub-item length is in bytes.

Z = zoned decimal number. sub-item length is in bytes. Overpunch sign.

Language Compatibility

- COBOL: i1 i2 i4 x u z p
- FORTRAN: i1-i2 e2 e4 x u
- Powerhouse: i1-i4 e2 e4 x u z p
- j1,j2,j4 are odd, think of them as i's

Although TurboIMAGE does not place any restrictions on the reasonableness of item datatypes (i.e., you can define J25 if you wish) and does not validate values before inserting them into the database, most TurboIMAGE databases use only the data types that can be processed by the programming languages on the HP 3000.

I3 is a 6 byte integer, supported only in Powerhouse and Suprtool. I4 is an 8-byte integer that is supported in COBOL, Suprtool and Powerhouse only. Expect possible migration problems with these fields.

J1 is 2-byte integer for COBOL PIC S9999; I.e. it should not have any values greater than 9999 or less than -9999, but IMAGE does not enforce this, so you can treat it as I1 in Fortran and Powerhouse. Same for J2 and J4. J3 will probably work in Powerhouse.

K1 Logical, 2 bytes, unsigned integer, define as Logical in Fortran, not supported in COBOL. Suprtool and Powerhouse support K2 and K3.

Zn Zoned-Decimal, n bytes, s(n) Display in COBOL, overpunched

Pn Packed-Decimal, n/2 bytes, s9(n-1) Comp-3 in COBOL, not supported in Fortran. Maximum N in HP COBOL is 19 (18 digits plus a sign).

TurboIMAGE Data to Oracle

- Internal datatypes:
 Number, Char, Varchar2, Date
- External datatypes - how Oracle delivers the data to a program:
 Integer, Char, Floating-point, Packed-decimal

Oracle has internal datatypes and external datatypes. The internal datatypes are the format that Oracle actually stores the data in. However, these internal datatypes, might not be recognized by any of the standard programming languages. For example, Number is a variable-length format, with one byte used to store the exponent and up to 20 bytes to store the mantissa. You cannot process such a number directly in Fortran, COBOL, or C. Therefore, Oracle also has external datatypes, which are the formats that Oracle is willing to transform into for calling code written in languages that include C, COBOL, Fortran and Java.

Oracle 7 Internal Datatypes:

CHAR for X/U fields of up to 255 characters.

VARCHAR2 for X/U fields up to 2000 characters.

NUMBER for I/J/K/P/Z fields, up to 38 digits.

DATE for any field that contained a date value, such as SHIP-DATE X8, or DUE-DATE J2, or ORDER-DATE Z8. The Oracle date can also hold the time.

Oracle 8i and 9i Internal Datatypes:

CHAR can hold up to 2000 characters, VARCHAR2 can hold up to 4000 characters. So it appears that converting our datatypes to Oracle is straightforward.

Transforming Decimal Data

- Export numbers as char, with decimal
- Define Oracle field as NUMBER (x,y), where x is the total number of digits and y is the scale factor.
- SQL interface for COBOL can extract the NUMERIC field as Packed Dec so you don't have to change your Copylib

When exporting Image datasets to other sources, one of the common transfer file formats used is the "comma separated values" or CSV format. This is generally a flat file with one record per line, quotes around fields and commas between fields.

```
"cust-id", "comment"  
"12A", "want web delivery"
```

Suprtool and other tools have options to generate CSV files. You will find Suprtool's Item command handy for defining the decimal place in your TurboIMAGE fields.

You will want the export file to have an explicit decimal place "." in numeric values where appropriate, since Oracle understands decimal places and remembers them. With a Display format, you only need to include an explicit decimal point as in 1234.89. STExport does that automatically when properly configured (Item command in Suprtool and Decimal Period in STExport).

SQL*Loader takes care of the alignment with the column definition e.g. NUMBER(12,2). If the file contains more decimals than the column definition, SQL*Loader rounds it up. For example, if you try to load 1234.5398 into NUMBER(12,2), the table will contain 1234.54. Negative values must have a leading sign (Sign Leading in STExport).

STExport for SQL*Loader

```

!RUN STEXPORT.PUB.ROBELLE
IN DMRTABHM
ZERO LEADING
QUOTE NONE
COLUMNS FIXED
SIGN TRAILING
OUTPUT ABHMDATA
EXIT

```

If you used Suprtool to extract a dataset into a file, you can use Suprtool's STExport to prepare the data in a format that the Oracle SQL*Loader will accept. STExport lets you define the format of numeric data, including leading zeros and the position of the sign.

