EyeDB Administration Guide

COLLABORATORS			
	TITLE : EyeDB Administration Guide		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	François Déchelle	May 27, 2010	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

List of Figures

List of Tables

List of Examples

Chapter 1

Introduction

1.1 Administration of the EyeDB object database management system

As for every database management system, using EyeDB implies some administration operations, ranging from the simplest ones, such as creating or deleting a database, to the most advanced, such as managing objects and indexes locations for improved performances.

This manual describes in details all EyeDB administration operations and include the following topics:

- server administration: how to start and stop the server, get its status, configure it...
- database administration: creating and deleting databases, renaming, moving, exporting databases...
- user administration: adding and deleting users, managing user rights...
- advanded database administration: managing data partitionning
- database tuning: how to create large databases, managing indexes, collections and data locations for better performance

This manual is directed to a large audience, ranging from regular users experimenting with EyeDB to database administrators managing large databases. The first three chapters (server, database and user administration) are usefull for every EyeDB users; the last two chapters (advanced administration and tuning) are directed to advanced EyeDB users who want to understand in-depth databases organisation and tune their EyeDB installation for large databases and increased performances.

Most of the EyeDB administration operations are performed using the **eyedbadmin** command. This command will be described briefly in the next section.

1.2 The eyedbadmin command

The **eyedbadmin** command is used for most of the EyeDB administration operations. It is installed in `/usr/bin` for standard EyeDB installations.

1.2.1 eyedbadmin command syntax

The **eyedbadmin** command has the following syntax:

`eyedbadmin` *`topic`* *`command`* options... where *`topic`* is one of the subsystem adressed by the command (database, user, index...) and *`command`* is the command relative to this subsystem.

Help on the **eyedbadmin** command can be obtained with the **help** command for general help (Example 1.1) or ***`topic`*** **help** (Example 1.2):

Example 1.1 eyedbadmin help

```
eyedbadmin help
eyedbadmin usage:
  eyedbadmin --help
  eyedbadmin --help-eyedb-options
  eyedbadmin TOPIC --help
  eyedbadmin TOPIC COMMAND --help
  eyedbadmin TOPIC COMMAND OPTIONS

where TOPIC is one of the following:
  database | db
  datafile | dtf
  dataspace | dsp
  user | usr
  index | idx
```

Example 1.2 eyedbadmin help on topic

```
eyedbadmin database help
eyedbadmin database usage:

  eyedbadmin database --help
  eyedbadmin database COMMAND OPTIONS

where COMMAND is one of the following:
  create
  delete
  list
  move
  copy
  rename
  defaccess
  export
  import
  setobjcount
  getobjcount
  setlogsize
  getlogsize
```

1.2.2 eyedbadmin standard options

eyedbadmin accepts a list of options shared by all the topics, described in Example 1.3.

Example 1.3 eyedbadmin help on topic

```
eyedbadmin --help-eyedb-options
Common Options:
  -U USER|@, --user=USER|@          User name
  -P [PASSWD], --passwd[=PASSWD]    Password
  --host=HOST                       eyedbd host
  --port=PORT                       eyedbd port
  --inet                            Use the tcp_port variable if port is not set
  --dbm=DBMFILE                     EYEDBDBM database file
  --conf=CONFFILE                   Client Configuration file
  --server-conf=CONFFILE            Server Configuration file (used for local opening)
  --default-file-mask=MASK          Default file mask
  --default-file-group=GROUP        Default file group
  --maximum-server-memory-size=SIZE Maximum server memory size (in Mb)
  --logdev=LOGFILE                  Output log file
  --logmask=MASK                    Output log mask
  --logdate=on|off                  Control date display in output log
  --logtimer=on|off                 Control timer display in output log
  --logpid=on|off                   Control pid display in output log
  --logprog=on|off                  Control progname display in output log
  --error-policy=VALUE              Control error policy: status|exception|abort|stop|echo
  --trans-def-mag=MAGORDER          Default transaction magnitude order
  --arch                            Display the client architecture
  -v, --version                     Display the version
  --help-logmask                    Display logmask usage
  --help-eyedb-options              Display this message
```

The following options are the most usefull:

- --user=*user* : specifies the user name for EyeDB authentication; if user is "@", then the Unix user name is used
- --passwd=*passwd* : specifies the user password for EyeDB authentication
- --host=*host* : specifies the host name of the machine running the EyeDB server
- --port=*port* : specifies the TCP port number for a TCP connection to the EyeDB server

1.2.3 eyedbadmin shortcuts

A set of shortcuts have been defined for **eyedbadmin** most used topics and subcommands. These shortcuts are symbolic links, created at installation time and located in binaries installation directories. Shorcuts can be used as any shell command and accept the same arguments and options as their **eyedbadmin** equivalents.

The list of **eyedbadmin** shortcuts is the following:

Shortcut	**eyedbadmin equivalent**
eyedb_dbcreate	eyedbadmin database create
eyedb_dbdelete	eyedbadmin database delete
eyedb_dblist	eyedbadmin database list
eyedb_dbrename	eyedbadmin database rename
eyedb_dbexport	eyedbadmin database export
eyedb_dbimport	eyedbadmin database import
eyedb_dtfcreate	eyedbadmin datafile create
eyedb_dtfdelete	eyedbadmin datafile delete
eyedb_dtflist	eyedbadmin datafile list
eyedb_dspcreate	eyedbadmin dataspace create
eyedb_dspdelete	eyedbadmin dataspace delete
eyedb_dsplist	eyedbadmin dataspace list
eyedb_useradd	eyedbadmin user add
eyedb_userdelete	eyedbadmin user delete
eyedb_userlist	eyedbadminuser list
eyedb_userpasswd	eyedbadmin user passwd

Table 1.1: List of **eyedbadmin** shortcuts

Chapter 2

Server administration

The first step in EyeDB administration adresses the server administration: as all existing database management systems, EyeDB adopts a client/server architecture to provide databases integrity and concurrent accesses. Every database operation requires that a server is up and running; this chapter explains how to start and stop the server, get its status and manage its configuration.

2.1 The eyedbctl command

The EyeDB server is started, stopped and examined using the **eyedbctl** command. This command is installed in `/usr/sbin` for standard EyeDB installations.

The **eyedbctl** command has the following syntax:

`eyedbctl` *`command`* [options...] where **command** is one of **start**, **stop** or **status**.

The following options are the most usefull:

- --listen= [[*`hostname:`*] *`port`* | *`unix socket name`*] , ... specifies the port the server is listening on. This port can be either a TCP port, specified by a hostname (defaults to "localhost") and a TCP port number, or a Unix socket, specified by a file name.
- --access-file=*`file`* specifies the server access file for network configuration (see Section 2.5)
- --server-conf=*`file`* specifies the server configuration file (see Section 2.4.1)
- --nod no daemon mode, do not close file descriptors 0, 1 band 2

2.2 Starting and stopping the server

2.2.1 Starting EyeDB server

The EyeDB server is started using the following command:

`eyedbctl start` [options...]

Example 2.1 eyedbctl start

```
eyedbctl start
Starting EyeDB Server
 Version      V2.8.8
 Compiled     Feb 14 2009 00:16:45
 Architecture i686-pc-linux-gnu
 Program Pid  19433
```

By default, the EyeDB server listens to incoming client connections on 2 ports:

- a TCP/IP socket bound to port 6240
- a Unix socket bound to file name `/var/lib/eyedb/pipes/eyedbd` for standard installation, or `PREFIX/var/lib/eyedb/pipes/eyedbd` where `PREFIX` is the installation prefix that has been set during configuration

The listening ports can be set:

- using **eyedbctl** command options --listen (see eyedbctl --listen option [5])
- using server configuration file (see Section 2.4.1)

Example 2.2 eyedbctl start with --listen option

```
eyedbctl start --listen=localhost:10000,/var/tmp/eyedb-socket
Starting EyeDB Server
 Version      V2.8.8
 Compiled     Feb 14 2009 00:16:45
 Architecture i686-pc-linux-gnu
 Program Pid  19560
```

2.2.2 Stopping EyeDB server

The EyeDB server is stopped using the following command:

`eyedbctl stop` [options...]

Example 2.3 eyedbctl stop

```
eyedbctl stop
Killing EyeDB Server Pid 19433
```

If the server was started with the **--listen** option to set the listening port, the *same* option must be passed to **eyedbctl** when stopping the server.

