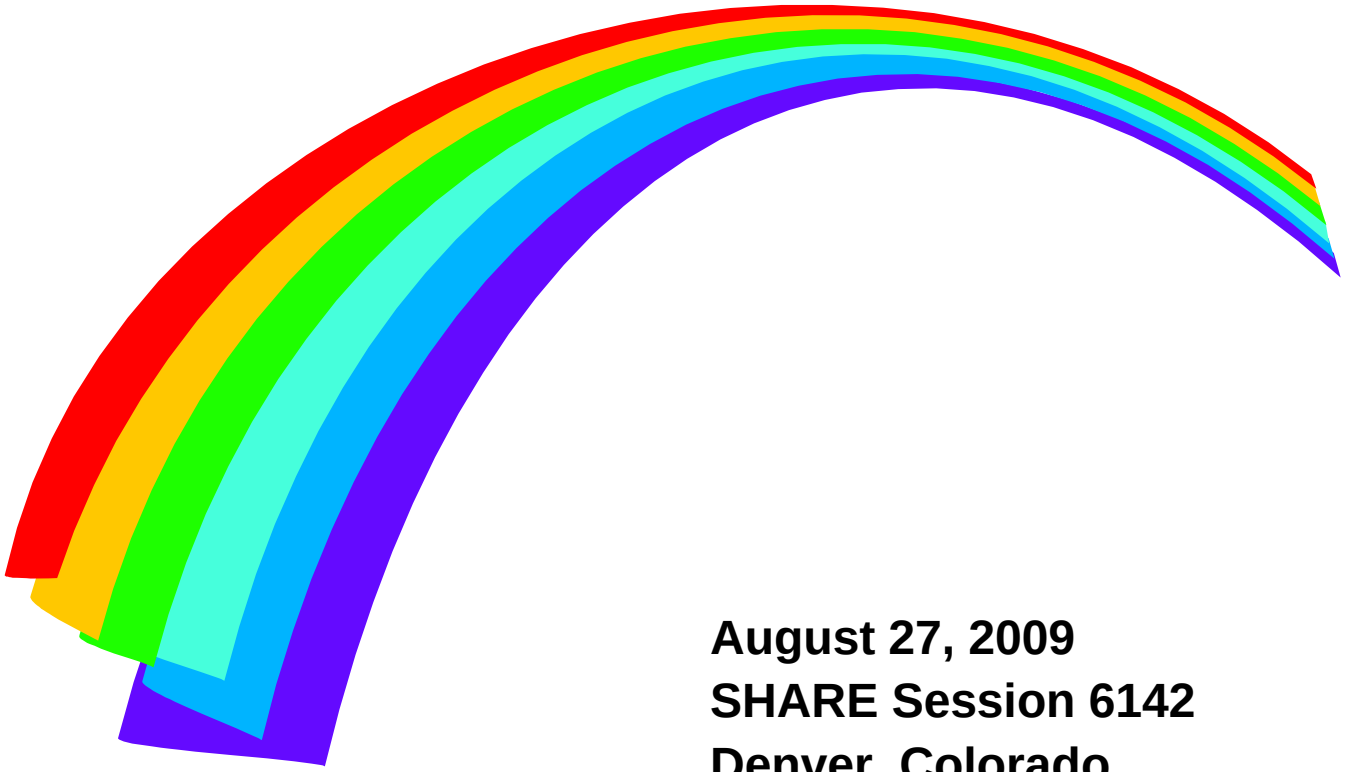
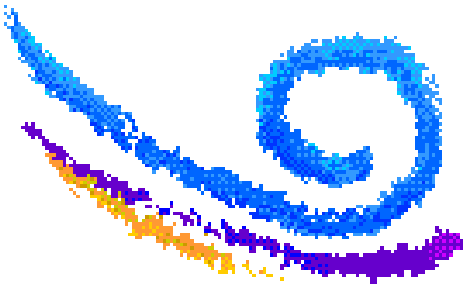
DB2 LOBs - A Practical Application



August 27, 2009
SHARE Session 6142
Denver, Colorado

Presented by Hunter Cobb for The Trainer's Friend, Inc.
Written by Steve Comstock for The Trainer's Friend, Inc.



Each new release of DB2 introduces new features

- Many are clearly beneficial, but others may not seem relevant to your shop
  - So you may be in no hurry to keep current
    - Who needs the extra hassle and work?

- Today I would like to explore one of the areas that may seem esoteric but actually has great potential:

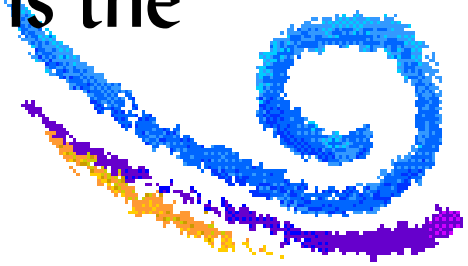
  - LOBs (Large OBjects) - a way to store and present media and documents

BLOBs, CLOBs, and DBCLOBs

- Large OBjects (LOBs) come in three flavors, but they are all ways to store unstructured data
  - BLOBs - Binary Large OBjects include photos, movies, sound, signatures
  - CLOBs - Character Large OBjects include documents such as policies, contracts, specifications
  - DBCLOBs - Double Byte Character Large OBjects are CLOBs composed of CJK (Chinese, Japanese, Korean) characters encoded in DBCS or UTF-16

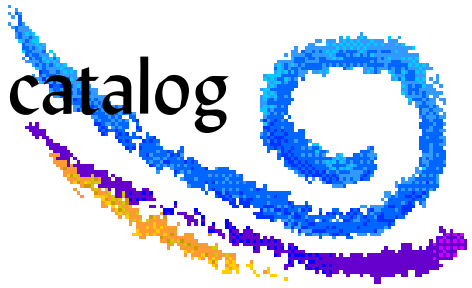
But what use are LOBs?

