



Image, Audio & Video in DB2

(Assignment 3)

Based on Roger Holmberg's document
Relational Database Design
*62/2i1056/2i1071/2i4110 autumn term 2004

v. 3.2

Table of contents

1 INTRODUCTION	3
2 MULTIMEDIA & DB2	3
2.1 MULTIMEDIA.....	3
2.2 MULTIMEDIA DATA AND DB2	4
3 DATABASE.....	5
4 EXERCISES.....	6
4.1 STARTING THE IAV EXTENDER SERVICE	6
4.2 CREATING THE DATABASE AND THE TABLES	7
4.3 QBIC (QUERY BY IMAGE CONTENTS)	8
4.4 POPULATING THE DATABASE.....	9
4.5 RUNNING QUERIES	9
4.6 ASSIGNMENTS	16
4.7 WHEN THINGS HAVE GONE BAD!	16
5 INTERNET RESOURCES	17
6 EPILOGUE	17

Table of figures

FIGURE 1 MULTIMEDIA.....	4
FIGURE 2 HANDLES AND META TABLES	5
FIGURE 3 MUSIC DATABASE.....	5

1 Introduction

This compendium contains the following:

- An introduction to multimedia
- An introduction to DB2's facilities for handling multimedia data
- Exercises on using DB2 for managing multimedia data

The environment in which you will perform this assignment is IBM DB2 Universal Database version 7.2, with IAV (Image Audio Video) extender. The following facilities of DB2 will be used:

- DB2 Command Window
- DB2 Command Center
- DB2 Information Center

More information on DB2 and its facilities can be found in the "Introduction to DB2 v7.2"-compendium. You are also expected to read that compendium before you start with this assignment, since there it contains some necessary information about how to use the removable disks and log into the DB-SRV-1 server.

2 Multimedia & DB2

In this chapter you can find a short introduction to multimedia and an introduction to DB2's facilities for managing multimedia data.

2.1 Multimedia

Image, audio and video data are sometimes referred to as *multimedia* data. This document introduces some of the multimedia capabilities of DB2.



Figure 1 Multimedia

2.2 Multimedia data and DB2

DB2 provides support for storing image, audio and video data (*IAV data*). The functionality for this is provided in *extenders* that can be introduced in DB2. Each extender provides some new user defined types (*UDTs*) and a number of user defined functions (*UDFs*) for operations on these *UDTs*. Moreover, there is a special command line processor, the DB2EXT Command Line Processor, where you can administer the IAV extenders.

DB2 can store single data objects up to 2 gigabytes (*Gb*) in size inside the database. This means roughly 160 minutes video in MPEG-1 format. Larger objects can also be handled, but must be stored outside DB2. A single row in a table can contain at most 24 Gb, where there may be several columns with IAV data. A single table can hold up to 4 terabytes (~ 4000 Gb).

To enable querying on complex objects, DB2 keeps administrative data in separate tables. These tables are called *meta tables* (or *administrative support tables*). They contain information about formats of IAV objects, such as GIF, sizes of IAV objects in bytes, and many other features of IAV objects. They may also contain the actual IAV objects, if the objects were specified to be copied into the database. It is also possible to store files outside DB2, and have DB2 refer to those files instead of actually copying them into the database.

When an IAV object is inserted into the database, a *handle* is created that refers to that IAV object. The handle may refer to the meta tables or to some file outside the database. This handle is then stored in the actual user-created IAV column. The following picture shows an image column, as an example.

