

AIT DBManager User's Guide

Version 1.0

Table of Contents

*Search All Your Favorite Engines from a Single Source with **tyBit!!!***
([Download Now](#))

I. Introduction

- 1.1 [Provisioning of Reseller DBManager and End User DBManager](#)
- 1.2 [Accessing DBManager](#)
- 1.3 [Basic Features](#)

II. Creating an Example Database

- 2.1 [Database definition](#)
- 2.2 [Entering Administration area](#)
- 2.3 [Creating a new database](#)
- 2.4 [Creating a table in the new database](#)
- 2.5 [Adding fields to the table](#)
- 2.6 [Adding records to the table](#)
- 2.7 [Selecting data from your database](#)
- 2.8 [Editing data](#)

III. Provisioning DBManager for your Virtual Host (resellers only)

- 3.1 [Copying files](#)
- 3.2 [Editing configurations](#)
- 3.3 [Password protecting](#)

IV. Common Errors

I. Introduction

Welcome to AIT's DBManager tutorial. This tutorial assumes that you already have MySQL and the DBManager installed onto your Virtual Server. In this tutorial we will create a very simple and small database using the GUI interface the DBManager provides.

1.1 Provisioning of Reseller DBManager and End User DBManager

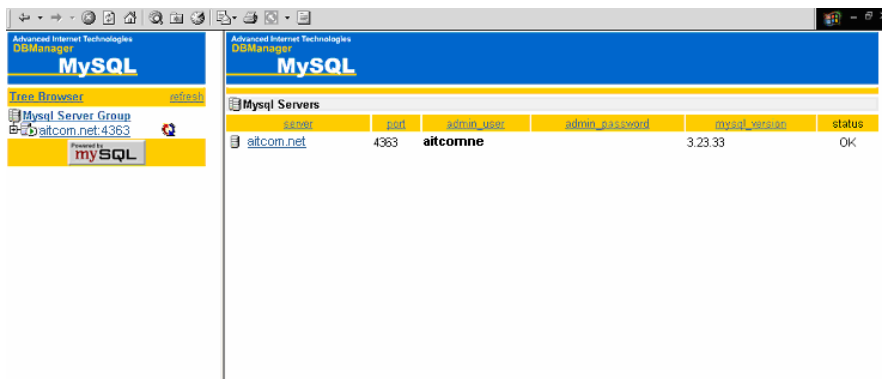
If you want to have DBManager on your server, you must understand how AIT provisions this service. You must either have the Reseller DBManager, which is specifically for Server 2 accounts and above so that you can resell this offering to your customers, or the End User DBManager, which is specifically for Server 1 accounts. This feature can not be resold, since Server 1's do not have virtual hosts.

1.2 Accessing DBManager

First, access your DBManager in the following manner:

<http://domain.com/dbmanager/>

Of course, you would replace the domain.com with your virtual server domain name, and your username and password would be the same as your control panel. For resellers who provision the DBManager for virtual hosts, we will cover how to access and password protect it later in this tutorial. Once you are inside of your DBManager, you should see a screen such as this one:



1.2 Basic Features

In the left frame you will see your domain name and the port number that MySQL is using to listen for connections. If you had a database or more than one database created, a small 'plus' symbol would appear next to the domain name, and if you clicked on it, the listing would expand and the databases currently running would be shown. In the right hand frame, the following information will be displayed:

1. The name of the MySQL server.
2. The port number that MySQL is using to listen for connections.
3. The administrative username recognized by MySQL.

4. The administrative password if any that is set.
5. The version of MySQL that is being run on the Virtual Server.
6. The current status of the MySQL server.

If the status is stopped, you can start the MySQL daemon by simply clicking on the icon next to the server's name in the left frame of your browser. When running, the status should be listed as "OK".

II. Creating an Example Database

In this tutorial, we will create a small database that will contain people's names, email addresses, phone numbers, and an ID number that will be assigned to people in the database. This database can have quite a few practical uses to someone running a commercial website, and this is a very simple example of the kind of database that can be put into use.

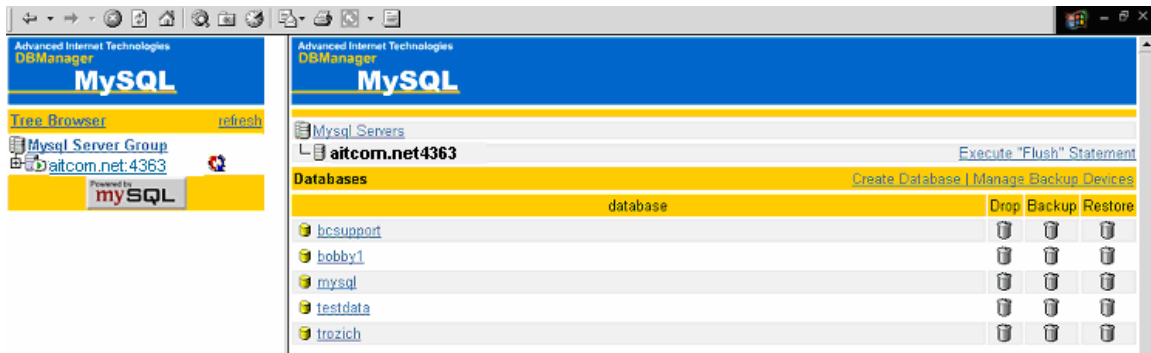
2.1 Database Definition

A database is simply a collection of organized data. A relational database organizes data into tables, which makes the collection of this data much easier to categorize. The tables will consist of fields which will have different records inserted into them.

