1 Installing MySQL under Windows

You can download MySQL[1] from `http://www.mysql.com/`. You will need to run the setup program, which installs MySQL in `c:\mysql` directory. You then have to 'install' the MySQL service. You do this by running: `c:\mysql\bin\mysqld-nt --install`

To uninstall it, you'd do: `c:\mysql\bin\mysqld-nt --remove`

Once you've *install*ed the service, you can start it by running: `net start mysql`. To stop the service, you'd do: `net stop mysql`

Once the service is installed, it will be started automatically next time you boot the machine. MySQL doesn't take up a lot of resources, so it is usually not a major issue to have it running in the background.

You can now add `c:\mysql\bin` to your `PATH` environment variable.

2 Installing MySQL under Linux/UNIX

If you're using a major Linux distribution, like RedHat, chances are, MySQL is already installed (and possibly running) on your system. If you're using Gentoo (and don't have MySQL installed, you'd just do: `emerge mysql`).

To manually compile MySQL, download source code from `http://www.mysql.com/`, uncompress, change to that directory, and run `./configure`, then `make`, and then `make install`. Of course, you have to be logged in as `root` to install MySQL.

Once installed, you'll have to make sure that MySQL starts next time you reboot the machine. That's sometimes in the `/etc/init.d/` or `/etc/rc.d/` directories, depending on your flavor of Linux.

3 Changing `root` Password.

Right after installation, MySQL has 'root' account[2] with an empty password. So to login (and obviously change the administrator password) you'd just do: `mysql -uroot`.

There are several ways of changing the `root` password. A simple one is with an update statement. To do that, switch to use `mysql` database and:

```
mysql> use mysql;
Database changed
mysql> update user set password=PASSWORD('12345') where user='root';
```

The above would change the password for user `root` to `12345`[3]. You then need to logout, and run `mysqladmin -uroot reload`. This will 'reload' the permissions table, and allow you to use your new password. [4]

[1] Get MySQL version 4.

[2] This is separate from the operating system login.

[3] This is a pretty stupid password.

[4] Note that there is also a way to change the password via the `mysqladmin` program.

4 Creating Databases and Users

Usually, you'd want to have many users, each of whom will have access to their own database. For example, if you're development a website, that's itself a database user, etc.[5]

4.1 Creating Databases

To create a database named `webdb` you'd do something like:

```
mysql> create database webdb;
Query OK, 1 row affected (0.00 sec)
```

Usually, that's all there is to it. For a more advanced case, you can also specify many different options when a database is created. Look those up in the MySQL manual (found on `http://www.mysql.com/`).

4.2 Creating Users

You can create users just by granting them privileges. For example, to add a user (and give them all rights on `webdb` that we just created), you can do:

```
mysql> grant all on webdb.* to johndoe identified by '12345';
Query OK, 0 rows affected (0.09 sec)
```

Which will add user named 'johndoe' with password '12345'; and give them all rights on the 'webdb' database. If you don't specify a password, then none is assigned.

You use a similar technique to grant existing user's privileges to various databases; except you're usually a bit more specific about what you're granting and on what tables.

It is very important that you 'reload' the permission table after you add a user; you can do that by running:

```
mysqladmin -uroot -p reload
```

Note the `-p` option. It tells `mysqladmin` to ask you for your password. Without it, `mysqladmin` will assume that the password is blank, and will usually fail (since we *did* change the `root` password).

5 Using MySQL

To use MySQL, login as the user you've created via:

```
> mysql -ujohndoe -p webdb
Enter password: *****
```

Note that we are specifying username with `-u`, we are asking `mysql` to ask us for a password, and we are also telling it that we want to use `webdb` database.

Once in, we see a prompt:

[5] It is relatively common for database users to be applications, and not 'humans'.

```
mysql>
```

Where you can type any SQL statements to create tables, insert records, etc. You can also type 'help' for help on various commands (like how to import data from files, etc.)

5.1 Simple Session

Let's create a `user` table, that's typical of web sites:

```
mysql> CREATE TABLE user (
    ->  id INT UNSIGNED AUTO_INCREMENT,
    ->  username VARCHAR(64) NOT NULL,
    ->  password VARCHAR(64) NOT NULL,
    ->  name VARCHAR(64) NOT NULL,
    ->  email VARCHAR(64) NOT NULL,
    ->  PRIMARY KEY(id),
    ->  UNIQUE(username)
    -> );
Query OK, 0 rows affected (0.10 sec)
```

Having that table, we can now add records to it:

```
mysql> INSERT INTO user (username,password,name,email)
    -> VALUES ('johndoe','12345','John Doe','jdoe@yahoo.com');
Query OK, 1 row affected (0.00 sec)
```

We can now retrieve the user we've added:

```
mysql> SELECT * FROM user;
+----+----------+----------+----------+----------------+
| id | username | password | name     | email          |
+----+----------+----------+----------+----------------+
|  1 | johndoe  | 12345    | John Doe | jdoe@yahoo.com |
+----+----------+----------+----------+----------------+
1 row in set (0.00 sec)
```

Obviously, most of the time when interacting with databases, users don't actually use this `mysql` command line client to add or retrieve records. This is just a tool meant for administrators or developers; i.e.: 'us'.

A typical website would present a user with a 'create user account' page, which would have the user fill out a form and submit it to the server. The server (which may be running PHP, Java, Perl, ASP, etc.) then takes the form data and calls on the database to create a record. In other words, the user never sees the database.