Example 2.4 eyedbctl stop with --listen option

```
eyedbctl stop
No EyeDB Server is running on localhost:6240
eyedbctl stop --listen=localhost:10000,/var/tmp/eyedb-socket
Killing EyeDB Server Pid 19560
```

2.3 Getting server information

Once EyeDB server has been launched, information can be obtained on server status (server uptime, listening ports, clients...) using **eyedbctl**.

Independently from server status, the EyeDB server can be launched with logging option, that allows to trace precisely all server actions (client connections, transactions...).

2.3.1 Getting EyeDB server status

The EyeDB server status can be obtained using the following command:

`eyedbctl status` [options...] This command will print on its standard output informations about the EyeDB server currently running: server PID and user, version, listening ports and list of connected clients.

Example 2.5 eyedbctl status

```
eyedbctl status

EyeDB Server running since Thu Mar  5 16:08:45 2009

  Version       V2.8.8
  Date          Feb 14 2009 00:16:45
  Architecture  i686-pc-linux-gnu
  Program Pid   19433
  Running Under francois

  Listening on  localhost:6240
                localhost:/home/francois/projects/eyedb/install/var/lib/eyedb/pipes/eyedbd
  Datafile Directory /home/francois/projects/eyedb/install/var/lib/eyedb/db

  No Clients connected.
```

Same as for stopping server, if the server was started with the **--listen** option to set the listening port, the *same* option must be passed to **eyedbctl** when getting server status.

Example 2.6 eyedbctl status with --listen option

```
eyedbctl status --listen=localhost:10000,/var/tmp/eyedb-socket
EyeDB Server running since Thu Mar  5 16:09:07 2009

  Version       V2.8.8
  Date          Feb 14 2009 00:16:45
  Architecture  i686-pc-linux-gnu
  Program Pid   19560
  Running Under francois

  Listening on  localhost:10000
                localhost:/var/tmp/eyedb-socket
  Datafile Directory /home/francois/projects/eyedb/install/var/lib/eyedb/db

  No Clients connected.
```

If clients are connected, the server status will show the list of connected clients, giving for each client a list of open databases.

Example 2.7 eyedbctl status with connected clients

```
eyedbctl status
EyeDB Server running since Thu Mar  5 16:46:55 2009

  Version        V2.8.8
  Date           Feb 14 2009 00:16:45
  Architecture   i686-pc-linux-gnu
  Program Pid    20332
  Running Under  francois

  Listening on   localhost:6240
                 localhost:/home/francois/projects/eyedb/install/var/lib/eyedb/pipes/eyedbd
  Datafile Directory /home/francois/projects/eyedb/install/var/lib/eyedb/db

  1 Client connected

Client #0
  Connected on Thu Mar  5 16:53:01 2009
  Host:Port    localhost:/home/francois/projects/eyedb/install/var/lib/eyedb/pipes/eyedb
  User Name    francois
  Program Name eyedboql
  Client Pid   20377
  EyeDB Server Pid 20378
  Open Databases 'EYEDBDBM'  [mode=sread/local]
                   'EYEDBDBM'  [mode=sread/local]
                   'EYEDBDBM'  [mode=sread/local]
                   'test'  [mode=sread] [userauth=francois]
```

2.3.2 Configuring EyeDB server logging

The EyeDB server can log its activities with a fine grain control over what is logged (client connections, transactions, index...). This logging can be done into a file or into a special file such as `/dev/stderr`.

The following options of **eyedbctl** are used to control server logging:

- --logdev=*`logfile`* specifies output log file (can be either a plain file, or a special file like `/dev/stderr`
- --logmask=*`mask`* specifies log mask (see log mask values [8] for log mask values)
- --logdate= [on | off] controls display of date in output log
- --logtimer= [on | off] controls display of time in output log
- --logpid= [on | off] controls display of process pid in output log
- --logprog= [on | off] controls display of program name in output log

Log mask is either an hexadecimal number or a combination of symbols prefixed by + (to enable the corresponding log) or - (to disable the corresponding log). The list of allowed symbols can be obtained by running **eyedbctl start --logmask=help**. For instance, specifying **--logmask=+server+connection** will log all server and connection events.

Example 2.8 shows an example of launching EyeDB server with logging enabled, running an EyeDB command (here: **eyedbadmin**) and examining the log file (here `/var/tmp/eyedbd.log`) with the **tail**.

Example 2.8 Launching EyeDB server with logging enabled

```
eyedbctl start --logdev=/var/tmp/eyedbd.log --logdate=on --logpid=on --logprog=on --logmask ←
    =+server+connection
Starting EyeDB Server
 Version      V2.8.8
 Compiled     Feb 14 2009 00:16:45
 Architecture i686-pc-linux-gnu
 Program Pid  7484
eyedbadmin database list
...
tail -f /var/tmp/eyedbd.log
Thu Mar  5 19:19:04 2009 [thread 7484#-1216465216] eyedbd : rpc_multiConnManage 0
Thu Mar  5 19:19:04 2009 [thread 7484#-1216465216] eyedbd : rpc_multiConnManage 1
Thu Mar  5 19:19:04 2009 [thread 7484#-1216465216] eyedbd : rpc_multiConnManage 2
Thu Mar  5 19:19:04 2009 [thread 7484#-1216465216] eyedbd : new connection : fd = 0, fd =  ←
    1, fd = 2
Thu Mar  5 19:19:04 2009 [thread 7522#-1216465216] eyedbd : rpc_makeThread which=0, fd=0
CONN Thu Mar  5 19:19:04 2009 [thread 7522#-1225118832] eyedbd : new thread -1225118832 [fd ←
     = 0, which=0], stack = 0xb6fa239c
CONN Thu Mar  5 19:19:04 2009 [thread 7522#-1225118832] eyedbd : connected host='localhost ←
    :/home/francois/projects/eyedb/install/var/lib/eyedb/pipes/eyedbd', username='francois', ←
     progname='eyedbadmin', pid=7521
Thu Mar  5 19:19:04 2009 [thread 7522#-1216465216] eyedbd : rpc_makeThread which=1, fd=1
CONN Thu Mar  5 19:19:04 2009 [thread 7522#-1564058736] eyedbd : new thread -1564058736 [fd ←
     = 1, which=1], stack = 0xa2c6539c
Thu Mar  5 19:19:04 2009 [thread 7522#-1216465216] eyedbd : rpc_makeThread which=2, fd=2
CONN Thu Mar  5 19:19:04 2009 [thread 7522#-1580647536] eyedbd : new thread -1580647536 [fd ←
     = 2, which=2], stack = 0xa1c9339c
Thu Mar  5 19:19:04 2009 [thread 7522#-1564058736] eyedbd : -1564058736 thread EXIT
Thu Mar  5 19:19:04 2009 [thread 7522#-1564058736] eyedbd : rpc_garbClientInfo(which = 0,  ←
    fd = 1, ci = 0x9ce8100)
Thu Mar  5 19:19:04 2009 [thread 7522#-1564058736] eyedbd : refcnt = 3, fd_cnt = 3
Thu Mar  5 19:19:04 2009 [thread 7522#-1564058736] eyedbd : close connection fd=1
...
```

2.4 Server configuration

The EyeDB server has a number of *configuration switches* that allow to specify server parameters such as listening ports, directory for temporary files, databases directory...

Each EyeDB server configuration switch value is determined using, in this order:

- **eyedbctl** options
- server configuration file
- environment variables
- compile-time defaults

2.4.1 Server configuration file

The server configuration file is a file that contains a list of configuration switches definition. The syntax of the configuration file is given informaly in Figure 2.1.

```
file : line ...
line : comment | definition
comment : # LINE
definition: VARIABLE = VALUE ;
```

Figure 2.1: EyeDB server configuration file syntax

A variable value can contain variable expansions; a variable expansion is a variable name surrounded by the character `%` as in: `%variable%`.

Example 2.9 gives an example of a EyeDB server configuration file.