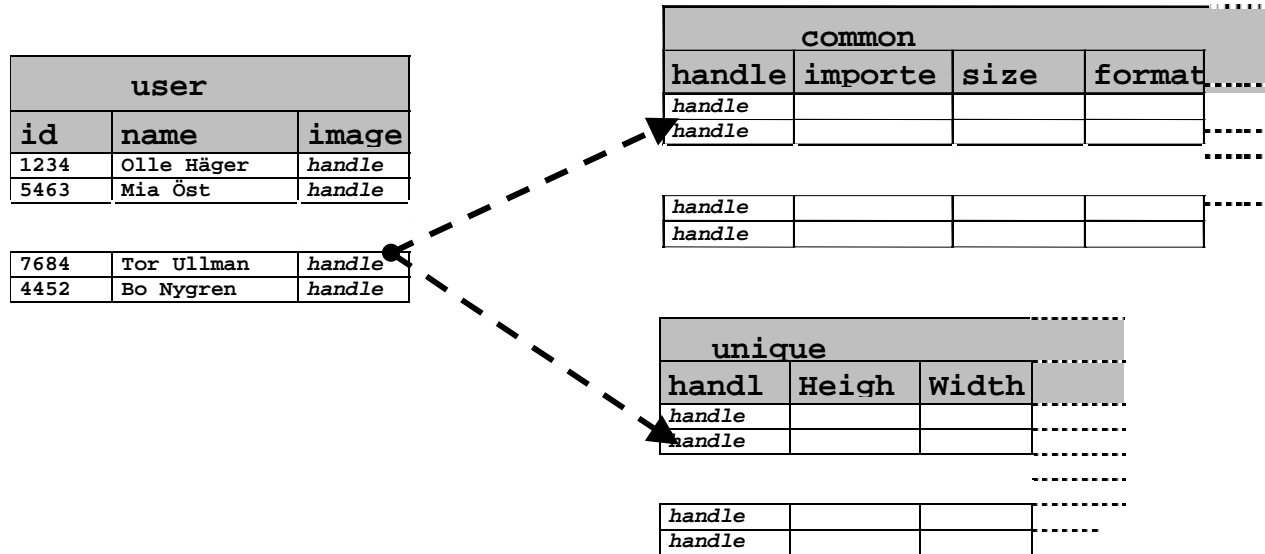


Figure 2 Handles and meta tables

Figure 2 shows an image example, but the principle is the same for audio and video data. It is not necessary to perform selects on the meta tables to get the attribute data; attribute data is supplied through UDFs. This will be exemplified later. In addition to the administrative support tables in the figure above, there are also administrative tables containing the names of tables and columns that hold IAV data, and also what kind of IAV data they hold. For image data there is also a special structure, called QBIC catalogue, which will be discussed in a later chapter.

3 Database

In the exercises in chapter 4 we will work with IAV data stored in a database. This database consists of three tables, containing songs, artists and music instruments.

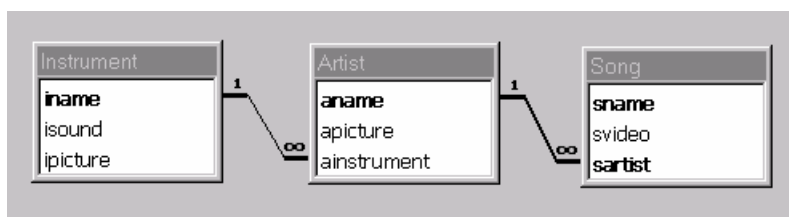


Figure 3 Music database

The column isound in the table instrument contains audio data.
The column ipicture in the table instrument contains image data.
The column apicture in the table artist contains image data.
The column svideo in the table song contains video data.

You can find scripts for creating and populating the database (we will do that in chapter 4), as well as multimedia files in your removable disk in the folder D:\Labb3 or at the DB-SRV-1 server: [\\Db-srv-1\StudKursInfo\x62 ht2004\Labb3\Music Database](http://Db-srv-1\StudKursInfo\x62 ht2004\Labb3\Music Database). See the *Introduction to DB2* compendium for information about how to log in and get access to files on the server.

4 Exercises

In this chapter we will:

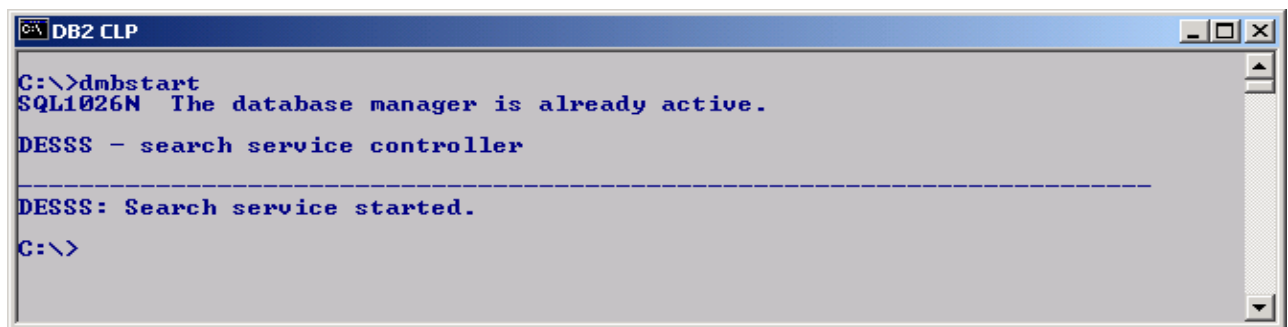
1. create a database according to the definition in chapter 3 and enable it for multimedia.
2. fill the database with "meaningful" data
3. run queries against the database (including the multimedia data)

4.1 Starting the IAV extender service

To make the IAV functionality available, we have to start a special process. To start this process, open a command prompt, and run the following command:

DMBSTART

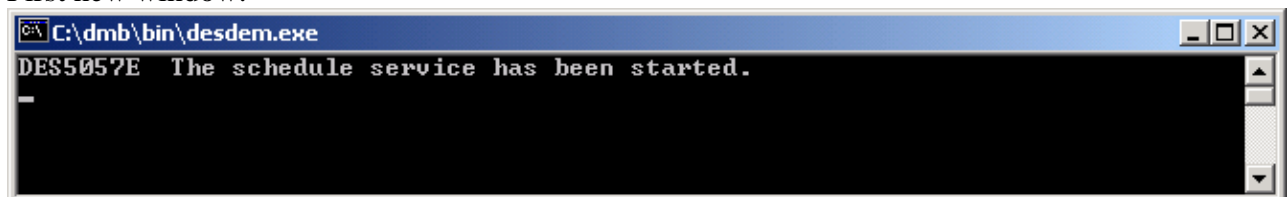
This starts a special process, called *extender service*, which must be running on the database server for DB2 to handle IAV data. This process needs only be started once after you log in to your workstation. The command prompt window will show the following and two more windows will pop up:



```
DB2 CLP
C:\>dmbstart
SQL1026N The database manager is already active.
DESSS - search service controller

-----
DESSS: Search service started.
C:\>
```