- If DB2 is your backend data server to the Web, you can store product images in BLOBs to produce online catalogs including pictures, for example
- You can store document chunks in CLOBs, then combine them to build tailored contracts and policies
- Data items > 32K in size must be stored in LOBs, although you can store smaller items; 2GB is the maximum size per LOB



Using LOBs - an Example

- Suppose we have an inventory database that contains the following fields (plus more, but we'll ignore them):
  - PartNo - a nine-byte field of the form "Part*nnnnn*"
  - Description - a 30 byte field
  - QOH (Quantity on Hand) - an integer
  - Picture - an image of the item, for our online catalog



Using LOBs - an Example, 2

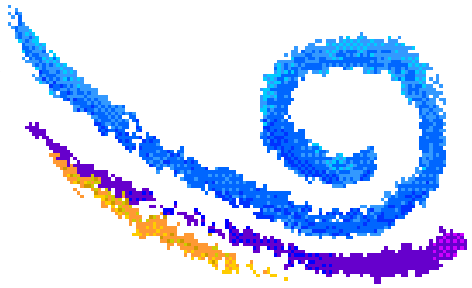
- The first thing we have to do is define the table, and it might be something like this (using SPUFI):

```
create table Items
  (PartNo       char(9)   not null,
   Description  char(30)  not null,
   QOH          int       not null,
   ItemRowID    rowid     not null,
   Picture      blob(4m),
constraint EnsurePartNo primary key (PartNo)  );


create unique index PartNoIndex on Items(PartNo);
```

Notes:

*** A table with any kind of LOB must have a ROWID column**

*** Every table should have a primary key, and this requires a unique index**

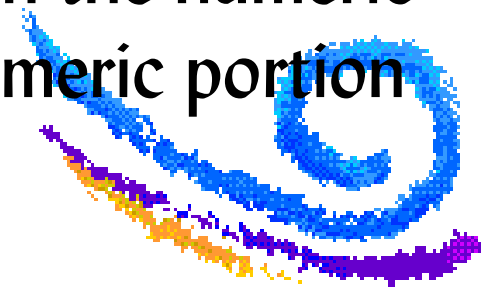


Using LOBs - an Example, 3

- Each LOB column requires a LOB table space, auxiliary table, and auxiliary table index
- DB2 Version 9.1 helps: if you issue CREATE TABLE that includes one or more LOB columns, and do not specify a tablespace, DB2 automatically creates the LOB table space, auxiliary table, and auxiliary table index
  - Perhaps something DBAs would rather do explicitly, but at least it's easy to try things

Using LOBs - an Example, 4

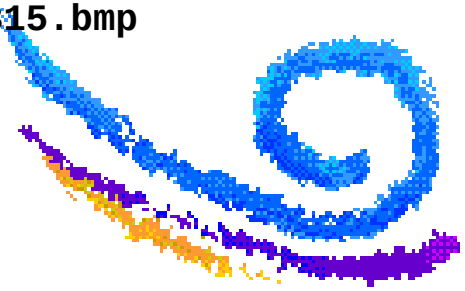
- Of course, we need to get our images into our database LOB column in a standard graphics format such as .bmp, .jpg, .png, .gif, and so on
  - Perhaps simply a digital camera shot, perhaps something more professional, but it needs to end up as an image file on our workstation
  - We named the images "part*nnnnn*.bmp", with the numeric portion of the name corresponding to the numeric portion of the item's part number



Using LOBs - an Example, 5

- Next, upload the image files to our z/OS system, preferably into our HFS or zFS
  - We created a directory /u/scomsto/blobs and uploaded all the image files into that directory (as binary, of course), using ftp; result something like this:

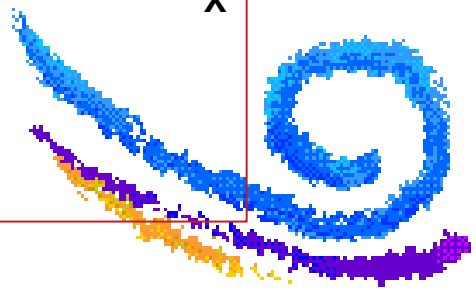
```
/u/scomsto/blobs
-rw-r--r--   1 SCOMSTO  STUDENT   129558 Oct  2  2008 part00105.bmp
-rw-r--r--   1 SCOMSTO  STUDENT   130726 Oct  2  2008 part00240.bmp
-rw-r--r--   1 SCOMSTO  STUDENT   308278 Oct  2  2008 part00273.bmp
-rw-r--r--   1 SCOMSTO  STUDENT   128086 Oct  2  2008 part00315.bmp
.
.
.
```



Using LOBs - an Example, 6

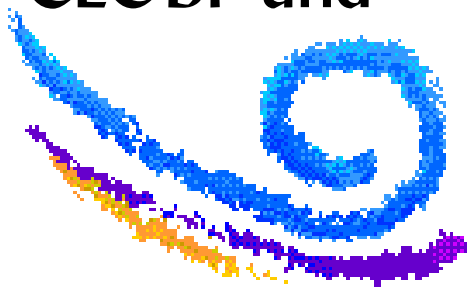
- Next, we have to populate our table
  - We used the LOAD DB2 utility, something like this:

```
//SCOMSTO  JOB ...
//STEP1 EXEC DSNUPROC,UID=SCOMSTO.LOADLOB
//SYSUT1   DD  UNIT=SYSALLDA,SPACE=(TRK,(20,20))
//SORTOUT  DD  UNIT=SYSALLDA,SPACE=(TRK,(20,20))
//SYSIN    DD  *
LOAD DATA RESUME NO REPLACE CONTINUEIF(72:72)='X' INTO TABLE ITEMS
    (PARTNO       POSITION (01) CHAR(9),
     DESCRIPTION  POSITION (10) CHAR(30),
     QOH          POSITION (42) INTEGER EXTERNAL(2),
     PICTURE      POSITION (80) CHAR(30) BLOBF )
//SYSREC DD *
Part00105Keys to the Kingdome          10                              X
/u/scomsto/blobs/part00105.bmp
Part00240Overtime Hours                17                              X
/u/scomsto/blobs/part00240.bmp
.
.
.
```



Using LOBs - an Example, 7

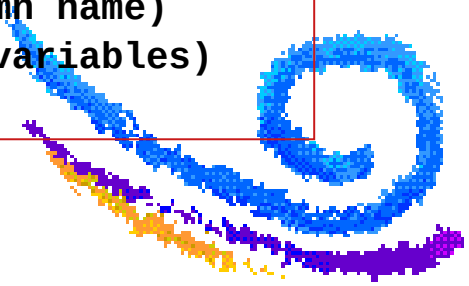
- DB2 Version 9.1 helps: the LOAD utility can now use HFS file names (can also be sequential files or members of PDS or PDSE) to hold names of files containing BLOB data
  - (See the BLOBF data type in the statements on the previous page)
  - Also works for CLOBs and DBCLOBs (that is, CLOBF and DBCLOBF can be used)



Using LOBs - an Example, 8

- At this point, we wanted to prepare for writing code against this table, so we used DCLGEN, like this:

