Image, Audio and Video extenders for DB2 Universal Database v6.1

– An Introduction

Department of Computer and Systems Sciences,
Stockholm University & The Royal Institute of Technology
2000

Contents

Introduction

Image, audio and video data are sometimes referred to as *multimedia* data. This document introduces you to some of the multimedia capabilities of **DB2 Universal Database 6.1** (*DB2* for short, in the rest of this document). You will work using the standard applications included with DB2, such as **Command Center** etc. You will not be required to write any special application code in Java, or such.

These are some of the things you will learn from this introduction:

- Basics of how DB2 manages image, audio and video data.
- How to create a database, table and columns able to hold image, audio and video data.
- How to store and extract image, audio and video data.
- How to search on image contents (Query By Image Contents, QBIC).

Once you have followed all instructions, you should do some rudimentary assignments given at the end of this document. Notice that you must first follow all the instructions before you can do these assignments.

Among the things you will ***not*** learn in this document are:

- *All* the multimedia capabilities of DB2.
- How to install multimedia extenders in DB2.
- How to write embedded SQL code that handles image, audio and video data. Some IAV functionality is only available through embedded SQL (such as video scene change detection).
- How to extend DB2 to handle more multimedia file formats.
- …

Any interested may refer to the manuals.

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 (*UDT*s) and a number of user defined functions (*UDF*s) for operations on these UDTs. In addition there is a special command line processor, the **db2ext** command line processor, for administering IAV data. Internally in DB2, the extenders provide special storage structures for IAV data.

Some statistics: DB2 can store single data objects up to 2 gigabyte (*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 terra byte (4 000 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, in case 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.

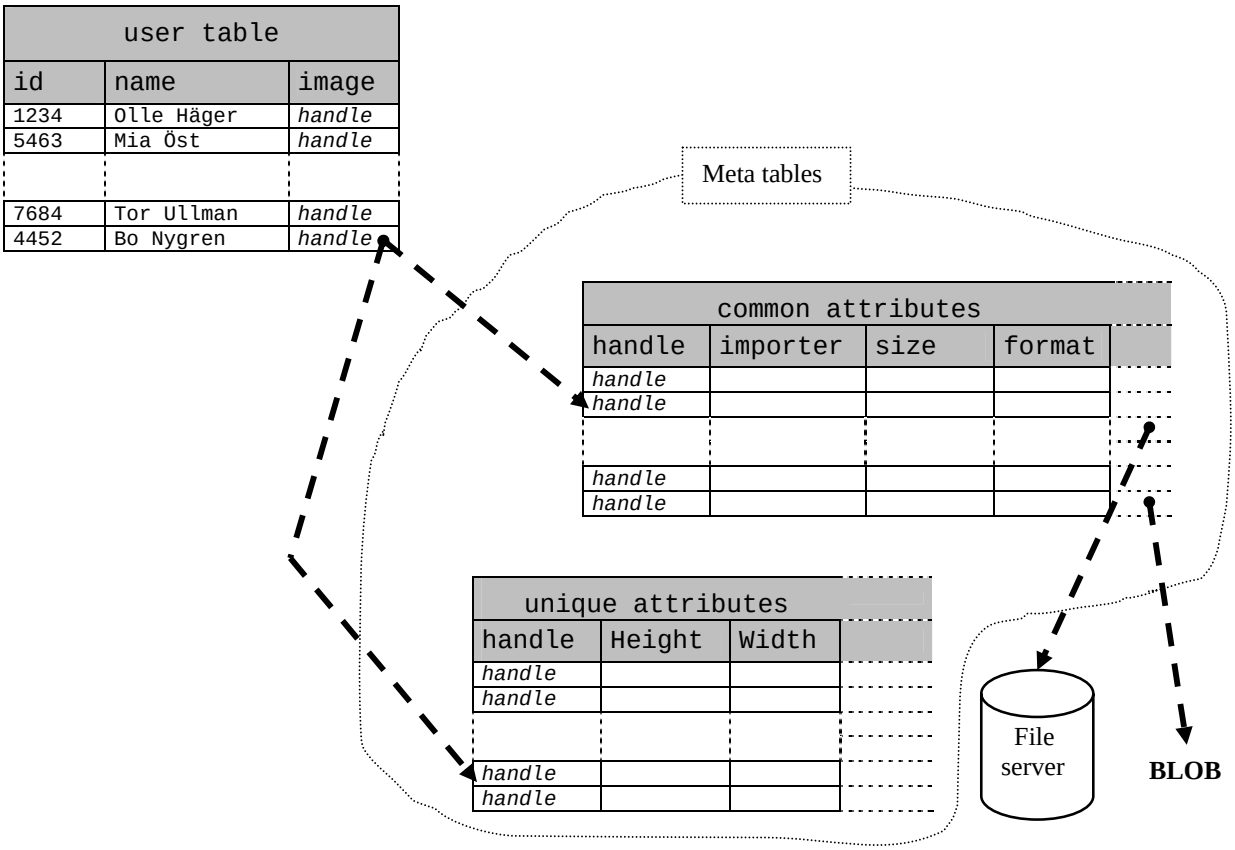

Fig 1. Handles and meta tables.

The figure above 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.

Incidentally, all handles for IAV objects actually have the same data type, as shown in the table below.

UDT	Data type
DB2IMAGE	VARCHAR(250)
DB2AUDIO	VARCHAR(250)
DB2VIDEO	VARCHAR(250)

Table 1. IAV UDT data types.

How you will work with this document (your assignment)

This document is separated into chapters dealing with different subjects related to IAV data management. However, throughout the document a series of 30 to 40 sequentially numbered *steps* are provided. **You** should follow these steps, in the order they appear. The steps will gradually introduce you to more aspects of IAV data management. When you

have completed these steps, you should provide answers to a few questions. Your first step will be to download a collection of IAV files from the course directory to your own DB2 account.

Good luck!

Download the IAV files from the course directory

1) Download the IAV files. In the course catalogue, there is a subdirectory `IAV_DATA`. Copy this entire subdirectory to your own DB2 account, for example using the Explorer.

   The result should be a subdirectory `IAV_DATA` in your own DB2 account.

Start the extender services

2) Start the extender services. Open a command prompt (dos window), and run the following command:

   ```
   C:\>DMBSTART
   ```

   This starts a special process, called *extender service*, that must be running on the database server for DB2 to handle IAV data. This process need only be started once every time DB2 is started on this server.

