CET Software

Commercial Application Specialists

MySQL Support for Linux
21 May 2007

1. Overview

The runtime system and library for CET BASIC for Linux has been enhanced to provide support for MySQL databases. The document “CET W/32 MySQL support module” (that can be found on http://www.cet-software.com/downloadable.html) describes most of the what is needed to convert CET indexed files. The following sections will note the differences in implementation between our Linux and Windows version.

2. Using Native MySQL Support

CET BASIC programs can be converted to use MySQL tables instead of indexed files. The only change required in the CET BASIC programs is to add the new keyword “SQL” to the open statement, for example:

OPEN #1, “/opt/data/isamfile”, INDEXED UPDATE, SQL

The presence of the SQL keyword causes the CET runtime (/lib/Brtm) to perform operations on a MySQL database.

3. Name Mapping

CET BASIC for Linux uses a mapping file to associate CET file names and MySQL tables. The environment variable B_SQLMAPFILE must have a value equaling the name of the mapping file. For example, in a csh .login file, one may include:

setenv B_SQLMAPFILE /opt/newbas/sqlmapfile

This file must contain the name of the BASIC file, followed by the name of the MySQL table, for example:

```
/opt/data/testsql,testsql
/opt/data/bonkers,matsql
```

When the OPEN statement is processed, each OPEN statement with a SQL keyword must have a file name (such as “/opt/data/testsql”) followed by the name of the MySQL table.

4. Converting Indexed Files

To convert indexed files for use by MySQL, it is necessary to run the bsqlconv program. Currently, this program is only available for Windows, but CET indexed files can be copies from Linux to Windows and the conversion performed there.

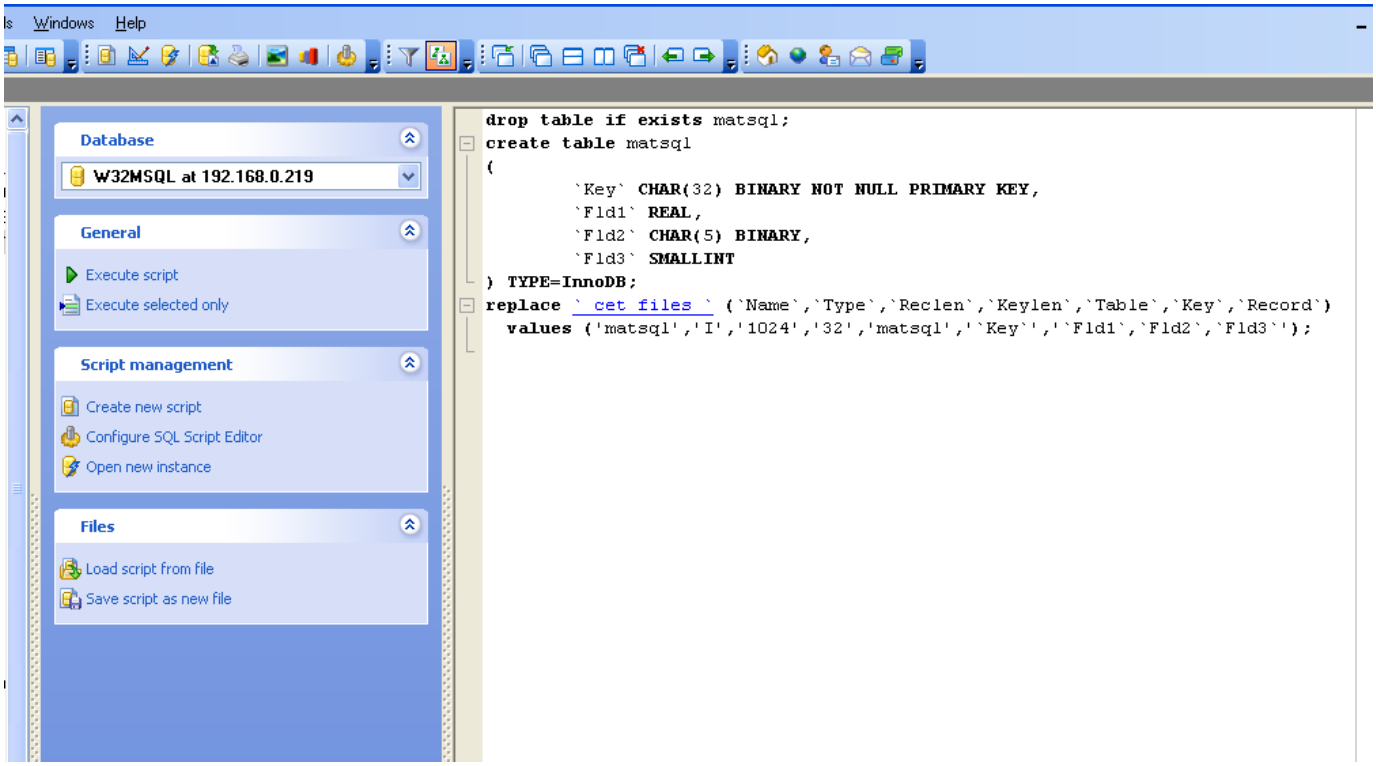
The bsqlconv utility performs two functions. The first is to produce the SQL script that will create the MySQL table. For example, consider a CET BASIC indexed file that has a key field, followed by a decimal floating-point, a string, and an integer. A BLIST of the indexed file looks like:

```
Blist ./matsql
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
./matsql (Indexed)                                  May 21 11:39:23 2007 Page 1
Key    Length = 32
Record Length = 1024
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
1                               :.55,stg1,1
10                              :5.5,stg10,10
2                               :1.1,stg2,2
3                               :1.65,stg3,3
4                               :2.2,stg4,4
5                               :2.75,stg5,5
6                               :3.3,stg6,6
7                               :3.85,stg7,7
8                               :4.4,stg8,8
9                               :4.95,stg9,9
```