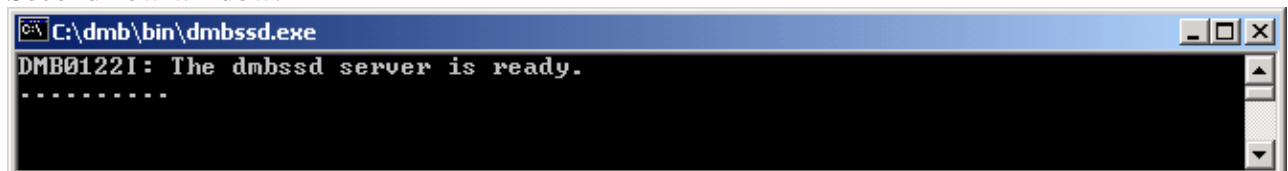
First new window:



```
C:\dmb\bin\desdem.exe
DES5057E The schedule service has been started.
```

⚠ Do not close this window! (You may minimize it, if you like)

Second new window:



```
C:\dmb\bin\dmbssd.exe
DMB0122I: The dmbssd server is ready.
.....
```

⚠ Do not close this window either!

4.2 Creating the database and the tables

To create the database we will need a DB2 Command Center window, or a DB2 Command Window. In the example that follows we use a DB2 Command Center for issuing DB2 commands. In addition to the Command Center window, we will need a DB2 extender Command Line processor window. To start the DB2EXT Command Line Processor, choose DB2 AIV Extenders Command Line Processor from the Start Menu (under DB2 extenders).

At this point there should be five windows open:

- A Command Prompt window (from where we started the extender service).
- Two windows where the DB2 extender service is running.
- A DB2 Command Center window (where we can run SQL statements, both DDL and DML statements)
- A DB2EXT Command Line Processor window (where we can run DB2 extender specific commands, e.g. for enabling a database for multimedia)

We start by creating a new database called **music**. Run the following command in the Command Center:

```
CREATE DATABASE music
```

❗ If something goes wrong during this exercise, you may need to undo certain steps. For example you can do **DROP TABLE** to undo a **CREATE TABLE**. A complete list of such commands that can be used for undoing things can be found in chapter 4.7.

When the database has been created, we can enable it for multimedia with the following commands issued in the DB2EXT Command Line Processor window (one at a time):

```
CONNECT TO music  
enable database for DB2AUDIO, DB2IMAGE, DB2VIDEO
```

When the database has been enabled, new items have been added: system tables, UDFs and UDTs. This gives us the possibility to create columns of special multimedia types and also provides functions for managing multimedia data. But the database is not ready to receive multimedia data yet.

We can now create tables with special multimedia columns. To create the database tables, use the **CREATE TABLE** statements from the "music.create.script" file. Run them in the Command Center!¹

When the tables have been created, we can continue by enabling them for the specific multimedia types that they contain. Moreover we have to enable each multimedia column for their multimedia type. We do that with the **ENABLE TABLE** and **ENABLE COLUMN** commands (that can also be found in the "music.create.script" file).

¹ If DB2 is configured to expect a command terminator character, you will need to either add this character at the end of each command, or reconfigure DB2 Command Center not to expect a command terminator character.

The database is now ready to receive multimedia data. However, there are certain features that we have not yet activated. Those features have to do with images only and are referred to as QBIC (Query By Image Contents).

4.3 QBIC (Query By Image Contents)

QBIC provides a more sophisticated form of querying on images. QBIC makes it possible to search on color distribution and texture. To enable QBIC we must do the following:

- ❖ Create a QBIC administrative catalogue for a specific image column. Run the following command in the DB2EXT Command Line Processor:

```
CREATE QBIC CATALOG artist apicture ON
```

This creates a special catalogue for the column `apicture` of the table `artist`, called the QBIC catalogue. The QBIC catalogue will contain the administrative (or meta) data associated with the images in the column `apicture`. These metadata are used when searching for images by content.

The `ON` parameter specifies that when a new image is inserted, the QBIC catalogue should be automatically updated.