   Do not close this window, keep it open throughout the laboration! (You may minimise it, if you like)

   The result should look something like this:

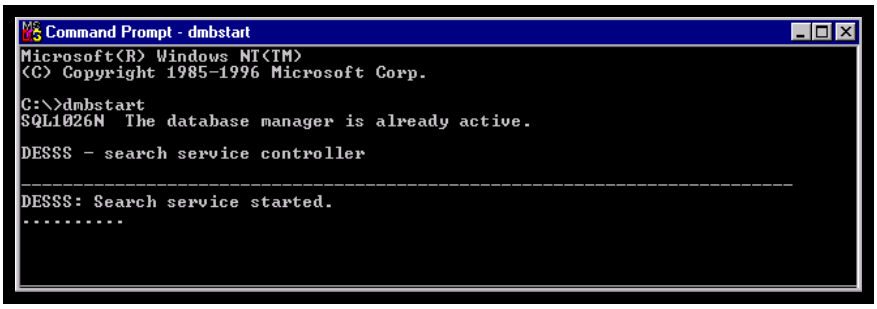

Fig 2. Starting the extender services.

Create the database and the table

1) Create the database. Start the **Control Center** and create a database IAV_LAB. Make it identical to the one created in the following three images below.

Once you have done this, minimise the **Control Center** window.

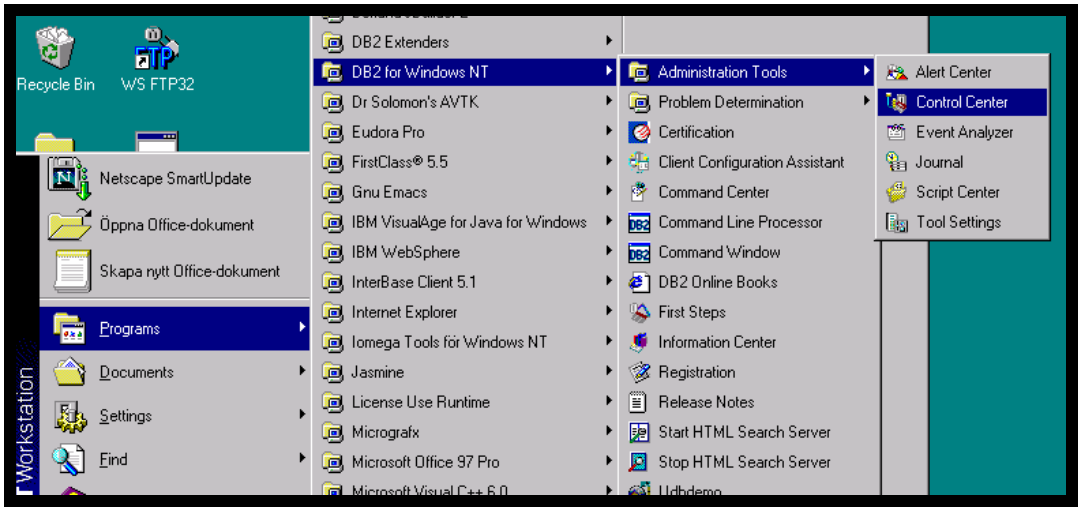

Fig 3. Start the **Control Center**.

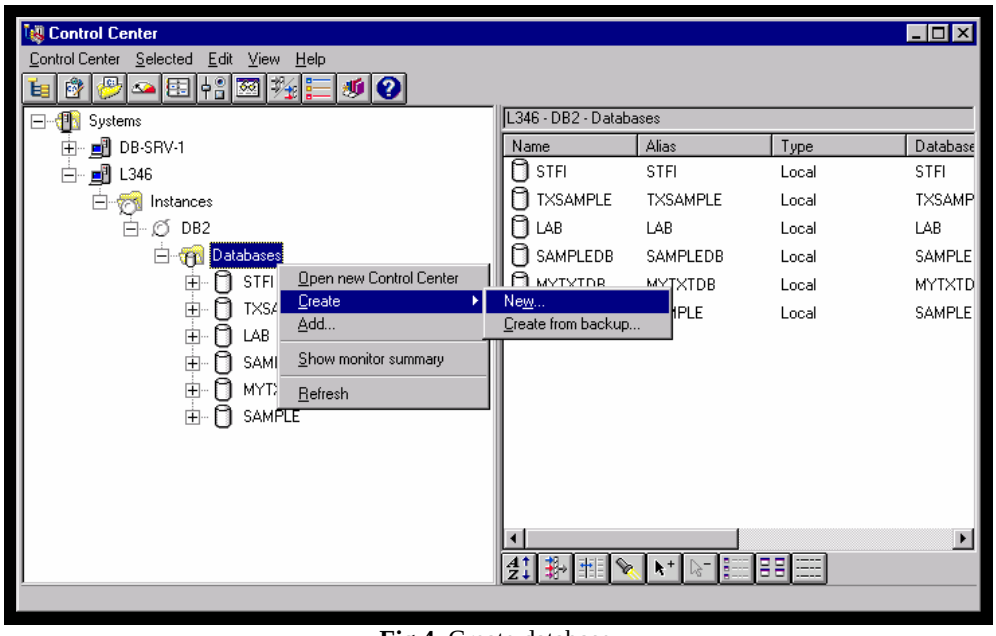

Fig 4. Create database.

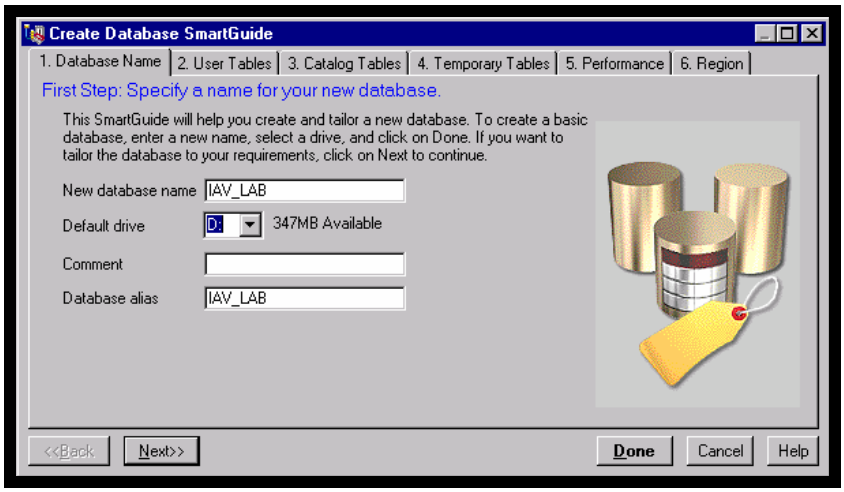

Fig 5. Name the new database.