Example 2.9 EyeDB server configuration file

```
# EyeDB server configuration file
#

# Bases directory
databasedir = /home/francois/projects/eyedb/install/var/lib/eyedb/db;

# Temporary files directory
tmpdir = /home/francois/projects/eyedb/install/var/lib/eyedb/tmp;

# Path to .so files
sopath = /home/francois/projects/eyedb/install/lib/eyedb;

# Default EYEDBDBM database
default_dbm = /home/francois/projects/eyedb/install/var/lib/eyedb/db/dbmdb.dbs;

# Granted EYEDBDBM database
granted_dbm = %default_dbm%;

# Default file mask
default_file_mask = "0";

# Default file group
default_file_group = "";

# Maximum server memory size in Mb
maximum_memory_size = 0;

# Server access file
access_file = /home/francois/projects/eyedb/install/etc/eyedb/Access;

# Smd connection port
smdport = /home/francois/projects/eyedb/install/var/lib/eyedb/pipes/smd;

# Server listen ports and sockets list
listen = localhost:6240,/home/francois/projects/eyedb/install/var/lib/eyedb/pipes/eyedbd;

# OQL files path
oqlpath = /home/francois/projects/eyedb/install/lib/eyedb/oql
```

The EyeDB server will try to load one of the following configuration files, in this order:

- the file specified by **eyedbctl** --server-conf option
- the file specified by `EYEDBDCONF` environment variable

- user configuration file: the file `~/.eyedb/eyedbd.conf` where ~ represents the home directory of the user running the **eyedbctl** command
- the file `eyedbd.conf` located in the directory specified by `EYEDBCONFDIR` environment variable
- system-wide configuration file: the file `eyedbd.conf` located in directory `sysconfdir/eyedb`, where `sysconfdir` is the system configuration directory specified at compilation time using the **configure** script. By default, the system-wide configuration file is `/etc/eyedb/eyedbd.conf`

2.4.2 Server configuration switches list

The list of configuration switches that are use by the EyeDB server is given in Table 2.1. This table gives the variable name that can appear in a configuration file, the corresponding **eyedbctl** option if applicable, the name of the environment variable and a description of the configuration switch meaning.

Configuration variable name	**eyedbctl option**	**environment variable**
databasedir	--databasedir	EYEDBDATABASEDIR
tmpdir	NA	EYEDBTMPDIR
sopath	NA	EYEDBSOPATH
default_dbm	--default-dbm	EYEDBDEFAULT_DBM
granted_dbm	--granted-dbm	EYEDBGRANTED_DBM
default_file_mask	--default-file-mask	EYEDBDEFAULT_FILE_MASK
default_file_group	--default-file-group	EYEDBDEFAULT_FILE_GROUP
maximum_memory_size	--maximum-server-memory-size	EYEDBMAXIMUM_MEMORY_SIZE
access_file	NA	EYEDBACCESS_FILE
smdport	NA	EYEDBSMDPORT
listen	--listen	EYEDBLISTEN
oqlpath	NA	EYEDBOQLPATH

Table 2.1: List of EyeDB server configuration switches

2.5 Server network access file

When a EyeDB client command or application attempts to connect to the EyeDB server using a TCP connection, an *access file* is used to determine wheter the connection is authorized.

The location of the network access file is given by the `access_file` configuration switch, that can be specified either as a configuration variable in a configuration file or as an environment variable.

The syntax of this access file is the following:

- comments: all characters following a # are skipped
- access rule, composed of an IP address or subnet mask `address`, followed by a list of user authorizations, where a user authorization can be:
  - a EyeDB user name: the designated user is authorized to connect to the EyeDB server from the machine identified by `address`
  - a EyeDB user name prefixed by the ! character: the designated user is *not* authorized to connect to the EyeDB server from the machine identified by `address`
  - a EyeDB user name prefixed by the = character: the designated user will be used for any connection opened from the machine identified by `address` if no authentication is provided
  - a + character: any EyeDB user is authorized to connect from the machine identified by `address`

Example 2.10 gives an example of a EyeDB server network access file.

Example 2.10 EyeDB server network access file

```
# All users are authorized from domain localdomain:
.localdomain       +

# Any user, except user ’eyedbroot’, are authorized from machine r2.somewhere.org:
r2.somewhere.org   + !eyedbroot

# User ’joe’ is authorized from domain ircam.fr:
.ircam.fr          joe

# Users ’tom’ and ’guest’ are authorized from subnet mask 192.168.1.0; if no authenticatino ←-
     is provided, user ’guest’ is used:
192.168.1.0        tom =guest
```

Chapter 3

Database administration

This chapter covers the database administration commands used for:

- creating and deleting databases
- getting information on databases
- moving, copying, renaming databases
- importing and exporting databases

All these operations are performed using the **eyedbadmin** command and its **database** topic.

3.1 Creating and deleting a database

3.1.1 Creating a database

Creating a database is done using the **eyedbadmin** command:

`eyedbadmin database create` [options...] *`database`*

Options are:

- --dbfile=*`file`* specifies the database file (a file, usually with `.dbs` extension, that stores the database structure.
- --filedir=*`directory`* specifies the directory that will contains the database data files
- --max-object-count=*`count`* specifies the maximum number of objects that can be store in the database

Example 3.1 shows several examples of database creation.

Example 3.1 eyedbadmin database create

```
# create database 'test1'
eyedbadmin database create test1
# create database 'test2' giving the database file
eyedbadmin database create --dbfile=/var/tmp/test2.dbs test2
ls -l /var/tmp/test2.dbs
-rw------- 1 francois francois 250696 2009-03-13 11:55 /var/tmp/test2.dbs
# create database 'test3' giving the database files directory
mkdir /var/tmp/test3
eyedbadmin database create --filedir=/var/tmp/test3 test3
ls -l /var/tmp/test3
total 1504
-rw------- 1 francois francois  8471424 2009-03-13 11:57 test3.dat
-rw------- 1 francois francois   250696 2009-03-13 11:57 test3.dbs
-rw------- 1 francois francois    70000 2009-03-13 11:57 test3.dmp
-rw------- 1 francois francois        0 2009-03-13 11:57 test3.lck
-rw------- 1 francois francois    61440 2009-03-13 11:57 test3.omp
-rw------- 1 francois francois 67108864 2009-03-13 11:57 test3.shm
```

3.1.2 Deleting a database

Deleting a database is done using the **eyedbadmin** command:

`eyedbadmin` database delete *database*...

This command has no options (except the **eyedbadmin** command standard options).

Example 3.2 eyedbadmin database delete

```
# delete database 'test1'
eyedbadmin database delete test1
```

3.1.3 Listing databases

Obtaining a list of existing databases or obtaining information on a specific database is done using the **eyedbadmin** command:

`eyedbadmin database list` [options...] [~]*database*...

Options are:

- --dbname lists database names
- --dbid lists database identifier
- --dbfile lists database file
- --max-object-count lists database max object count
- --datafiles lists database datafiles
- --defaccess lists database default access
- --useraccess lists user database accesses
- --stats lists database statistics
- --all lists all database info

Command arguments are:

- [~]*database*... database(s) to list (~ means regular expression)

Example 3.3 shows several examples of database listing.

Example 3.3 eyedbadmin database list

```
# list all databases
eyedbadmin database list
Database Name
  test
Database Identifier
  2
Database File
  /home/francois/projects/eyedb/install/var/lib/eyedb/db/test.dbs
Max Object Count
  10000000
Datafiles
  Datafile #0
    Name      DEFAULT
    Dataspace #0
    File      test.dat
    Maxsize   ~2048Mb
    Slotsize  16b
    Server Access read/write
Default Access
  NO_DBACCESS_MODE
Database Access
  User Name  francois
  Access Mode ADMIN_DBACCESS_MODE
Database Name
  EYEDBDBM
....
### output truncated for readability
# list only database names
eyedbadmin database list --dbname
test
EYEDBDBM
test3
test2
# list non existing database
eyedbadmin database list --dbname foo
Database ’foo’ not found
# list statistics for a given database
eyedbadmin database list --stats test
Statistics
  Maximum Object Number 10000000
  Object Number         2574
  Maximum Slot Count    134217728
  Busy Slot Count       532202
  Maximum Size          2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
  Busy Slot Size        8515232b, ~8315Kb, ~8Mb
  Disk Size Used        76007288b, ~74225Kb, ~72Mb
  Disk Block Size Used  10452992b, ~10208Kb, ~10Mb
  Used                  0.40%
# list database default access right
eyedbadmin database list --defaccess test
NO_DBACCESS_MODE
# list with regular expression (note shell quoting)
eyedbadmin database list --dbname ’~test*’
test
test3
test2
# list with another regular expression (note shell quoting)
eyedbadmin database list --dbname --dbfile ’~test[0-9]’
test3
/var/tmp/test3/test3.dbs
test2
/var/tmp/test2.dbs
```

3.1.4 Setting database default access rights

Database access rights are defined at two levels:

- default database access, applying for all EyeDB users
- per user database access, applying only for specified user

Setting default access right is done use the **eyedbadmin** command:

`eyedbadmin` database defaccess *`database`* *`mode`*

Allowed values for *`mode`* are:

- **r**: read-only
- **rw**: read/write
- **rx**: read/execute
- **rwx**: read/write/execute
- **admin**: administrate
- **no**: access forbiden

Example 3.4 eyedbadmin database defaccess

```
# list database default access right
eyedbadmin database list --defaccess test
NO_DBACCESS_MODE
# set default access right to read-only
eyedbadmin database defaccess test r
# list database default access right
eyedbadmin database list --defaccess test
READ_DBACCESS_MODE
# set default access right to read/write
eyedbadmin database defaccess test rw
# list database default access right
eyedbadmin database list --defaccess test
READ_WRITE_DBACCESS_MODE
```

Note
Only an EyeDB user that has the **admin** database access for a given database can set a default database access for this database. The user that has created the database has automatically the **admin** database access.