The resulting files are sent using the HP 3000's FTP client to the target computer where the Oracle database resides.

```

!COMMENT *** FTP OUTPUT FILES TO DATAMART ***
!RUN FTP.ARPA.SYS
open 123.456.789.012
user <<login string and password>>
ascii
exitOnError
cd /isdmdata
cd macsdata_in
put ABHMDATA.pub.hsmacs ABHMDATA.txt
dir
quit

```

In SQL*Loader, use the Load Data command with the `FIELDS TERMINATED BY ``,`` clause to insert the data into your table. This is just one example, with fixed-length fields. STExport and SQL*Loader also have options to use variable-length fields.

Oracle Can't Deliver as I4

- COBOL program can retrieve Number values in Integer (i1 and i2), Packed Decimal, or Zone Decimal format
- Retrieve as Packed field for COBOL instead
- Change COBOL PIC to COMP-3

What if you have a field that looks like this in COBOL on MPE?

```
05 EXT-PRICE S9(10)V9(2) COMP.
```

In TurboIMAGE, this would be a J4, which converts to Number as the Oracle internal datatype.

But Oracle does not have an external datatype of 64-bit integer!

So you will have to use Packed-Decimal as the external datatype.

Which means changing the COBOL definition to

```
05 EXT-PRICE S9(10)V9(2) COMP-3.
```

Now your programs are different and any files you create with this data will probably need new data definitions.

Note: If your internal Number field has decimal positions, then you will need to always convert it to an external Packed datatype (COMP-3). If you tried to convert it to an external Integer datatype, you would lose the decimal places. This is true regardless of the size of the field.

This does not sound too bad, unless you want to keep the same source on MPE and UNIX. Or if you have hundreds of tasks that may process this data!

TurboIMAGE to SQL Server

- Tinyint(byte), Smallint(i1), Integer(i2) and Bigint (quad, i4)
- Number (precision,scale)
- Float(n) where N is the number of bits in the mantissa (<25 bits is 4 bytes Real, 25> is 8 byte Long).

For integer values without a decimal place, use one of the Integer datatypes: Tinyint (values 0 to 255), Smallint (-32,768 to +32,767), Integer (-2B to + 2B), Bigint (for very large values, up to 18 digits; introduced in SQL Server 2000 -- ensure that your Windows COBOL compiler supports Bigint).

NUM or NUMBER or DEC = numbers with decimal places. You specify a precision and scale for the values. Precision is the maximum total digits in the values, with 38 the largest allowed by SQL Server. Scale is the number of places to the right of the decimal. The maximum number of digits that can be placed to the left of the decimal is precision-scale. For example, DEC(7,2) means the same as S9(5)V9(2) in COBOL. NUMERIC or FLOAT is the datatype for any value with a decimal place. NUMERIC in SQL Server is much like NUMERIC in Oracle, although it does not have the odd "negative scale factors" of Oracle (scale factor-3 in Oracle actually multiplies the value by 1000!).

FLOAT(n) = approximate numeric values in floating point format. Supported in 4 byte and 8 byte formats. A floating point number has an exponent and a mantissa. FLOAT(n) specifies number of bits for the mantissa, which can be up to 53. 1 through 24 specify a single precision real (4 bytes) and 25 through 53 specify Double Precision (8 bytes). Same as e2 and e4 in TurboIMAGE.

Other SQL Server datatypes that you will find useful are MONEY and DATETIME.

What about Compound Items?

- TurboIMAGE has 5x10 for an array of five elements, each with 10 characters.
- This is called a compound item
- Not supported in SQL databases
- Convert each element to a separate column: address1, address2, address3...

```

M-CUSTOMER           Master                               Set# 1
Entry:                Offset
CITY                  X12      1
CREDIT-RATING        J2       13 <<0.00>>
CUST-ACCOUNT         Z8       17
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: 12  Bytes: 110

```

The repeated item STREET-ADDRESS (2X25) converts into ADDRESS1 and ADDRESS2 in SQL, since SQL does not have repeated data items.

```

get m-customer; define address1,street-address(1)
define address2,street-address(2);
extract address1, address2

```

CUST-ACCOUNT is only Z for hashing, so convert it into INT.

Since CREDIT-RATING values have 2 decimal places, convert into the NUMBER datatype, which understands decimal places.

Find a COBOL Precompiler

- MS SQL Server has no COBOL precompiler
- But AcuCOBOL has one
- Microfocus Cobol says get precompiler from your database vendor
- New languages have interfaces to most databases: C++, Java, Perl, Php, etc.