2) Create the table. Start the **db2** command line processor (see fig), and run the following commands in it[1]:

```
db2 => CONNECT TO iav_lab
db2 => CREATE TABLE employee (id INTEGER, name VARCHAR(50))
```

Do not close this window, keep it open throughout the laboration! (You may minimise it, if you like)

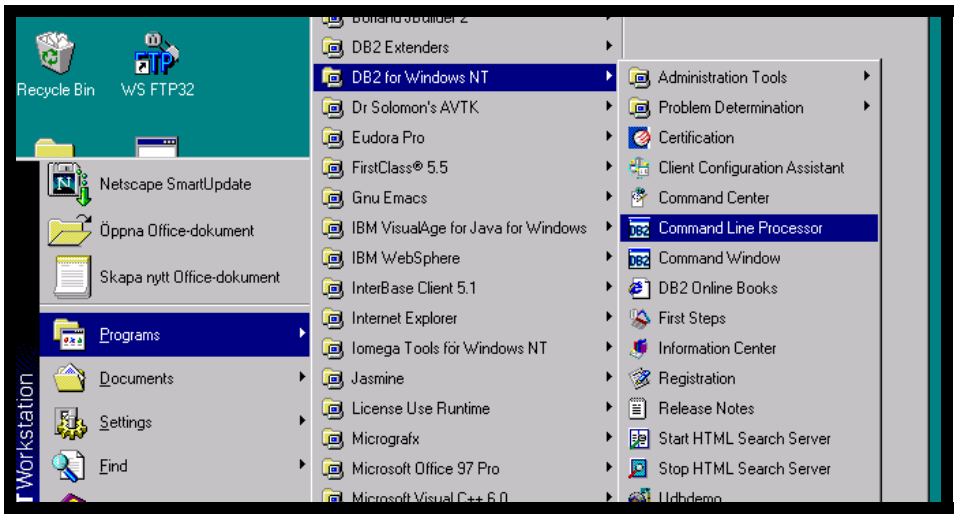

Fig 6. Start the **db2** command line processor.

We now proceed to handle image data in our database.

[1] You will use the **db2** command line processor for some tasks, instead of the **Control Center**, because our experience is that sometimes **Control Center** does not respond properly to changes introduced by IAV extenders. Typically, **Control Center** sometimes does not adopt IAV UDTs for databases despite these databases being enabled for such UDTs. You may find notes about this later in this document. Of course, **Control Center** *should* handle this properly, and also does many times.

Image data

DB2 can handle a variety of image formats (see Appendix) and provides content based search for images.

Set-up for image data

3) Enable the database for image data. Start the **db2ext** command line processor (see fig). In the **db2ext** command processor, connect to iav_lab and enable the database for image data (see fig).

Do not close this window, keep it open throughout the laboration! (You may minimise it, if you like)

The result of these operations is that
i) a UDT named DB2IMAGE is created for image objects
ii) meta tables are created for image objects and
iii) a set of UDFs for image objects is created.

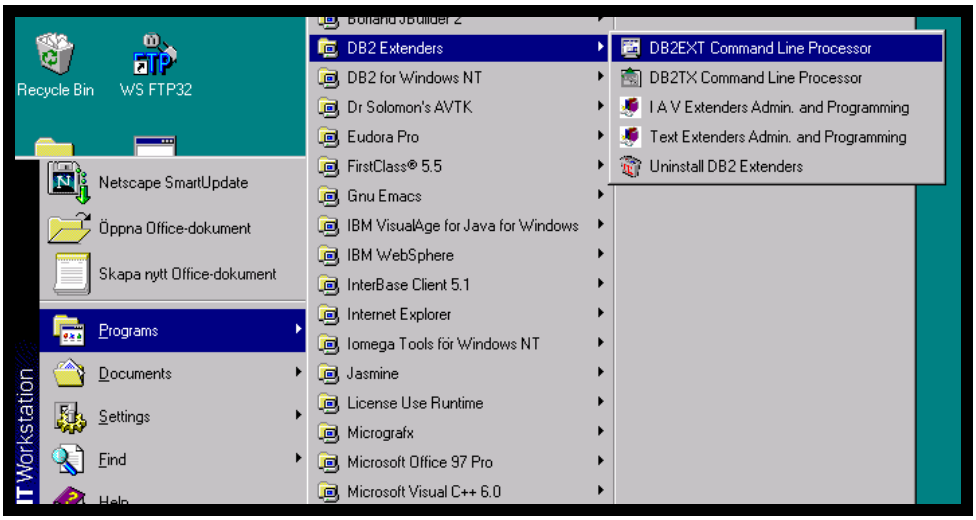

Fig 7. Start the **db2ext** command line processor.

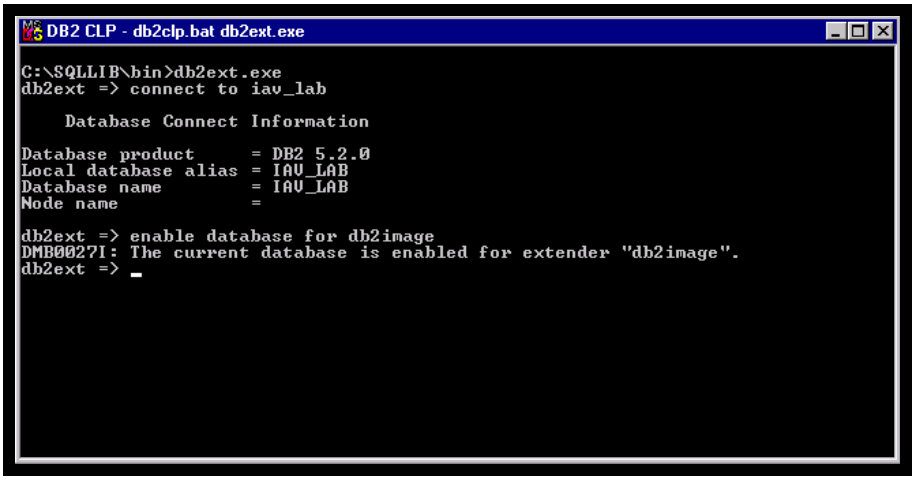

Fig 8. Connect to database and enable it for image data.