2.2 Entering Administration Area

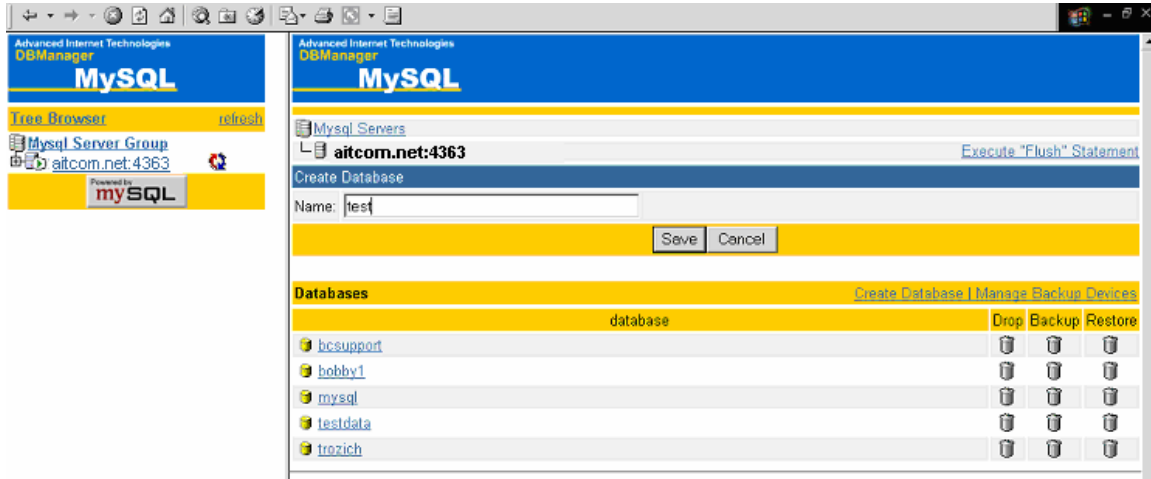
First let us create a database using the DBManager's GUI interface. After you login, click on the server name. In the right frame you will see the area where the database creation/administration is done. The right frame will be divided into four main sections under the MySQL server name:

1. Databases currently created (if any)
2. Users that have been created (your main user will be the default)
3. Status Variables
4. Process List

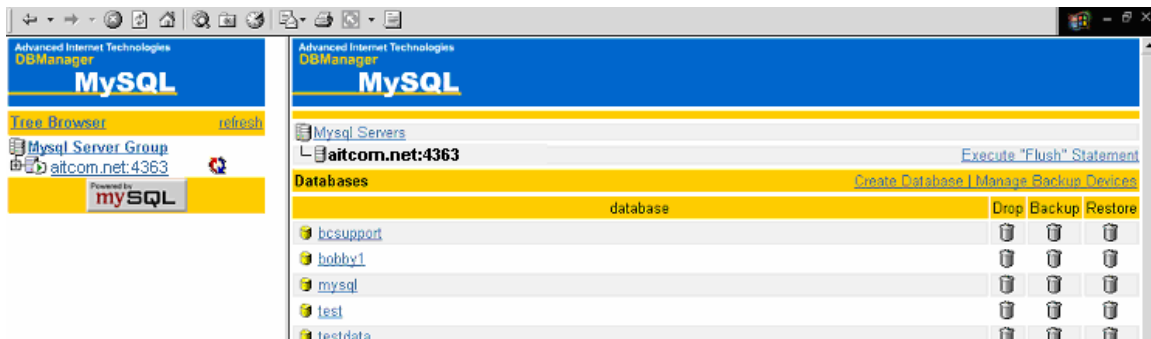


2.3 Creating a database

In the section that holds the database listing (above), click on the hyperlink labeled “create database” which will bring you to the following screen:



You simply need to type the name of the database you want to create in the field labeled “name” and click the gray button below it labeled “save”. In this example, we will create a database called “test”. This will bring you back to the screen with the database listings, and you should see your database listed. In the screenshot below you should see the database called “test”.

