

## **mSQL to MySQL Conversion**

The purpose of this document is to assist in migrating data from mSQL2 to MySQL.

Brief introduction to mSQL and MySQL data structure:

*MySQL is:*

- a.) ODBC Compliant
- b.) Multi-Treaded
- c.) Many different data types
- d.) Fast Database can handle large amounts of data

*mSQL2 is:*

- a.) Simple database with few data types (field types)
- b.) Does not support primary keys.
- c.) Does not support alter table commands.
- d.) Is NOT ODBC complaint.

***How mSQL2 syntax is similar and different from MySQL...***

Corresponding column types for mSQL and MySQL:

**mSQL**

- 1.) CHAR(len)
- 2.) TEXT(len)
- 3.) INT
- 4.) REAL
- 5.) UINT
- 6.) DATE
- 7.) TIME
- 8.) MONEY

### **MySQL**

- 1.) CHAR(len)
- 2.) TEXT(len).        len is the maximal length. And LIKE works.
- 3.) INT.        With many more options.
- 4.) REAL. Or FLOAT.        Both 4- and 8-byte versions are available.
- 5.) INT UNSIGNED
- 6.) DATE.        Uses ANSI SQL format rather than mSQL's own format.
- 7.) TIME
- 8.) DECIMAL(12,2).        A fixed-point value with two decimals.

### **Additional types supported by MySQL that are not supported by mSQL**

ENUM  
SET  
BIGINT  
UNSIGNED

ZEROFILL  
AUTO\_INCREMENT  
DEFAULT  
PRIMARY KEY

### *SQL differences between mSQL and MySQL...*

#### **Index Creation:**

##### MySQL

Indexes may be specified at table creation time with the CREATE TABLE statement.

##### mSQL

Indexes must be created after the table has been created, with separate CREATE INDEX statements.

#### **To Insert a Unique Identifier into a Table:**

##### MySQL

uses AUTO\_INCREMENT as a column type specifier.

##### MSQL

Create a SEQUENCE on a table and select the \_seq column.

#### **To Obtain a Unique Identifier for a Row:**

##### MySQL

Add a PRIMARY KEY or UNIQUE key to the table and use this. If the PRIMARY or UNIQUE key consists of only one column and this is of type integer, one can also refer to it as \_rowid.

### mSQL

Use the `_rowid` column. Observe that `_rowid` may change over time depending on many factors.

### **To Get the Time a Column Was Last Modified:**

### MySQL

Add a `TIMESTAMP` column to the table. This column is automatically set to the current date and time for `INSERT` or `UPDATE` statements if you don't give the column a value or if you give it a `NULL` value.

### mSQL

Use the `_timestamp` column.

### **NULL Value Comparisons:**

### MySQL

MySQL follows ANSI SQL, and a comparison with `NULL` is always `NULL`.

### mSQL

In mSQL, `NULL = NULL` is `TRUE`. You must change `=NULL` to `IS NULL` and `<>NULL` to `IS NOT NULL` when porting old code from mSQL to MySQL

### **String Comparisons:**

### MySQL

Normally, string comparisons are performed in case-independent fashion with the sort order determined by the current character set (ISO-8859-1 Latin1 by default). If you don't like this, declare your columns with the `BINARY` attribute, which causes comparisons to be done according to the ASCII order used on the MySQL server host.

### mSQL

All string comparisons are performed in case-sensitive fashion with sorting in ASCII order.

### **Case-insensitive Searching:**

### MySQL

LIKE is a case-insensitive or case-sensitive operator, depending on the columns involved. If possible, MySQL uses indexes if the LIKE argument doesn't start with a wild-card character.

### mSQL

Retains trailing space.

### **WHERE Clauses:**

### MySQL

MySQL correctly prioritizes everything (AND is evaluated before OR). To get mSQL behavior in MySQL, use parentheses (as shown in an example below).

### mSQL

Evaluates everything from left to right. This means that some logical calculations with more than three arguments cannot be expressed in any way. It also means you must change some queries when you upgrade to MySQL. You do this easily by adding parentheses. Suppose you have the following mSQL query:

```
MySQL> SELECT * FROM table WHERE a=1 AND b=2 OR a=3 AND b=4;
```

To make MySQL evaluate this the way that mSQL would, you must add parentheses:

```
MySQL> SELECT * FROM table WHERE (a=1 AND (b=2 OR (a=3 AND (b=4))));
```

## Access Control:

### MySQL

Has tables to store grant (permission) options per user, host, and database.

### mSQL

Has a file `mSQL.acl' in which you can grant read/write privileges for users.

## *How to convert a mSQL database to a MySQL database...*

Depending on the size of the database the customer will have a few choices.

- A) Use the database tools in the control panel to dump the contents of the database to their browser.
- 1) Copy and paste the information from their browser to a text file [WordPad/notepad]
  - 2) Use the “find and replace” feature of notepad/WordPad to update the field types.
  - 3) Manually scan the data to verify that the information matches the correct formats.
  - 4) Copy and paste the one table definition at a time into the monitor for MySQL.
    - i) This will create the table.
    - ii) Then copy the insert statements to add the data....
    - iii) Depending on size the customer may have to chop up the tables into sections.
    - iv) Check for errors before moving on to the next table.
- B) If the browser times out while dumping the database or completes but the dump is incomplete, then the customer can contact support to have the database dumped as a plain text to the /u/web/\$userid directory.
- 1) Copy and paste the information from their browser to a text file [WordPad/notepad]
  - 2) Use the “find and replace” feature of notepad/WordPad to update the field types.
  - 3) Manually scan the data to verify that the information matches the correct formats.
  - 4) Copy and paste the one table definition at a time into the monitor for MySQL.
    - i) This will create the table.
    - ii) Then copy the insert statements to add the data....
    - iii) Depending on size the customer may have to chop up the tables into sections
    - iv) Check for errors before moving on to the next table.
- C) The customer would like they can use Perl or PHP to create a script that will convert the database. (We recommend against using the -c option.)
- 1) Read database for table names.
  - 2) Read each table for the table definitions.

- 3) Use a case structure to create the same tables using MySQL definitions.
- 4) If the table does not have an index field that is a unique integer for each entry, and increments for each entry, you might want to create one. This will make it easier to track which row you've exported and what rows you've successfully imported
- 5) Depending on size....
  - i) Have a variable that you can use to remember how many rows into the update you are.
    - (a) PHP 30 seconds time-out.
    - (b) Perl XXX seconds timeout.
  - ii) Then run "Select \* FROM \$table\_name".
  - iii) Then you will need to set up an "if" statement to check if the value of the data from mSQL matches the data value expected for MySQL.
    - (a) Matches --> then insert the new row into the table
    - (b) Doesn't Match --> then convert the data type to match the new database.
  - iv) Repeat until either all of the data has been converted or the script times-out.

For examples of data types in mSQL and the similar matches in MySQL, [click here](#).