3.2 Renaming, copying and moving a database

Renaming, copying and moving a database cannot be simply done by renaming the files that are associated with the database, because EyeDB maintains databases informations in a database itself. Specific commands are therefore required to rename, copy and move a database.

3.2.1 Renaming a database

Renaming a database is done using the **eyedbadmin** command:

`eyedbadmin` database rename *`database`* *`new_database`*

Command arguments are:

- *`database`* name of database to rename
- *`new_database`* new database name

Example 3.5 eyedbadmin database rename

```
eyedbadmin database list test
Database Name
  test
Database Identifier
  2
Database File
  /home/francois/projects/eyedb/install/var/lib/eyedb/db/test.dbs
Max Object Count
  10000000
Datafiles
  Datafile #0
    Name      DEFAULT
    Dataspace #0
    File      test.dat
    Maxsize   ~2048Mb
    Slotsize  16b
    Server Access read/write
Default Access
  READ_WRITE_DBACCESS_MODE
Database Access
  User Name  francois
  Access Mode ADMIN_DBACCESS_MODE
eyedbadmin database rename test newtest
eyedbadmin database list newtest
Database Name
  newtest
Database Identifier
  2
Database File
  /home/francois/projects/eyedb/install/var/lib/eyedb/db/test.dbs
Max Object Count
  10000000
Datafiles
  Datafile #0
    Name      DEFAULT
    Dataspace #0
    File      test.dat
    Maxsize   ~2048Mb
    Slotsize  16b
    Server Access read/write
Default Access
  READ_WRITE_DBACCESS_MODE
Database Access
  User Name  francois
  Access Mode ADMIN_DBACCESS_MODE
```

Database renaming and database file names
Example 3.5 shows that renaming a database does *not* rename the database files. Rename the database data files is done using the **eyedbadmin datafile rename** command described in Example 5.4.

3.2.2 Copying a database

Copying a database is done using the **eyedbadmin** command:

`eyedbadmin database copy` [options...] *`database`* *`new_database`*

Options are:

- --dbfile=*`file`* specifies the database file (a file, usually with `.dbs` extension, that stores the database structure.
- --filedir=*`directory`* specifies the directory that will contains the database data files

Command arguments are:

- *`database`* name of database to copy
- *`new_database`* new database name

Example 3.6 eyedbadmin database copy

```
eyedbadmin database list test3
Database Name
  test3
Database Identifier
  4
Database File
  /var/tmp/test3/test3.dbs
Max Object Count
  10000000
Datafiles
  Datafile #0
    Name      DEFAULT
    Dataspace #0
    File      test3.dat
    Maxsize   ~2048Mb
    Slotsize  16b
    Server Access read/write
Default Access
  NO_DBACCESS_MODE
Database Access
  User Name  francois
  Access Mode ADMIN_DBACCESS_MODE
mkdir /var/tmp/test3copy
eyedbadmin database copy --filedir=/var/tmp/test3copy test3 test3copy
eyedbadmin database list test3copy
Database Name
  test3copy
Database Identifier
  4
Database File
  /var/tmp/test3copy/test3copy.dbs
Max Object Count
  10000000
Datafiles
  Datafile #0
    Name      DEFAULT
    Dataspace #0
    File      /var/tmp/test3copy/test3copy.dat
    Maxsize   ~2048Mb
    Slotsize  16b
    Server Access read/write
Default Access
  NO_DBACCESS_MODE
```

3.2.3 Moving a database

Moving a database is a different operation than renaming a database: moving a database means moving the database files to a new directory, without changing the database name.

Moving a database is done using the **eyedbadmin** command:

`eyedbadmin database move` [options...] *`database`*

Options are:

- --dbfile=*`file`* specifies the database file (a file, usually with `.dbs` extension, that stores the database structure
- --filedir=*`directory`* specifies the directory that will contains the database data files. This option is *mandatory*.

Command arguments are:

- `database` name of database to move

Example 3.7 eyedbadmin database move

```
eyedbadmin database list test2
Database Name
  test2
Database Identifier
  3
Database File
  /var/tmp/test2.dbs
Max Object Count
  10000000
Datafiles
  Datafile #0
    Name      DEFAULT
    Dataspace #0
    File      test2.dat
    Maxsize   ~2048Mb
    Slotsize  16b
    Server Access read/write
Default Access
  NO_DBACCESS_MODE
Database Access
  User Name  francois
  Access Mode ADMIN_DBACCESS_MODE
mkdir /var/tmp/test2
eyedbadmin database move --filedir=/var/tmp/test2 test2
eyedbadmin database list test2
Database Name
  test2
Database Identifier
  3
Database File
  /var/tmp/test2/test2.dbs
Max Object Count
  10000000
Datafiles
  Datafile #0
    Name      DEFAULT
    Dataspace #0
    File      test2.dat
    Maxsize   ~2048Mb
    Slotsize  16b
    Server Access read/write
Default Access
  NO_DBACCESS_MODE
```

3.3 Exporting and importing a database

Database export and import is used to make a copy of a database content in a single file. It can be used to backup a database or to copy a database between computers that do not share file systems.

3.3.1 Exporting a database

Exporting a database is done using the **eyedbadmin** command:

`eyedbadmin database export` *`database file`*

Command arguments are:

- *`database`* name of database to export
- *`file`* file for export (- means export to standard output)

Example 3.8 eyedbadmin database export

```
eyedbadmin database export test2 /var/tmp/test2.dump
```

3.3.2 Importing a database

Importing a database is done using the **eyedbadmin** command:

`eyedbadmin database import` [options...] *`database file`*

Options are:

- --filedir=*`directory`* specifies the directory that will contains the database files
- --mthdir=*`directory`* specifies the directory that will contains the database methods
- --list list only, do not import

Command arguments are:

- *`database`* name of database to import
- *`file`* file for import (- means import from standard output)

Example 3.9 eyedbadmin database import

```
eyedbadmin database import test2 /var/tmp/test2.dump
```

Chapter 4

User administration

As any other DBMS, EyeDB provides the concept of *user* that is the basis for rights management: database read or write access, database administration...

4.1 The different user types

An EyeDB user is identified by:

- a name
- an optional password
- a user identifier, managed by EyeDB and for internal use only

EyeDB users can have the following types:

- *eyedb* user type (the default type): users of this type are not mapped to Unix users and can use EyeDB password authentication only
- *unix*: users of this type are mapped to Unix standard users (a Unix user of the same name must exist) and can use EyeDB password authentication or EyeDB Unix identification
- *strict unix*: users of this type are mapped to Unix standard users, and can use only EyeDB Unix authentication

The EyeDB Unix authentication is a mode of authentication that can be used only when using a named pipe connection between the EyeDB client and the EyeDB server; this authentication does not require a password and uses a safe challenge-based authentication supported by the Unix operating system.

4.2 Managing users

4.2.1 Adding a user

Adding user is done using the **eyedbadmin** command:

`eyedbadmin user add` [options...] [*`user`*] [*`password`*]

Options are:

- [--unix | --strict-unix] specify the user type, if unix or strict-unix type

Command arguments are:

- [user] user name, if not unix or strict-unix type (if unix or strict-unix type, the user name is the user running the eyedbadmin command)
- [password] password, if not strict-unix type. If no password is provided on command line, **eyedbadmin** will ask for a password on standard input.