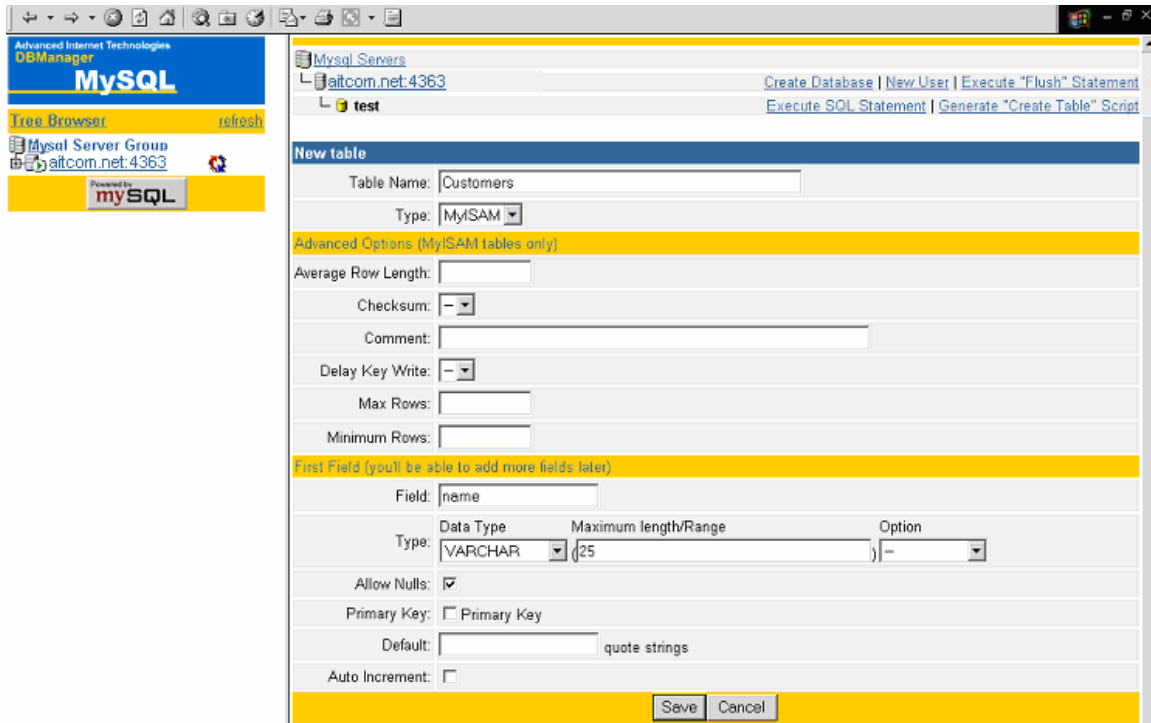


2.4 Creating a table in the new database

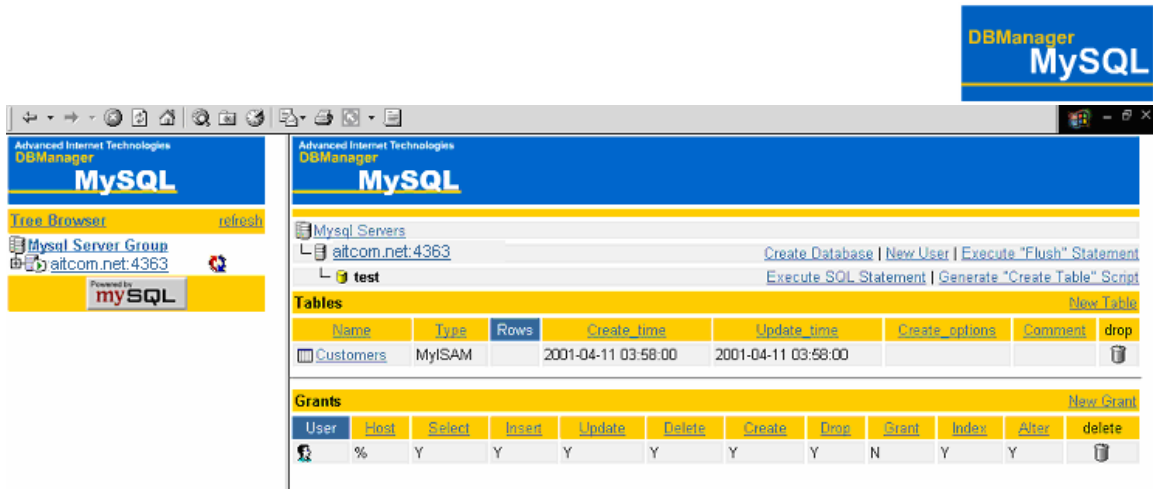
Click on the hyperlink of your database, which in this case is “test” and this will bring you to the following screen:



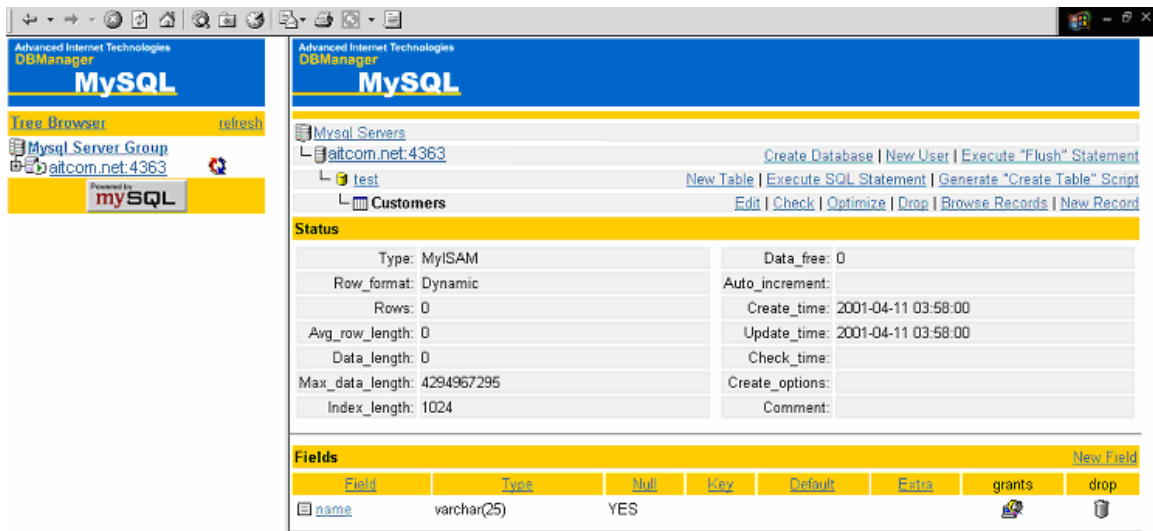
There will be a listing for Tables that are currently set up in your database. Click on the hyperlink on the right of the table listing which reads “New Table” and a new screen will be presented. You will be required to enter not only the name of the table, but you will also need to enter in one of the fields that you will be using in your table. In this example, we will create the table “Customers” and the initial field being created will be “name”. Take a look at the screen shot:



Under the “New Table” heading you will see the table name field. The table we are creating is going to be called “Customers” so that is what you will enter inside the field. The field called “type” is the type of table your data will be stored in. ISAM used to be the old default, but MyISAM tables are the new default for MySQL and hold a number of advantages over ISAM table, so select MyISAM as your table type. We will leave the advanced options blank, and skip down to the first field and create the “name” field. The fields are what will hold the data such as names, email addresses, etc. Select “name” as your initial field and then go down to the field labeled “type”. For the “Data Type” we will select “VarChar” which will enable us to use 1-225 characters and will store the data as a variable length string instead of a fixed length. For the Maximum length, we will use “25” characters although you could have set it to 255 characters. Lastly, we will check the box labeled “Allow Nulls” which will accept a null value into the name field. Click save and the table and initial field will be created. You will be returned to the following screen:

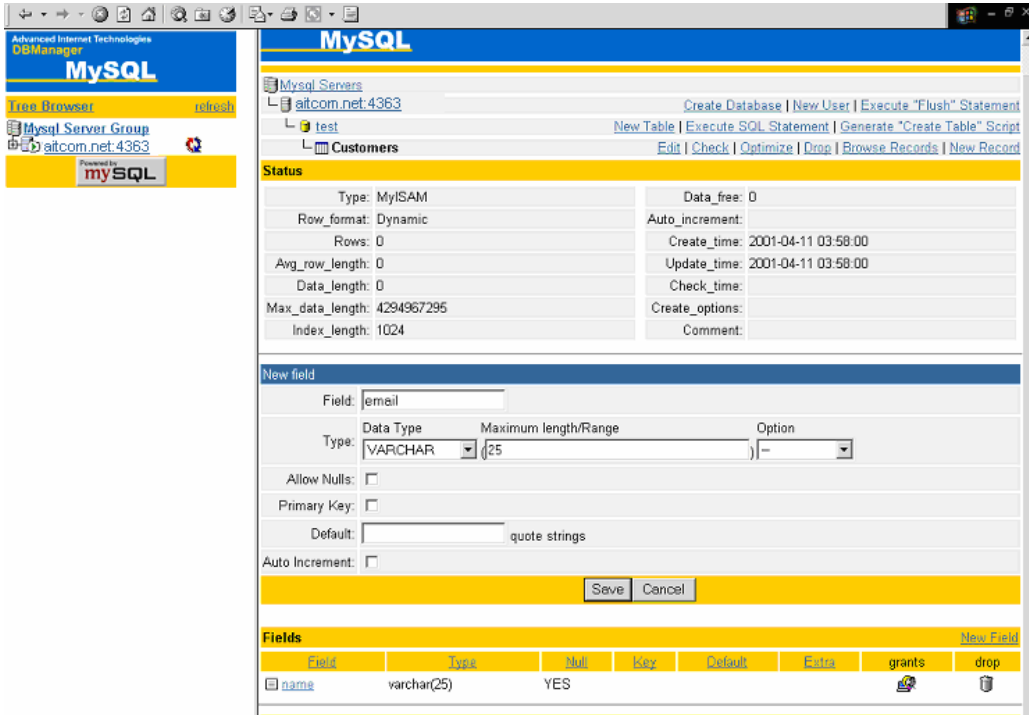


Now you will see the table “customers” under your table listing. It will show the type of table it is, the time it was created, and the last time it was updated. Clicking on the hyperlink “Customers” will generate a page that shows the status of the table, and what fields currently exist in that table.



