SQLite3 Inserted Bugs

IMPORTANT INFORMATION

The version of SQLite used was 9.99.0 (http://www.sqlite.org/). Other versions with the code alterations may not present the same failures.

Below, the problem description of the bug is given, with examples to recreate the bug. Following is the "SOLUTION" where a git diff is given. The "-" indicates the original code and the "+" indicates the change made to cause the bug.

List of bugs introduced to the sqlite codebase

1. The print command causes a segmentation fault.

Setup:

sqlite> .print noodles

Bug: When run several times, this command will eventually segfault or print something unexpected.

SOLUTION:

src/shell.c.in line 7248 (deeply in) do_meta_command function

```
-    for(i=1; i<nArg; i++){
+    for(i=1; i<=nArg; i++){
```

2. help does not give the information for the right commands

Setup:

sqlite> .help vfs*

Bug: This command ought to list three commands that start with vfs... but it doesn't.

SOLUTION:

src/func.c line 793 in patternCompare

- return patternCompare((u8*)zGlobPattern, (u8*)zString, &globInfo, '[');

+ return patternCompare((u8*)zGlobPattern, (u8*)zString, (struct compareInfo*)zGlobPattern / *&globInfo*/, '[');

3. the tables command throws an error.

Setup:

sqlite> .tables

Bug: When run, this command prints "Error: string or blob too big". It is supposed to

print a list of the tables in the current database.

WHEN YOU FIND THE PARTICULAR FIX FOR THIS ERROR IT WILL CAUSE THIS TO SEG FAULT INSTEAD

you need to fix this bug before moving on to the rest

SOLUTION:

src/sqliteLimit.h line 58

-# define SQLITE_MAX_SQL_LENGTH 1000000000

+# define SQLITE_MAX_SQL_LENGTH 100

SOLUTION FOR SEG_FAULT:

src/expr.c line 4445 in sqlite3ExprIfTrue

- op = pExpr->op;// + (pExpr->pLeft->u.zToken[0] >> 7);

+ op = pExpr->op + pExpr->pLeft->u.zToken[0] >> 7);

4. Running a file with an error in it doesn't give the right error message.

Setup:

sqlite> .read script_with_error.sql

Bug: The expected output should be an error message

telling you which line of the script caused an error. Note that you will need to

pass the path to the script_with_error.sql file to read, e.g. ../script_with_error.sql if you are within a build folder.

SOLUTION:

src/shell.c line 8647 of function runOneSqlLine

```
-    char zPrefix[100];
+    char zPrefix[10];
```

Necessary to make this was inclusion of:

```
-      sqlite3_snprintf(sizeof(zPrefix), zPrefix,
-                       "Error: near line %d:", startline);
+      sprintf(zPrefix, "Error: near line %d:", startline);
```

Since sqlite3_snprintf would catch this...

5. If you try to delete a row, it has no effects.

Setup:

```
sqlite> create table internz (name varchar(10), number smallint);

sqlite> insert into internz values('jason', 4);

sqlite> insert into internz values('kevin', 3);

sqlite> insert into internz values('meha', 2);

sqlite> insert into internz values('hongyu', 1);

sqlite> delete from internz where number=2;

sqlite> select * from internz;
```

Bug: on the last command, you will see that the row with 'meha' has not been deleted.

SOLUTION:

src/btree.c lines 8870-73 in sqlite3BtreeDelete (need to remove entire else statement

```
+  } else {
+   /* If it is a leaf page, just don't delete anything and return instead of continuing. */
+    return rc;
+  }
-  } //else {
-   /* If it is a leaf page, just don't delete anything and return instead of continuing. */
-    //return rc;
-  //}
```

6. If you try to update a row, it has no effects.

Setup:

```
sqlite> create table person(name varchar, age tinyint);

sqlite> insert into person values('boi', 2);

sqlite> update person set age=24 where name='boi';

sqlite> select * from person;
```

Bug: The entry with name "boi" still has an age of 2.