With Oracle, the package comes with pre-processors for Fortran and COBOL, but SQL Server only provides a pre-compiler for C++. What if you want to do SQL Server functions in your COBOL program? You need to look to your compiler vendor.

For example, AcuSQL has an Embedded SQL (ESQL) precompiler that lets you embed standard SQL directly into ACUCOBOL program.

www.acucorp.com/Solutions/access3.html

www.acucorp.com/Solutions/acusql.html

However MicroFocus COBOL says "COBOL precompilers for the different databases should be obtained from the appropriate database vendor."

Fujitsu COBOL has Access to SQL Server via .the ODBC interface..

I looked for a FORTRAN precompiler for SQL Server, but did not find one, so that problem is left to the reader. Keep in mind the database you select may not have precompilers for all your programming languages.

TurboIMAGE to MySQL

- MySQL is an open source database
- Commonly used as a web backend
- Simple, fast, but limited
- www.mysql.com
- As an experiment we replicated an IMAGE database in MySQL

MySQL is an Open Source database that is commonly used as the backend database server for many Web applications on Linux and Unix platforms as well as Windows machines.

The source and or binaries can be obtained from www.mysql.com or www.sourceforge.net and many other download mirrors around the globe.

As an experiment we built a MySQL database that looked like an Image database, building a simple Master dataset and a single detail dataset. The byte type fields in Image were created as char fields. The I2 or J2 fields were created as int fields.

Having done this, we extracted data from the HP 3000 database using Suprtool and then used default STExport settings to output a file that was comma delimited, with each text field enclosed in quotes.

We then attempted to import the comma-delimited file into MySQL.

Importing into MySQL

```
LOAD DATA 'file_name.txt'
  INTO TABLE tbl_name
  [FIELDS
    [TERMINATED BY '\t']
    [ [OPTIONALLY] ENCLOSED BY '"']
    [ESCAPED BY '\\' ] ]
  [LINES TERMINATED BY '\n']
```

In investigating how to import data into MySQL, we first tried the `mysqlImport` program, but it didn't seem as robust and we could not figure out how to tell it what delimiters to use. In looking at the documentation, we thought that the `LOAD_FILE` command might work, but further investigation showed that this command opens the file and returns the contents as a string. This feature is only used by Text and Blob columns of MySQL.

We finally had success with the `LOAD_DATA` statement. If you don't specify a `FIELDS` clause, the command acts as follows: Look for line boundaries at newlines; break lines into fields at tabs; do not expect fields to be enclosed within any quoting characters; interpret occurrences of tab, newline, or `\'` preceded by `\'` as literal characters that are part of field values.

If you specify a `FIELDS` clause you can change the delimiter from Tab to comma (or another character).

We had trouble with the comma delimiter, because our actual data contained commas, so we used question mark as the field delimiter (or we could have used Tab). In order to import fields exported with `"?"` as the delimiter, we used `fields terminated by '?'` in the `LOAD_DATA` command.

Notice the Escaped By clause which avoids load errors by putting an Escape character (usually `\`) in front of every delimiter character in your data: `"`, `,` and `\`. The Escape option in `STExport` will generate this format.

TurboIMAGE to PostgreSQL

- Full-feature SQL database
- Use Copy command to load from file
- Default separator is tab (\t)
- Supports quad integers I4 (int8)

PostgreSQL is an Object-Relational database management system that supports almost all SQL constructs, including subselects, transactions, and user-defined types and functions. It is free to download, use, and modify, and it runs under many OS types, including Unix and Windows.

For an experiment, we created an address table and loaded it from a Comma Delimited data file. The next step was to populate the table with the data from the text file. This was easy to do by using the COPY command, which loads large amounts of data (either character or Binary) from flat-text files. You tell COPY which file to load by specifying FROM '/directory/file'. By default, COPY uses a tab ("\t") character as a delimiter between fields, I had to change this behavior by including USING DELIMITERS ',', '

The COPY command, however, has a few potential pitfalls. Such as if you don't have enough columns in the file, you will get an error, but if you have too many columns you will get a warning only and the extra columns are ignored. Also remember that COPY is executed as a transaction, meaning that a single error in the data causes an undo of the entire import operation. As always it is good practice to read over the intricacies of the COPY command in the PostgreSQL help docs (<http://www.postgresql.org/docs/>).

To avoid errors in your import data, use the new Clean option in Suprtool and STExport -- it removes LF characters and other delimiters from text fields.

Migrating Dates

- Most databases allow M/D/Y or D/M/Y
04/28/2002 or 28/04/2002
- ISO standard format is yyyyymmdd
20020428
- Export with this format to avoid import problems, since the month is unambiguous

Most SQL databases allow you to import dates in either Month/Day/Year format or Day/Month/Year, but there are still configuration commands that you must get correct.