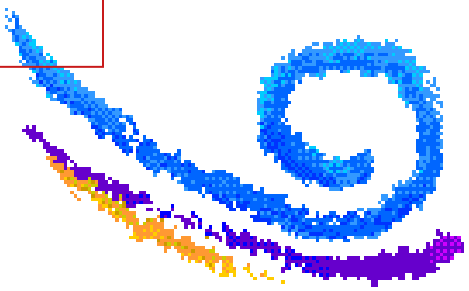
```
Enter table name for which declarations are required:
 1  SOURCE TABLE NAME ===> items
 2  TABLE OWNER ..... ===> SCOMSTO
 3  AT LOCATION ..... ===>                                  (Optional)
Enter destination data set:          (Can be sequential or partitioned)
 4  DATA SET NAME ... ===> TR.COBOL(ITEMS)
 5  DATA SET PASSWORD ===>           (If password protected)
Enter options as desired:
 6  ACTION .......... ===> REPLACE   (ADD new or REPLACE old declaration)
 7  COLUMN LABEL .... ===> NO        (Enter YES for column label)
 8  STRUCTURE NAME .. ===>                                  (Optional)
 9  FIELD NAME PREFIX ===> host-                            (Optional)
10  DELIMIT DBCS .... ===> YES       (Enter YES to delimit DBCS identifiers)
11  COLUMN SUFFIX ... ===> YES       (Enter YES to append column name)
12  INDICATOR VARS .. ===> NO        (Enter YES for indicator variables)
13  RIGHT MARGIN .... ===> 72        (Enter 72 or 80)
```



Using LOBs - an Example, 9

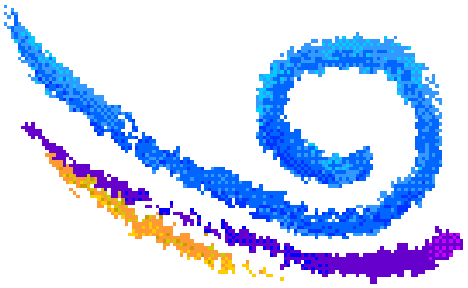
- The resulting statements, after a little tweaking:

```
     EXEC SQL DECLARE SCOMSTO.ITEMS TABLE
     ( PARTNO                          CHAR(9) NOT NULL,
       DESCRIPTION                     CHAR(30) NOT NULL,
       QOH                             INTEGER NOT NULL,
       ITEMROWID                       ROWID NOT NULL,
       PICTURE                         BLOB(4194304)
     ) END-EXEC.
************************************************************
* COBOL DECLARATION FOR TABLE SCOMSTO.ITEMS                 *
************************************************************
 01  DCLITEMS.
     10 HOST-PARTNO           PIC X(9).
     10 HOST-DESCRIPTION      PIC X(30).
     10 HOST-QOH              PIC S9(9) USAGE COMP.
     10 HOST-ITEMROWID        USAGE SQL TYPE IS ROWID.
     10 HOST-PICTURE          USAGE SQL TYPE IS BLOB-LOCATOR.
```



Using LOBs - an Example, 10

- Our goal was to serve web pages that included these BLOBs on demand; this required:
  - Designing and coding a web page where the user could request information about an item
  - Coding a COBOL program to run as a CGI that would accept the request and return a web page containing the image of the requested item



Using LOBs - an Example, 1 1

- The page we designed looked like this:

Welcome to The Trainer's Fiend Novelty Products Division

"We provide the finest products you can't find anywhere else!"

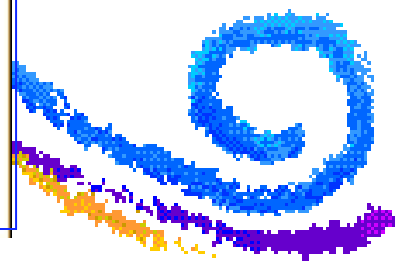
Product Catalog

Here is the list of items in our catalog; Select the one item you want to see by clicking on it.

Descriptions

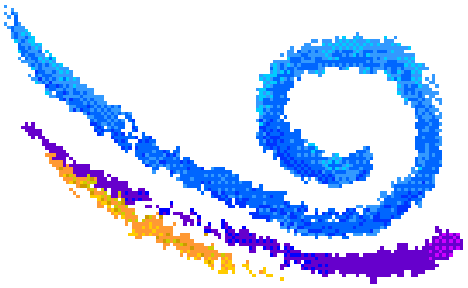
- Keys to the Kingdome
- Overtime Hours
- Anodized Tennis Ball
- Colo. Creeping Crud
- Plastic Food

Submit request | Clear request



Using LOBs - an Example, 12

- The HTML to create that page is available in the supplementary materials
  - As are the bmp files, the stylesheet, and the other supporting code and control data
  - This is not the place to explore HTML (some other time, perhaps)



Using LOBs - an Example, 13

- However, here are few notes regarding how we set up the files we provide in our environment - you need to make your own adjustments:

  - **Our HTTP configuration sets "public_html" as our starter directory for each user's HTML; we placed our HTML (and stylesheet and animated gif) in the subdirectory /u/scomsto/public_html/db2work, to keep other work separate, so to start, we pointed our browser at:**

    //http://www.ttfi.biz/~scomsto/db2work/getitem1.html

  - **The animated GIF logo we found at http://www.animationlibrary.com/**

  - **In our example, we hardcoded the entries in the list box; in a real world example, you would generate this page through another CGI, so you always create a current list**

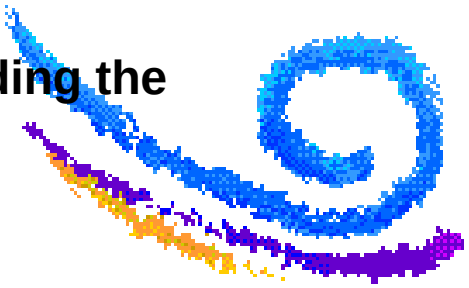
Using LOBs - an Example, 14

- More notes on the supplementary files:

  - **Our HTTP configuration maps /SCOMSTO/* to /u/scomsto/CGI/*, where I place my own CGIs**

  - **The logic is: whenever an item in the drop down list is clicked, the variable "ano" is set to contain the corresponding part number; when the "submit" button is clicked, the CGI named "coblob1" in my CGI directory will get invoked, being passed the string: ano=Part*nnnnn***

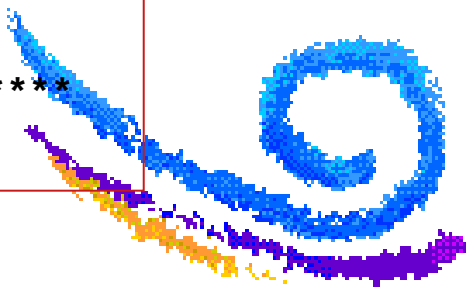
  - **Program "coblob1" is a COBOL program that runs as a CGI:**
    *** Obtains the value in the "ano" variable supplied from a Web form and assigns it to the trans-partno variable**
    *** Uses trans-partno as a host variable in the WHERE clause of a SQL SELECT statement to retrieve the assocated row**
    *** Generates an HTML page to display the information, including the BLOB that is the picture of the corresponding item**



