Министерство образования Московской области
Государственный университет «Дубна»

Филиал «Протвино»

# Материалы XV научно-практической конференции филиала «Протвино» Государственного университета «Дубна»,

посвященной 55-й годовщине космического полета Ю.А. Гагарина

(г. Протвино, 6 — 13 апреля 2016 г.)

Под общ. редакцией канд. техн. наук А.П. Леонова

Дубна
2016

УДК 62+3
ББК 94.3я431
М 34-1

**М 34-1** **Материалы XV научно-практической конференции филиала «Протвино» Государственного университета «Дубна»**, посвященной 55-й годовщине космического полета Ю.А. Гагарина (г. Протвино, 6—13 апреля 2016 г.) : сб. материалов / под общ. ред. к.т.н., с.н.с. А.П. Леонова. — Дубна : Гос. ун-т «Дубна», 2016. — 118 [1] с.

ISBN 978-5-89847-494-2

В сборнике представлены результаты научных исследований студентов и преподавателей филиала «Протвино» Государственного университета «Дубна» в областях автоматизации технологических процессов и производств, информационных технологий, экономики, гуманитарных и социальных наук.

УДК 62+3
ББК 94.3я431

ISBN 978-5-89847-494-2

*И.О. Ковцова, А.В. Морозевич*

# НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ БД *SQLITE*

*Филиал «Протвино» государственного университета «Дубна»*
*Кафедра информационных технологий*

*В статье рассматривается встраиваемая реляционная, кроссплатформенная база данных SQLite, описываются ее достоинства и недостатки. Приводятся основные функции библиотек sqlite3.h.*

Сегодня задача автоматизации является актуальной для любого типа производства. Автоматизированная система управления технологическими процессами — комплекс, предназначенный для управления различными процессами в рамках технологического процесса, производства, предприятия, в состав которого входит как персонал, так и аппаратные и программные средства [1]. Решения принимаются на основе собранной и обработанной информации, представленной в удобном виде. Информация собирается с первичных устройств — датчиков и счётчиков электрической энергии, терминалов релейной защиты и автоматики и т.п. Информация с этих устройств, как правило, поступает на специализированный контроллер — устройство сбора и передачи данных (УСПД), которое и обеспечивает хранение данных в течение заданного времени. При разработке таких контроллеров приходится решать задачу организации хранения данных. Одним из решений является использование встраиваемых баз данных (БД).

На текущем рынке встраиваемых БД представлено много продуктов от различных производителей, но только один из них поставляется с открытыми исходными кодами (на языке *C*), не требует лицензионных сборов и спроектирован исключительно как встраиваемая БД — это *SQLite*.

*SQLite* — встраиваемая реляционная, кроссплатформенная база данных, которая поддерживает достаточно полный набор команд *SQL (structured query language* — «язык структурированных запросов»*)*. Эта база данных не поддерживает все особенности *SQL* и уступает в функциональности другим развитым СУБД (системам управления базами данных), но вполне подходит для хранения и извлечения информации. *SQLite* широко используется разработчиками для организации локального хранения данных и эффективной работы с ними в разрабатываемых программных продуктах.

*SQLite* не использует парадигму «клиент-сервер» — а именно: движок *SQLite*, с которым взаимодействует прикладная программа, не выделяется в отдельный процесс. Вместо этого предоставляется библиотека, с которой программа компонуется и движок становится составной частью программы. Для работы *SQLite* используются вызовы функций (*API — application programming interface*) библиотеки *SQLite* (набор функций *sqlite3_XXX* на языке *C*). Заголовки этих функций находятся в библиотеке *sqlite3.h*.

Такой подход уменьшает накладные расходы, время отклика и упрощает программу. *SQLite* хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется. Требования *ACID* (*atomicity* — атомарность, *consistency* — согласованность, *isolation* — изолированность, *durability* — долговечность) достигаются, в том числе, за счёт создания файла журнала. Т. е. по своей сути, БД *SQLite* — обычный файл. Поэтому нет никакой необходимости в дополнительных средствах администрирования, требующих много времени на создание сложной структуры прав доступа для защиты пользовательских баз данных. Всё это уже автоматически поддерживается организацией прав доступа в самой файловой системе, это также подразумевает (в области ограничения пространства), что не требуется никаких специальных правил для отслеживания заполнения дискового пространства пользователем. Преимущество для пользователей состоит в том, что появляется возможность создать необходимое количество баз данных, в зависимости от задачи, плюс к этому — абсолютный контроль над ними.

Можно выделить следующие преимущества *SQLite*:

— файловая структура — вся база данных состоит из одного файла, поэтому её очень легко переносить на разные машины;

— используемые стандарты — *SQLite* использует *SQL*;

— простота использования при разработке и тестировании.

К недостаткам *SQLite* можно отнести:

— отсутствие системы пользователей (Более крупные СУБД включают в свой состав системы управления правами доступа пользователей. Обычно применения этой функции не так критично, так как эта СУБД используется в небольших приложениях. Это можно рассматривать и как преимущество);

— отсутствие возможности увеличения производительности (Опять же, исходя из проектирования, довольно сложно выжать что-то более производительное из этой СУБД).