When bsqlconv is run on this file, it produces:

```
$ bsqlconv matsql
Input file: matsql
# Indexed file 'matsql' Key Length 32 Record Length 1024
No inconsistencies found
drop table if exists matsql;
create table matsql
(
        `Key` CHAR(32) BINARY NOT NULL PRIMARY KEY,
        `Fld1` REAL,
        `Fld2` CHAR(5) BINARY,
        `Fld3` SMALLINT
) TYPE=InnoDB;
replace `_cet_files_` (`Name`,`Type`,`Reclen`,`Keylen`,`Table`,`Key`,`Record`)
  values ('matsql','I','1024','32','matsql','`Key`','`Fld1`,`Fld2`,`Fld3`');
load data infile 'c:/tmp/matsql.txt' into table matsql
  fields terminated by ',' enclosed by '"' lines terminated by '\r\n';
```

This SQL script can be used to create the table and load it.



This script will create the table and store its field descriptions into a special table called _cet_files_, which CET uses during execution to perform SQL queries.

5. Modifying Table Descriptions

The bsqlconv program attempts to pick reasonable data types for each of the fields that it analyzes in the indexed file. However, the CET runtime supports many data types, including:

TINYINT
SMALLINT
INTEGER
MEDIUMINT
BIGINT
DECIMAL or NUMERIC
FLOAT
DOUBLE or REAL
YEAR
CHAR or BINARY
VARCHAR or VARBINARY

For example, the sample script produced by bsqlconv could be changed to:

```
create table matsql
(
        `Key` CHAR(128) BINARY NOT NULL PRIMARY KEY,
        `Fld1` DECIMAL(10,2),
        `Fld2` VARCHAR(1024) BINARY,
        `Fld3` INTEGER
) TYPE=InnoDB;
```

The changes from CHAR to VARCHAR, for example will save space in the MySQL database. The change from REAL to DECIMAL will preserve decimal precision. Similarly, the change from SMALLINT to INT will permit 4-byte integers to be stored in the database.

6. Describing the MySQL Database

To describe the name and location of the MySQL database, as well as the username and password, set an environment variable, such as:

```
B_MYSQL=192.168.0.219:root:JohnDave:W32MSQL
```

7. Data Conversion at RunTime

As always, CET BASIC is very forgiving when a READ or WRITE statement specifies a type different than the underlying datatype of the indexed file or MySQL table. A file can be written with a statement such as:

WRITE #1,KEY$: A, A$, A%

and be read back as

WRITE #1,KEY$: A$, B$, A

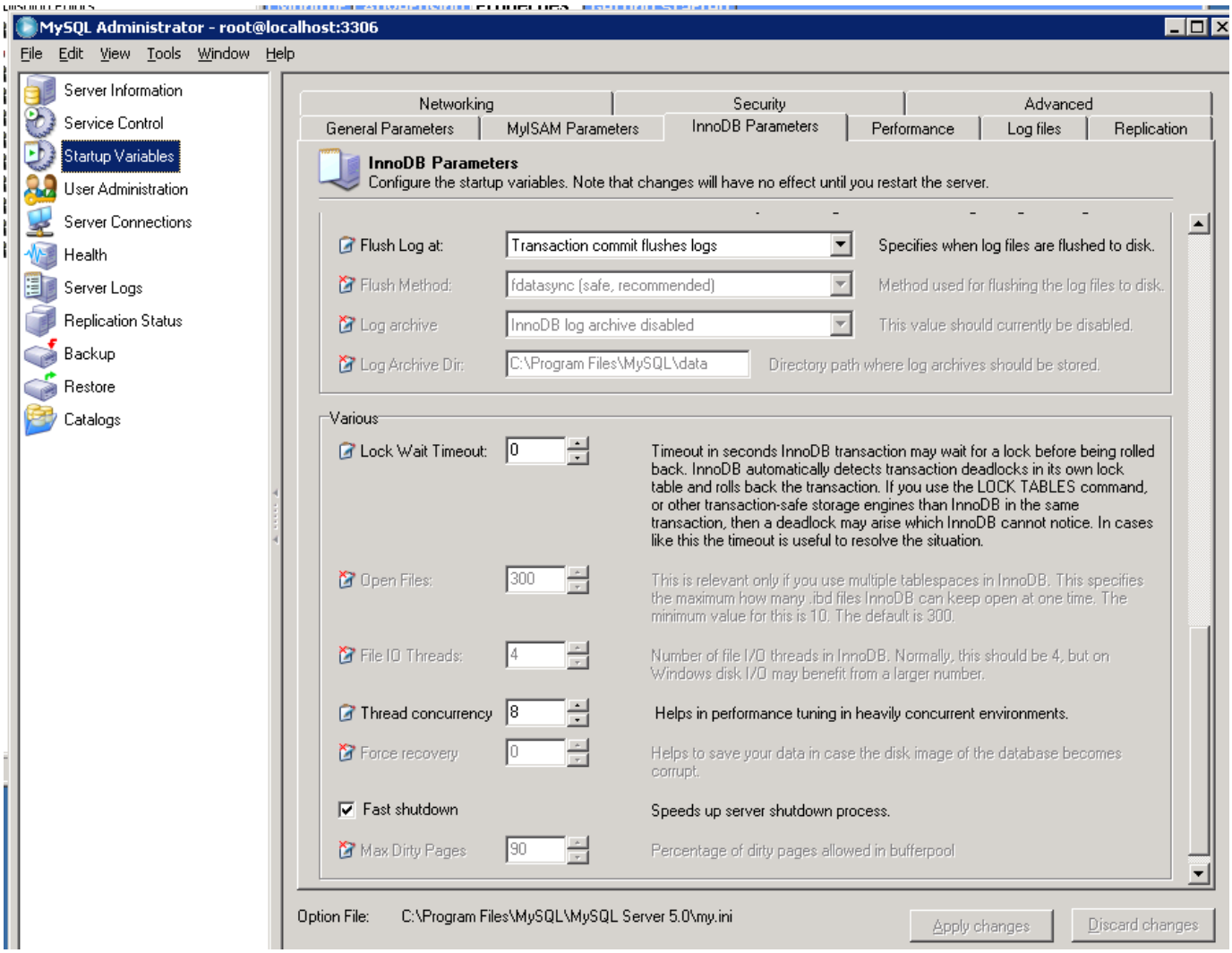
CET will automatically create a string A$ to contain the decimal number, and convert the integer value to the decimal variable A.

CET BASIC also support floating-point variables such as A! and double precision variables such as B!!. These will be converted as necessary to fit the MySQL datatypes.

8. Record and File Locking

Record and file locking for MySQL tables work exactly as they do for indexed files, except locking is performed on a row basis. The ON ERROR and ON LOCK statements may be used to specify a lock handler. The B_OPTLOCK and B_THLOCK environment variables perform the same function that they do with indexed files.

The only difference in locking is that the MySQL InnoDB locking method is slower to timeout than our C-ISAM indexed files. One way to make the return to your ON ERROR trap faster is to set the MySQL InnoDB Lock Wait Timeout to 1 or 0. This can be done from the MySQL Administrator as shown below:



9. Manually Converting CET Indexed Files

CET indexed files can be converted easily to MySQL tables if they have a regular structure. For example, consider the following program:

```
open #1:"/opt/data/testsql", input indexed
open #2:"/opt/data/testsql", update indexed, SQL
while 1
        read  #1,k$: decval, stgval$, intval%
        if EOF(1) THEN QUIT
        write #2,k$: decval, stgval$, intval%
wend
```

This program reads in the indexed file /opt/data/testsql and writes to the table “matsql” that is defined in the map file discussed in an earlier section.

10. Caveats

The following list of restrictions are currently necessary:

a. The key field of any table must be CHAR or VARCHAR and can be no longer than 255 bytes
b. You may have to decide which precision to use with DECIMAL MySQL data types to properly preserve the decimal precision for your data.