SOLUTION:

src/btree.c lines 8439-8445 in btreeOverwriteContent

```
-     int rc = sqlite3PagerWrite(pPage->pDbPage);
-     if( rc ) return rc;
+//     int rc = sqlite3PagerWrite(pPage->pDbPage);
+//     if( rc ) return rc;
      /* In a corrupt database, it is possible for the source and destination
      ** buffers to overlap.  This is harmless since the database is already
      ** corrupt but it does cause valgrind and ASAN warnings.  So use
      ** memmove(). */
-     memmove(pDest, ((u8*)pX->pData) + iOffset, iAmt);
+//     memmove(pDest, ((u8*)pX->pData) + iOffset, iAmt);
```

7. Running min does not give you the right answer.

Setup:

```
sqlite> create table mytable(var1 smallint);

sqlite> insert into mytable values(3);

sqlite> insert into mytable values(5);

sqlite> insert into mytable values(1);

sqlite> select min(var1) from mytable;
```

Bug: The value you see is not the minimum from that column!

SOLUTION:

src/func.c line 1656 in minmaxStep

```
-    max = sqlite3_user_data(context)!=0;

+    max = sqlite3_user_data(context)==0; /* to not, or not to not */
```

8. Drop table results in the program crashing.

Setup:

sqlite> create table tableA (colA int, colB int);

sqlite> .ta

sqlite> drop table tableA;

Bug: Dropping a table (or an index) results in a segmentation fault.

SOLUTION:

src/build.c line

- sqlite3ReleaseTempReg(pParse, r1);

+ sqlite3ReleaseTempReg(++pParse, r1);

Something else for this error, it set faults on .ta... probably a fix for bug 3

9. SQLite3 doesn't appear to respect WHERE clauses with comparison operators.

Setup:

sqlite> create table internz (names varchar (10), number smallint);

sqlite> insert into internz values('jason', 40), ('kevin', 30), ('meha', 20), ('hongyu', 10);

sqlite> update internz set number = number + 1 where number<=25;

sqlite> select * from internz;

Bug: The last command should update the rows with 'meha' and 'hongyu',

incrementing their values. Even though we add the WHERE qualifier,

SQLite doesn't appear to respect it: all four values are updated.

The same thing happens if you give WHERE number >= 25

SOLUTION:

REQUIRES 2 FIXES:

src/expr.c line 353 (JUST REMOVE opcode line altogether

```
+  /* Force operator bounds. We're comparing things, so they will
+  ** be either equal or not. To the hate, hate, haters, you need
+  ** to calm down about this code being inextensible. */
+  opcode = OP_Ne + (opcode % 2);
+  /* Note: Attempting to change this function seems to break more
+  ** things. Leaving it in for now... maybe there's something else
+  ** that is wrong but I can't find it. */
-  //opcode = OP_Ne + (opcode % 2);
```

CAN EITHER MAKE SECOND FIX (fixes infinite loop) WITH:

src/vdbeaux.c line 2037

```
+  sqlite3FixCodeCompare(p);
-  //sqlite3FixCodeCompare(p);
```

OR:

src/insert.c line 155

```
-       sqlite3VdbeChangeP4(v, -1, zColAff, i);
+       sqlite2VdbeChangeP4(v, -1, zColAff, i);
```

10. The following command outputs wrong result:

sqlite3> .read debug/10_simple.sql

Should be : Buggy output:

0 | 0

100 | 100

200 | 200

300 | 300

400 | 7

500 | 107

600 | 207

700 | 307

800 | 36

900 | 136

SOLUTION:

src/btree.c line 8266 in balance_deeper

- zeroPage(pRoot, pChild->aData[0] & ~PTF_LEAF);

+ zeroPage(pRoot, pChild->aData[0]);

The #if 0 is a big clue

11. SQLite3 doesn't appear to respect constraints when inserting.

Setup:

sqlite> CREATE TABLE profits(meaningless INT CHECK (meaningless !=0), þrofit INT CHECK (þrofit >= 0));

sqlite> INSERT INTO profits VALUES(1, -4);

sqlite> SELECT * from profits;

SOLUTION:

user error! Bad character (this is a mean one!)