2.5 Adding fields to the table

Notice that next to the server name, you will have a list of options allowing you to create a new database or user. Across from the database name “test” you will have a list of options that are used to manipulate your database including creating new tables. Beside your table called “Customers” you will have the option to manipulate the table. Now, we’ll finish entering our fields that will be used with the table “Customers”. Under the Fields listing, click on the hyperlink labeled “New Field” and you will come to the following screen:



A section labeled “New Field” will appear and it is the same as the previous section we used to enter in our field labeled “name”. In the field labeled “Field” type in “email”. This field we are creating is going to store customers’ email addresses. Email addresses can vary in size depending on the customer’s username and domain name. For this reason, we want to select “VARCHAR” as the Data Type. For the maximum length we will choose “25” although you could make this much larger to accommodate larger email addresses. Click the gray button labeled “Save” and the field will be created. We will repeat the process to create the field for phone numbers. The field name will need to be “phone_numbers” since the field names do not allow for spaces. As for the Data Type you will select “VARCHAR” so that you can use numbers and dashes. Set the field for “Maximum Length” to 12 so that this will give enough room for the area code and phone number. Last, we are going to have a field for “ID” created so that each customer in the database will have an identification number assigned to them. We will also create the field so that as new records are added, this particular field will automatically assign a new number to the customer. To do this click on the “new field” link as you have done before, and use the name “ID” as the field name. For the Data Type, select “INT” since numbers will be used and leave the maximum length blank to use the default. Check the box labeled “primary key” and this option will make sure that no two ID’s for any of the records will ever be the same. Also, check the box labeled “Auto Increment” which will ensure that the field is automatically updated as new entries are made. Now all the fields will be created and your screen should represent the following:

The screenshot shows the DBManager MySQL interface. The left sidebar contains navigation links like 'Free Browser', 'MySQL Server Group', and 'aitcom.net:4363'. The main content area displays the 'Customers' table details.

Status

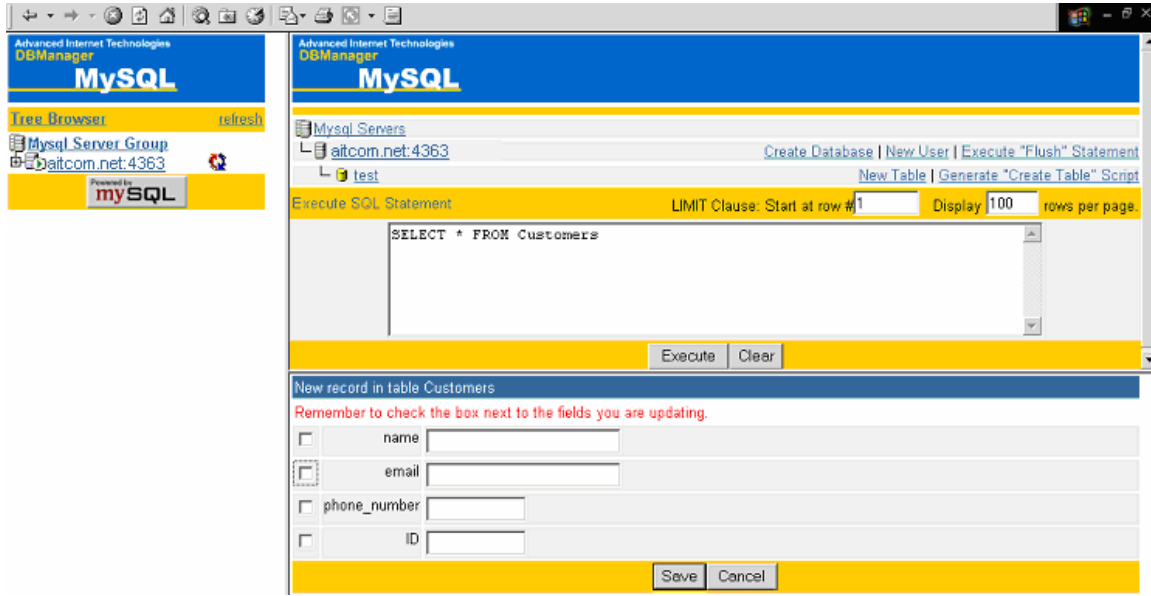
Type: MyISAM	Data_free: 44
Row_format: Dynamic	Auto_increment: 4
Rows: 0	Create_time: 2001-04-11 05:07:52
Avg_row_length: 0	Update_time: 2001-04-11 05:08:39
Data_length: 44	Check_time:
Max_data_length: 4294967295	Create_options:
Index_length: 2048	Comment:

Fields

Field	Type	Null	Key	Default	Extra	grants	drop
name	varchar(25)	YES					
email	varchar(25)						
phone_number	varchar(12)			0			
ID	int(11)		PRI		auto_increment		

