MonetDB/XQuery Installation and Running

Disclaimer

This handout refers to procedures for installing and starting to use MonetDB server Version 4 with XQuery support. This information is correct as of **October 10, 2007**. Due to an upcoming release of a new version of MonetDB/XQuery, some of this information may soon become out-of-date.

MonetDB/XQuery: overview

MonetDB is a lightweight DBMS designed by a team of researchers at CWI (Center for Mathematics and Computer Science, the Netherlands). MonetDB is an open-source product with licensing terms derived from *Mozilla General License*.

MonetDB uses technology called BATs - Binary Association Tables to store data on disk. This approach is drastically different from traditional relational DBMS storage. MonetDB server accepts communications in its internal language called MIL.

MonetDB comes with a a client program (`MapiClient`) which allows for the use of `mil`, SQL or XQuery query langauge to access data stored in the database. There are two versions of the MonetDB server. Due to current implementation status, MonetDB/XQuery (i.e., MonetDB server plus MonetDB XQuery client) works only with **MonetDB Server Version 4**.

MonetDB/XQuery provides full support for storage of XML data and is fully XQuery-compliant. In addition (XQuery is a query language, it does not change the state of the database), MonetDB server allows (using mil) to insert/delete XML files to/from XML databases maintained by the MonetDB server.

MonetDB/XQuery installation

MonetDB/XQuery is available under both Linux and Windows platforms.

Linux Installation

There are two ways to install MonetDB on a linux system: using RPM packages or compiling from sources.

To install RPM packages root access is required, and thus, this route is unavilable for CSL installations.

Installation from source is simple and can be done on any CSL machine [1].

- Download the installation script. The original is available at

  http://monetdb.cwi.nl/projects/monetdb//download.php?target=/projects/monetdb//Assets/monetdb-install.sh

  A local copy of this file is available from the course web page:

  http://www.csc.calpoly.edu/~dekhtyar/560-Fall2007/monetdb-install.sh

  and in linked to from the `index.html` file.

- Create a directory for MonetDB installation. I use `~/MonetDB`. Copy `monetdb-install.sh` to this directory.

- Run

  ```
  > monetdb-install.sh --help
  ```

  You should see the following output:

  ```
  usage: monetdb-install.sh < OPTS ... >

  where OPTS are:
     --prefix=path     install into location path, defaults to ~/MonetDB
     --build=path      use path as (temporary) build directory, defaults
                       to /var/tmp/MonetDB-XXXXXXXXX
     --enable-sql      build the MonetDB/SQL server
     --enable-xquery   build the MonetDB/XQuery server
     --nightly=target  download and install a nightly snapshot of the stable
                       or current branch, target must be 'stable' or 'current'
     --cvs             checkout a CVS snapshot of the current branch
     -j[X]             use parallel make with the optionally given limit
     --enable-debug    compile with debugging support via e.g. gdb
     --enable-optimise compile with high optimisation flags, enabling this
                       option increases compilation time considerably but
                       often yields in a faster MonetDB server
     --enable-optimize alias for --enable-optimise
     --quiet           suppress output going to stdout
     --help            this message
     --devhelp         special help for developers
     --version         show revision number and quit
  ```

- Run

  ```
  > monetdb-install.sh --enable-xquery
  ```

  This starts the process of installation of MonetDB/XQuery. The installation is performed from a stable build checked out of MonetDB CVS repository. The download of source files, compilation and installation take a bit of time (anywhere from 10 to 30 mins, depending on workstation workload).

  When the script stops running, MonetDB/XQuery is installed on your Linux system!

[1] Please note, that configurations on two legacy CSL servers: `falcon` and `hornet` are different from the configurations on the new machines, and servers such as `vogon`. This leads to the fact that the linux installation from source will work only on one of the systems: either on vogon/rest of the CSL machines or on falcon/hornet **only**. I recommend installing on a "regular" CSL machine.

