# 插入操作时间性能测试（优化）

## 1、环境

172.18.8.36

SQLite V3.7.13

## 2、方法

使用bind()、reset()优化方式，循环插入1000000条记录。

（1）磁盘文件模式，测试10次

（2）内存驻留模式，测试20次

## 3、参数

插入表格与对应数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试表格** | | | | |
| **字段** | ID(int) | NAME(text) | GOAL(double) | NOTE(text) |
| **1** | 1 | Myname | 2.3 | Hello, nice to see you! |
| **……** | …… | …… | …… | …… |
| **1000000** | 1000000 | Myname | 2.3 | Hello, nice to see you! |

insert into inserttest values(?, ?, ?, ?);

## 4、测试代码

### 4.1文件模式

/home/liangxx/learning/src/study\_sqlite/time\_test/insert\_test/insert\_test.cpp

//multi\_operation.cpp

#include <stdio.h>

#include <string>

#include <sqlite3.h>

#include <sys/time.h>

#define TESTTIME 1000000

int main()

{

sqlite3 \* db;

sqlite3\_stmt \* stmt;

char \* zErrMsg = 0;

int rc;

register int i;

char \* sql\_insert = "insert into inserttest values(?, ?, ?, ?);";

const char \* insertnote = "Hello, nice to see you!";

const char \* insertname = "myname";

struct timeval begin;

struct timeval end;

long int waste;

rc = sqlite3\_open("./mydb.db", &db);

if (rc) {

fprintf(stderr, "Can't open database: %s\n", sqlite3\_errmsg(db));

sqlite3\_close(db);

return 1;

}

rc = sqlite3\_prepare(db, sql\_insert, strlen(sql\_insert), &stmt, 0);

if (rc) {

fprintf(stderr, "Can't open statement: %s\n", sqlite3\_errmsg(db));

sqlite3\_close(db);

return 1;

}

**//---------------------------------------------------------------------**

**gettimeofday(&begin, NULL);**

**for (i = 1; i <= TESTTIME; i ++)**

**{**

**//bind**

**sqlite3\_bind\_int(stmt, 1, i);**

**sqlite3\_bind\_text(stmt, 2, insertname, strlen(insertname), SQLITE\_TRANSIENT);**

**sqlite3\_bind\_double(stmt, 3, 2.3);**

**sqlite3\_bind\_text(stmt, 4, insertnote, strlen(insertnote), SQLITE\_TRANSIENT);**

**while (sqlite3\_step(stmt) == SQLITE\_ROW);**

**//reset**

**sqlite3\_reset(stmt);**

**// printf("Inserted: %8d\n", i);**

**}**

**gettimeofday(&end, NULL);**

**//---------------------------------------------------------------------**

sqlite3\_finalize(stmt);

sqlite3\_close(db);

waste = end.tv\_sec - begin.tv\_sec;

printf("The Waste Of Time Is: %d s\n", waste);

return 0;

}

/\* vim: set tabstop=4 shiftwidth=4 expandtab: \*/

### 4.2内存模式

/home/liangxx/learning/src/study\_sqlite/time\_test/insert\_test/insert\_test\_mem.cpp

//insert\_test\_mem\_cpp

#include <stdio.h>

#include <string>

#include <sqlite3.h>

#include <sys/time.h>

#define TESTTIME 1000000

int main()

{

//db obj

sqlite3 \*db;

//stmt obj

sqlite3\_stmt \*stmt;

char \* sql\_insert = "insert into inserttest values(?, ?, ?, ?);";

char \* sql\_create = "create table inserttest(ID integer primary key, \

NAME text, GOAL real, NOTE text);";

char \* sql\_query = "select \* from inserttest where ID = 251314;";

const char \*insertnote = "Hello, nice to see you!";

const char \*insertname = "myname";

register int i;

int id;

unsigned char \* name;

double goal;

unsigned char \* note;