- ❖ Open the QBIC catalogue. Run the following command to open the QBIC catalogue:

```
OPEN QBIC CATALOG artist apicture
```

- ❖ Add desired features to the QBIC catalogue. There are four possible features available:
 - `QbColorFeatureClass` enables searches based on average color of image.
 - `QbColorHistogramFeatureClass` enables searches based on comparisons against a spectrum of 64 colors. For instance, an image may consist of 20% green, 5% blue and 75% black.
 - `QbDrawFeatureClass` enables searches based on average color in specified areas of images.
 - `QbTextureFeatureClass` enables searches based on the coarseness, contrast and directionality of images. Coarseness indicates the size of repeating elements in an image, contrast identifies variations in brightness in images and directionality indicates whether a direction predominates in an image or not (for instance, an image of a striped surface or an image of an evenly colored surface.)

These features can be added to the QBIC catalogue with the `ADD QBIC FEATURE` command:

```
ADD QBIC FEATURE QbColorFeatureClass
ADD QBIC FEATURE QbColorHistogramFeatureClass
ADD QBIC FEATURE QbDrawFeatureClass
ADD QBIC FEATURE QbTextureFeatureClass
```

It is not necessary to use QBIC features at all. It is also possible to select only a few of them. In this exercise we will use the two first ones. (You can experiment with the rest on your own!)

- ❖ Check the QBIC catalogue info. The following command can be used to see which QBIC features are active in the QBIC catalogue:

GET QBIC CATALOG INFO

4.4 Populating the database

The database is now more than ready for data. There is a script file for populating the database (music.populate.script). In this file there are INSERT statements that refer to multimedia files on the local file system. It is assumed that all the multimedia files lie under d:\tmp. It is essential that the multimedia files exist in the corresponding directory. If you choose not to place the multimedia files under d:\tmp, then you will have to exchange the location in the INSERT statements with the correct one. The multimedia files can be found on your removable disk in the folder D:\Labb3\Music Database or at the DB-SRV-1 server: [\\Db-srv-1\StudKursInfo\x62ht2004\Labb3\Music Database](http://Db-srv-1/StudKursInfo/x62ht2004/Labb3/Music Database). See the *Introduction to DB2* compendium for information about how to log in and get access to files on the server.

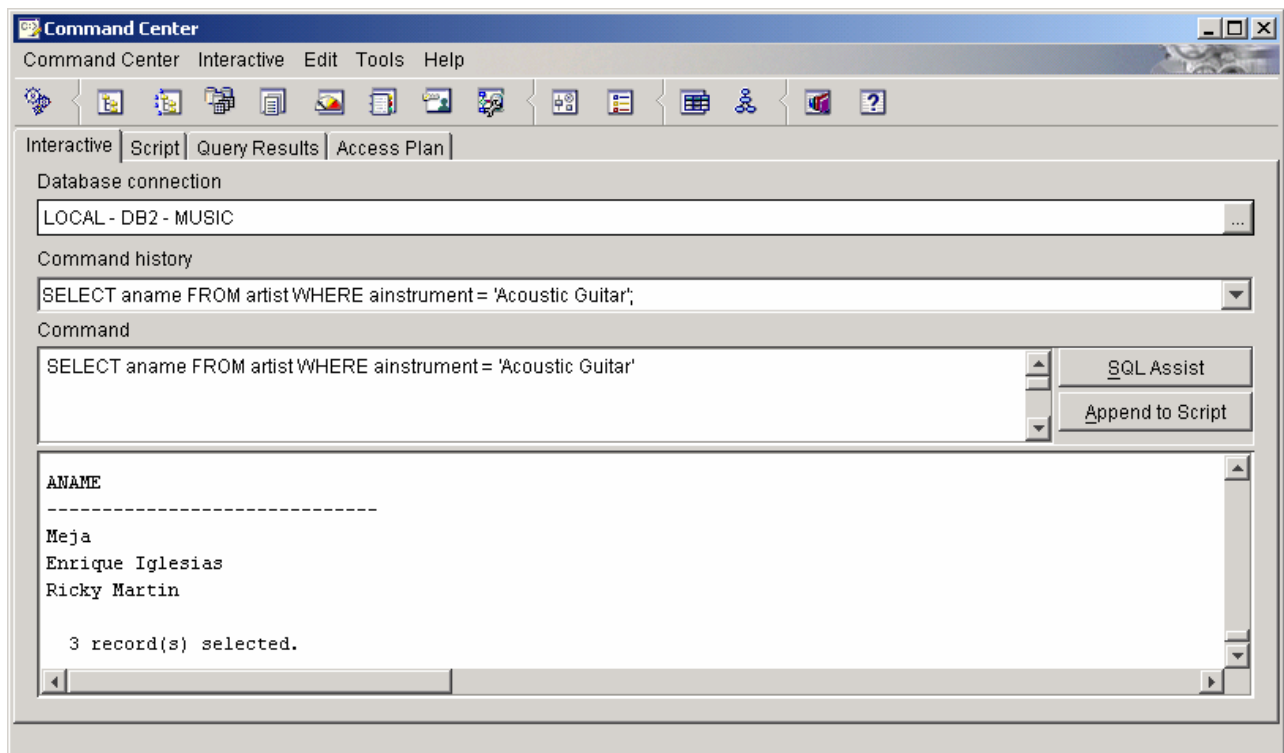
4.5 Running queries

Now that the database has some "meaningful" data, we can run some queries. We can of course ask ordinary questions, like "Show all the *artists* that play *Acoustic Guitar*!"