2.6 Adding records to the table

Now with all the fields created, we will add records to the database. In the above screen, across from the table "Customers" you will see some hyperlinked options. We will select "New Records". After selecting that option, the following screen will appear:



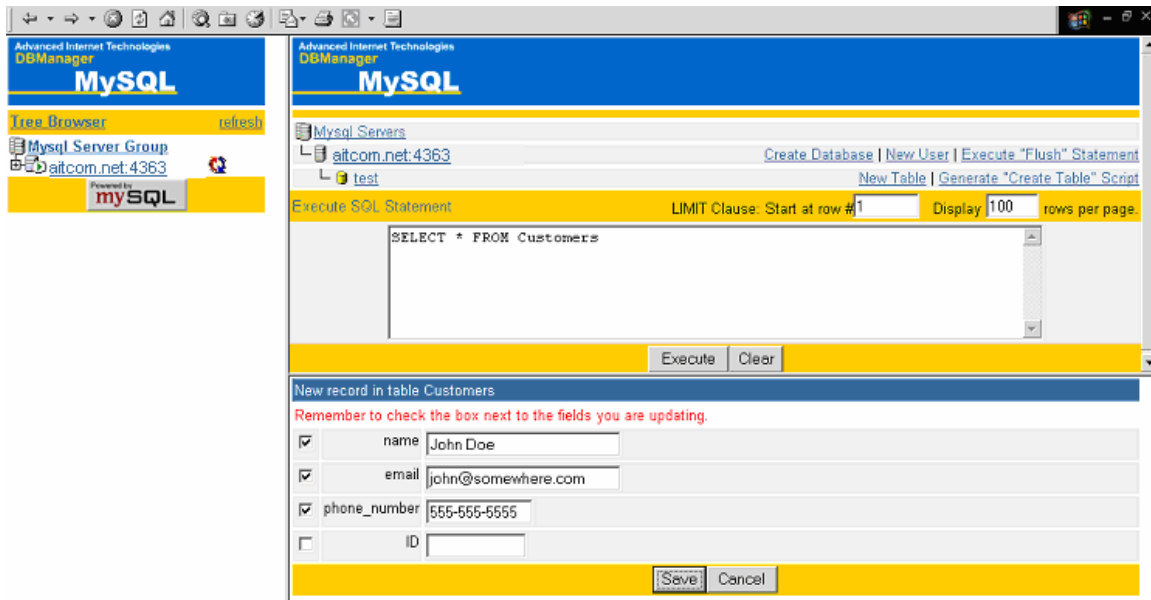
For our example we will enter our first record using the following customer data:

Name: John Doe

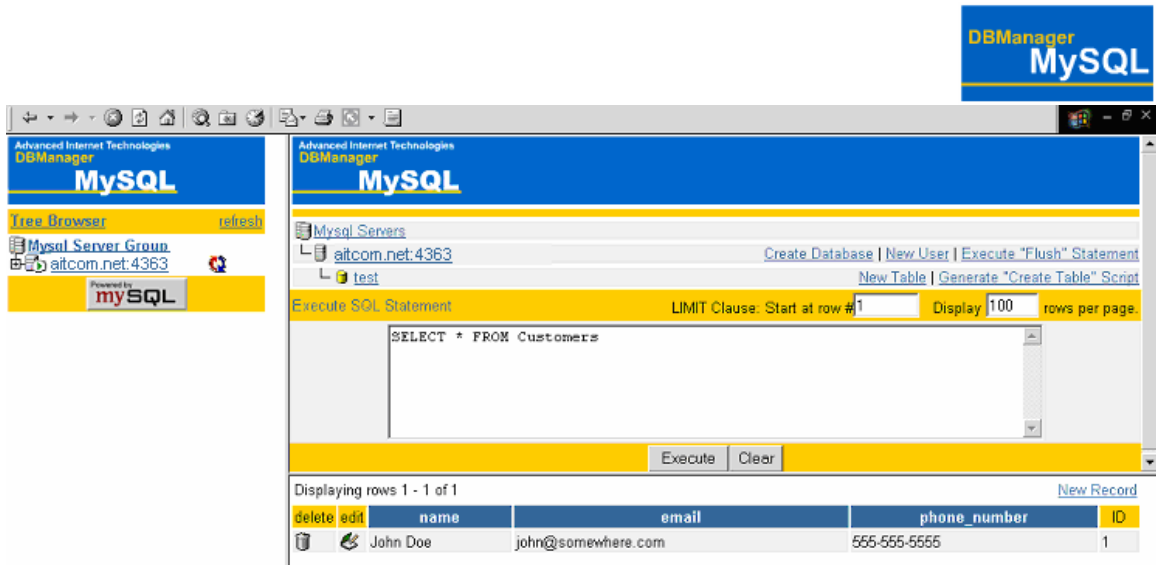
Email: john@somewhere.com

Phone Number: 555-555-5555

In order to properly edit the fields, you will need to place a **check mark next to the field name that you are updating or adding**. Once the checkmarks have been placed next to the appropriate fields, the information should be entered as shown in the screen shot below:



Please note that the ID field is left blank, this is because it was already set to increment automatically. Once the information has been entered, click on the gray button labeled “save” and the records will be entered into the appropriate fields. The screen will then show the record as it was entered into the database:



You now have the option of entering other records into the table by clicking on the link labeled “New Record” and repeating the whole process again. It is very easy to create a database using the MySQL DBManager and this tutorial does not even scratch the surface of what can be done using the DBManager interface.