4) Add the image column. Run the following commands in the **db2** command line processor[2]:

db2 => `ALTER TABLE employee ADD picture MMDBSYS.DB2IMAGE`

5) Enable the table for image data. Run the following command in the **db2ext** command line processor:

db2ext => `ENABLE TABLE employee FOR DB2IMAGE`

6) Enable the columns for IAV data. Run the following command in the **db2ext** command line processor:

db2ext => `ENABLE COLUMN employee picture FOR DB2IMAGE`

You are now ready to put data into the table.

Inserting image data

7) Insert a row, with image data into the table employee. Start the **Command Center** (see fig), and run the following commands:

```
CONNECT TO iav_lab

INSERT INTO employee VALUES (
    1,
    'flowers1',
    MMDBSYS.DB2IMAGE (
        CURRENT SERVER,
        'm:/db2account/IAV_DATA/flower1.gif',
        'ASIS',
        1,
        'a flower arrangement'
    )
)
```

OBS! Replace *db2account* with your own DB2 account name, such as db2100 or similar.

*Do not close the **Command Center** window, keep it open throughout the laboration! (You may minimise it, if you like)*

CURRENT SERVER is a register containing the name of the current database, that is the database that you are issuing the command in.

'ASIS' means that the image, fetched from '/employee/images/holger.bmp' will be stored in the database in the format it already is, which is GIF.

The '1' signals that the image should be copied into the database as a BLOB. The image is NOT actually inserted into the table employee, but into the meta tables for the table employee. In the employee.picture field a handle is stored, that refers to the image BLOB in the meta tables. By instead specifying a '2' the image will not be stored in the database, but instead on a separate file server[3]. In that case a filename referring to the image is stored in the meta tables.

[2] For some reason, this seems not to work in the **Command Center**.

[3] There are actually two constants MMDBSYS.MMDB_STORAGE_TYPE_INTERNAL=1 and MMDBSYS.MMDB_STORAGE_TYPE_EXTERNAL=2 that correspond to these values but it appears that it is not possible to specify these interactively via the command center because they are longer than 18 characters.

The last string 'a flower arrangement' is just a comment that will be attached to the image.

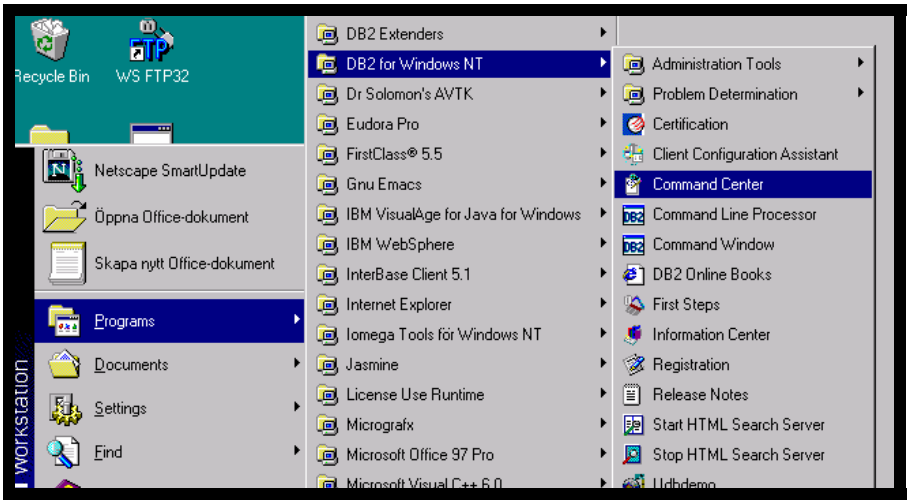

Fig 9. Start **Command Center**.

Extracting image data

Just as we may want to store an image in the database, we may want to extract it from the database.

8) Extract a stored image from the database. Run the following command in the **Command Center**.

```
SELECT MMDBSYS.content(picture,'m:/db2account/img.gif',1)
FROM employee
WHERE name='flowers1'
```

OBS! Replace *db2account* with your own DB2 account name, such as db2100 or similar.

In this case flowers1 is stored in a file called img.gif. The ”1” signals that DB2 should overwrite the file img.gif if it already exists. A ”0” would prevent DB2 from doing this.

Querying image data

It is possible to issue simple queries through, for instance, the **Command Center**.

9) A simple query on image data. Run the following query in the **Command Center**.

```
SELECT name, MMDBSYS.SIZE(picture)"IMAGE SIZE"
FROM employee
```

The result is the names of all employees and size of their images.

Query By Image Contents (QBIC)

Query by image content (QBIC) provides a more sophisticated form of querying on images. QBIC makes it possible to search on colour distributions and textures. Some preparations must be made to enable QBIC.

Preparations for QBIC

10) Create the QBIC administrative catalogue. Run the following command in the **db2ext** command line processor.

db2ext => `CREATE QBIC CATALOG employee picture ON`

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

The 'on' specifies that when a new image is inserted, the QBIC catalogue should be automatically updated.

11) Open the QBIC catalogue. Run the following command in the **db2ext** command line processor:

db2ext => `OPEN QBIC CATALOG employee picture`

12) Add desired features to the QBIC catalogue. Run the following commands in the **db2ext** command line processor:

db2ext => `ADD QBIC FEATURE QbColorFeatureClass`
db2ext => `ADD QBIC FEATURE QbColorHistogramFeatureClass`
db2ext => `ADD QBIC FEATURE QbDrawFeatureClass`
db2ext => `ADD QBIC FEATURE QbTextureFeatureClass`

This specifies that DB2 should keep information about these four different features in the QBIC catalogue.

QbColorFeatureClass enables searches based on average colour of image. QbColorHistogramFeatureClass enables searches based on comparisons against a spectrum of 64 colours. For instance, an image may consist of 20% green, 5% blue and 75% black. QbDrawFeatureClass enables searches based on average colour 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 coloured surface.)

These are in fact all image features that DB2 supplies.