Windows Installation

Windows version of MonetDB comes in a different package, and some executables have different file names. While this section provides directions to installing MonetDB/XQuery on a Windows system, in the rest of the handout, I will refer to program names used in the Linux distribution.

- Open the following URL in a browser:

  http://monetdb.cwi.nl/projects/monetdb//Download/

  (linked to from the course web page.)

- Download `MonetDB4-XQuery-i686-0.18.4.msi` file or MonetDB4-XQuery-x86_64-0.18.4.msi for 64-bit systems.

- Run `MonetDB4-XQuery-i686-0.18.4.msi`. Follow the installation instructions (directory selection, etc...).

- once the installer finished working, MonetDB/XQuery is installed on your system. By default, the `Start` menu will contain a directory called `MonetDB`, which includes both the client and the server launch links.

Starting the work with MonetDB/XQuery

Note: For some reason, I am still not able to **shred** (i.e., **load**) into a MonetDB database an XML file residing locally in the file system. At the same time, I can shred any XML available on the world wide web file via the http protocol.

I will assume that MonetDB is installed in `~/MonetDB`.

- change directory to `~/MonetDB/bin`. `ls` the directory.

- Two executable files are of interest to us: `Mserver` and `MapiClient`. The first executable starts the server, the second executable is the query language front-end (client).

- run

  ```
  > Mserver --help
  ```

  The following will be returned:

  ```
  sage: Mserver [ options ] [ script+ ]
  Options are:
   -c <config_file>    | --config=<config_file>
                         --dbname=<database_name>
                         --dbfarm=<database_directory>
                         --dbinit=<MIL_statement>
   -d<debug_level>     | --debug=<debug_level>
   -s <option>=<value> | --set <option>=<value>
   -?                  | --help
  ```

- Out of the options above, we only need to worry about two for now. `--dbname` specfies the name of the database, which MonetDB server will use for the session (MonetDB server creates a new database if database with a given name does not exist). Anoter option, `dbinit` will allow us to start the server in desired state (rather than typing individual commands upon starting the MonetDB server). All other options have appropriate default values.

- to start Mserver:

  ```
  > Mserver --dbname = test
  ```

  (you may use any name you like). As a result, you should see something like this:

  ```
  !WARNING: GDKlockHome: created directory /home/dekhtyar/MonetDB/var/MonetDB4/dbfarm/test01/
  !WARNING: GDKlockHome: ignoring empty or invalid .gdk_lock.
  !WARNING: BBPdir: initializing BBP.
  # Monet Database Server V4.18.2
  # Copyright (c) 1993-2007, CWI. All rights reserved.
  # Compiled for i686-redhat-linux-gnu/32bit with 32bit OIDs; dynamically linked.
  # Visit http://monetdb.cwi.nl/ for further information.
  MonetDB>
  ```

  (the first three lines appear only for a new database.)

  The server starts its operation.

- Open another terminal/xterm and change to the "*sim*/MonetDB/bin" directory.

- run

  ```
  > MapiClient --help
  ```

  You should see the following:

  ```
  Usage: MapiClient [ options ]
  Options are:
   -b t/f      | --blocked=true/false /* blocked mode */
   -c config   | --config=file    /* config filename */
   -C colname  | --collection=name /* collection name */
   -e          | --error          /* exit on error */
   -H          | --history        /* load/save cmdline history (default off) */
   -h hostname | --host=hostname  /* host to connect to */
   -i          | --interactive    /* read stdin after command line args */
   -l language | --language=lang  /* {mal,sql,mil,xquery} */
   -P passwd   | --passwd=passwd  /* password */
   -p portnr   | --port=portnr    /* port to connect to */
   -d database | --database=database /* database to connect to */
   -o format   | --output=format  /* output format for xquery (dm (default) or xml) */
   -q          | --quiet          /* don't print welcome message */
   -s stmt     | --statement=stmt /* run single statement */
   -T          | --time           /* time commands */
   -t          | --trace          /* trace mapi network interaction */
   -u user     | --user=user      /* user id */
   -?          | --help           /* show this usage message */
  ```

  The default values of the hostname (`localhost`), portname (50000) etc. are ok for us. One parameter that must be included is the query langauge MapiClient should be using. Four choices are given, two of them are of interest:

  `>MapiClient -l xquery` starts the MonetDB client prepared to deal with XQuery statements.