//time

struct timeval begin;

struct timeval end;

long int waste;

int rc;

rc = sqlite3\_open(":memory:", &db);

if (rc) {

fprintf(stderr, "Can't open database: %s\n", sqlite3\_errmsg(db));

sqlite3\_close(db);

return 1;

}

//create table

sqlite3\_prepare(db, sql\_create, strlen(sql\_create), &stmt, 0);

while (sqlite3\_step(stmt) == SQLITE\_ROW);

sqlite3\_finalize(stmt);

//insert

rc = sqlite3\_prepare(db, sql\_insert, strlen(sql\_insert), &stmt, 0);

if (rc) {

fprintf(stderr, "Can't open statement: %s\n", sqlite3\_errmsg(db));

sqlite3\_close(db);

return 1;

}

**//---------------------------------------------------------------------**

**gettimeofday(&begin, NULL);**

**for (i = 1; i <= TESTTIME; i ++)**

**{**

**//bind**

**sqlite3\_bind\_int(stmt, 1, i);**

**sqlite3\_bind\_text(stmt, 2, insertname, strlen(insertname), SQLITE\_TRANSIENT);**

**sqlite3\_bind\_double(stmt, 3, 2.3);**

**sqlite3\_bind\_text(stmt, 4, insertnote, strlen(insertnote), SQLITE\_TRANSIENT);**

**while (sqlite3\_step(stmt) == SQLITE\_ROW);**

**//reset**

**sqlite3\_reset(stmt);**

**// printf("Inserted: %8d\n", i);**

**}**

**gettimeofday(&end, NULL);**

**sqlite3\_finalize(stmt);**

**//---------------------------------------------------------------------**

//query

sqlite3\_prepare(db, sql\_query, strlen(sql\_query), &stmt, 0);

while (sqlite3\_step(stmt) == SQLITE\_ROW)

{

id = sqlite3\_column\_int(stmt, 0);

name = (unsigned char \*)sqlite3\_column\_text(stmt, 1);

goal = sqlite3\_column\_double(stmt, 2);

note = (unsigned char \*)sqlite3\_column\_text(stmt, 3);

printf("%d %s %f %s\n", id, name, goal, note);

}

sqlite3\_finalize(stmt);

sqlite3\_close(db);

waste = (end.tv\_sec - begin.tv\_sec) \* 1000000 + (end.tv\_usec - begin.tv\_usec);

printf("The Waste Of Time Is: %ld us\n", waste);

return 0;

}

/\* vim: set tabstop=4 shiftwidth=4 expandtab: \*/

## 5、测试结果

### 4.1文件模式（单位s）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3568 | 5098 | 4673 | 5823 | 5598 |
| 5950 | 3258 | 3330 | 5700 | 3498 |
| 3767 | 3158 | 4316 | 5612 | 5030 |
| 3352 | 5983 | 3723 | 3565 | 3783 |

平均：4439.25s

### 4.2内存模式（单位us）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7717460 | 7762232 | 7761384 | 6943695 | 7036881 |
| 9870924 | 8312949 | 7444938 | 8152426 | 7333887 |
| 7448746 | 8908814 | 8224034 | 7764799 | 7142657 |
| 7135006 | 7386375 | 7755169 | 7319889 | 6731809 |

平均值：7707703.7us

## 6、结论

文件模式：插入单条记录4439.25us

内存模式：插入单条记录7.71us

内存模式的插入速度是文件模式的576倍。

# 查询操作时间性能测试（优化）

## 1、环境

172.18.8.36

SQLite V3.7.13

## 2、方法

使用bind()、reset()优化方式循环查询1000条记录，各测试20次。

（1）磁盘文件模式

（2）内存驻留模式

## 3、参数

查询：1000条数据

查询表格与对应数据

查询条件：ID%1000==0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试表格** | | | | |
| **字段** | ID(int) | NAME(text) | GOAL(double) | NOTE(text) |
| **1** | 1 | Myname | 2.3 | Hello, nice to see you! |
| **……** | …… | …… | …… | …… |
| **1000000** | 1000000 | Myname | 2.3 | Hello, nice to see you! |

select NOTE from inserttest where ID = ?;