It is not necessary to use all these features as we do here. It is possible to select only a few of them if desired. Also it is possible to add on more features successively.

13) Check the QBIC catalogue info (optional step). You may check that all is ok by running the following command in the **db2ext** command line processor:

db2ext => `GET QBIC CATALOG INFO`

14) Close the QBIC catalogue. Run the following command in the **db2ext** command line processor:

db2ext => `CLOSE QBIC CATALOG`

This closes the QBIC catalogue for employee picture. This means that you may not add features to the column.

15) Cataloguing images that have not been catalogued before. There is however a slight problem that need to be resolved before QBIC is possible. The images that we have already stored will *not* be automatically catalogued when a feature is added to the QBIC catalogue for some column[4]. To solve this problem you will have to re-insert all previously inserted images in the column. Thus, run the following commands in the **Command Center**:

```
DELETE FROM EMPLOYEE

INSERT INTO employee VALUES (
    1,
    'flowers1',
    MMDBSYS.DB2IMAGE (
        CURRENT SERVER,
        'm:/db2account/IAV_DATA/flower1.gif',
        'ASIS',
        1,
        'a flower arrangement'
    )
)
```

This is only necessary for images that have not been catalogued before, for images that have been catalogued before it suffices to issue the command `CATALOG QBIC COLUMN FOR ALL`.

Which clears the entire employee table, and then inserts the same data as was in the table previously. The difference is that this time the QBIC catalog will be updated with feature data.

You are now ready to do some QBIC…

Doing some QBIC

16) Querying, with QbScoreFromStr, on average color provided explicitly. Run the following command in the **Command Center**:

```
SELECT name,
    MMDBSYS.QbScoreFromStr (
        'QbColorFeatureClass color=<0,255,0>',
        picture
    )
FROM employee
```

An explanation is clearly needed here.

QbScoreFromStr[5] is a UDF for DB2IMAGE data that takes a condition string (in this case 'QbColorFeatureClass color=<0,255,0>') and an image handle (in this case the one in the column picture) and returns a *score* indicating

[4] There is a command CATALOG QBIC COLUMN FOR ALL but this command only recatalogs images that have already been catalogued, it does not catalogue images that have not been catalogued before The manual actually indicates that all images are catalogued, regardless of whether they have been so before or not (see manual p 135). However, when I try it does not seem to work.

[5] QbScoreFromString in DB2 v5.0

how well the selected images match the condition in the condition string. This score is a value greater than or equal to 0 (zero), where *a lower score indicates greater resemblance to the condition*. In this case the images are scored according to how close to green their average color is. Thus, a very green image would have a score close to 0.

Of course, the call to the UDF QbScoreFromStr could have appeared in a condition in the WHERE clause just as well. Notice also that the condition is provided to the UDF as a string (hence the name QbScoreFrom**Str**).

17) Querying, with QbScoreFromStr, on an average color provided from another image. Run the following command in the **Command Center**:

```
SELECT name, MMDBSYS.QbScoreFromStr (
                 'QbColorFeatureClass
                      file=<server,m:/db2account/IAV_DATA/flower1.gif>',
                 picture
             )
FROM employee
```

This query displays all employee names and how close the average colours of employee images are to the average color of the image photo.jpg. The flag server indicates to DB2 that the image file is on the server. Instead of server, one may specify client, to make DB2 look for the file on the client computer.

18) Querying, with QbScoreFromStr, on histogram color provided explicitly. Run the following command in the **Command Center**:

```
SELECT name, MMDBSYS.QbScoreFromStr (
                 'QbColorHistogramFeatureClass
                     histogram=<(10,255,0,0),(50,0,255,0),(40,0,0,255)>',
                 picture
             )
    FROM employee
```

The score returned from the UDF, for each image in the picture column, indicates how well the image correspond to the specified histogram. The histogram in this example, states that the image is 10% red, 50% green and 40% blue. Lower scores indicate higher resemblance.

As with querying on average color, it is also possible to specify a file outside the database instead of a specific histogram. The next point demonstrates this.

19) Querying, with QbScoreFromStr, on a histogram color provided from another image. Run the following command in the **Command Center**:

```
SELECT name, MMDBSYS.QbScoreFromStr(
                 'QbColorHistogramFeatureClass
                     file=<server,m:/db2account/IAV_DATA/flower1.gif>',
                 picture
             )
FROM employee
```

Here a histogram is first automatically extracted from the image photo.jpg and the used to compute the score of each image in the column.

20) Querying on Positional Color with QbScoreFromStr. Run the following command in the **Command Center**:

```
SELECT name,
    MMDBSYS.QbScoreFromStr (
        'QbDrawFeatureClass
            file=<server,m:/db2account/IAV_DATA/flower1.gif>',
        picture
    )
FROM employee
```

This query demonstrates how to query on positional color. Unlike the previous UDFs, it is not possible to give any kind of explicit specify of positional color in the query. Rather, one must always refer to an image file.

21) Querying on Texture with QbScoreFromStr. Run the following command in the **Command Center**:

```
SELECT namn,
       MMDBSYS.QbScoreFromStr (
             'QbTextureFeatureClass
                 file=<server,m:/db2account/IAV_DATA/flower1.gif>',
             picture
       )
FROM employee
```

This query demonstrates how to query on texture. Again, it is not possible to explicitly provide a texture specification in the condition string, the texture must be provided as an image file.

More about QBIC

In the previous section, you tried some examples on QBIC querying. This section will describes some additional functionality plus some more in depth description about DB2 image extender. Although some examples are given, you are not required to try them out. You may, however, need to at least skim this section in order to solve the assignments.

Using the UDF QbScoreTBFromStr

The UDF QbScoreTBFromStr returns a table with image handles and scores, according to a specified string condition, for all images in a column. The columns of the returned table are IMAGE_ID (type DB2IMAGE) and SCORE (type DOUBLE), respectively. The score is, as before, from zero and up, where low values indicate greater resemblence[6]. If a score is –1 then the corresponding image has not been catalogued. The following query returns the ten darkest images (the ones with color closest to black):

```
SELECT name
FROM employee
WHERE CAST (picture AS VARCHAR(250)) IN
    ( SELECT CAST (IMAGE_ID AS VARCHAR(250) ) FROM
      TABLE ( MMDBSYS.QbScoreTBFromStr (
                 'QbColorFeatureClass color=<0,0,0>',
                 'employee',
                 'picture',
                 10
            )
    )
    AS score_table)
```

Notice that the table and column names are provided as strings.