2.7 Selecting data from your database

To select data from your database, click on the database and then the table you want to select from. From there you will need to click on ‘Execute SQL Statement’.

The SQL statement to select data is as follows:

Select *[record]* from *[table]* where *[field]*='[item]'

The SELECT statement retrieves data from the database. To retrieve the data, you specify a field list, a table list, a list of fields to sort by, and the sort order. The parts of an SQL statement are called clauses. A basic SELECT statement has up to four clauses. For example:

```
SELECT (field1 [, field2]...)
FROM (table1 [, table2]...)
WHERE (condition1 [, condition2]...)
ORDER BY (field1 [ASC|DESC] [, field2 [ASC|DESC]]...)
```

The WHERE and ORDER BY clauses are optional. If you omit the WHERE clause, the query returns all rows from the specified tables. If you omit the ORDER BY clause, SQL retrieves rows in the sequence in which they’re stored in a table. By default, when you retrieve data from multiple tables, SQL uses the row order from the first specified field. At the most basic level, you can obtain all the information from a table using an asterisk (*) as a shorthand way of specifying all fields. Of course, you don’t have to select all fields. You can specify the exact fields and field order that you wish.

Programmers moving from file-based databases to relational databases often make the mistake of thinking that the simple SELECT statement is all they need. They are accustomed to scrolling (moving sequentially from field to field) through a set of records to find the info they need. That’s absolutely the wrong way to approach relational databases. Don’t search for records yourself — let the database do the work. That’s what the

WHERE clause does — it limits the returned records to exactly the ones you need by searching by comparison.

The options used with the WHERE clause are '=' or 'LIKE'. You would use '%' as a wild card when using LIKE in the WHERE clause. The following SELECT statements include = and LIKE:

Perhaps you want to find all Student ID's for lastname of Thompson and Thomas. Try this:
 SELECT StudentID FROM Students WHERE Lastname LIKE 'Thom%'

Or you want to find all Student ID's with the first name of Thomas.
 SELECT StudentID FROM Students WHERE Firstname='Thomas'

The ORDER BY clause of the SELECT statement controls the order of the records returned by the query.

The fields in the ORDER BY clause do not have to appear in the selected field list. The default sort order is ascending (ASC), but you can retrieve fields in reverse order by specifying the DESC keyword after the appropriate field name. You don't have to select all the fields, and you may select them in any order you desire.

The following SELECT statement includes all the basic SELECT clauses:

```
SELECT StudentID, LastName, FirstName
FROM Students
ORDER BY Grade DESC
```

2.9 Editing data

Syntax	INSERT INTO tablename (field1 [, field2]...) VALUES (value1 [, value2]...) INSERT INTO tablename (field1 [, field2]...) SELECT query
	Delete from table_name where [search_condition]
	UPDATE (tablename) SET field1=(value expression) [, field2=(value expression)]... FROM (table query source) WHERE (condition)

SQL INSERT statements add one or more new rows to a table. The INSERT statement has two variations. The first variation adds one row by assigning values to a specified list of columns in a specified table. The values you want to insert follow a VALUES statement. You put parentheses around both the field list and the values list.

For example:

```
INSERT INTO tablename (field1 [, field2]...)  
VALUES (value1 [, value2]...)
```

You must provide a value for all fields that cannot accept a null value and do not have a default value. You do not have to provide values for identity columns.

The second variation lets you add multiple rows using a SELECT query in place of the VALUES list, as follows:

INSERT INTO tablename (field1 [, field2]...) SELECT query

If you're inserting data into all the columns in the target table, you can omit the field list. The SELECT statement you use to obtain the data you want to insert can include any clause or condition discussed in the previous section, including calculated fields and a GROUP BY clause.

UPDATE statements change data in one or more columns and in one or more rows. The UPDATE statement is dangerous, however because if you forget to specify conditions, your database will happily update all the rows in the table. You should always specify a WHERE condition when updating data. The UPDATE statement has the following syntax:

```
UPDATE (tablename)
SET field1=(value|expression) [, field2=(value|expression)]...
FROM (table|query source)
WHERE (condition)
```

The UPDATE statement has four clauses. In the UPDATE clause, you must specify a table name containing the fields to update. You may not update multiple tables simultaneously.

The SET clause contains the list of fields you wish to update. You separate the list with commas. Each item in the list consists of a field name, an equals sign, and a new value. You can use a constant, a variable, a field from another table, or an expression for the value on the right-hand side of the equals sign.

