cubeSQL

ReadMe

Preface 3

System Requirements 4

Default installation paths 4

Remove REAL Server 4

Five Minutes Guide 5

MacOS X 5

Windows 5

Linux 6

SSL 7

What is SSL? 7

How do I set up SSL? 7

MVCC 9

JSON 11

CONNECT 11

EXECUTE 11

SELECT 12

DISCONNECT 12

Preface

cubeSQL is a fully featured and high performance relational database management system (RDBMS) built on top of the sqlite database engine. It's incredible fast, has a small footprint and it is very scalable. It can runs on Windows, Linux and MacOS X in both 32bit or 64bit versions.

Some features includes:

- Multi-core and multiprocessor aware.
- Strong AES encryption (128, 192 and 256 bit).
- SSL support.
- Supports unlimited connections (For each supported operating system, cubeSQL uses a state of the art event API, kqueue on Mac OS X, epool on Linux and I/O Completion Ports on Windows).
- Full ACID (Atomic, Consistent, Isolated, Durable) compliant.
- Platform independent storage engine.
- Full support of triggers and transactions.
- Journal engine for crash recovery.
- Supports databases of 2 terabytes.
- Supports sqlite 3 databases.
- Automatic logging.
- Automatic compression.
- Multiversion concurrency control (MVCC).
- Plugins for extending the SQL language and the custom commands supported by the server.
- Restore and backup support.
- Mac OS X, Windows and Linux support.
- Native 32bit and 64bit supports.

... and much more

cubeSQL can be access by:

- REALbasic
- PHP
- C/C++/ObjC (whit the C SDK)
- DLL
- ODBC (available soon)
- any JSON client

Read the Five Minutes Guide chapter in order to be able to quickly setup and running cubeSQL and then read the Admin Manual in order to fully understand how to administer and register your server, finally read the Language Reference in order to know all the custom commands recognized by the server.

System Requirements

MacOS X:
MacOS 10.5 or higher (PPC or i386) with a 32bit or 64bit processor.

Windows:
Windows XP SP 2 or higher/VISTA/7/8 with a 32bit ot 64bit processor.

Linux:
Linux kernel 2.6.2 or higher with a 32bit ot 64bit processor.

Default installation paths

MacOS X:
Setting file in /Library/Preferences/cubesql.settings
Everything else inside /Library/cubesql/

Windows:
Everything inside:
Win XP C:\Documents and Settings\All Users\Application Data\cubesql
Win VISTA/7/8 C:\ProgramData\cubesql

Linux:
Setting file in /etc/opt/cubesql.settings
Everything else inside /var/opt/cubesql

Remove REAL Server

If you have installed an old copy of the REAL Server (or REAL SQL Server) just use:

- on OSX: Remove REALServer AppleScript Application
- on Windows: Official Uninstaller from Windows Control Panel
- on Linux: uninstall_realserver script

your databases and settings will NOT be deleted!.

Five Minutes Guide

MacOS X

1. Install cubeSQL using cubeSQL.pkg installer.

2. If you decided to install the StartupScripts then cubeSQL will automatically be launched when your computer starts-up. Note that in order to start/stop server you can use the cubeSQL PreferencePane installed in your system.

3. Once the server is running (started by either a restart of your Mac or using the Preference Pane) you can connect to it using the Admin application located inside the main cubeSQL folder (installed into your Application folder)

4. Connect to a running server using the default values (default username is **admin** and default password is **admin**).

5. Now you can read the Admin Manual in order to fully understand how to administer your server and how to register it and then read the Language Reference in order to know all the custom commands recognized by the server. Please remember that the server is by default in autotransaction mode, that means that all your sql statement must be finalized with a COMMIT sql command.

Windows

1. Install cubeSQL using cubeSQL installer.

2. cubeSQL will be installed as a Service on your system (you can manage it under Control Panel -> Administrative Tools).

3. Once the server is running you can connect to it using the Admin application located inside the main cubeSQL folder (installed into your Application folder)

4. Connect to a running server using the default values (default username is **admin** and default password is **admin**).

5. Now you can read the Admin Manual in order to fully understand how to administer your server and how to register it and then read the Language Reference in order to know all the custom commands recognized by the server. Please remember that the server is by default in autotransaction mode, that means that all your sql statement must be finalized with a COMMIT sql command.

Linux

1. Install cubeSQL using the appropriate installer for your Operating System.

2. cubeSQL will automatically be launched when your computer starts-up. Note that in order to manually start/stop server you can use the cubesqlctl utility installed in your system (with sudo privileges).

3. Once the server is running (started by either a restart or using the cubesqlctl utility) you can connect to it using the Admin application located inside the main cubeSQL folder (installed into your /opt/cubesql)

4. Connect to a running server using the default values (default username is **admin** and default password is **admin**).