На разных аппаратных платформах скорость работы с базой данных (транзакции *INSERT*, *UPDATE*, *DELETE*) может различаться, поэтому необходимо приложение, которое позволит оценить производительность БД в каждом конкретном случае. Целью данной работы является разработка приложения, осуществляющего нагрузочное тестирование базы данных *SQLite*. Первоначальный этап — это создание простого приложения, которое создает БД, делает вставку, выборку и изменение данных.

Приложение разрабатывается на языке *C++*, с использованием свободной интегрированной среды разработки — *KDevelop 4.7.1*.

Для работы с базой используются основные функциии *sqlite3_XXX*:

— *sqlite3_open* — для открытия соединения к базе;

— *sqlite3_exec* — для выполнения запроса *sql* (Интерфейс *sqlite3_exec* является оболочкой *sqlite3_prepare_v2() , sqlite3_step ()* , и *sqlite3_finalize ()* , что позволяет приложению запускать несколько заявлений *SQL* без необходимости использования большого количества кода *C*.);

— *sqlite3_close* — для закрытия базы.

В ходе заполнения таблиц изначально применялась функция *sqlite3_exec*, но в связи с длительностью процесса, было принято решение использовать функции выполнения запросов *sqlite3_prepare_v2() , sqlite3_step ()* , и *sqlite3_finalize ()*, что увеличило скорость заполнения таблицы примерно вдвое. На рис. 1 представлена заполненная данными таблица. На рис. 2 приведен код заполнения данной таблицы. В ходе работы необходимо создать несколько заполненных таблиц разного объема (10Мб, 100 Мб, 200Мб, 500Мб,1000Мб).

На основе полученных таблиц будет осуществляться тестирование.

Тест 1 представляет собой заполнение БД данными с фиксацией времени. После заполнения всех испытуемых БД необходимо построить результирующий график.

Тест 2 представляет собой выборку одиночных значений в БД (начало, середина, конец): во всех испытуемых БД после выполнения необходимо построить результирующий график.

Тест 3 представляет собой выборку 50 записей по двум критериям во всех испытуемых БД: после выполнения необходимо построить результирующий график.

| | IEDIndex | ParamIndex | SQN | Value | Status |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 7761360 | 38 | 71 |
| 2 | 1 | 2 | 7761360 | 68 | 71 |
| 3 | 1 | 3 | 7761360 | 77 | 71 |
| 4 | 1 | 4 | 7761360 | 51 | 71 |
| 5 | 1 | 5 | 7761360 | 39 | 71 |
| 6 | 1 | 6 | 7761360 | 53 | 71 |
| 7 | 1 | 7 | 7761360 | 68 | 71 |

Рис.1 — Код заполнения таблицы

```
mysql.openDB("test");
//connect(&db);

 query = "CREATE TABLE IF NOT EXISTS ProfileTableD (IEDIndex INT,ParamIndex
 "INT,SQN INT,Value INT,Status INT, PRIMARY KEY (IEDIndex,ParamIndex,SQN));";
  char *x = new char[query.length() + 1]; // or
           std::strcpy(x, query.c_str());
           mysql.request(x);
           delete[] x;

   IEDIndex = 10;
   ParamIndex = 15;
   SQN = 7890720;
   Status = 17;
   clock_gettime(CLOCK_REALTIME, &tstart);  //фиксируем время
  for(int m=1;m<48;m++)
  {
     strSQN = ToString(SQN);

     for(int i=1; i<IEDIndex; i++)
     {

       strIEDIndex = int_to_string(i);
         for(int j=1;j<ParamIndex;j++)
         {
           strParamIndex = ToString(j);//std::to_string(j);
           Value = rand() %100;
           strValue = int_to_string(Value);
           strStatus = int_to_string(Status);

           query = "INSERT OR REPLACE INTO ProfileTableD VALUES ('"; //начало
           query += strIEDIndex + "', '";
           query += strParamIndex + "', ";
           query += strSQN + ", ";
           query += strValue + ", ";
           query += strStatus + ");";

           char *y = new char[query.length() + 1]; // or
           std::strcpy(y, query.c_str());
           mysql.request(y);
           delete[] y;


         }
     }

     SQN=SQN+30;
  }

  clock_gettime(CLOCK_REALTIME, &tstop); //фиксируем время
  sec = tstop.tv_sec - tstart.tv_sec;
  printf("Затрачено времени в секундах: %d\n", sec);
```

Рис. 2 — Код заполнения таблицы

**Библиографический список**


1. Автоматизированная система управления // Википедия. [2015—2015]. Дата обновления: 16.11.2015. URL: http://ru.wikipedia.org/?oldid=74550149 (дата обращения: 16.11.2015).
2. *Ухова М.В.* Организация сбора, хранения и обработки информации на объектах электроэнергетики в соответствии со стандартом IEC 61850. / М.В. Ухова., А.В. Мандрик. // Автоматизация и современные технологии, 2015 г., № 4. — С. 7—11.
3. SQLite // Википедия. [2015—2015]. Дата обновления: 21.08.2015. URL: http://ru.wikipedia.org/?oldid=72856081 (дата обращения: 21.08.2015).