That is, in SQL:

```
SELECT aname FROM artist WHERE ainstrument = 'Acoustic Guitar'
```

Here is the result:



If we want to ask the database something about the multimedia data stored in the database, we have to use special functions. Here is a list of some useful functions that apply to multimedia data:

UDF	Description	Applies to			Page in Manual
		Image	Audio	Video	
Comment	Returns or updates a comment stored with an image, audio, or video.	X	X	X	206
Duration	Returns the duration (that is, playing time in seconds) of a WAVE or AIFF audio, or video.		X	X	228
Filename	Returns the name of the server file that contains the contents of an image, audio, or video.	X	X	X	229
Format	Returns the format of an image, audio, or video.	X	X	X	232
FrameRate	Returns the throughput of a video in frames per second.			X	233
Height	Returns the height, in pixels, of an image or video frame.	X		X	236
NumColors	Returns the number of colors in an image.	X			242
NumFrames	Returns the number of frames in a video.			X	243
Size	Returns the size of an image, audio, or video, in bytes.	X	X	X	257
Width	Returns the width in pixels of an image or video frame.	X		X	264

The syntax of these functions is quite simple:

MMDBSYS.function(columnname)

Where:

MMDBSYS is the name of the schema that owns the functions (it is always the same)

Function is the name of the function (for example Width)

Columnname is the name of the column that contains the multimedia data (for example apicture)

There are also four special UDFs for QBIC attributes:

UDF	Description	Page in Manual
QbScoreFromName	Returns the score of an image (uses a named query object).	245
QbScoreFromStr	Returns the score of an image (uses a query string).	247
QbScoreTBFromName	Returns a table of scores from an image column (uses a named query object).	248
QbScoreTBFromStr	Returns a table of scores from an image column (uses a query string).	250

Of these four functions, we will only use the second one (QbScoreFromStr). The syntax of this function is

MMDBSYS.QbScoreFromStr (queryString, imgHandle)

Where:

QueryString is a string containing a query:

Examples

- The following query string specifies an average color of red²:

² Colors are denoted with the amount of red, green and blue. The value of each color must be between 0 and 255. A bluish green for example would be <0,255,140>

`QbColorFeatureClass color=<255, 0, 0>`

- The following query string specifies a histogram comprised of 10% red, 50%, green, and 40% blue:
`QbcolorHistogramFeatureClass histogram=<(10, 255, 0, 0), (50, 0, 255, 0),(40, 0, 0, 255)>`

`ImgHandle` is a column containing images to be evaluated according to the `queryString`

- ❗ For more details on the syntax of a `queryString` and for a complete specification of values (for example `color`) available in Feature classes (for example `QbColorFeatureClass` and `QbcolorHistogramFeatureClass`), refer to the IAV manual, page 143.

We can now go back to solving queries...

For example we could have the following query:

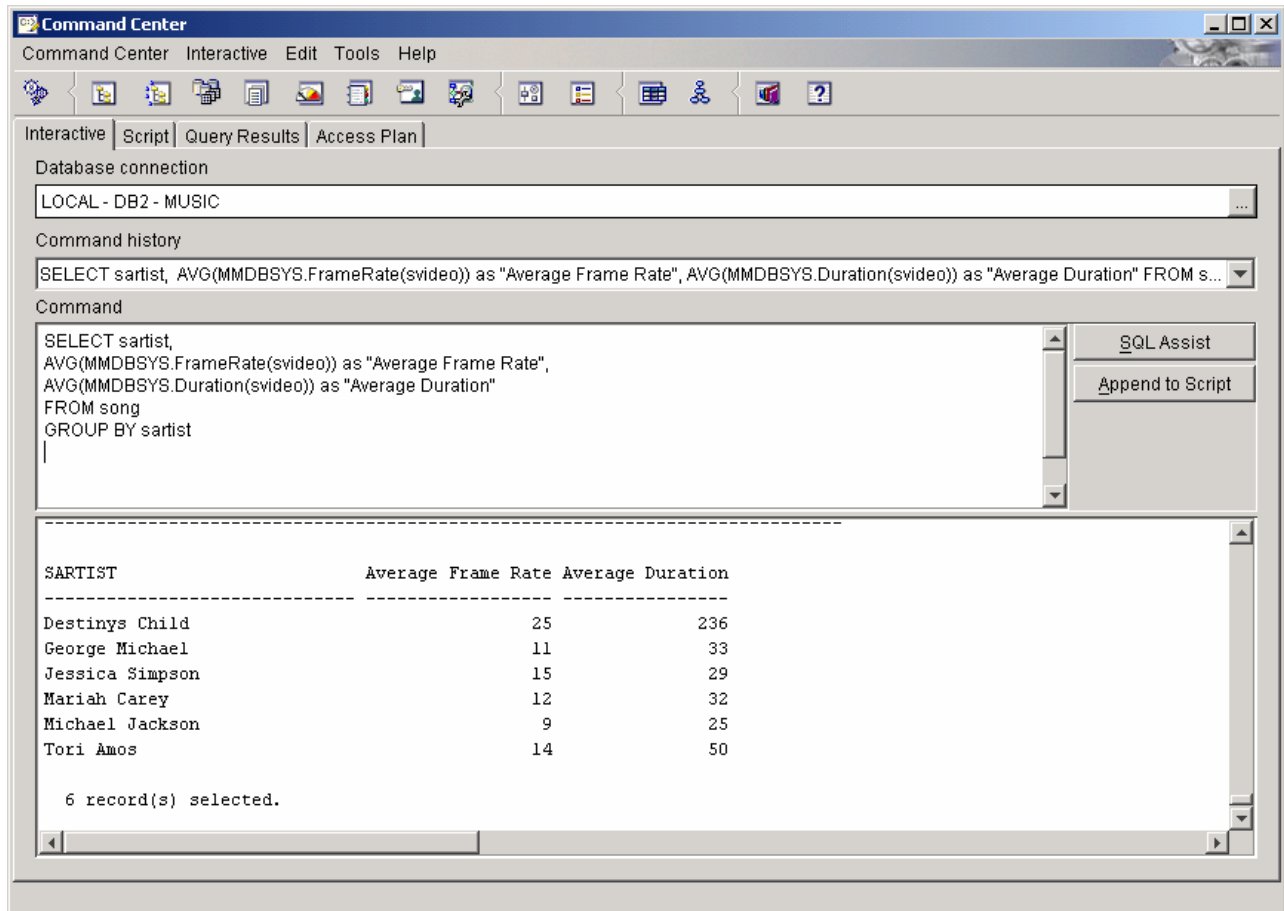
Show the *average quality* (Frames per second) and *average duration* of all the *videos*, for each *artist*!

To do this we have to use two UDFs: `FrameRate` and `Duration`. Then we have to find the average of those for every artist (artists can have more than one video).

Here is an SQL statement that does exactly that:

```
SELECT sartist,  
       AVG(MMDBSYS.FrameRate(svideo)) as "Average Frame Rate",  
       AVG(MMDBSYS.Duration(svideo)) as "Average Duration"  
FROM song  
GROUP BY sartist
```

Here is the result:



Another query can be the following:

Calculate how many *pixels* there are in every *artist's picture*!

To solve this query we can use the `height` and `width` functions, which return the amount of pixels of the dimensions of an image. And the SQL statement could look like this:

```
SELECT aname, (MMDBSYS.Height(apicture) * MMDBSYS.width(apicture)) as Pixels
FROM artist
```

And if we want to sort the results so that the biggest picture comes first:

```
SELECT * FROM
    (SELECT aname, (MMDBSYS.Height(apicture) * MMDBSYS.width(apicture))
    as Pixels FROM artist) as temptable
ORDER BY Pixels DESC
```

Or simply:

```
SELECT aname, (MMDBSYS.Height(apicture) * MMDBSYS.width(apicture)) as Pixels
FROM artist
order by 2 desc
```

where 2 refers to the column *Pixels* (the *second* column in the result), but before the column alias has been assigned.

Here is the result:

ANAME	PIXELS
Mariah Carey	330504
Meja	309246
Michael Jackson	246372
Ricky Martin	161976
Madonna	128000
Tori Amos	106932
Destinys Child	106400
Jessica Simpson	86700
Will Smith	61875
Enrique Iglesias	56160
Jennifer Lopez	46725
Lenny Kravitz	19200
George Michael	18300
Britney Spears	15400

14 record(s) selected.

We can now try a query that requires a QBIC feature. Let's try to solve the following query:

List all the *artists* according to the *blackness* of their *picture*!

To answer this query we need to use the `QbColorFeatureClass`. This class provides the possibility to compare a given color to the average color of an image. So we can compare all the images to black and order all the artists according to the result. The queryString would therefore be:

```
QbColorFeatureClass color=<0,0,0>
```