5. Now you can read the Admin Manual in order to fully understand how to administer your server and how to register it and then read the Language Reference in order to know all the custom commands recognized by the server. Please remember that the server is by default in autotransaction mode, that means that all your sql statement must be finalized with a COMMIT sql command.

SSL

Starting from version 4.3 cubeSQL fully support the SSL protocol. SSL is automatically loaded if OpenSSL is installed on your system. OpenSSL is freely available from: http://www.openssl.org

What is SSL?

Within cubeSQL SSL tries to do two things:

- Encrypt and verify the integrity of traffic between the browser and the server.

- Verify that the browser is talking to the correct server. In practice, this usually means verifying that the owner of the domain and the owner of the server are the same entity. This helps prevent man-in-the-middle attacks. Without it there's no guarantee that you're encrypting traffic to the right recipient.

How do I set up SSL?

You first need to create a SSL certificate or just purchase it from a trusted company like godaddy, cheapssls, verisign or digicert. OpenSSL offers a command line tool that helps you create your own digital SSL certificate. Here you go a step by step tutorial:

```
Generate the private key
# openssl genrsa -des3 -out cubesql.key 1024

Generate the CSR (certificate signing request)
# openssl req -new -key cubesql.key -out cubesql.csr

Generate the self signed certificate
# openssl x509 -req -days 365 -in cubesql.csr -signkey cubesql.key -out cubesql.crt

Remove the passphrase from the key (important step!)
# cp cubesql.key cubesql.key.copy
# openssl rsa -in cubesql.key.copy -out cubesql.key

Generate pem file (a pem file contains the certificate and the private key)
# cat cubesql.crt cubesql.key > cubesql.pem
```

Then copy cubesql.pem in the main cubesql folder located in:

```
MacOS X: /Library/cubesql/
Linux: /var/opt/cubesql
Windows XP: C:\Documents and Settings\All Users\Application Data\cubesql
Windows VISTA/7/8: C:\ProgramData\cubesql
```

On Windows the command:

```
cp cubesql.key cubesql.key.copy
```

can be replaced with

```
copy cubesql.key cubesql.key.copy
```

and the command:

```
cat cubesql.crt cubesql.key > cubesql.pem
```

with

```
copy /b cubesql.crt+cubesql.key cubesql.pem
```

The same process must be followed to create a certificate to be used by clients that needs to connect to the server in SSL mode. Once a certificate has been created then you need to pass its path to the C library using the proper functions. Please refer to the C SDK documentation or to the REAL Studio documentation about the correct way to setup a SSL connection.

MVCC

In order to increase concurrency cubeSQL supports MultiVersion Concurrency Control (MVCC), a standard technique for avoiding conflicts between reads and writes of the same object. MVCC guarantees that each transaction sees a consistent view of the database.

MVCC is a fairly common technique in database implementation and all the modern DBMS adopt an implementation of the MVCC algorithm developed by Jim Starkey while he was working at DEC (he is the database architect who developed InterBase, the first relational database to support multi-versioning and he is currently working for MySQL AB).

Implements a complete MVCC algorithm inside a database adds a big overhead so a lot of DBMS uses a sort of relaxed version in order to achieve better performance. cubeSQL uses an optimistic variant of MVCC that provides execution time consistency. (PostgreSQL uses locks -- a pessimistic scheme.) Instead of raising a ReadConflictError to signal a consistency problem, cubeSQL automatically reads non-current data that provides consistency, in other words when MVCC is ON read operations are always UNCOMMITTED.

When MVCC is ON server is able to offer a much better concurrency in situations where there is a very large number of concurrent writers. In general Write operations are slower when MVCC is ON and Read is always UNCOMMITTED. If you are upgrading from REAL Server 2009/2010 you probably want MVCC to be turned ON.

When MVCC is OFF server works in safer mode. The behavior is identical to the one offered by the standard sqlite engine and Write operations are always performed at full speed. Read operations are COMMITTED so for some applications this could be the preferred and safer behavior. If you are upgrading from REAL SQL Server 2008 you probably want MVCC to be turned OFF.