## 4、测试代码

### 4.1文件模式

/home/liangxx/learning/src/study\_sqlite/time\_test/query\_test/query\_test.cpp

//query\_test\_cpp

#include <stdio.h>

#include <sqlite3.h>

#include <string>

#include <sys/time.h>

int main()

{

sqlite3 \* db;

sqlite3\_stmt \* stmt;

char \* zErrMsg = 0;

int rc;

char \* sql = "select NOTE from inserttest where ID = ?;";

int id;

unsigned char \* name;

double goal;

unsigned char \* note;

int queryline = 1000000;

int i;

struct timeval begin;

struct timeval end;

long int waste;

rc = sqlite3\_open("./mydb.db", &db);

if (rc)

{

fprintf(stderr, "Can't open database: %s\n", sqlite3\_errmsg(db));

sqlite3\_close(db);

return 1;

}

rc = sqlite3\_prepare(db, sql, strlen(sql), &stmt, 0);

if (rc)

{

fprintf(stderr, "Can't open statement: %s\n", sqlite3\_errmsg(db));

sqlite3\_close(db);

return 1;

}

**//---------------------------------------------------------------------**

**gettimeofday(&begin, NULL);**

**for (i = 1; i <= queryline; i ++)**

**{**

**if (i % 1000 == 0)**

**{**

**sqlite3\_bind\_int(stmt, 1, i);**

**while (sqlite3\_step(stmt) == SQLITE\_ROW)**

**{**

**// id = sqlite3\_column\_int(stmt, 0);**

**// name = (unsigned char \*)sqlite3\_column\_text(stmt, 1);**

**// goal = sqlite3\_column\_double(stmt, 2);**

**note = (unsigned char \*)sqlite3\_column\_text(stmt, 0);**

**}**

**sqlite3\_reset(stmt);**

**}**

**else {}**

**}**

**gettimeofday(&end, NULL);**

**waste = (end.tv\_sec - begin.tv\_sec) \* 1000000 + (end.tv\_usec - begin.tv\_usec);**

**printf("the time is :%d us\n", waste);**

**//---------------------------------------------------------------------**

sqlite3\_finalize(stmt);

sqlite3\_close(db);

return 0;

}

/\* vim: set tabstop=4 shiftwidth=4 expandtab: \*/

### 4.2内存模式

/home/liangxx/learning/src/study\_sqlite/time\_test/query\_test/query\_test\_mem.cpp

//query\_test\_mem\_cpp

#include <stdio.h>

#include <string>

#include <sqlite3.h>

#include <sys/time.h>

#define TESTTIME 1000000

int main()

{

//db obj

sqlite3 \*db;

//stmt obj

sqlite3\_stmt \*stmt;

char \* sql\_insert = "insert into inserttest values(?, ?, ?, ?);";

char \* sql\_create = "create table inserttest(ID integer primary key, NAME text, GOAL real, NOTE text);";

char \* sql\_query = "select NOTE from inserttest where ID = ?;";

const char \* insertnote = "Hello, nice to see you!";

const char \* insertname = "myname";

register int i;

unsigned char \* note;

//time

struct timeval begin;

struct timeval end;

long int waste;

int rc;

rc = sqlite3\_open(":memory:", &db);

if (rc)

{

fprintf(stderr, "Can't open database: %s\n", sqlite3\_errmsg(db));

sqlite3\_close(db);

return 1;

}

//create table

sqlite3\_prepare(db, sql\_create, strlen(sql\_create), &stmt, 0);

while (sqlite3\_step(stmt) == SQLITE\_ROW);

//insert data

sqlite3\_prepare(db, sql\_insert, strlen(sql\_insert), &stmt, 0);

for (i = 1; i <= TESTTIME; i ++)