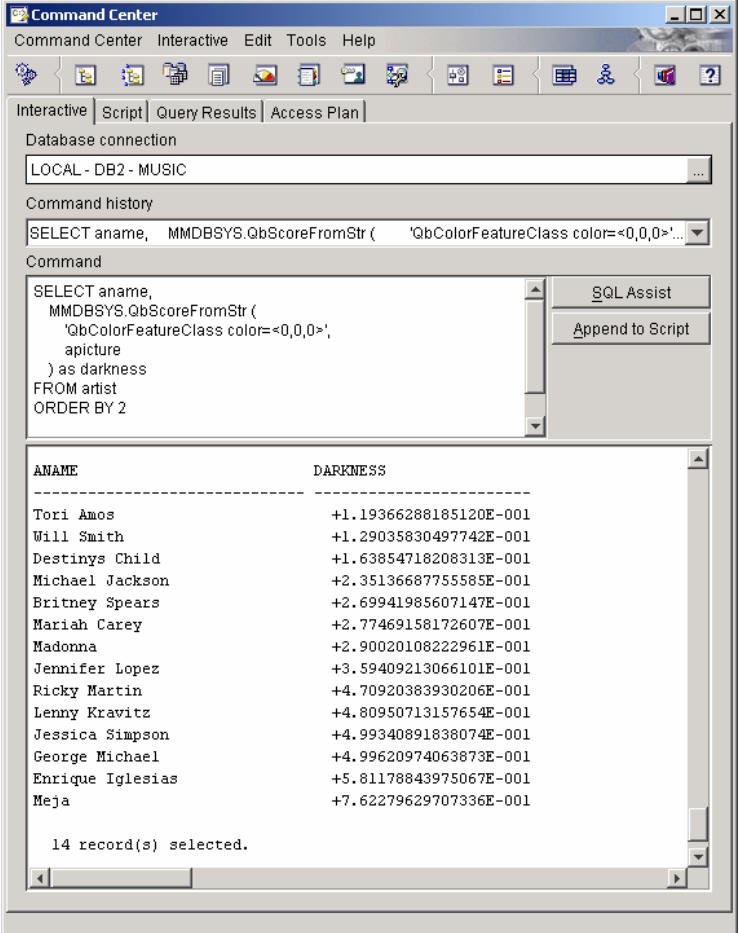
This queryString is the first parameter that is required in the `MMDBSYS.QbScoreFromStr` function. The second parameter is the image that we want to compare, which is in the `apicture`

column of the artist table. What the function returns is a numeric value between zero and infinity, or -1. -1 indicates that the image compared has not been cataloged (Normally the default is to catalog all the images automatically. If this hasn't been done, images can be cataloged with the CATALOG QBIC COLUMN FOR NEW command. This command should be executed while the QBIC catalogue is open.). If the result is 0 then the image matches exactly the criteria in the queryString. The greater the value gets, the more the image does not match the criteria. In our case, a dark picture should get a score close to zero, while a bright image should give a greater score.

Here is, in any case, an SQL statement:

```
SELECT aname,  
       MMDBSYS.QbScoreFromStr (  
         'QbColorFeatureClass  
color=<0,0,0>'  
       , apicture  
       ) as darkness  
FROM artist  
ORDER BY 2
```

And on the right you see how the result would look:



ANAME	DARKNESS
Tori Amos	+1.19366288185120E-001
Will Smith	+1.29035830497742E-001
Destinys Child	+1.63854718208313E-001
Michael Jackson	+2.35136687755585E-001
Britney Spears	+2.69941985607147E-001
Mariah Carey	+2.77469158172607E-001
Madonna	+2.90020108222961E-001
Jennifer Lopez	+3.59409213066101E-001
Ricky Martin	+4.70920383930206E-001
Lenny Kravitz	+4.80950713157654E-001
Jessica Simpson	+4.99340891838074E-001
George Michael	+4.99620974063873E-001
Enrique Iglesias	+5.81178843975067E-001
Meja	+7.62279629707336E-001

14 record(s) selected.