[6] The returned table is actually ordered descendingly on SCORE.

Multiple feature conditions

It is possible to specify more than one conditions in a condition string. The following example demonstrates.

```
SELECT name, MMDBSYS.QbScoreFromStr (
                'QbColorHistogramFeatureClass
                    file=<server,m:/db2account/IAV_DATA/flower1.gif>
                    and QbColorFeatureClass color=<60,12,0>',
                picture
            )
FROM employee
```

In this query images are scored against a histogram from image photo.jpg as well as against the average color with RGB values 60, 12 and 0 respectively. It is *not* possible to specify an ”or”condition. See the next subsection for a description of how the score is calculated.

Weighing features

When a multiple feature condition is used, as in the previous subsection, the score returned from the UDF will be the sum of the scores of each individual condition divided by the number of conditions. For instance, in the example in the previous subsection, if the score for QbColorHistogramFeatureClass file=<server,c:\images\photo.jpg> is 3.76 and the score for QbColorFeatureClass color=<60,12,0> is 8.07, the total score is (3.76+8.07)/2 = 5.92.
Sometimes, however, it may be desirable to give higher weight to one of the conditions. This can be done by specifying weights in the condition string, as the following example demonstrates:

```
SELECT name, MMDBSYS.QbScoreFromStr (
                'QbColorHistogramFeatureClass
                    file=<server,m:/db2account/IAV_DATA/flower1.gif> weight=1.0
                    and QbColorFeatureClass color=<60,12,0> weight=2.0',
                picture
            )
FROM employee
```

In this example, the second condition will have twice as much impact on the score as the first condition. For instance, if the individual scores are 3.76 and 8.07, as before, then the total score will be (1.0*3.76+2.0*8.07)/2 = 9.95.

A note about displaying retrieved images

It would be nice to be able to display images directly in the query result. This would, however, require a special application program as Command Center cannot do it. Unfortunately, no such application is provided with the DB2 extenders. Of course, anyone with decent programming skills and enough time can write one (IBM does not necessarily have to do this.)

Audio data

Set-up for audio data

22) Enable the database for audio data. Run the following commands in the **db2ext** command line processor:

db2ext => `ENABLE DATABASE FOR DB2AUDIO`

23) Add the audio column: Run the following commands in the **db2** command line processor[7]:

db2 => `ALTER TABLE employee ADD voice MMDBSYS.DB2AUDIO`

[7] For some reason, this seems not to work in the **Command Center**.

24) Enable the table for audio data: Run the following commands in the **db2ext** command line processor:

db2ext => `ENABLE TABLE employee FOR DB2AUDIO`

This causes the audio extender to i) identifies the employee table for use ii) create administrative support tables that hold attribute information for audio object in enables columns.
Note that it is *not* possible to write ”mmdbsys.db2audio”.

25) Enable the column for audio data: Run the following commands in the **db2ext** command line processor:

db2ext => `ENABLE COLUMN employee voice FOR DB2AUDIO`

Causes audio extender to identify the voice column for use and creates some triggers. These triggers update the adm sup tables in response to insert, update and delete operations on the employee table.

26) Check extender status (optional step):

db2ext => `GET EXTENDER STATUS`

Should show something like this:

```
EXTENDER TABLESPACE                          TABLE
-------- ----------------------------------- --------------------------
DB2IMAGE USERSPACE1,USERSPACE1,USERSPACE1 db2account .EMPLOYEE
DB2AUDIO USERSPACE1,USERSPACE1,USERSPACE1 db2account .EMPLOYEE
DMB0024I: The current database is enabled for "2" extenders.
```

Inserting audio data

27) Insert audio data into the table. In this example, we also insert an image. Run the following command in the **Command Center**:

```
UPDATE EMPLOYEE
SET voice = MMDBSYS.DB2AUDIO (
                CURRENT SERVER,
                'm:/db2account/IAV_DATA/sound1.wav',
                'WAV',
                1,
                'A sound'
            )
WHERE id=1
```

The parameters are similar to insertion of image data, where, for instance, the '1' signals that the audio clip should be copied into the database etc.

Querying audio data

28) A simple query on audio data. Run the following query in the **Command Center**.

```
SELECT MMDBSYS.FORMAT(voice)
FROM employee
WHERE id=1
```

The result is the names of all employees and size of their images. This is a very simple query. There is no

functionality corresponding to QBIC for audio, nor for video.

Video data

Set-up for video data

29) Enable database for video data: Run the following command in the **db2ext** command processor:

db2ext => `ENABLE DATABASE FOR DB2VIDEO`

30) Add the video column: Run the following command in the **db2** command processor:

db2 => `ALTER TABLE employee ADD greeting MMDBSYS.DB2VIDEO`

31) Enable table for video data: Run the following command in the **db2ext** command processor:

db2ext => `ENABLE TABLE employee FOR DB2VIDEO`

32) Enable column for video data: Run the following command in the **db2ext** command processor:

db2ext => `ENABLE COLUMN employee greeting FOR DB2VIDEO`

33) Check the extender status (optional step):

db2ext => `GET EXTENDER STATUS`

The result should look something like this, provided you have performed all the steps in this document. DB2100 will be replaced by your user id.

```
EXTENDER TABLESPACE                          TABLE
-------- ----------------------------------- --------------------------
DB2IMAGE USERSPACE1,USERSPACE1,USERSPACE1 db2account .EMPLOYEE
DB2AUDIO USERSPACE1,USERSPACE1,USERSPACE1 db2account .EMPLOYEE
DB2VIDEO USERSPACE1,USERSPACE1,USERSPACE1 db2account .EMPLOYEE
DMB0024I: The current database is enabled for "3" extenders.
```

Inserting video data

34) Insert video data. In this example we also insert an image and an audio clip. Run the following command in the **Command Center**:

```
UPDATE EMPLOYEE
SET greeting = MMDBSYS.DB2VIDEO (
                    CURRENT SERVER,
                    'm:/db2account/IAV_DATA/sound1.wav',
                    'AVI',
                    1,
                    'A video clip'
               )
WHERE id=1
```

This is analogous to insertion of image and audio data.