{

//bind

sqlite3\_bind\_int(stmt, 1, i);

sqlite3\_bind\_text(stmt, 2, insertname, strlen(insertname), SQLITE\_TRANSIENT);

sqlite3\_bind\_double(stmt, 3, 2.3);

sqlite3\_bind\_text(stmt, 4, insertnote, strlen(insertnote), SQLITE\_TRANSIENT);

while (sqlite3\_step(stmt) == SQLITE\_ROW);

//reset

sqlite3\_reset(stmt);

}

**//---------------------------------------------------------------------**

**sqlite3\_prepare(db, sql\_query, strlen(sql\_query), &stmt, 0);**

**register int count = 21;**

**while (-- count)**

**{**

**gettimeofday(&begin, NULL);**

**for (i = 1; i <= TESTTIME; i ++)**

**{**

**if (i % 1000 == 0) {**

**//bind**

**sqlite3\_bind\_int(stmt, 1, i);**

**while (sqlite3\_step(stmt) == SQLITE\_ROW)**

**{**

**note = (unsigned char \*)sqlite3\_column\_text(stmt, 0);**

**}**

**//reset**

**sqlite3\_reset(stmt);**

**}**

**}**

**gettimeofday(&end, NULL);**

**waste = (end.tv\_sec - begin.tv\_sec) \* 1000000 + (end.tv\_usec - begin.tv\_usec);**

**printf("The Waste Of Time Is:%2d ==== %d us\n", (20 - count), waste);**

**}**

sqlite3\_finalize(stmt);

sqlite3\_close(db);

return 0;

}

/\* vim: set tabstop=4 shiftwidth=4 expandtab: \*/

## 5、测试结果

### 5.1文件模式（单位us）

48100 | 47604 | 30044 | 34647 | 35185 |

35820 | 41492 | 45622 | 41425 | 37621 |

41992 | 34869 | 47555 | 38538 | 54993 |

36499 | 46013 | 35592 | 46333 | 38144 |

平均值：40904.4us

### 5.2内存模式（单位us）

6655 | 6490 | 6401 | 6294 | 6341 |

6330 | 6284 | 6327 | 6040 | 6177 |

6564 | 6461 | 6380 | 6435 | 6351 |

6376 | 6172 | 6407 | 6386 | 6462 |

平均值：6366.65us

## 6、结论

文件模式：查询单条记录40.9044us

内存模式：查询单条记录6.36665us

内存模式速度是文件模式的6.42倍。

# 删除操作时间性能测试（优化）

## 1、环境

172.18.8.36

SQLite V3.7.13

## 2、方法

使用bind()、reset()优化方式循环删除1000000条记录。

（1）磁盘文件模式：测试1次

（2）内存驻留模式：测试1次

## 3、参数

删除表格与对应数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试表格** | | | | |
| **字段** | ID(int) | NAME(text) | GOAL(double) | NOTE(text) |
| **1** | 1 | Myname | 2.3 | Hello, nice to see you! |
| **……** | …… | …… | …… | …… |
| **1000000** | 1000000 | Myname | 2.3 | Hello, nice to see you! |

delete from inserttest where ID = ?;

## 4、测试代码

### 4.1文件模式

/home/liangxx/learning/src/study\_sqlite/time\_test/delete\_test/delete\_test.cpp

//delete\_test.cpp

#include <stdio.h>

#include <string>

#include <sqlite3.h>

#include <sys/time.h>

#include <boost/progress.hpp>

int main()

{

sqlite3 \* db;

sqlite3\_stmt \* stmt;

boost::progress\_display d(100);

char \* zErrMsg = 0;

int rc;

char \* sql\_create = "create table inserttest(ID integer primary key, NAME text, GOAL real, NOTE text);";

char \* sql\_insert = "insert into inserttest values(?, ?, ?, ?);";

char \* sql\_delete = "delete from inserttest where ID = ?;";

const char \*insertnote = "Hello, nice to see you!";

const char \*insertname = "myname";

int id;

unsigned char \*name;

int goal;

int insertline = 1000000;

int i;

struct timeval begin;

struct timeval end;

long int waste\_insert;

long int waste\_delete;

rc = sqlite3\_open("./mydb.db", &db);

if (rc) {

fprintf(stderr, "Can't open database: %s\n", sqlite3\_errmsg(db));

sqlite3\_close(db);

return 1;