The FROM clause is optional. If you're updating a single row with constant values, you can omit the FROM clause. You need the FROM clause when you're updating data in one table from values stored in a different table (or in another place in the same table). Fortunately, the FROM clause is identical to the FROM clause you saw earlier in The SELECT Statement section. You may update from multiple tables using JOIN statements as appropriate.

The WHERE clause (important: don't forget the WHERE clause!), again, is a condition that identifies the rows in the target table you wish to update.

The DELETE statement is the simplest of all, but quite powerful. You can use the DELETE statement to delete one or more rows in one or more tables. The DELETE statement is just as dangerous as the UPDATE statement, because it cheerfully deletes data without prompting. If you accidentally run a DELETE statement, it's difficult to recover your data. You should rarely use a DELETE statement without a WHERE clause. If you want to delete all the data from a table it's much more efficient to use a different type of statement, one of a group of statements that alters the database itself—the TRUNCATE TABLE statement.

Truncating a table removes all the data and resets the identity column value to its default.

You should rarely use DELETE without a WHERE clause. There is one reason to do so. The TRUNCATE statement is not logged — that means you can't recover if you use it automatically, whereas the DELETE statement is a logged operation. That's the reason TRUNCATE is so much more efficient—it avoids the log operations, but it also means the data is unrecoverable from the transaction log.

III. [Provisioning DBManager for your Virtual Host \(resellers only\)](#)



The dbmanager is composed of two main files, index.cgi and index.cpl. To give your virtual host access to their own dbmanager, simply copy the dbmanager directory into the virtual host's directory. Then you will need to add an alias for that directory.

3.1 Copying files

To add a virtual dbmanager, you will need to FTP to your server and download the dbmanager directory from /www/htdocs/ to your local hard drive. This directory contains one directory (images) and three files. (.htaccess, index.cgi, and index.cpl) Change to your virt's directory and create a directory called dbmanager. Upload the images directory and the index.cpl in binary mode and the other two files in ASCII mode. Next you must set the permissions on these files. Index.cpl and index.cgi should be set to 755.

3.2 Editing configurations

To access your dbmanager from your virtual host, you will need to create an alias. To do this you will need to ftp to your server and change to /www/conf/. Open the srm.conf file for editing. At the very bottom of the file add the following line:

```
Alias /virtdb /www/htdocs/virt/dbmanager/
```

For example, If you are creating a dbmanager for *abcd.com*, you would add the following line:

```
Alias /abcddb /www/htdocs/abcdcom/dbmanager/
```

3.3 Password protecting

To password protect your virtual dbmanager, you will need to FTP to your server and open /www/htdocs/virt/.../dbmanager/.htaccess for editing. You should see something similar to the following.

```
AuthType Basic
AuthName "authentication required"
AuthUserFile /usr/local/etc/httpd/htdocs/dbmanager/.htpasswd
AuthGroupFile /dev/null
Require username
```

Change the AuthUserFile line to read

```
AuthUserFile /etc/passwd
```

Replace the username in the require line to the username of the virtual host. Once this is done, you will be able to log in, create and manage databases for your virtual hosts.

IV. Common Errors

If you see this in the dbmanager status (on the website):

Can't connect to MySQL server on 'domain.com' (111)

the domain may have a resolution issue or is not pointed to us...try this:

[open /www/conf/dbmanager.conf for editing](#)
[change the domain.com in](#)



```
$MysqlTool::servers{1}->{'server'}    = 'domain.com';  
to  
$MysqlTool::servers{1}->{'server'}    = 'localhost';
```

If you see this in dbmanager status (on the website):

Can't connect to local MySQL server through socket '/tmp/mysql.sock' (111)

try this:

start the mysql daemon :)

If the daemon is currently running, then stop the daemon and start it again.

This error is sometimes caused by trying to start the mysqld when it's already started and /tmp/mysql.sock is removed. By stopping and then starting it again, you are recreating the sock file.

If you see something like the following in the apache error_log:

at /usr/lib/perl5/site_perl/5.005/MysqlTool.pm line 116
install_driver(mysql) failed: Can't load '/usr/lib/perl5/site_perl/5.005/i686-linux/auto/DBD/mysql/mysql.so' for module DBD::mysql: libmysqlclient.so.6: cannot open shared object file: No such file or directory at /usr/lib/perl5/5.00502/i686-linux/DynaLoader.pm line 168.

Or this:

Can't locate strict.pm in @INC (@INC contains: /usr/lib/perl5/i686-linux/5.00404 /usr/lib/perl5 /usr/lib/perl5/site_perl/i686-linux /usr/lib/perl5/site_perl .) at - line 3.
BEGIN failed--compilation aborted at - line 3.
httpd: [Wed Apr 4 00:36:26 2001] [error] [client

Or this:

usr/lib/perl5/site_perl/5.005/i686-linux/auto/DBI/DBI.so: undefined symbol: __b
zero httpd: [Sat Mar 31 20:58:12 2001] [error] [client 208.234.1.225] Premature end of script headers: /usr/local/etc/httpd/htdocs/dbmanager/index.cgi

contact Tech support at (877) 209 5184 so it can be resolved.