Querying video data

35) Simple query on video data: Run the following command in the **Command Center**:

```
SELECT name, MMDBSYS.FORMAT(greeting)
FROM employee
WHERE id=1
```

UDB also has functionality for detecting video scene changes in supported video formats. A scene change occurs when two consecutive images in a video sequence differ significantly from each other. This indicates that a clip occurred in the video sequence. However, this functionality is available only in embedded SQL, and not through ad hoc querying. Thus, you will not be able to test this in this tutorial.

Questions

Now that you have completed the introductory steps, it is time for you to answer the questions.

The purpose of this laboration, up till now, has been to show how to manage IAV data in DB2, step by step. For simplicity, you have only managed one single row in the table you created. You have added an image, an audio clip and a video clip to this row. Before you proceed with the questions, you will first add 50 additional rows of IAV data to your table. To do this, proceed as follows:

Preparations

1) Change to your **db2** command line processor window.

2) Issue the `QUIT` command in that window. You now come to the dos prompt in that window.

3) Change directory to `IAV_DATA`, by issuing

   ```
   cd m:\db2account\IAV_DATA
   ```

   followed by

   ```
   m:
   ```

4) Issue the following command

   ```
   insert_flowers
   ```

   This causes a batch file, insert_flowers.bat, in the current directory to be runned. This batch fills your table with IAV data.

5) When the batch has stopped, start the **db2** command line processor again, by issuing

   ```
   db2
   ```

   Note: You do *not* have to connect to IAV_LAB again.

Questions

Your database contains a set of images of flower arrangements, with some audio and video clips as well.

1) What is the duration, in seconds, of the audio clip associated with flower arrangement with id=15 ? (show the query also)

2) How many frames does the video clip associated with the flower arrangement with id=32 consist of ? (show the query also)

3) Which flower arrangement is the most red ? (also show the query that finds this arrangement)

4) In your `IAV_DATA` directory, you will find a bmp image with a hand-sketch of a desired flower arrangement in the file `sketch.bmp`. What flower arrangement in the database has color distribution most similar to the sketch ? (also show the query that finds this arrangement)

5) Describe how would you go about finding which two flower arrangements that are most similar to each other in the PICTURE column. Can you find any particular limitation of QbScoreFromStr and QbScoreTBFromStr in relation

to this ? (Hint: At least one of the images compared in a QbScoreFromStr and QbScoreTBFromStr must be specified as a filename.)

Once your answers have been OK:ed by your seminar leader, you should delete all your work and stop the extender services. This is described in the following, final chapters.

Deleting the table and the database

36) Disable column, table and database. Run the following commands in the **db2ext** command line processor:

```
db2ext => OPEN QBIC CATALOG employee picture
db2ext => DELETE QBIC CATALOG employee picture
db2ext => CLOSE QBIC CATALOG

db2ext => DISABLE COLUMN employee picture FOR DB2IMAGE
db2ext => DISABLE COLUMN employee picture FOR DB2AUDIO
db2ext => DISABLE COLUMN employee picture FOR DB2VIDEO

db2ext => DISABLE TABLE employee FOR DB2IMAGE
db2ext => DISABLE TABLE employee FOR DB2AUDIO
db2ext => DISABLE TABLE employee FOR DB2VIDEO

db2ext => DISABLE DATABASE FOR DB2IMAGE
db2ext => DISABLE DATABASE FOR DB2AUDIO
db2ext => DISABLE DATABASE FOR DB2VIDEO
```

This causes the extender services to delete all meta tables for the columns, including the QBIC catalogue, and also to delete the meta tables for the table and the database.

37) Delete the table and the database. You may now delete the table, and the database. You can do this using the **Control Center**.

Shutting down the extender services

38) And, finally, When extender service is no longer needed, it should be stopped by issuing the following command at the prompt on the server:

C:\>DMBSTOP

This will disable all extender-enabled databases. When the extender services is started again the databases will be automatically enabled again.

You have now completed the IAV laboration, congratulations!

Appendix: short reference

Managing extender services

Issued at prompt on database server:

Syntax	Description
DMBSTART	Start extender services on server
DMBSTOP	Stop extender services on server
DMBSTAT	Status of extender services on server
GET SERVER STATUS	Show status of extender server

UDTs

The following UDT for IAV data is provided by the extender:

UDT	Data type
DB2IMAGE	VARCHAR(250)
DB2AUDIO	VARCHAR(250)
DB2VIDEO	VARCHAR(250)

UDFs

The following three tables show all available IAV UDFs, for image, audio and video respectively. Many of these UDFs have many alternative syntaxes. Such syntaxes are not shown here, only the simplest syntaxes are shown. All of these UDFs must be preceeded by ”MMDBSYS.” when called.

Image UDFs		
UDF	Short explanation	Sample syntax, description
Comment	Manage user comment	See manuals.
Content	Image content	See chapter "Image data / Extract image data"
DB2Image	Store image	See chapter "Image data / Inserting image data"
Filename	Image filename	Filename(*handle*)
Format	'GIF', for instance	Format(*handle*)
Height	Heigh in pixels	Height(*handle*)
Importer	ID of Image importer	User ID of importer of image
Importertime	Image import time	Importertime(*handle*)
NumColors	Num. colors in image	NumColors(*handle*)
QbScoreFromStr	Image similarity	See chapter "Image data / Doing some QBIC"
QbScoreTBFromStr	Images similari*ties*	See chapter "Image data / More about QBIC"
QbScoreFromName	Image similarity	As QbScoreFromStr but with named query. Used in embedded SQL.
QbScoreTBFromName	Images similari*ties*	As QbScoreTBFromName but with named query. Used in embedded SQL.
Replace	Update image content	See manuals.
Size	Image size, in bytes	Size(*handle*)
Thumbnail	Image as thumbnail	Thumbnail(*handle*)
Updater	ID of Image updater	Get the user ID of the updater of the image
Updatetime	Image update time	Get the time when an image was updated
Width	Width in pixels	Width(*handle*)