Example 4.1 eyedbadmin user add

```
## Adding a 'eyedb' user: password is required
eyedbadmin user add joe
joe password: 123

retype joe password: 123

## Adding a 'unix' user with password on command line: Unix user does not exist, error
eyedbadmin user add --unix tom 456
eyedb error: username 'tom' is an unknown unix user

## Adding a 'unix' user: Unix user already exists, password is required
eyedbadmin user add --unix eric
eric password: 456

retype eric password: 456

## Adding a 'strict-unix' user: no password is required
eyedbadmin user add --strict-unix mimi
```

4.2.2 Deleting a user

Deleting user is done using the **eyedbadmin** command:

`eyedbadmin user delete` *`user`*

Command arguments are:

- *`user`* EyeDB user name (can be of any user type)

Example 4.2 eyedbadmin user delete

```
## Deleting existing user
eyedbadmin user delete joe
## Deleting unknown user
eyedbadmin user delete r2d2
delete user error: user entry 'r2d2' does not exist
```

4.2.3 Setting user password

Setting user password is done using the **eyedbadmin** command:

`eyedbadmin user passwd` *`user`* [*`password`*] [*`new_password`*]

Command arguments are:

- [user] EyeDB user name (can be of type 'eyedb' or 'unix')

- [password] old password (will be asked if not provided on command line)
- [new_password] new password (will be asked if not provided on command line)

Example 4.3 eyedbadmin user passwd

```
## Setting password with prompt
eyedbadmin user passwd eric
user old password: 456

user new password: 123
retype user new password: 123
## Setting password from command-line
eyedbadmin user passwd eric 123 789
```

4.2.4 Listing users

Listing existing EyeDB users is done using the **eyedbadmin** command:

`eyedbadmin user list [user]`

Command arguments are:

- [*user*] EyeDB user name

Example 4.4 eyedbadmin user list

```
## Listing all users
eyedbadmin user list
name      : "mimi" [strict unix user]
sysaccess : NO_SYSACCESS_MODE

name      : "eric" [unix user]
sysaccess : NO_SYSACCESS_MODE

name      : "francois" [strict unix user]
sysaccess : SUPERUSER_SYSACCESS_MODE
dbaccess  : (dbname : "newtest", access : ADMIN_DBACCESS_MODE)
            (dbname : "test3", access : ADMIN_DBACCESS_MODE)
## Listing a specific user
eyedbadmin user list mimi
name      : "mimi" [strict unix user]
sysaccess : NO_SYSACCESS_MODE
```

4.3 Managing user access rights

4.3.1 The different access rights

EyeDB access rights are managed at two levels:

- system level, for global operations (for instance, adding a user, creating a database...)
- database level, for database specific operations (for instance, reading objects with a query, inserting objects...)

Each EyeDB user has therefore:

- a set of system access rights, controling the authorization to perform the following system operations:
  - creating a database
  - adding a user
  - deleting any user
  - setting password for any user
- a set of database access rights, controling the authorization to perform the following operations on a given database:
  - read from this database
  - write to this database
  - execute any method in this database
  - delete this database

Note that a system access right is not attached to a particular database, whilst a database access right is attached to a particular database.

Table 4.1 presents the different system access rights defined by EyeDB and the corresponding authorized operations.

System Access Right	**Authorized operation**
no	no authorized system operation
dbcreate	can create a database
adduser	can add any user
deleteuser	can delete any user
setuserpasswd	can set the password of any user
admin	can create a database and add a user
superuser	all operations

Table 4.1: System access rights and corresponding authorized operations

Table 4.2 presents the different database access rights defined by EyeDB and the corresponding authorized operations.

Database Access Right	**Authorized operation**
no	no database access
read	have read access on this database
write	have write access on this database
exec	have execute access on this database
admin	have read/write/execute access on this database and can delete this database

Table 4.2: Database access rights and corresponding authorized operations

4.3.2 Setting system access rights

Setting system access rights for a EyeDB user is done using the **eyedbadmin** command:

`eyedbadmin user sysaccess` *`user`* [dbcreate | adduser | deleteuser | setuserpasswd | admin | superuser | no]

Command arguments are:

- *`user`* EyeDB user name
- [dbcreate | adduser | deleteuser | setuserpasswd | admin | superuser | no] system access right

Example 4.5 eyedbadmin user sysaccess

```
## List user system access rights
eyedbadmin user list eric
name      : "eric" [unix user]
sysaccess : NO_SYSACCESS_MODE
## Set user system access rights
eyedbadmin user sysaccess eric dbcreate
eyedbadmin user list eric
name      : "eric" [unix user]
sysaccess : DB_CREATE_SYSACCESS_MODE
```

4.3.3 Setting database access rights

Setting database access rights for a EyeDB user is done using the **eyedbadmin** command:

`eyedbadmin user dbaccess` *`user database access`*

Command arguments are:

- *`user`* EyeDB user name
- *`database`* database name
- *`access`* database access right, that can be one of the following values:
  [r | rw | rx | rwx | admin | no]

Example 4.6 eyedbadmin user dbaccess

```
## List user database access rights
eyedbadmin user list eric
name      : "eric" [unix user]
sysaccess : NO_SYSACCESS_MODE
## Set user database access rights for database test2
eyedbadmin user dbaccess eric test2 rw
eyedbadmin user list eric
name      : "eric" [unix user]
sysaccess : DB_CREATE_SYSACCESS_MODE
dbaccess  : (dbname : "test2", access : READ_WRITE_DBACCESS_MODE)
## Set user database access rights for database test3
eyedbadmin user dbaccess eric test3 admin
eyedbadmin user list eric
name      : "eric" [unix user]
sysaccess : DB_CREATE_SYSACCESS_MODE
dbaccess  : (dbname : "test2", access : READ_WRITE_DBACCESS_MODE)
            (dbname : "test3", access : ADMIN_DBACCESS_MODE)
```

Chapter 5

Database advanced administration

This chapter explains EyeDB databases advanced administration, namely the *datafiles* and *dataspaces* concepts that allow to create multi-volumes large EyeDB databases.

The first section describes EyeDB databases architecture; the following sections deal with datafiles and dataspaces management commands.

5.1 EyeDB databases architecture

EyeDB databases organisation is based on two concepts:

- *dataspaces*: logical organization units for organizing where classes, objects, indexes... will be located
- *datafiles*: physical files where data are stored

A database is composed of several dataspaces (at least one). When creating a database with no options, a dataspace named DEFAULT is automaticaly created.

A datafile is always associated to a *single* dataspace, whilst a dataspace can contain several datafiles (up to 32 datafiles currently). When creating a database with no options, a datafile named DEFAULT, whose filename is <<database_name>>.dat and size is 2 Gb, is automaticaly created; this datafile is associated with the default dataspace.

After database creation, new datafiles and dataspaces can be added at any time using the commands described in this section. Datafiles can be resized, moved, deleted and updated. Dataspaces can be deleted, renamed and updated.

5.2 Managing datafiles

This section describes how to manage datafiles: create, list and delete datafiles, renaming, moving, resizing and defragmenting datafiles.

A datafile is identified by:

- an id (a number)
- a name, that can be used in place of the id to identify the datafile
- a file name, that is the operating system file name

Datafile name and file name
The datafile name and its file name are two different things: the file name is simply the operating system file name, as in `/var/tmp/foo.dat` and the name is an identifier known by EyeDB that can be used to identify the datafile as in `FOO`.

5.2.1 Creating, listing and deleting datafiles

Creating a datafile is done using the **eyedbadmin** command:

`eyedbadmin datafile create` [options] *`database`* *`datafile`*

Options are:

- --filedir=*`directory`* datafile directory
- --name=*`name`* datafile name (see Note [28])
- --size=*`size`* datafile size in Mbytes
- --slotsize=*`slotsize`* the allocation slot size in bytes
- --physical to specify that this datafile will contain physical Oids (see Chapter 6)

Command arguments are:

- *`database`* the database to which the datafile will be attached
- *`datafile`* the datafile name (.dat file name extension is recommended)

Example 5.1 eyedbadmin datafile create

```
## create a datafile
eyedbadmin datafile create test3 data1.dat
## create a datafile with a name
eyedbadmin datafile create --name=DATA2 test3 data2.dat
## create a datafile with a size of 30 Gb and a directory
eyedbadmin datafile create --filedir=/var/tmp --size=30000 test3 data3.dat
## list the database
eyedbadmin database list --datafiles test3
test3.dat
data1.dat
data2.dat
/var/tmp/data3.dat
```

The list of the datafiles associated with a database can be obtained using the **eyedbadmin** command:

`eyedbadmin datafile list` [options] *`database`* [*`datafile_id`* | *`datafile_name`*]...