Using LOBs - an Example, 15

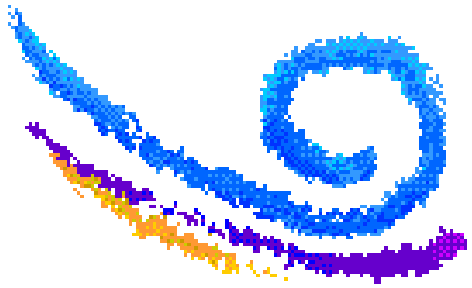
- Some highlights from the COBOL CGI (just looking at the DB2 part, especially handling the BLOBs):

```
     EXEC SQL INCLUDE SQLCA END-EXEC.
     EXEC SQL DECLARE SCOMSTO.ITEMS TABLE
     ( PARTNO                          CHAR(9) NOT NULL,
       DESCRIPTION                     CHAR(30) NOT NULL,
       QOH                             INTEGER NOT NULL,
       ITEMROWID                       ROWID NOT NULL,
       PICTURE                         BLOB(4194304)
     ) END-EXEC.
*****************************************************************
 01  DCLITEMS.
     10 HOST-PARTNO           PIC X(9).
     10 HOST-DESCRIPTION      PIC X(30).
     10 HOST-QOH              PIC S9(9) USAGE COMP.
     10 HOST-ITEMROWID        USAGE SQL TYPE IS ROWID.
     10 HOST-PICTURE          USAGE SQL TYPE IS BLOB-LOCATOR.
*****************************************************************
 01  blob-file-var usage is sql type is blob-file.
```



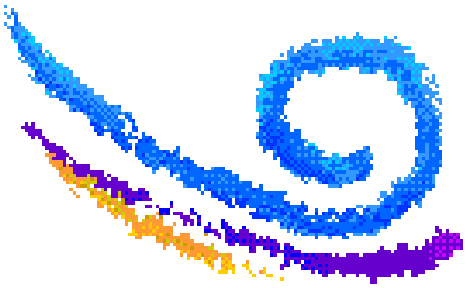
Using LOBs - an Example, 16

- DB2 Version 9.1 helps: the ability to reference LOBs using file reference variables is new:
  - In COBOL, specify a USAGE as SQL TYPE BLOB-FILE (or CLOB-FILE or DBCLOB-FILE)
  - If you use file reference variables for retrieval of a LOB, DB2 copies the data in the LOB column into the file you specify; on update or insert, the contents of a named file will be copied into the database LOB column



Using LOBs - an Example, 17

- Use file reference variables when you want to move the entire contents of a single row's LOB column value to / from an external file
  - Since BLOB data is unstructured (but probably well-defined), you certainly don't want to change part of the value!



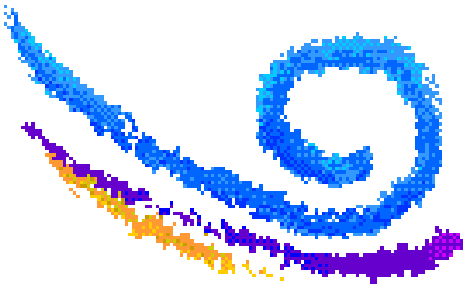
Using LOBs - an Example, 18

- When you specify a file reference variable, as we did with:

```
01  blob-file-var usage is sql type is blob-file.
```

  - … the pre-processor or co-processor generates four sub fields, using a well-defined naming convention; in our case we get:

```
01  BLOB-FILE-VAR.
    49 BLOB-FILE-VAR-NAME-LENGTH PIC S9(9) COMP-5 SYNC.
    49 BLOB-FILE-VAR-DATA-LENGTH PIC S9(9) COMP-5.
    49 BLOB-FILE-VAR-FILE-OPTION PIC S9(9) COMP-5.
    49 BLOB-FILE-VAR-NAME PIC X(255).
```

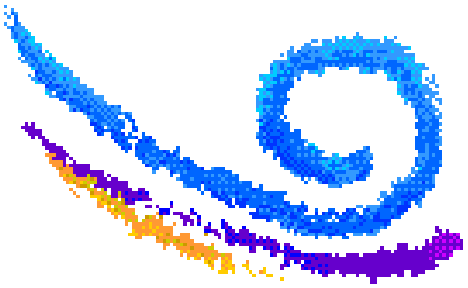


Using LOBs - an Example, 19

- Working with file reference variables requires you to:
  - Set a value into blob-file-var-file-option to indicate if you intend to:
    - 2 - read an existing file
    - 8 - create a new file (and if it exists that is an error)
    - 16 - overwrite an existing file (and if it doesn't exist, create it)
    - 32 - append to an existing file (and if it doesn't exist, create it)
  - Data items are generated with these values, if you care to use them

Using LOBs - an Example, 20

- Working with file reference variables requires you to:
  - Build the file name in blob-file-var-name
  - Put the length of the file name in blob-file-var-name-length
  - If you are retrieving a LOB, the SELECT service will populate blob-file-var-data-length for you



Using LOBs - an Example, 2 I

- Here is the code we used to retrieve our BLOB:

```
move sql-file-overwrite to blob-file-var-file-option
move 1 to blob-file-var-name-length
string '/u/scomsto/public_html/db2work/'
       trans-partno '.bmp' delimited by size
   into blob-file-var-name
   with pointer blob-file-var-name-length
subtract 1 from blob-file-var-name-length
exec sql
   select description, qoh,  picture
   into  :host-description, :host-qoh, :blob-file-var
   from scomsto.items
   where  partno = :trans-partno
end-exec
```

Notes:

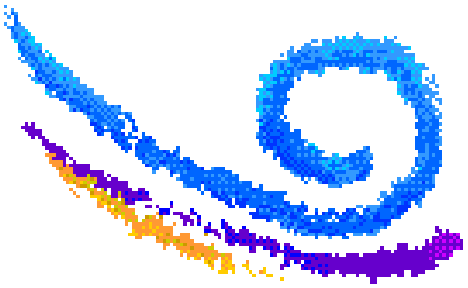
*** SQL-FILE-OVERWRITE is the name of the generated field with value 16**

*** The other generated fields are:**

SQL-FILE-READ with value 2

SQL-FILE-CREATE with value 8

SQL-FILE-APPEND with value 32

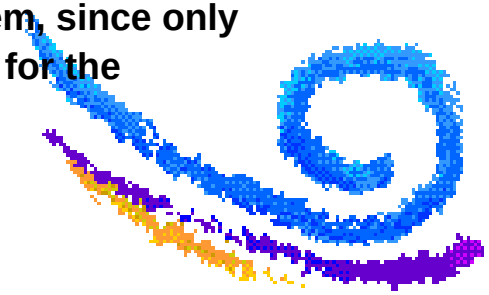


Using LOBs - an Example, 22

- Given that we know the part number we are after, we can construct the name for the blob file by appending .bmp and placing this in an available directory
  - We ended up with /u/scomsto/public_html/db2work/Part*nnnnn*.bmp

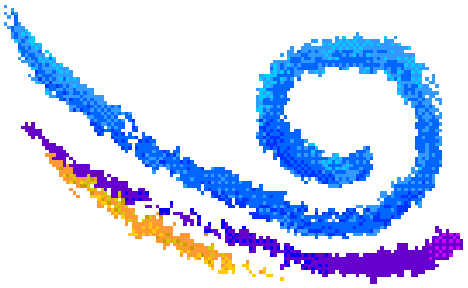
Note:

*** You can have indicator variables for LOB columns, for managing NULL values when retrieving / updating these columns; in our application we didn't need them, since only valid part numbers can be requested and all rows have non-NULL values for the PICTURE column**



Using LOBs - an Example, 23

- If the SELECT is successful, we have retrieved the information we need
  - We can now build an HTML page to send back to the requestor
    - In COBOL, you do this with a series of "display" statements
- Then we're done!