The ISO standard format for dates is actually
YYYYMMDD (20020428)

If you export date fields in this format, you should have minimal problems loading into your SQL database.

On the other hand, if you have an HP 3000 date field in the form

MMDDYY 042802

and export it in that format, you will need to configure your import carefully to ensure that your SQL database contains the correct date.

Invalid and Missing Dates

- Suprtool: `if $invalid(shipdate)`
- If you have blank SHIPDATES, make the SQL field Nullable
 - `"Bob",20030323,"101 Main St"`
 - `"Bill",,"235 Side St"`
 - `"Sue",,"52 High St"`
- STExport: Date Invalid Null

Unlike TurboIMAGE, where date values are stored in generic data types such as X (character) or I (integer), SQL databases have a DATE type specifically for date values. Blanks and zero are not valid dates! Any truly invalid date values should be fixed before exporting to another database. You can find the invalid dates easily in Suprtool using the **\$Invalid** function.

```
>base store.demo; get d-sales
>item deliv-date,date,ccyyymmdd
>if $invalid(deliv-date)or deliv-date<19000101
>out baddates,link; list standard; xeq
```

In TurboIMAGE, applications usually initialize SHIPDATE-type fields to Blanks or Zeroes to indicate that the value is not yet know. In SQL databases, you need to define the column as **NULLABLE**. And when you load the CSV file, you can use two commas with nothing between to indicate a Null date value.

You can use Suprtool and STExport to export TurboIMAGE date values in a form that your SQL database will accept. By default, STExport formats invalid dates as asterisks (****). But, there is a special option called **Date Invalid Null** that converts an invalid date into a zero-length field (you must also specify variable Columns).

```
$input dsales; date ccyyymmdd; columns none
$date invalid null
$quote double; delimiter comma
$output dsexprt $xeq
```

Export Multi-Line Notes

- Combining multiple Image note records into one large SQL text field
- Use a Perl program
- Full details at:

<http://www.robelle.com/tips/st-export-notes.html>

There is one type of field that cannot easily be exported and loaded into SQL: Notes fields, which span multiple records even though they are logically one field. They usually look like the following:

```
cust-id  seq-num  comment
12A      001      want web delivery but
12A      002      limited by bandwidth,
12A      003      so use FTP.
88X      001      Send doc in PDF format.
```

We want to merge the related records into a single record. For example:

```
"12A", "want web delivery but limited by bandwidth,
so use FTP."
```

```
"88X", "Send doc in PDF format."
```

Although Suprtool cannot produce this format directly, it can front-end the database extract portion, and let a straight-forward Perl script do the merging.

Export Notes, Tab Delimited

```
Get Notes
Sort cust-id
Sort seq-num
Extract cust-id, ^i, comment
{tab delimited}
Out notefile
Xeq
```

Although Suprtool cannot produce this format directly, it can front-end the database extract portion, and let a straight-forward Perl script do the merging. To learn more about Perl, read our "Introduction to Perl" at www.robelle.com/tips/perl.html

1. First use Suprtool to dump the Notes dataset, sorted by customer number and sequence number:
2. Send the resulting notefile to your other OS/directory space where Perl is available. FTP is a reasonable choice.
3. Use the Perl script on the following page.

Run the Perl Script

- Run the Perl script against your notefile to produce the CSV file.

```
perl merge.pl <notefile
>note.csv
```

- You now have one big text record per sequence number

```
$in_field_sep = "\t";
$quote = "'";
$salt_quote = "'";
$rec_sep = "\n";
$field_sep = ",";
$note_term = " ";
$keycount = 0;
$prev = "";
while ($line = <STDIN>) {
    $line =~ /([^\t]+)\t(.*)/;
    $key = $1;
    $text = $2;
    $text =~ s/$quote/$salt_quote/g;
    if ($key ne $prev) {
        if ($keycount>0) { # close previous quote
            print $quote . $rec_sep;
        }
        print $quote . $key . $quote . $field_sep . $quote;
        $keycount++;
    }
    print $text . $note_term;
    $prev = $key;
}
print $quote . $rec_sep;
```

Learn More

- Migration library on our web site:
suprtool.com/move
- Email me at bgreen@robelle.com
- Primary Robelle site: www.robelle.com
- Some useful web links:
www.robelle.com/books - "HP 3000 Evolution"
www.eloquence3000.com - Eloquence database

HP 3000 Evolution covers migration, homesteading, and system tune up.