Options are:

- --all list all informations about datafile
- --stats list only statistics on datafile

Command arguments are:

- *`database`* the database to which the datafile belongs
- [*`datafile_id`* | *`datafile_name`*]... the datafiles to be listed, that can be specified either by its id or by its name

Example 5.2 eyedbadmin datafile list

```
## List
eyedbadmin datafile list --all test3
Datafile #0
Name      DEFAULT
Dataspace #0 DEFAULT
File      test3.dat
Maxsize   2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize  16
Oid Type  Logical

Object Number        2561
Total Busy Size      8439065b, ~8241Kb, ~8Mb
Average Size         3295b, ~3Kb

... output deleted

Datafile #1
Name      <unnamed>
File      data1.dat
Maxsize   2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize  16
Oid Type  Logical

... output deleted

Datafile #2
Name      DATA2
File      data2.dat
Maxsize   2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize  16
Oid Type  Logical

... output deleted

Datafile #3
Name      <unnamed>
File      /var/tmp/data3.dat
Maxsize   31457280000b, ~30720000Kb, ~30000Mb, ~29Gb
Slotsize  16
Oid Type  Logical

... output deleted
```

Deleting a datafile is done using the **eyedbadmin** command:

`eyedbadmin datafile delete` [*`datafile_id`* | *`datafile_name`*]

Command arguments are:

- *`database`* the database to which the datafile belongs
- [*`datafile_id`* | *`datafile_name`*] the datafile to be deleted, that can be specified either by its id or by its name

Example 5.3 eyedbadmin datafile delete

```
## Delete datafile by id
eyedbadmin datafile delete test3 1
## Delete datafile by name
eyedbadmin datafile delete test3 DATA2
## List
eyedbadmin datafile list test3
Datafile #0
Name      DEFAULT
Dataspace #0 DEFAULT
File      test3.dat
Maxsize   2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize  16
Oid Type  Logical

... output deleted

Datafile #3
Name      <unnamed>
File      /var/tmp/data3.dat
Maxsize   31457280000b, ~30720000Kb, ~30000Mb, ~29Gb
Slotsize  16
Oid Type  Logical

... output deleted
```

5.2.2 Renaming and moving datafiles

A datafile has a name (see Note [28]); changing this name is done using the **eyedbadmin** command:

`eyedbadmin datafile rename` *database* [*datafile_id* | *datafile_name*] *new_name*

Example 5.4 eyedbadmin datafile rename

```
## Rename datafile
eyedbadmin datafile rename test3 3 DATA3
## List
eyedbadmin datafile list test3
Datafile #0
Name      DEFAULT
Dataspace #0 DEFAULT
File      test3.dat
Maxsize   2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize  16
Oid Type  Logical

... output deleted

Datafile #3
Name      DATA3
File      /var/tmp/data3.dat
Maxsize   31457280000b, ~30720000Kb, ~30000Mb, ~29Gb
Slotsize  16
Oid Type  Logical

... output deleted
```

Moving a datafile is done using the **eyedbadmin** command:

`eyedbadmin datafile move` [options] *`database`* [*`datafile_id`* | *`datafile_name`*] *`new_datafile`*

Options are:

- --filedir=*`directory`* datafile directory

Command arguments are:

- *`database`* the database to which the datafile belongs
- [*`datafile_id`* | *`datafile_name`*] the datafile to be moved, that can be specified either by its id or by its name
- *`datafile`* the new datafile file name (.dat file name extension is recommended)

Example 5.5 eyedbadmin datafile move

```
## Create destination directory
mkdir /var/tmp/test3
## Move datafile
eyedbadmin datafile move --filedir=/var/tmp/test3 test3 DATA3 newdata3.dat
## List
eyedbadmin datafile list test3
Datafile #0
Name      DEFAULT
Dataspace #0 DEFAULT
File      test3.dat
Maxsize   2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize  16
Oid Type  Logical

... output deleted

Datafile #3
Name      DATA3
File      /var/tmp/test3/newdata3.dat
Maxsize   31457280000b, ~30720000Kb, ~30000Mb, ~29Gb
Slotsize  16
Oid Type  Logical

... output deleted
```

5.2.3 Resizing and defragmenting datafiles

Datafile size is set at creation time; once created, the size of a datafile can be changed using the **eyedbadmin** command:

`eyedbadmin datafile resize` *`database`* [*`datafile_id`* | *`datafile_name`*] *`new_size`*

Command arguments are:

- *`database`* the database to which the datafile belongs
- [*`datafile_id`* | *`datafile_name`*] the datafile to be resized, that can be specified either by its id or by its name
- *`new_size`* the new datafile size, in Mbytes

Example 5.6 eyedbadmin datafile resize

```
## List
eyedbadmin datafile list test3 DATA3
Datafile #3
Name      DATA3
File      /var/tmp/test3/newdata3.dat
Maxsize   31457280000b, ~30720000Kb, ~30000Mb, ~29Gb
Slotsize  16
Oid Type  Logical

... output deleted
## Set new size to 40Gb
eyedbadmin datafile resize test3 DATA3 40000
## List
eyedbadmin datafile list test3 DATA3
Datafile #3
Name      DATA3
File      /var/tmp/test3/newdata3.dat
Maxsize   41943040000b, ~40960000Kb, ~40000Mb, ~39Gb
Slotsize  16
Oid Type  Logical

... output deleted
```

Datafiles can become fragmented because of object deletion; the allocator used inside datafile is a bitmap allocator for better efficiency, but as a consequence there can be holes in the datafile that can impact performance. Defragmenting can be done using the **eyedbadmin** command:

`eyedbadmin datafile defragment` *`database`* [*`datafile_id`* | *`datafile_name`*]

Command arguments are:

- *`database`* the database to which the datafile belongs
- [*`datafile_id`* | *`datafile_name`*] the datafile to be defragmented, that can be specified either by its id or by its name

Example 5.7 eyedbadmin datafile defragment

```
## datafile can be given by their id
eyedbadmin datafile defragment test3 0
## datafile can also be given by their name
eyedbadmin datafile defragment test3 DEFAULT
## giving a non-existing datafile
eyedbadmin datafile defragment test3 42
eyedb error: datafile #42 not found in database test3
```

5.3 Managing dataspaces

After being created, datafiles must be attached to *dataspaces*. Before attaching a datafile to a dataspace, the dataspace must have been created if it is not the default dataspace (which is created when creating the database).

5.3.1 Creating, listing, and deleting dataspaces

This section explains how to create, listing and delete dataspaces. When creating or updating a dataspace, the list of datafiles contained in this dataspace must be specified; datafiles must therefore have been created before, refer to Section 5.2 for explainations on datafiles management.

Creating a dataspace is done using the **eyedbadmin** command:

`eyedbadmin dataspace create` *`database`* *`dataspace`* [*`datafile_id`* | *`datafile_name`*]...