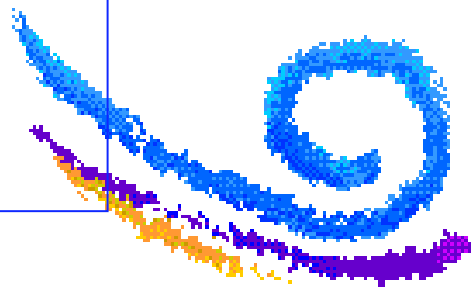
Using LOBs - an Example, 24

- Here is a sample response from our CGI:

Welcome to The Trainer's Fiend Novelty Products Division

"We provide the finest products you can't find anywhere else!"

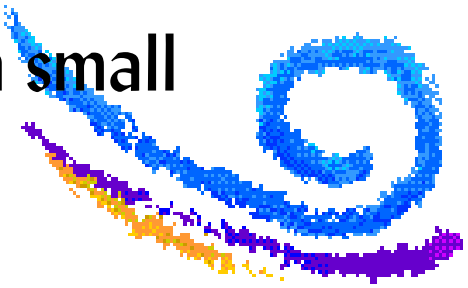
Here is the item called User Friendly Toys
Quantity on hand: 38



LOBs - a Summary

- BLOBs, CLOBs, and DBCLOBs provide unique opportunities to handle nontraditional data

- DB2 V9.1 provides new facilities to make it easier to store and work with documents and multimedia as LOBs in your database tables:
  - Automatic create of supporting infrastructure
  - File reference variables, usable in utilities such as LOAD as well as from programming languages
  - FETCH CONTINUE to access parts of LOBs in small chunks (not discussed here)



References

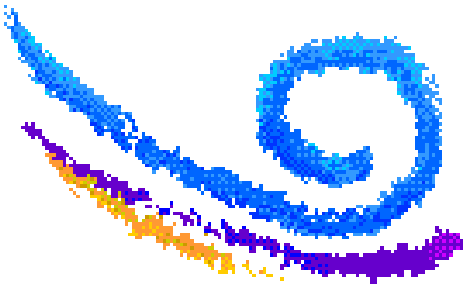
DB2 V9.1 IBM manuals found at:
  http://www-03.ibm.com/systems/z/os/zos/bkserv/zswpdf/#db2v91

IBM Redbooks on DB2:
  http://publib-b.boulder.ibm.com/abstracts/sg246826.html?Open
  http://publib-b.boulder.ibm.com/abstracts/sg247427.html?Open
  http://publib-b.boulder.ibm.com/abstracts/sg247330.html?Open

World Wide Web Consorition (W3C)
  http://www.w3.org/ (follow links here to XML, HTML, etc.)

Free technical papers from The Trainer's Friend
  http://www.trainersfriend.com/General_content/Book_site.htm
Includes "Introduction to Unicode"
         "Introduction to XSLT"
         "Hosting a Web Site on z/OS"
and more

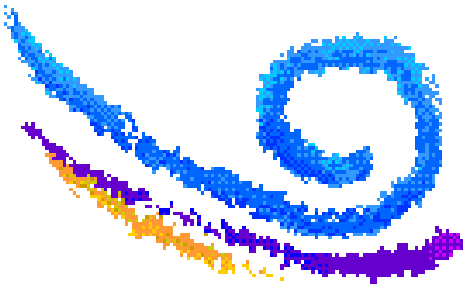
Training on DB2 and other technologies discussed here
  http://www.trainersfriend.com



Conclusion

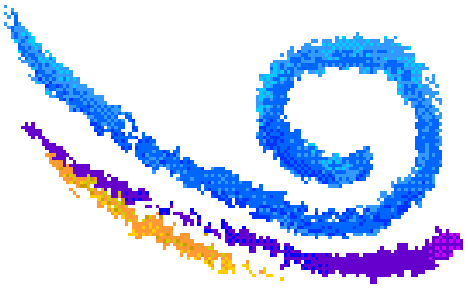
- DB2 Version 9.1 is a significant release of this flagship product, with lots of useful enhancements
- Today we have seen how you can create a table that uses LOBs, and how to store and retrieve LOBs from a DB2 database

- Questions, time permitting



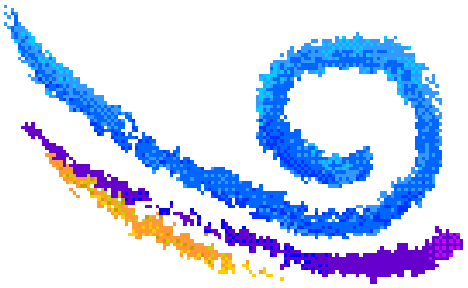
HTML - getitem1.html

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!--   Copyright (C) 2008 by Steven H. Comstock            Ver1   -->
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

<head>
<link rel="stylesheet" href="db2style.css" type="text/css" />
<title>Novelty Products Division Product Catalog </title>
<script type="text/javascript">
var pno; </head>

<body>

<h1>Welcome to

<span class="logo">
The Trainer's Fiend
</span>
<span class="logo2">
     Novelty Products Division
</span>
```



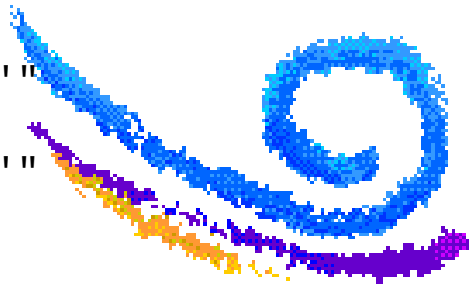
HTML - getitem1.html, 2

```
<p>
<img src="TrainersFiend.gif" alt="GIF did not display" align="center" />
"We provide the finest products you can't find anywhere else!"
</p>
<hr />

<div class="catalog">
<h1>Product Catalog</h1>
</div>

<form action="/SCOMSTO/coblob1" method="get" id="oform" >

<h2>Here is the list of items in our catalog; Select the one item you want to see by
clicking on it.</h2>