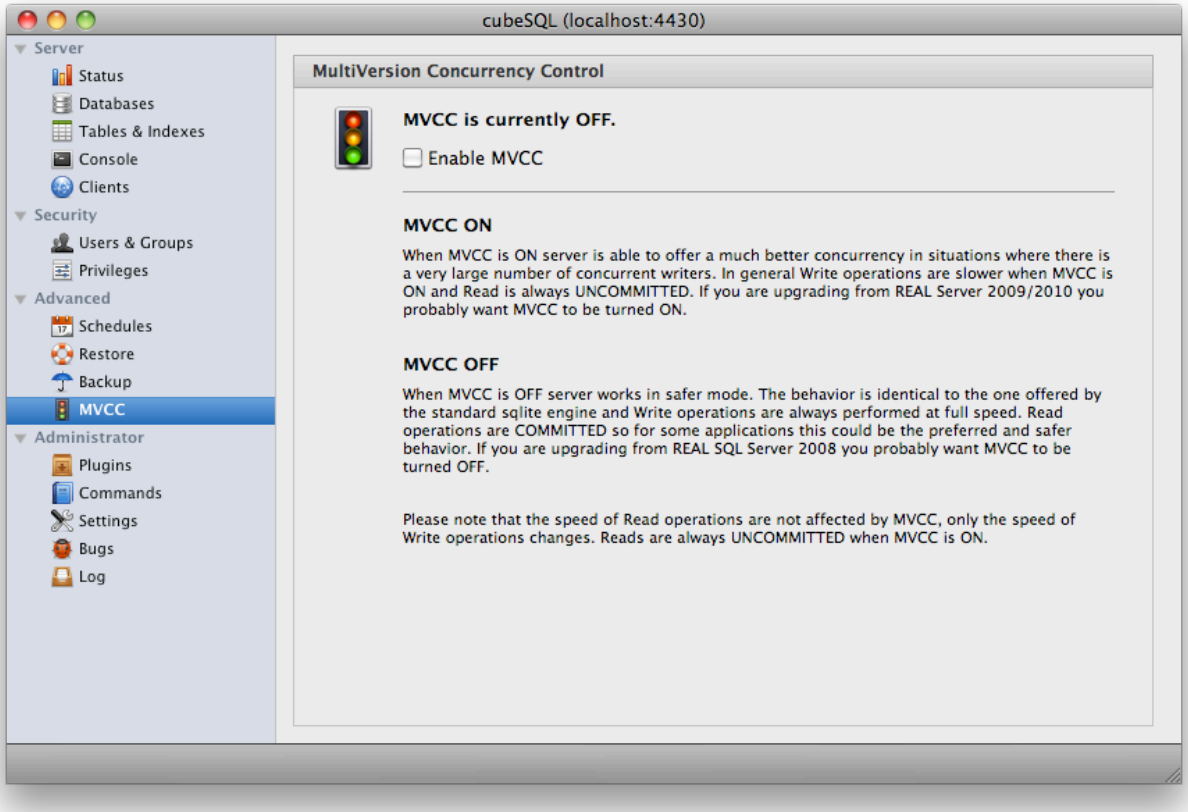
There are several MVCC related custom commands, one to query for its status:

```
SHOW MVCC
```

and two to enable/disable it:

```
ENABLE MVCC
DISABLE MVCC
```

You can run these commands from REAL Studio, or using the C SDK or PHP or any other supported client. An easy way to play with the MVCC is to just use the graphical interface provided by the Admin application:



REALbasic example
This example enabled MVCC on the server.

```
db = New CubeSQLServer
db.Host = "localhost"
db.port = 4430
db.UserName = "admin"
db.Password = "admin"

If (db.connect = false) then
    MsgBox "Connect error: " + db.ErrorMessage
    return
end if

db.SQLExecute("ENABLE MVCC;")
If db.error then MsgBox "An error occurred: " + db.ErrorMessage
```

JSON

In order to try to supports as much heterogeneous clients as possible cubeSQL fully supports the JSON open standard protocol. JSON is a lightwave text based protocol and is built-into any major language (like PHP, Ruby, LiveCode and so on). In this version only JSON over TCP/IP is supported, next version will also support JSON over HTTP.

For a complete and working JSON implementation we strongly suggest you to take a look at the cubeSQLServer.php class.

For basic operations are supported by our JSON implementation, connect, execute, select and disconnect.

CONNECT

```
{
     "command":"CONNECT",
     "username":"admin",
     "password":"admin",
     "randpool":"12345"
}
```

This is the first command that is required in order to open a JSON connection with the server. username is SHA1*(randpool + username), password is BASE64(SHA1(SHA1(password))). randpool is any random integer array.

+ is the string concatenation symbol).
*SHA1 for username is in HEX mode

In case of error any JSON command returns:

```
{
     "errorCode":"7047",
     "errorMsg":"An error occurred..."
}
```

In case of a successful execution errorCode is set to 0.

EXECUTE

This command executes an sql statement on the server:

```
{
     "command":"EXECUTE",
     "username":"UPDATE foo SET col1='test';"
```

```
}
```

SELECT

This command executes an sql query on the server and returns a cursor using the JSON protocol:

```
{
     "command":"SELECT",
     "username":"SELECT * FROM foo;"
}
```

DISCONNECT

This command close current connection with the server:

```
{
     "command":"DISCONNECT"
}
```