Command arguments are:

- *`database`* the database
- *`dataspace`* the name of the dataspace to be created
- [*`datafile_id`* | *`datafile_name`*]... the datafiles to be attached to the dataspace, that can be specified either by its id or by its name

Example 5.8 eyedbadmin dataspace create

```
## create 2 datafiles
eyedbadmin datafile create test4 data1.dat
eyedbadmin datafile create test4 data2.dat
## create a dataspace, attaching the 2 datafiles to this dataspace
eyedbadmin dataspace create test4 DSP1 1 2
```

Obtaining the list of dataspaces is done using the **eyedbadmin** command:

`eyedbadmin dataspace list` *`database`* *`dataspace`*

Command arguments are:

- *`database`* the database
- *`dataspace`* the name of the dataspace to be listed

Example 5.9 eyedbadmin dataspace list

```
## List all the dataspaces of database ’test4’
eyedbadmin dataspace list test4
Dataspace #0
Name DEFAULT
Datafile #0
Name     DEFAULT
File     test4.dat
Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize 16
Oid Type Logical
Dataspace #1
Name DSP1
Datafile #1
File     data1.dat
Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize 16
Oid Type Logical
Datafile #2
File     data2.dat
Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize 16
Oid Type Logical
```

Deleting a dataspace is done using the **eyedbadmin** command:

`eyedbadmin delete` *`database`* *`dataspace`*

Command arguments are:

- *database* the database
- *dataspace* the name of the dataspace

Example 5.10 eyedbadmin dataspace delete

```
## delete dataspace named 'NEWDSP1'
eyedbadmin dataspace delete test4 NEWDSP1
## list all dataspaces
eyedbadmin dataspace list test4
Dataspace #0
Name DEFAULT
Datafile #0
Name     DEFAULT
File     test4.dat
Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize 16
Oid Type Logical
```

5.3.2 Updating and renaming dataspaces

Updating a dataspace is done using the **eyedbadmin** command:

`eyedbadmin dataspace update` *database dataspace* [*datafile_id* | *datafile_name*]...

Command arguments are:

- *database* the database
- *dataspace* the name of the dataspace
- [*datafile_id* | *datafile_name*]... the datafiles to be attached to the dataspace, that can be specified either by its id or by its name

Example 5.11 eyedbadmin dataspace update

```
## create 2 more datafiles
eyedbadmin datafile create test4 data3.dat
eyedbadmin datafile create test4 data4.dat
## update the dataspace, attaching the 4 datafiles to the dataspace
eyedbadmin dataspace update test4 DSP1 1 2 3 4
## list the updated dataspace
eyedbadmin dataspace list test4 DSP1
Dataspace #1
Name DSP1
Datafile #1
File     data1.dat
Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize 16
Oid Type Logical
Datafile #2
File     data2.dat
Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize 16
Oid Type Logical
Datafile #3
File     data3.dat
Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize 16
Oid Type Logical
Datafile #4
File     data4.dat
Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize 16
Oid Type Logical
## update dataspace, leaving only the last 2 datafiles
eyedbadmin dataspace update test4 DSP1 3 4
## list updated dataspace
eyedbadmin dataspace list test4 DSP1
Dataspace #1
Name NEWDSP1
Datafile #3
File     data3.dat
Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize 16
Oid Type Logical
Datafile #4
File     data4.dat
Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize 16
Oid Type Logical
```

Renaming a dataspace is done using the **eyedbadmin** command:

`eyedbadmin dataspace rename` *`database dataspace new_dataspace`*

Command arguments are:

- *`database`* the database
- *`dataspace`* the name of the dataspace
- *`new_dataspace`* the new name of the dataspace

Example 5.12 eyedbadmin dataspace rename

```
## rename dataspace
eyedbadmin dataspace rename test4 DSP1 NEWDSP1
## list renamed dataspace
eyedbadmin dataspace list test4 NEWDSP1
Dataspace #1
Name NEWDSP1
Datafile #3
File     data3.dat
Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize 16
Oid Type Logical
Datafile #4
File     data4.dat
Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize 16
Oid Type Logical
```

5.3.3 Getting and setting current datafile

A dataspace maintains an internal pointer to its *current* datafile, i.e. the datafile in which new data will be stored when creating an object, an index that hits into this dataspace.

When a datafile is full, the pointer to the current datafile is moved to the next datafile in the dataspace that has space left for insertion. If all datafiles in the dataspace are full, then the insertion ends with an error.

Using **eyedbadmin**, it is possible to examine and set the pointer to the current datafile.

Examining the current datafile is done using the **eyedbadmin** command:

`eyedbadmin dataspace getcurdat` *`database dataspace`*

Command arguments are:

- *`database`* the database
- *`dataspace`* the name of the dataspace

Example 5.13 eyedbadmin dataspace getcurdat

```
eyedbadmin dataspace getcurdat test4 DSP1
Datafile #1
Name      <unnamed>
Dataspace #1 DSP1
File      data1.dat
Maxsize   2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
Slotsize  16
Oid Type  Logical
```

Setting the current datafile is done using the **eyedbadmin** command:

`eyedbadmin dataspace setcurdat` *`database dataspace`* [*`datafile_id`* | *`datafile_name`*]...

Command arguments are:

- *`database`* the database
- *`dataspace`* the name of the dataspace

- [`datafile_id` | `datafile_name`]... the datafiles to be attached to the dataspace, that can be specified either by its id or by its name

Example 5.14 eyedbadmin dataspace setcurdat

```
      eyedbadmin dataspace setcurdat test4 DSP1 3
      eyedbadmin dataspace getcurdat test4 DSP1
      Datafile #3
      Name      <unnamed>
      Dataspace #1 DSP1
      File      data3.dat
      Maxsize   2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
      Slotsize  16
      Oid Type  Logical
      francois@revard$
```

5.3.3.1 Getting and setting default dataspace for a database

A database maintains an internal pointer to a *default* dataspace; this default dataspace will be used when inserting data that do not have an associated dataspace Section 6.4.

Using **eyedbadmin**, it is possible to examine and set the pointer to the default dataspace for a database.

Getting and setting default dataspace
Getting and setting default dataspace is done using the *database* topic of the **eyedbadmin** command Section 1.2.1, because the default dataspace is a property of the *database* and not of a particular dataspace.

Examining the default dataspace is done using the **eyedbadmin** command:

`eyedbadmin database getdefdsp` *`database`*

Command arguments are:

- *`database`* the database

Example 5.15 eyedbadmin database getdefdsp

```
eyedbadmin database getdefdsp test4
Dataspace #0
Name DEFAULT
   Datafile #0
     Name     DEFAULT
     File     test4.dat
     Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
     Slotsize 16
     Oid Type Logical
```

Setting the default dataspace is done using the **eyedbadmin** command:

`eyedbadmin database setdefdsp` *`database dataspace`*

Command arguments are:

- *`database`* the database
- *`dataspace`* the name of the dataspace that must become the default dataspace

Example 5.16 eyedbadmin database setdefdsp

```
eyedbadmin database setdefdsp test4 DSP1
eyedbadmin database getdefdsp test4
Dataspace #1
Name DSP1
   Datafile #1
     File     data1.dat
     Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
     Slotsize 16
     Oid Type Logical
   Datafile #2
     File     data2.dat
     Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
     Slotsize 16
     Oid Type Logical
   Datafile #3
     File     data3.dat
     Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
     Slotsize 16
     Oid Type Logical
   Datafile #4
     File     data4.dat
     Maxsize  2147483648b, ~2097152Kb, ~2048Mb, ~2Gb
     Slotsize 16
     Oid Type Logical
```

Chapter 6

Database tuning

This chapter explains how to tune EyeDB to improve performance, either system response time or databases size.

The first section describes how to build multi-volumes databases that can contain hundred of terabytes of data. The two following sections explains how to manage indexes and collections to improve system response time. The last section describes how to specify a location (i.e. a dataspace) for EyeDB entities such as class instances, collections...

6.1 Creating large databases

Creating large databases (i.e. larger than a terabyte) requires partitionning the database between several volumes, either physical disks or partitions of a disk. EyeDB supports multi-volumes databases using datafiles (see Section 5.2) and dataspaces (see Section 5.3).

6.1.1 Databases limits

6.1.2 Tuning the host system

An important notice is that the host system imposes to the processes running on it some resource limits, in particular on process memory size and process virtual memory size. As EyeDB file access relies heavily on virtual memory and file mapping, it is mandatory to either unset these limits or set them to a high value.

On Linux, the resource limits can be retrieved and set using functions `getrlimit` and `setrlimit`, that can be used from a C or C++ program. For a further description of these functions, consult the getrlimit(2) and setrlimit(2) manual pages.

When using command line tools, an easy way to retrieve and set resource limits is to use the bash **ulimit** command. For a further description of this command, consult the bash(1) manual page.

The two mandatory resource limits that must be set are:

- the maximum memory size, which can be retrieved using command **ulimit -m**
- the maximum virtual memory size, which can be retrieved using command **ulimit -v**

Example 6.1 Retrieving and setting resource limits using **ulimit**

```
ulimit -m
1024
ulimit -m unlimited
ulimit -m
unlimited
ulimit -v
8192
ulimit -v unlimited
ulimit -v
unlimited
```

6.1.3 Creating multi-volumes databases using datafiles and dataspaces

6.2 Managing indexes

As a relational database management system does, EyeDB provides indexes to improve drastically the time needed to retrieve an objet in a database and to improve therefore the performances of queries.