<input type="hidden" name="ano" id="ano" value="Unset;" />
```



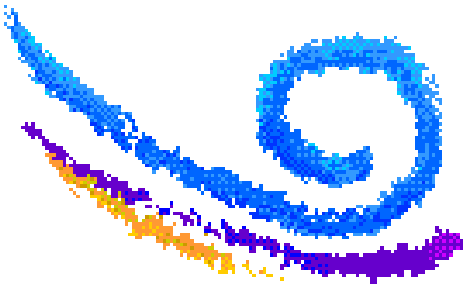
HTML - getitem1.html, 3

```
<label>Descriptions
<select name="description" id="description"
size="5">
   <option name="Part00105" id="Part00105" onclick="pno='Part00105'"
   >Keys to the Kingdome </option>
   <option name="Part00240" id="Part00240" onclick="pno='Part00240'"
   >Overtime Hours   </option>
   <option name="Part00273" id="Part00273" onclick="pno='Part00273'"
   >Anodized Tennis Ball </option>
   <option name="Part00315" id="Part00315" onclick="pno='Part00315'"
   >Colo. Creeping Crud </option>
   <option name="Part00453" id="Part00453" onclick="pno='Part00453'"
   >Plastic Food     </option>
   <option name="Part00456" id="Part00456" onclick="pno='Part00456'"
   >Sane Asylum      </option>
   <option name="Part00459" id="Part00459" onclick="pno='Part00459'"
   >Profits          </option>
   <option name="Part00681" id="Part00681" onclick="pno='Part00681'"
   >Company Policy   </option>
   <option name="Part00705" id="Part00705" onclick="pno='Part00705'"
   >Emergency Rations </option>
   <option name="Part00717" id="Part00717" onclick="pno='Part00717'"
   >Hardy Plants     </option>
   <option name="Part00735" id="Part00735" onclick="pno='Part00735'"
   >Neighborhood Quarks </option>
```



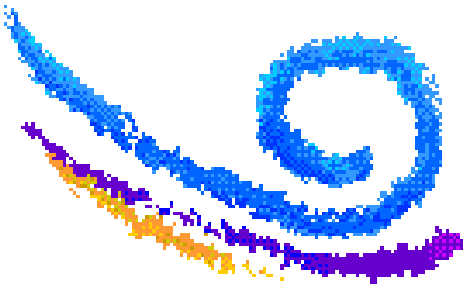
HTML - getitem1.html, 4

```
    <option name="Part03204" id="Part03204" onclick="pno='Part03204'"
    >Precision Parts  </option>
    <option name="Part03297" id="Part03297" onclick="pno='Part03297'"
    >Terminal Difficulties </option>
    <option name="Part03303" id="Part03303" onclick="pno='Part03303'"
    >Temporary Bendings </option>
    <option name="Part03369" id="Part03369" onclick="pno='Part03369'"
    >User Friendly Carpet Tacks </option>
    <option name="Part03402" id="Part03402" onclick="pno='Part03402'"
    >Indifferent Bankers </option>
    <option name="Part03450" id="Part03450" onclick="pno='Part03450'"
    >Diddle Tats      </option>
    <option name="Part03600" id="Part03600" onclick="pno='Part03600'"
    >Impartial Palace </option>
    <option name="Part03666" id="Part03666" onclick="pno='Part03666'"
    >User Friendly Toys </option>
    <option name="Part03732" id="Part03732" onclick="pno='Part03732'"
    >Rounding Errors  </option>
</select>
</label>
```



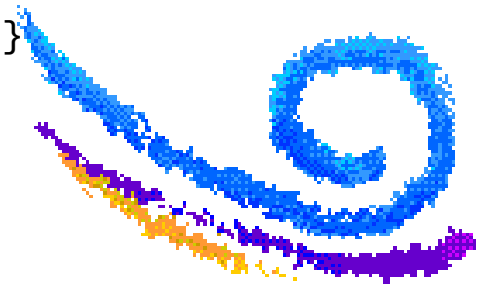
HTML - getitem1.html, 5

```
<br />

<input type="submit" name="sub_1" value="Submit request"
   onclick="getElementById('ano').setAttribute('value',pno);" />

<input type="reset" name="res_1" value="Clear request" />

</form>


</body>
</html>
```



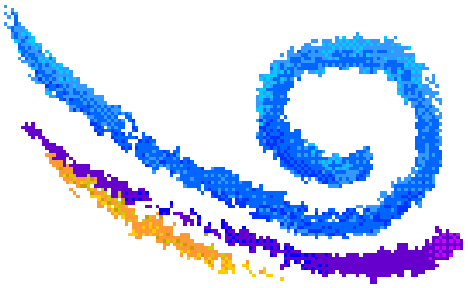
HTML - db2style.css

```
pre     { color: blue;
          font-weight: bold;
          font-size: 10pt;
          font-family: courier }
p        {font-size: 10pt; font-weight: bold}
p.pwd    {font-size: 12pt; font-weight: bold; }
label    {font-size: 12pt; font-weight: bold; }
span.logo {font-size: 14pt;
           font-weight: bold;
           border-width: 2;  border-style: solid; border-color: red;
           font-family: "copperplate gothic", fantasy, serif;
           color: red}
span.logo2 {font-size: 14pt; font-weight: bold; font-family: fantasy, serif;
            color: magenta;  align: center }
pre.h-name {color: red}
pre.h-val  {color: blue}
pre.a-name {color: green}
pre.a-val  {color: magenta}
  sz       {color: yellow}
div.catalog {background-color: #FFFF00 ; border-style: ridge;
             border-width: 5;  text-align: center; font-size: 12pt}
```



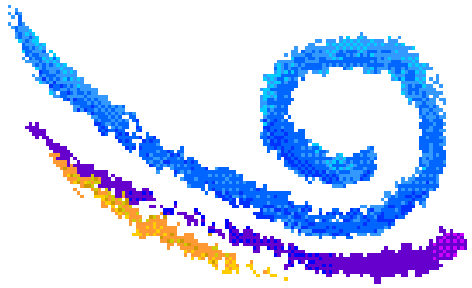
COBOL - coblob1

```
    process offset
* COBOL program to access a DB2 database from a CGI
* Copyright (C) 2008 by Steven H. Comstock
*
* Program is invoked from an HTML form, passing back a
* variable named 'ano' which has a value of the form
* Partnnnnn; this is value is used as a select into the
* Items table, retrieving Description, QOH, and Picture
* (which is a lob); copy lob data to file in HFS; build
* HTML response in the general style of displayItem.html
*
 id division.
 program-id.  coblob1.

 environment division.
 data division.
 working-storage section.