Audio UDFs		
UDF	Short explanation	Sample syntax, desciption
AlignValue	Bytes per sample	AlignValue(*handle*)
BitsPerSample	Bits per sample	BitsPerSample(*handle*)
BytesPerSec	Bytes per second	BytesPerSec(*handle*)
Comment	Manage comments	See manuals.
Format	File format	Format(*handle*)
Content	Audio clip content	Analogous to Content for DB2IMAGE. See See chapter "Image data / Extract image data"
DB2Audio	Store audio clip	See chapter "Audio data / Inserting audio data"
Duration	Playing time (sek)	Duration(*handle*)
Filename	Filename	Filename(*handle*)
FindInstrument	Track# of instrument	FindInstrument(*handle*,instrName), instrName is a varchar. Only for MIDI files.
FindTrackName	Track# of named track	FindTrackName(*handle*,trackName), trackName is a varchar. Only for MIDI files.
GetIntruments	All instrument names	Return names of all instruments in a MIDI file. See manuals for details.
GetTrackNames	All track names	Return names of all tracks in a MIDI file. See manual for details.
Importer	User ID of importer	Importer(*handle*)
ImportTime	Time of import	ImportTime(*handle*)
NumAudioTracks	Number of tracks	NumAudioTracks(*handle*). Only for MIDI files.
NumChannels	Number of channels	Only for WAV or AIFF.
Replace	Update audio contents	See manuals.
SamplingRate	Sampling rate (Hz)	SamplingRate(*handle*)
Size	Size in bytes	Size(*handle*)
TicsPerQNote	Recorded clock speed	TicsPerQNote(*handle*). In tics per quarter note. Only for MIDI files.
TicksPerSec	Recorded clock speed	TicksPerSec(*handle*). In tics per second. Only for MIDI files.
Updater	ID of audio updater	Updater(*handle*). Get the user ID of the updater of the audio
UpdateTime	Audio update time	UpdateTime(*handle*). Get the time when the audio was updated.

Video UDFs		
UDF	Purpose	Syntax
AlignValue	Bytes per sample	AlignValue(*handle*)
AspectRatio	Aspect ratio	AspectRatio(*handle*). Aspect ration of first track in video clip.
BitsPerSample	Bits per sample	BitsPerSample(*handle*)
Comment	Manage comments	See manuals.
CompressType	Compression type	CompressType(*handle*). Returns varchar(8). For example, ”MPEG-1”
Content	Video clip content	Analogous to Content for DB2IMAGE. See See chapter "Image data / Extract image data"
DB2Video	Store video clip	See chapter "Video data / Inserting video data"
Duration	Playing time (sek)	Duration(*handle*)
Filename	Filename	Filename(*handle*)
Format	File format	Format(*handle*)
FrameRate	In frames per second	FrameRate(*handle*)
Height	Height in pixels	Height(*handle*)
Importer	ID of importer	Importer(*handle*)
MaxBytesPerSec	Maximal troughput	MaxBytesPerSec(*handle*). Maximal troughput of a video, in bytes per second.
NumAudioTracks	# of audio tracks	NumAudioTracks(*handle*)
NumChannels	# channels	NumChannels(*handle*). Number of recorded audio channels.
NumFrames	Frame count	NumFrames(*handle*) number of frames in video.
NumVideoTracks	NumVideoTracks	NumVideotracks(*handle*).
Replace	Update video content	See manuals.
SamplingRate	Sampling rate (Hz)	SamplingRate(*handle*)
Size	Size in bytes	Size(*handle*)
Thumbnail	Video as thumbnail	Thumbnail(*handle*)
Updater	ID of updater	Updater(*handle*)
UpdateTime	Time of update	UpdateTime(*handle*)
Width	Width in pixels	Width(*handle*)

QBIC catalog commands

Issued at the db2ext command line processor (**`db2ext =>`**). Some of these commands can take additional parameters as well (see manual for details). For example, ENABLE DATABASE can take several IAV types separated by comma. Notation used in the table:

t = the table name

c = the column name, *db* = database name.

[*a*|*b*] = either *a* or *b* (not both)

featureName = one of QbColorFeatureClass, QbColorHistogramFeatureClass, QbDrawFeatureClass or QbTextureFeatureClass. It is also possible to specify AverageColor instead of QbColorFeatureClass, Histogram instead of QbHistogramFeatureClass, Draw instead of QbDrawFeatureClass and Texture instead of QbTextureFeatureClass. The result is the same, but the queries become slightly more readable.

Command
CONNECT TO *db*
ENABLE DATABASE FOR [DB2IMAGE | DB2AUDIO | DB2VIDEO]
DISABLE DATABASE FOR [DB2IMAGE | DB2AUDIO | DB2VIDEO]
ENABLE TABLE *t* FOR [DB2IMAGE | DB2AUDIO | DB2VIDEO]
DISABLE TABLE *t* FOR [DB2IMAGE | DB2AUDIO | DB2VIDEO]
ENABLE COLUMN *t c* FOR [DB2IMAGE | DB2AUDIO | DB2VIDEO]
DISABLE COLUMN *t c* FOR [DB2IMAGE | DB2AUDIO | DB2VIDEO]
CREATE QBIC CATALOG *t c* [ON|OFF]
DELETE QBIC CATALOG *t c*
OPEN QBIC CATALOG *t c*
SET QBIC AUTOCATALOG [ON|OFF]
ADD QBIC FEATURE *featureName*
REMOVE QBIC FEATURE *featureName*
GET QBIC CATALOG INFO
CATALOG QBIC COLUMN FOR ALL
CLOSE QBIC CATALOG
DELETE QBIC CATALOG *t c*
QUIT

Recognized IAV formats

The tables below lists the IAV data formats recognized by DB2 UDB, and also specifies if read and/or write of the format is supported. Notice that the format of an image need not be recognized for it to be saved in the database. But if it is not recognized all the feature data must be provided by the user. If ”Read” is supported it means that UDB DB2 can extract feature data from the format. If ”Write” is supported it means that UDB DB2 can convert to that format (from a format that it can ”Read”).

Image formats (√=supported)		
Format	Read	Write
_IM	√	
BMP	√	√
EPS		√
EP2		√
GIF	√	√
IMG	√	√
IPS	√	√
JPG	√	
PCX	√	√
PGM	√	√
PS		√
PSC		√
PS2		√
TIF	√	√
YUV	√	√

Audio formats (√=supported)		
Format	Read	Write
AIF/AIFF	√	
AIFFC	√	
AU	√	
MIDI	√	
MPG1 / MPEG1	√	
WAV / WAVE	√	

Video formats (√=supported)		
Format	Read	Write
AVI	√	
MPG1 / MPEG1	√	
MPG2 / MPEG2	√	
QT	√	