EyeDB supports two types of indexes, hash and b-trees.

As the description of indexes management needs that the database has a *schema*, the ODL database schema given in Example 6.2 will be used in the index administration commands examples.

Example 6.2 The ODL database schema

```
class Address {
  attribute int number;
  attribute string street;
  attribute string town;
  attribute int code;
};

class Person {
  attribute string firstName;
  attribute string lastName;
  attribute int age;
  attribute Address addr;
};
```

Example 6.3 Indexes: updating the schema

```
eyedbadmin database create test_index
eyedbodl -u -d test_index test_index.odl
Updating 'test' schema in database test...
Adding class Address
Adding class Person

Done
```

6.2.1 Creating and deleting indexes

Creating an index is done using the **eyedbadmin** command:

`eyedbadmin index create` [options] *`database`* *`attribute_path`* [*`hints`*]

Options are:

- --propagate= [on | off] propagation mode
  --type=*type* index type (supported types are: hash, btree)

Command arguments are:

- *database* the database
- *attribute_path* the path of the attribute on which the index is to be created
- *hints* index hints, i.e. index configuration values that are implementation specific

An attribute path is an expression built using the . (dot) operator and class names or attribute names.

Example 6.4 Examples of attribute paths

```
Person.firstName
Person.age
Address.number
Person.addr.town
```

Example 6.5 eyedbadmin index create

```
eyedbadmin index create test_index Person.age
Creating btree index on Person.age
eyedbadmin index create test_index Person.addr.town
Creating hash index on Person.addr.town
eyedbadmin index create test_index Person.addr.number
Creating btree index on Person.addr.number
eyedbadmin index create --type=hash test_index Person.addr.code
Creating hash index on Person.addr.code
```

Deleting an index is done using the **eyedbadmin** command:

`eyedbadmin index delete` *database attribute_path*

Command arguments are:

- *database* the database
- *attribute_path* the path of the attribute on which the index is to be created

Example 6.6 eyedbadmin index delete

```
eyedbadmin index delete test_index Person.addr.number
Deleting index Person.addr.number
eyedbadmin index delete test_index Person.addr.street
eyedbadmin: index 'Person.addr.street' not found
```

6.2.2 Listing, updating and moving indexes

Listing indexes is done using the **eyedbadmin** command:

`eyedbadmin index list` [options] *database* [[*attribute_path* | *class_name*]]

Command arguments are:

- *database* the database
- [[*attribute_path* | *class_name*]] the index to be listed, specified either as an attribute path or as a class name

Example 6.7 eyedbadmin index list

```
eyedbadmin index list test_index
btree index on Person.age
hash index on Person.addr.town
hash index on Person.addr.code
eyedbadmin index list test_index Person
btree index on Person.age
hash index on Person.addr.town
hash index on Person.addr.code
eyedbadmin index list test_index Person.age
btree index on Person.age
```

6.2.3 Getting statistics on indexes

Getting statistics on indexes is done using the **eyedbadmin** command:

`eyedbadmin index stats` [options] *database* [[*attribute_path* | *class_name*]]

Command arguments are:

- *database* the database
- [[*attribute_path* | *class_name*]] the index on which to print statistics, specified either as an attribute path or as a class name

Options are:

- --full displays all index entries statistics
- --format=*format* statistics format; *format* is a printf-like string where:
  - %n denotes the number of keys
  - %O denotes the count of object entries for this key
  - %o denotes the count of hash objects for this key
  - %s denotes the size of hash objects for this key
  - %b denotes the busy size of hash objects for this key
  - %f denotes the free size of hash objects for this key

Example 6.8 eyedbadmin index statistics

```
eyedbadmin index stats --full test_index
    Total hash object count: 1
    Total hash object size: 98376B, ~96KB
    Total hash object busy size: 0B
    Total hash object free size: 98376B, ~96KB
    Busy entry count: 0
    Free entry count: 4096
...
eyedbadmin index stats '--format=%n %O\n' test_index Person.addr.code
4090 0
4091 0
4092 0
4093 0
4094 0
4095 0
...
eyedbadmin index stats '--format=%n -> %O, %o, %s\n' test_index Person.addr.town
4090 -> 0, 0, 0
4091 -> 0, 0, 0
4092 -> 0, 0, 0
4093 -> 0, 0, 0
4094 -> 0, 0, 0
4095 -> 0, 0, 0
...
```

6.2.4 Getting and setting index default dataspace

An index has a *default dataspace* (see Chapter 5). This default dataspace determines in which dataspace index data will be inserted when index is updated, for instance when objects are inserted into the database.

Getting index default dataspace is done using command **eyedbadmin index getdefdsp**:

`eyedbadmin index getdefdsp` *database* [*attribute_path* | *class_name*]

Command arguments are:

- *database* the database
- [[*attribute_path* | *class_name*]] the index on which to print statistics, specified either as an attribute path or as a class name

Setting index default dataspace is done using command **eyedbadmin index setdefdsp**:

`eyedbadmin index setdefdsp` *database* [[*attribute_path* | *class_name*]] *dataspace*

Command arguments are:

- *database* the database
- [[*attribute_path* | *class_name*]] the index on which to print statistics, specified either as an attribute path or as a class name
- *dataspace* the name of the dataspace to be created

Example 6.9 eyedbadmin index getdefdsp/setdefdsp

```
# get default dataspace for index on Person.age
eyedbadmin index getdefdsp test_index Person.age
Default dataspace for index 'Person.age':
  Dataspace #0 (default)
   Name DEFAULT
   Composed of {
     Datafile #0
       Name DEFAULT
       File test_index.dat
   }
# create a datafile
eyedbadmin datafile create --name=DAT1 test_index test_index_data1.dat
# create a dataspace containing this file
eyedbadmin dataspace create test_index DSP1 DAT1
# set default dataspace for index on Person.age
eyedbadmin index setdefdsp test_index Person.age DSP1
# get default dataspace for index on Person.age
eyedbadmin index getdefdsp test_index Person.age
Default dataspace for index 'Person.age':
  Dataspace #1
   Name DSP1
   Composed of {
     Datafile #1
       Name DAT1
       File test_index_data1.dat
   }
```

6.3 Managing collections

6.3.1 EyeDB collection implementations

EyeDB supports several types of collections:

- `set`: unordered collection without element duplication
- `bag`: unordered collection with element duplication
- `array`: ordered collection

Independentely from its type, a collection has an *implementation*, i.e. the data structure that supports the collection. EyeDB supports several types of implementation, either indexed or non-indexed:

- hash index
- btree index

The choice of the collection implementation will have of course an impact on the collection performance for inserting, deleting retrieving objects. For example, having a `bag` collection containing a large number of elements and *not* using an indexed implementation will have poor performance for element insertion: when inserting an element in a `bag` collection, EyeDB must lookup the element in the collection to check for duplication and this will have poor performance if the lookup is not accelerated using an index.

Example 6.10 The ODL database schema for collections

```
class Person {
  attribute string firstName;
  attribute string lastName;
};

class Diary {
  attribute string name;
  attribute set<Person *> persons;
};
```

Example 6.11 Collections: updating the schema

```
## create database
eyedbadmin database create test_collection
eyedbodl -u -d test_collection test_collection.odl
Updating 'test_collection' schema in database test_collection...
Adding class Person
Adding class Diary

Done
```

6.3.2 Getting and setting collection default implementation

Example 6.12 eyedbadmin

```
## getting and setting default implementation
eyedbadmin collection getdefimpl test_collection Diary.persons
Default implementation on Diary.persons:
  System default
eyedbadmin collection setdefimpl --type=hashindex test_collection Diary.persons
eyedbadmin collection getdefimpl test_collection Diary.persons
Default implementation on Diary.persons:
  Type: Hash
```

6.3.3 Getting and updating a particular collection implementation

6.3.4 Getting statistics on a collection implementation

6.3.5 Getting and setting collection default dataspace

6.4 Managing locations

6.4.1 What is a location in EyeDB?

6.4.2 Managing instances locations

6.4.3 Managing attributes locations

6.4.4 Managing collections locations

6.4.5 Managing indexes locations