 01  blank-line pic x value x'15'.
```



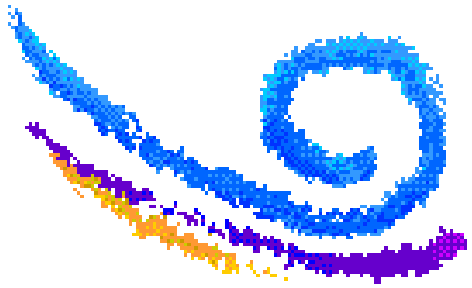
COBOL - coblob1, 2

```
* data items for displaying long strings    ------------

 01 html-stuff.
    02  style-file.
        03    pic x(23)
          value '<link rel="stylesheet" ' .
        03    pic x(38)
          value 'href="/~scomsto/db2work/db2style.css" '.
        03    pic x(20)
          value 'type="text/css" /> '.

* data items for call to bpx1opn and bpx1clo ---------

 01  bpx1chm-variables.
     02  options          pic x(4) value x'000000C3'.
     02  mode766          pic x(4) value x'000001F6'.
     02  file-descriptor  pic s9(9) binary  value 0.
     02  return-value     pic s9(9) binary  value 0.
     02  retrn-code       pic s9(9) binary  value 0.
     02  reason-code      pic s9(9) binary  value 0.
```



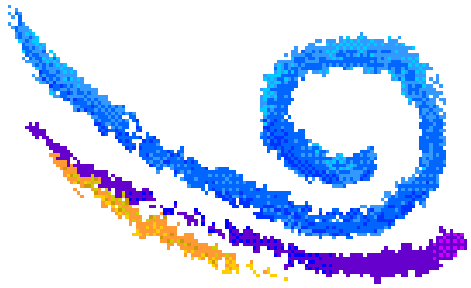
COBOL - coblob1, 3

```
* data items for environment variable work  ------------

 01  Envar-related-variables.
     02  var-name      pic x(13)   value z'QUERY_STRING'.
     02  env-ptr       pointer.
     02  len           pic s9(8)   binary value 0.
*
*     search-on value
*
 01  trans-partno                 pic x(9).

*
*     edit numeric field
*
 01  ed-qoh                       pic z99.
```



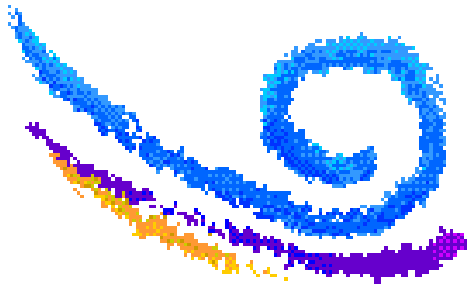
COBOL - coblob1, 4

```
*****************************************************************
*    db2 data areas                                             *
*****************************************************************

*
*     the following data items are used for invoking dsntiar
*
 01  message-area.
     05 err-len                  pic s9(4) comp value +1680.
     05 err-txt                  pic x(120) occurs 14 times
                                  indexed by err-index.
 01  err-txt-len                 pic s9(9) comp value +120.
 01  err-txt-ct                  pic s9(9) comp value +14.
 01  dsntiar-detail              pic x(120).


*
*    db2 sql communication area
*
     EXEC SQL INCLUDE SQLCA END-EXEC.
```



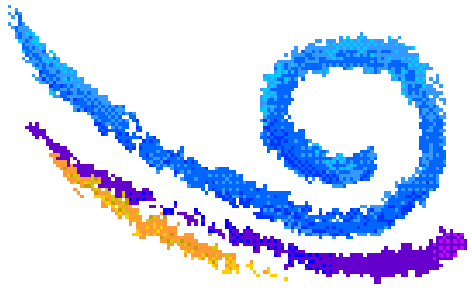
COBOL - coblob1, 5

```
*
*     db2 table declarations (from dclgen)
*
      EXEC SQL DECLARE SCOMSTO.ITEMS TABLE
      ( PARTNO                          CHAR(9) NOT NULL,
        DESCRIPTION                     CHAR(30) NOT NULL,
        QOH                             INTEGER NOT NULL,
        ITEMROWID                        ROWID NOT NULL,
        PICTURE                         BLOB(4194304)
      ) END-EXEC.
************************************************************
* COBOL DECLARATION FOR TABLE SCOMSTO.ITEMS                 *
************************************************************
 01  DCLITEMS.
     10 HOST-PARTNO          PIC X(9).
     10 HOST-DESCRIPTION     PIC X(30).
     10 HOST-QOH             PIC S9(9) USAGE COMP.
     10 HOST-ITEMROWID       USAGE SQL TYPE IS ROWID.
     10 HOST-PICTURE         USAGE SQL TYPE IS BLOB-LOCATOR.
************************************************************
* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 5  *
************************************************************

 01  blob-file-var usage is sql type is blob-file.
```



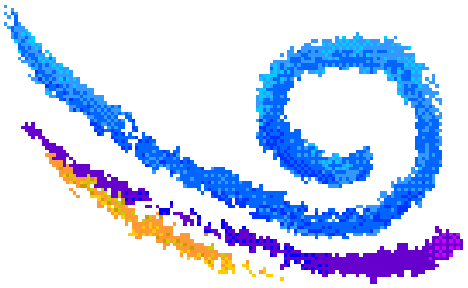
COBOL - coblob1, 6

```
01  rrs-stuff.
    05  req                    pic x(18).
    05  ssid                   pic x(4) value 'DB9G'.
    05  ribptr                 pointer.
    05  eibptr                 pointer.
    05  termecb                pic s9(9) comp value zero.
    05  startecb               pic s9(9) comp value zero.
    05  retcode                pic s9(9) comp value zero.
    05  reason                 pic s9(9) comp value zero.
    05  corrid                 pic x(12) value spaces.
    05  accttok                pic x(22) value spaces.
    05  acctint                pic x(6)  value 'COMMIT'.
    05  plan                   pic x(8)  value 'COBLOB1'.
    05  collid                 pic x(18) value spaces.
    05  reuse                  pic x(8)  value 'RESET'.
    05  dretcode               pic zzzzzzzz9.
    05  dreason                pic zzzzzzzz9.

linkage section.
01  var-value          pic  x(30).
```



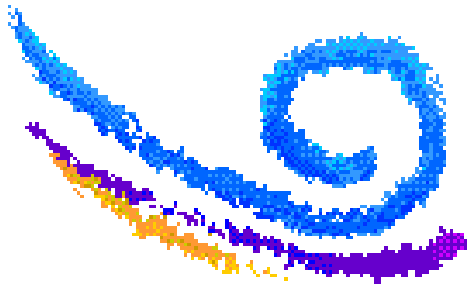
COBOL - coblob1, 7

```
 procedure division.