}

//insert

sqlite3\_prepare(db, sql\_insert, strlen(sql\_insert), &stmt, 0);

gettimeofday(&begin, NULL);

for (i = 1; i <= insertline; i ++)

{

if (i % 20000 == 0)

++ d;

sqlite3\_bind\_int(stmt, 1, i);

sqlite3\_bind\_text(stmt, 2, insertname, strlen(insertname), SQLITE\_TRANSIENT);

sqlite3\_bind\_double(stmt, 3, 2.3);

sqlite3\_bind\_text(stmt, 4, insertnote, strlen(insertnote), SQLITE\_TRANSIENT);

sqlite3\_step(stmt);

sqlite3\_reset(stmt);

}

gettimeofday(&end, NULL);

waste\_insert = (end.tv\_sec - begin.tv\_sec) \* 1000 + (end.tv\_usec - begin.tv\_usec) / 1000;

sqlite3\_finalize(stmt);

//delete

sqlite3\_prepare(db, sql\_delete, strlen(sql\_delete), &stmt, 0);

**//---------------------------------------------------------------------**

**gettimeofday(&begin, NULL);**

**for (i = 1; i <= insertline; i ++)**

**{**

**if (i % 20000 == 0)**

**++ d;**

**sqlite3\_bind\_int(stmt, 1, i);**

**sqlite3\_step(stmt);**

**sqlite3\_reset(stmt);**

**}**

**gettimeofday(&end, NULL);**

**waste\_delete = (end.tv\_sec - begin.tv\_sec) \* 1000 + (end.tv\_usec - begin.tv\_usec) / 1000;**

**//---------------------------------------------------------------------**

printf("the time insert is :%ld ms\n", waste\_insert);

printf("the time delete is :%ld ms\n", waste\_delete);

sqlite3\_finalize(stmt);

sqlite3\_close(db);

return 0;

}

/\* vim: set tabstop=4 shiftwidth=4 expandtab: \*/

### 4.2内存模式

/home/liangxx/learning/src/study\_sqlite/time\_test/delete\_test/delete\_test\_mem.cpp

//delete\_test\_mem\_cpp

#include <stdio.h>

#include <string>

#include <sqlite3.h>

#include <sys/time.h>

#define TESTTIME 1000000

int main()

{

//db obj

sqlite3 \*db;

//stmt obj

sqlite3\_stmt \*stmt;

char \*sql = "insert into inserttest values(?, ?, ?, ?);";

char \*creat = "create table inserttest(ID integer primary key, NAME text, GOAL real, NOTE text);";

char \*delet = "delete from inserttest where ID = ?;";

const char \*insertnote = "Hello, nice to see you!";

const char \*insertname = "myname";

register int i;

//time

struct timeval begin;

struct timeval end;

long int waste;

int testtime = 20;

int rc;

rc = sqlite3\_open(":memory:", &db);

if (rc) {

fprintf(stderr, "Can't open database: %s\n", sqlite3\_errmsg(db));

sqlite3\_close(db);

return 1;

}

sqlite3\_prepare(db, creat, strlen(creat), &stmt, 0);

while (sqlite3\_step(stmt) == SQLITE\_ROW);

for (register int c = 0; c < testtime; c ++)

{

rc = sqlite3\_prepare(db, sql, strlen(sql), &stmt, 0);

//---------------------------------------------------------------------

gettimeofday(&begin, NULL);

for (i = 1; i <= TESTTIME; i ++)