  `>MapiClient -l mil` starts the client using MonetDB's internal query langauge mil.

- Switch to the Mserver window. Prior to starting the client, the serve must be prepared to accept transactions from the client programs. This is achieved using the following sequence of commands (included here with the output):

```
MonetDB>module(pathfinder);
# XRPC administrative console at http://localhost:50001/admin
MonetDB>mil_start();
```

`module` command in `mil` uploads optional mil packages to the server. `Pathfinder` is the name of the XQuery query processing engine developed at CWI.

`mil_start()` prepares the server to accept incoming client inquiries.

- In the client window, run

  ```
  > MapiClient -l xquery
  xquery>
  ```

  The `xquery>` promt shows that the connectivity with the server has been established.

After this sequence of actions, you should have running MonetDB server and MapiClient.

Populating the database

XQuery is a query language, it does not by itself alter the state of an XML database. To insert new information into a database, one has to use mil. Data insertion can be done in two ways:

- running mil commands from within the MonetDB server.
- running mil commands from `MapiClient` which was started as follows:

  ```
  >MapiClient -l mil
  ```

In both cases, the mil command to put a new file into the database is the same:

shred_doc(URI, *InternalName*);

URI is the location of the XML file to be shredded and inserted into the database. *InternalName* is the name of the resource (external file) which will be used to refer to the XML document being shredded. *InternalName* can be used inXQuery startements.

Note: At this point the only acceptable *URI*s are **URL**s of the resources available through HTTP protocol.

The following command shreds a simple `hello.xml` file from my web page:

```
MonetDB>shred_doc("http://www.csc.calpoly.edu/~dekhtyar/hello.xml","hi.xml");
# Elapsed time = 167ms 440us [010ms 465us/node]
# Shredded 1 XML document (hi.xml), total time after commit=0.856s
```

If the file was found, retrieved and shredded successfully, you will recieve the message similar to the one above.

Querying the database

MapiClient running in XQuery mode will accept any valid XQuery query as input. Similar to Oracle's SQL*plus environment, MapiClient provides command-line interface and allows for entry of a single XQuery query into more than one line.

To complete a query in MapiClient, on linux move to a new line and then hit `<Ctrl>+D`. On Windows, the combination is `<Ctrl>+Z`.

Recall that XPath expressions are valid XQuery queries. To ensure that each XPath expression is applied to a specific XML document, we preface each XPath expression is a call to XQuery's built-in function `doc(DocName)`.

Consider the following simple XML file:

```
<?xml version="1.0" encoding="utf-8"?>
  <doc>
   <greet kind="informal">Hi</greet>
   <greet kind="casual">Hello</greet>
   <location kind="global">World</location>
   <location kind="local">Amsterdam</location>
  </doc>
```

The following is a sample session with MonetDB through the XQuery client.

```
xquery>doc("hi.xml")
more><?xml version="1.0" encoding="utf-8"?>
<doc>
   <greet kind="informal">Hi</greet>
   <greet kind="casual">Hello</greet>
   <location kind="global">World</location>
   <location kind="local">Amsterdam</location>
  </doc>

xquery>doc("hi.xml")/child::*/child::*
more><greet kind="informal">Hi</greet>,
<greet kind="casual">Hello</greet>,
<location kind="global">World</location>,
<location kind="local">Amsterdam</location>

xquery>doc("hi.xml")//greet[following-sibling::greet]
more><greet kind="informal">Hi</greet>
```