*
*   Emit the front lines of xhtml...
*
      display 'Content-Type: text/html'
      display blank-line

      display '<?xml version="1.0" encoding="UTF-8"?>'
      display '<!DOCTYPE html PUBLIC '
      display '"-//W3C//DTD XHTML 1.0 Strict//EN"'
      display '"http://www.w3.org/TR/xhtml1/DTD/' no advancing
      display 'xhtml1-strict.dtd">'
      display '<html xmlns="http://www.w3.org/1999/xhtml" '
      display 'xml:lang="en">'
      display '<head>'
      display style-file
      display '<title>Display one item from inventory</title>'
      display '</head>'
      display '<body>'
```



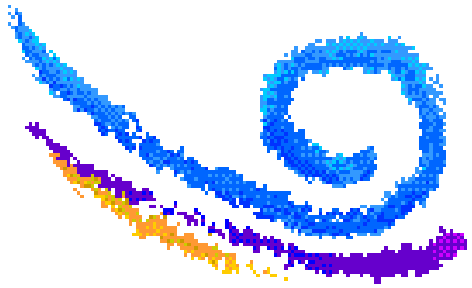
COBOL - coblob1, 8

```
* following lines extract query_string environment variable
*   then connect to DB2, retrieve the row, then disconnect from DB2

      call 'getenv' using var-name returning env-ptr
      set  address of var-value to env-ptr
      move var-value(5:9) to trans-partno

      move 'IDENTIFY' to req
      CALL 'DSNRLI' USING REQ SSID RIBPTR EIBPTR TERMECB STARTECB RETCODE REASON
      if retcode > 0 perform rrsafer goback end-if
      move 'SIGNON' to req
      CALL 'DSNRLI' USING REQ CORRID ACCTTOK ACCTINT RETCODE REASON
      if retcode > 0 perform rrsafer goback end-if
      move 'CREATE THREAD' to req
      CALL 'DSNRLI' USING REQ PLAN COLLID REUSE RETCODE REASON
      if retcode > 0 perform rrsafer goback end-if

      perform get-item

      move 'terminate thread' to req
      CALL 'DSNRLI' USING REQ RETCODE REASON
      if retcode > 0 perform rrsafer goback end-if
      move 'TERMINATE identify' to req
      CALL 'DSNRLI' USING REQ RETCODE REASON
      if retcode > 0 perform rrsafer goback end-if
```



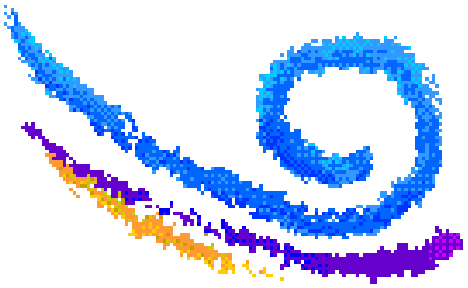
COBOL - coblob1, 9

```
*
*   Emit the back lines of html...
*

      display '</body>'
      display '</html>'

     goback.

 rrsafer.
     display '<h1> Had an RRS problem: ' req '</h1>'
     move retcode to dretcode
     display '<h2>Retcode:' dretcode '</h2>'
     move reason to dreason
     display '<h2>Reason:' dreason '</h2>'
     display '<h2>Halting execution with goback</h2>' .
```



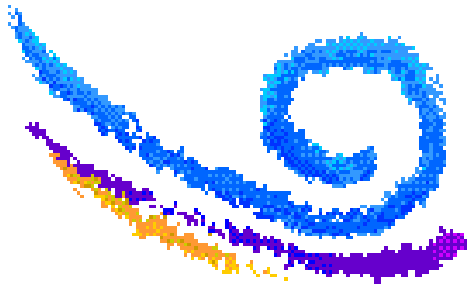
COBOL - coblob1, 10

```
 get-item.
     move sql-file-overwrite to blob-file-var-file-option
     move 1 to blob-file-var-name-length
     string '/u/scomsto/public_html/db2work/'
            trans-partno '.bmp' delimited by size
        into blob-file-var-name
        with pointer blob-file-var-name-length
     subtract 1 from blob-file-var-name-length

* open and close the file to create it with the correct permissions:

     call 'bpx1opn' using blob-file-var-name-length,
               blob-file-var-name, options,  mode766,
               file-descriptor, retrn-code, reason-code

     call 'bpx1clo' using file-descriptor, return-value,
               retrn-code, reason-code

     exec sql
        select description, qoh,  picture
        into  :host-description, :host-qoh, :blob-file-var
        from scomsto.items
        where  partno = :trans-partno
     end-exec
```



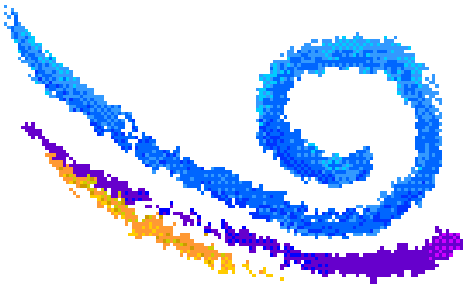
COBOL - coblob1, 11

```
if sqlcode =  zero

 display '<h1>Welcome to'
 display '<span class="logo">'
 display 'The Trainer' x'7d'  's Fiend'
 display '</span>'
 display '<span class="logo2">'
 display '  Novelty Products Division'
 display '</span>'
 display '<p>'
 display '<img src="/~scomsto/' no advancing
 display 'db2work/TrainersFiend.gif" '
 display 'alt="GIF did not display" align="center" />'
 display '"We provide the finest products you can' x'7d' 't '
 display 'find anywhere else!"'
 display '</p>'
 display '<hr />'
 display '<img src="/~scomsto/db2work/' no advancing
 display trans-partno '.bmp" align="middle"'
 display ' alt="BMP did not display" border="2"/>'
 display '<p class="pwd">'
 display 'Here is the item called '

 display  host-description
```



COBOL - coblob1, 12

```
      display '<br />'
      move host-qoh to ed-qoh
      display 'Quantity on hand: '  ed-qoh
      display '</p>'

     else if sqlcode = +100
          display 'Item not found'
     else perform dsntiar-call.


dsntiar-call.
     CALL 'DSNTIAR' USING SQLCA MESSAGE-AREA ERR-TXT-LEN
     perform dsntiar-print varying err-index
         from 1 by 1 until err-index > err-txt-ct.

dsntiar-print.
     move err-txt(err-index) to dsntiar-detail
     if dsntiar-detail not = spaces
          display '<pre>' dsntiar-detail '</pre>'.
```



6790 East Cedar Avenue, Suite 201
Denver, Colorado 80224
USA

http://www.trainersfriend.com
303.393.8716
Sales: kitty@trainersfriend.com
Technical: steve@trainersfriend.com
Technical: hunter@trainersfriend.com

Contact us for a copy of the files used to develop and test lecture points.