{

//bind

sqlite3\_bind\_int(stmt, 1, i);

sqlite3\_bind\_text(stmt, 2, insertname, strlen(insertname), SQLITE\_TRANSIENT);

sqlite3\_bind\_double(stmt, 3, 2.3);

sqlite3\_bind\_text(stmt, 4, insertnote, strlen(insertnote), SQLITE\_TRANSIENT);

while (sqlite3\_step(stmt) == SQLITE\_ROW);

//reset

sqlite3\_reset(stmt);

}

gettimeofday(&end, NULL);

waste = (end.tv\_sec - begin.tv\_sec) \* 1000000 + (end.tv\_usec - begin.tv\_usec);

printf("The Waste Of Insert Time Is: %ld s\n", waste);

//---------------------------------------------------------------------

sqlite3\_prepare(db, delet, strlen(delet), &stmt, 0);

**gettimeofday(&begin, NULL);**

**for (i = 1; i <= TESTTIME; i ++)**

**{**

**//bind**

**sqlite3\_bind\_int(stmt, 1, i);**

**while (sqlite3\_step(stmt) == SQLITE\_ROW);**

**//reset**

**sqlite3\_reset(stmt);**

**}**

**gettimeofday(&end, NULL);**

**waste = (end.tv\_sec - begin.tv\_sec) \* 1000000 + (end.tv\_usec - begin.tv\_usec);**

**printf("The Waste Of Delete Time Is: %ld us\n", waste);**

}

sqlite3\_finalize(stmt);

sqlite3\_close(db);

return 0;

}

/\* vim: set tabstop=4 shiftwidth=4 expandtab: \*/

## 5、测试结果

### 5.1文件模式（单位ms）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5024727 | 6458030 | 4399107 | 7297804 | 4470264 |
| 5224969 | 4648675 | 5161535 | 6398206 | 7293629 |
| 7609287 | 7832012 | 5517289 | 7421823 | 6539289 |
| 6753921 | 5729162 | 4872191 | 6652191 | 5419283 |

平均值：6036169.70ms

### 5.2内存模式（单位us）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9209465 | 7987816 | 10269509 | 7364937 | 8293791 |
| 6834946 | 8479504 | 7715321 | 8915214 | 8904720 |
| 8847275 | 7048427 | 8602543 | 7650822 | 8779752 |
| 8613658 | 7966454 | 7515766 | 8167565 | 8286321 |

平均值：8272690.3us

## 6、结论

文件模式：删除单条记录6.04ms

内存模式：删除单条记录8.27us

内存模式的删除速度是文件模式的729倍。

# 递归操作时间性能测试（优化）

## 1、环境

172.18.8.36

SQLite V3.7.13

## 2、方法

使用bind()、reset()优化方式递归1000000条记录。

（1）磁盘文件模式：测试20次

（2）内存驻留模式：测试20次

## 3、参数

表格与对应数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试表格** | | | | | |
| **字段** | ID(int) | ParentID(int) | NAME(text) | GOAL(double) | NOTE(text) |
| **1** | 1 | 2 | Myname | 2.3 | Hello, nice to see you! |
| **2** | 2 | 3 | Myname | 2.3 | Hello, nice to see you! |
| **3** | 3 | 4 | Myname | 2.3 | Hello, nice to see you! |
| **……** | …… | …… | …… | …… | …… |
| **1000000** | 1000000 | 0 | Myname | 2.3 | Hello, nice to see you! |

select \* from inserttest where ID = ?;

## 4、测试代码

### 4.1文件模式

/home/liangxx/learning/src/study\_sqlite/time\_test/iter\_test/iter\_test.cpp

//iter\_test\_cpp

#include <stdio.h>

#include <sqlite3.h>

#include <string>

#include <sys/time.h>

#include <iostream>

#define TESTTIME 1000000

int main()

{

sqlite3 \*db;

sqlite3\_stmt \*stmt;

char \*zErrMsg = 0;

int rc;

char \*sql = "select \* from inserttest where ID = ?;";

int id;

unsigned char \* name;

double goal;

unsigned char \* note;

int i;

struct timeval begin;

struct timeval end;

long int waste;