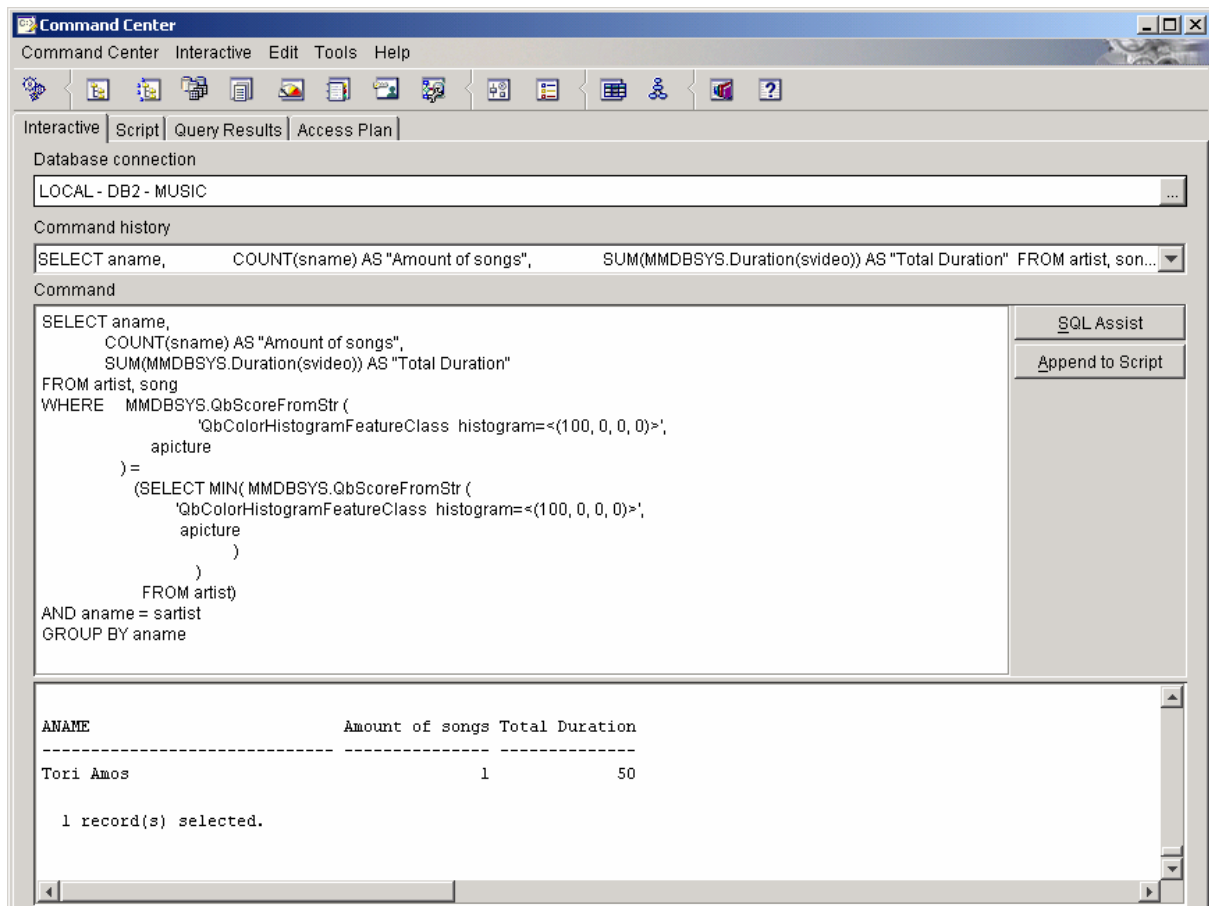
The score returned by the MMDBSYS.QbScoreFromStr function can also be used in an aggregate function or in the WHERE clause to build a criterion. Here is a little more advanced query that requires just that:

Show the *name* of the *artist* that has the *darkest picture* and the *amount of videos* that are associated with that artist and their *total length* in seconds! (Use the QbcolorHistogramFeatureClass instead of the QbColorFeatureClass!)

Here is the SQL statement for that:

```
SELECT aname,  
       COUNT(sname) AS "Amount of songs",  
       SUM(MMDBSYS.Duration(svideo)) AS "Total Duration"  
FROM artist, song  
WHERE  MMDBSYS.QbScoreFromStr (  
        'QbColorHistogramFeatureClass histogram=<(100, 0, 0, 0)>',  
        apicture) =  
       (SELECT MIN( MMDBSYS.QbScoreFromStr (  
        'QbColorHistogramFeatureClass histogram=<(100, 0, 0, 0)>',  
        apicture))  
        FROM artist)  
AND aname = sartist  
GROUP BY aname
```

And the result:



4.6 Assignments

The following queries should be solved sent in electronically in the FC conference “RELDES Assignments” (SQL statements and execution results):

1. List all the instruments, the length of their audio (in seconds) and the size of their picture (in bytes)!
2. Show the name and instrument of the artist that has the video with the best quality (most frames per second)!
3. List the instruments that their picture gives a score of less than 0.38 when the picture's average color is compared to blue, also list the dimensions of their picture and the amount of artists that are associated with that instrument! (It is OK if instruments that have no artist don't appear on the result.)

For the third exercise you will need to create a QBIC catalogue, add a feature in it and then catalog the images. Those commands (together with the responses you got when you ran them) must also be included in the solution that you send in!

4.7 When things have gone bad!

Sometimes you may need to undo certain steps, in order to correct something. In the table that follows you can find which commands that can be used to undo other commands.

Use this command	to undo this command
DROP DATABASE	CREATE DATABASE
DROP TABLE	CREATE TABLE
DISCONNECT	CONNECT TO
FORCE APPLICATIONS ALL	
DISABLE DATABASE FOR	ENABLE DATABASE FOR
DISABLE COLUMN	ENABLE COLUMN
DISABLE TABLE	ENABLE TABLE
DELETE QBIC CATALOG	CREATE QBIC CATALOG
CLOSE QBIC CATALOG	OPEN QBIC CATALOG
REMOVE QBIC FEATURE	ADD QBIC FEATURE

For the syntax of these commands, consult the Reference part of DB2 Information Center (general DB2 commands) or type ? in the DB2EXT Command Line Processor (DB2 Extender specific commands).

5 Internet Resources

DB2 IAV extender

<http://www-3.ibm.com/software/data/db2/extendere/aiv/index.html>

6 Epilogue

When all this is done, you should have an idea of how to use DB2 to manage multimedia data. There is much more that DB2 can do with multimedia data. If you are interested in learning more about this, then a good place to start is the DB2 IAV manual.

I hope you have enjoyed this compendium.

The Author

nikos dimitrakas