//file mode

rc = sqlite3\_open("./mydb.db", &db);

if (rc)

{

fprintf(stderr, "Can't open database: %s\n", sqlite3\_errmsg(db));

sqlite3\_close(db);

return 1;

}

rc = sqlite3\_prepare(db, sql, strlen(sql), &stmt, 0);

if (rc)

{

fprintf(stderr, "Can't open statement: %s\n", sqlite3\_errmsg(db));

sqlite3\_close(db);

return 1;

}

**//---------------------------------------------------------------------**

**gettimeofday(&begin, NULL);**

**for (i = 1; i <= TESTTIME; i ++)**

**{**

**sqlite3\_bind\_int(stmt, 1, i);**

**while (sqlite3\_step(stmt) == SQLITE\_ROW)**

**{**

**id = sqlite3\_column\_int(stmt, 0);**

**name = (unsigned char \*)sqlite3\_column\_text(stmt, 1);**

**goal = sqlite3\_column\_double(stmt, 2);**

**note = (unsigned char \*)sqlite3\_column\_text(stmt, 3);**

**}**

**sqlite3\_reset(stmt);**

**}**

**gettimeofday(&end, NULL);**

**waste = (end.tv\_sec - begin.tv\_sec) \* 1000000 + (end.tv\_usec - begin.tv\_usec);**

**printf("The iter\_test time is :%d us\n", waste);**

**//---------------------------------------------------------------------**

sqlite3\_finalize(stmt);

sqlite3\_close(db);

return 0;

}

/\* vim: set tabstop=4 shiftwidth=4 expandtab: \*/

### 4.2内存模式

/home/liangxx/learning/src/study\_sqlite/time\_test/iter\_test/iter\_test\_mem.cpp

//iter\_test\_mem\_cpp

#include <stdio.h>

#include <string>

#include <sqlite3.h>

#include <sys/time.h>

#define TESTTIME 1000000

int main()

{

//db obj

sqlite3 \* db;

//stmt obj

sqlite3\_stmt \* stmt;

char \* sql\_insert = "insert into inserttest values(?, ?, ?, ?);";

char \* sql\_create = "create table inserttest(ID integer primary key, \

NAME text, GOAL real, NOTE text);";

char \* sql\_iter = "select \* from inserttest where ID = ?;";

const char \* insertnote = "Hello, nice to see you!";

const char \* insertname = "myname";

register int i;

int id;

unsigned char \* name;

double goal;

unsigned char \* note;

//time

struct timeval begin;

struct timeval end;

long int waste;

int rc;

rc = sqlite3\_open(":memory:", &db);

if (rc)

{

fprintf(stderr, "Can't open database: %s\n", sqlite3\_errmsg(db));

sqlite3\_close(db);

return 1;

}

sqlite3\_prepare(db, sql\_create, strlen(sql\_create), &stmt, 0);

while (sqlite3\_step(stmt) == SQLITE\_ROW);

sqlite3\_prepare(db, sql\_insert, strlen(sql\_insert), &stmt, 0);

gettimeofday(&begin, NULL);

for (i = 1; i <= TESTTIME; i ++)

{

//bind

sqlite3\_bind\_int(stmt, 1, i);

sqlite3\_bind\_text(stmt, 2, insertname, strlen(insertname), SQLITE\_TRANSIENT);

sqlite3\_bind\_double(stmt, 3, 2.3);

sqlite3\_bind\_text(stmt, 4, insertnote, strlen(insertnote), SQLITE\_TRANSIENT);

while (sqlite3\_step(stmt) == SQLITE\_ROW);

//reset

sqlite3\_reset(stmt);

}

gettimeofday(&end, NULL);

waste = (end.tv\_sec - begin.tv\_sec) \* 1000000 + (end.tv\_usec - begin.tv\_usec);

printf("The insert\_test\_mem time is :%d us\n", waste);

**//---------------------------------------------------------------------**

**sqlite3\_prepare(db, sql\_iter, strlen(sql\_iter), &stmt, 0);**

**gettimeofday(&begin, NULL);**

**for (i = 1; i <= TESTTIME; i ++)**

**{**

**//bind**

**sqlite3\_bind\_int(stmt, 1, i);**

**sqlite3\_bind\_text(stmt, 2, insertname, strlen(insertname), SQLITE\_TRANSIENT);**

**sqlite3\_bind\_double(stmt, 3, 2.3);**

**sqlite3\_bind\_text(stmt, 4, insertnote, strlen(insertnote), SQLITE\_TRANSIENT);**

**while (sqlite3\_step(stmt) == SQLITE\_ROW)**

**{**

**id = sqlite3\_column\_int(stmt, 0);**

**name = (unsigned char \*)sqlite3\_column\_text(stmt, 1);**

**goal = sqlite3\_column\_double(stmt, 2);**

**note = (unsigned char \*)sqlite3\_column\_text(stmt, 3);**

**}**

**//reset**

**sqlite3\_reset(stmt);**

**}**

**gettimeofday(&end, NULL);**

**waste = (end.tv\_sec - begin.tv\_sec) \* 1000000 + (end.tv\_usec - begin.tv\_usec);**

**printf("The iter\_test\_mem time is :%d us\n", waste);**

**//---------------------------------------------------------------------**

sqlite3\_finalize(stmt);

sqlite3\_close(db);

return 0;

}

/\* vim: set tabstop=4 shiftwidth=4 expandtab: \*/

## 5、测试结果

### 5.1文件模式（单位us）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 40923878 | 47129810 | 37323136 | 38193470 | 42917329 |
| 39237846 | 42123189 | 39743290 | 39164898 | 43770312 |
| 44238724 | 36217378 | 41873202 | 40172386 | 39234703 |
| 43237923 | 38293013 | 47484739 | 41837429 | 39812379 |

平均值：41146451.7us

### 5.2内存模式（单位us）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4939718 | 5672393 | 4132345 | 4983762 | 4823689 |
| 5073892 | 4322389 | 4345675 | 5623868 | 5623979 |
| 4427181 | 4565232 | 5204198 | 6672513 | 6728265 |
| 4576239 | 6634424 | 5612890 | 4234622 | 5354132 |

平均值：5177570.3us

## 6、结论

文件模式：递归1000000记录41.15s

内存模式：递归1000000记录5.18s

内存模式的递归速度是文件模式的7.94倍。

# \*内存数据库、临时数据库与文件数据库性能对比

/home/liangxx/learning/src/study\_sqlite/mode\_configure/templib/

1、插入性能对比

内存数据库

**100：**1349 us、1348 us、1169 us、1517 us、1330 us

**1000：4944 us、5364 us、6715 us、10098 us、8101 us**

**10000：**55739 us、54711 us、54557 us、53846 us、53271 us

**100000：**497602 us、393154 us、429767 us、448017 us、425517 us

临时数据库

**100：7646 us**、4924 us、3126 us、3997 us、3277 us

**1000：**21981 us、30012 us、30870 us、30016 us、26871 us、28155 us、34483 us、36659 us

**10000：**227592 us、156978 us、178348 us、137131 us、161119 us、182308 us、442197 us

**100000：**1445486 us、1610571 us、2159278 us、1325063 us、1460177 us、1970883 us、1950561 us

文件数据库

**100：**317115 us、271684 us、277371 us、260661 us、322307 us

**1000：**2805561 us、3775442 us、2833744 us、2779821 us、2910210 us

**10000：**29670339 us、31132331 us、28644974 us、30834055 us、31973082 us

**100000：**325791831 us、321693665 us、338502059 us、351378342 us、673694482 us、951382463 us、980610979 us、924399383 us