УДК 004.254 (045)
Мельник В.М. , Гаджемура А.М.
Луцький національний технічний університет

ПЕРЕВАГИ ВИКОРИСТАННЯ СУБД SQLITE У ДОДАТКАХ WINDOWS

Мельник В. М., Гаджемура А. М. Переваги використання СУБД SQLite у додатках Windows. В даній роботі розглядаються переваги системи управління базами даних SQLite для Windows-додатків. Заодно детально розглядаються і обґрунтовуються самі механізми в реалізації цих переваг. В роботі також пропонується приклад Windows-додатку із демонстрацією використання SQLite в інтеграції з засобами середовища розробки Qt Creator та мови програмування високого рівня C++. Описані також способи підключення баз даних та основні операції запису даних і зчитування їх з бази.
Ключові слова: SQLite-переваги, система управління базами даних, середовище розробки Qt Creator, мова високого рівня C++, Windows-додатки.

Мельник В. М., Гаджемура А. М. Преимущества использования СУБД SQLite в приложениях Windows. В данной работе рассматриваются преимущества системы управления базами данных SQLite для Windows-приложений. Совместно рассматриваются и обосновываются сами механизмы в реализации этих преимуществ. В работе также предлагается пример Windows-приложения с демонстрацией использования SQLite в интеграции со средствами среды разработки Qt Creator и языка программирования высокого уровня C++. Описанные также способы подключения баз данных и основные операции записи данных и считывания их с базы.
Ключевые слова: SQLite-преимущества, система управления базами данных, среда разработки Qt Creator, язык высокого уровня C++, Windows-приложения.

Melnyk V. M., Gadzhemura A. M. Advantages in DBCS SQLite using for Windows applications. This paper considers the advantages of the database management system SQLite for Windows applications. There are jointly considered and justified mechanisms in these advantages implementation. The paper also offers an example of a Windows application that demonstrates the use of SQLite in integration with the development environment Qt Creator and the high-level programming language C++. Also ways of databases connecting and the basic operations of data writing and reading them from the database are described.
Keywords: SQLite-advantages, database management system, development environment Qt Creator, high-level language C++, Windows applications.

Вступ

Великі об'єми інформації та необхідність ними керувати призвели до створення баз даних та систем управління ними (СУБД). На базі розробок з залученням баз даних розгортаються цілі клієнт-серверні системи. Однак у невеликих додатках, де обчислювальні ресурси досить обмежені *актуально* використовувати не такі вже і складні бази даних. СУБД SQLite являє собою полегшену реляційну систему управління базами даних, яка може бути втілена у вигляді бібліотеки, де реалізовано багато принципів і механізмів зі стандарту SQL-92 [1]. Серцевинний код SQLite поширюється як суспільне надбання, тобто може використовуватися без обмежень та безоплатно з будь-якою метою [2].

Дана СУБД дуже поширена для додатків у середовищах операційних систем Android, Windows, Linux та UNIX. Проте *проблематикою даного напрямку* у наш час є досить не значна кількість прикладів та описів для застосування у підручниках, посібниках чи іншій навчальній літературі. З особливістю це прослідковується у середовищі розробки Qt Creator, яке в Україні не має високої популярності.

Теоретичні відомості

Сама бібліотека SQLite написана мовою C, а її особливістю є те, що вона не використовує парадигму клієнт-сервер. Це говорить про те, що рушій SQLite не є окремим процесом, з яким взаємодіє додаток, а він тільки надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Даний підхід зменшує накладні витрати, час відгуку і значно спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується додаток. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, який зберігає базу даних, блокується. ACID-функції (Atomicity – атомарність, Consistency – узгодженість, Isolation – ізольованість, Durability – довговічність) досягаються зокрема за рахунок створення файла-журналу. Кілька процесів або потоків можуть одночасно з легкістю читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів на

запис у цей час не обслуговується, інакше спроба запису закінчується невдачею і в програму-додаток повертається код помилки [3].

Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу. У комплекті постачання йде також функціональна клієнтська частина у вигляді файла виконання sqlite3, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина працює з командного рядка, і дозволяє звертатися до файла бази даних на основі типових функцій операційної системи. Завдяки архітектурі рушія можливо використовувати SQlite як на вбудованих системах, так і на виділених машинах з гігабайтними масивами даних.

Метою роботи послужила також розробка нескладного додатку у створенні та заповненні бази даних із демонстрацією можливостей СУБД SQLite як наведеного зразка. Сама програма буде являти собою Organizer із попередньо заданим спливаючим повідомленням. Дослідження в даному напрямку потребують великих удосконалення та ефективних універсальних програм для роботи з СУБД.

Виклад основного матеріалу

SQLite – це реляційна база даних. Поняття реляційності пов'язане з розробками відомого англійського спеціаліста в області систем баз даних Едгара Кодда[4]. Ця модель характеризується простотою структури даних, зручним для користувача табличним представленням і можливістю використання формального апарату алгебри відношень і реляційного обчислення для обробки даних. Реляційна модель орієнтована на організацію у вигляді двовимірних таблиць. Кожна реляційна таблиця являє собою двовимірний масив і володіє наступними властивостями:

- кожен елемент таблиці являє собою один елемент даних;
- всі комірки в стовпці таблиці однорідні, тобто всі елементи в стовпці мають однаковий тип;
- кожен стовбець має унікальне ім'я;
- однакові рядки в таблиці повинні бути відсутні;
- порядок наступності рядків і стовбців може бути довільним.

Для того, щоб представити інформацію, що міститься в таблиці бази даних, у середовищі Qt Creator використовується кілька класів:

– QSqlQueryModel – це клас моделі, яка формує таблицю шляхом задавання сирого SQL-запиту. Вона може бути корисна при формуванні особливо витончених фільтрів і компіляції інформації з різних таблиць бази даних. Про неї докладніше можна отримати інформацію в.

– QSqlTableModel – це клас-основа нашого обговорення в даній статті. Дана модель формує таблицю за іменем тієї таблиці, яка існує в базі даних. Як недостаток можна відзначити відсутність методів підключення зв'язків з іншими таблицями для того, щоб підставляти значення в поля з інших таблиць через параметр ID.

– QSqlRelationalTableModel – це клас, який дозволяє формувати таблицю зі зв'язками з іншими таблицями, підміняючи значення таблиці, яку представляє дана модель, через ID-записи, що містяться в інших таблицях.

Функціональні можливості додатка

Модуль QtSql використовує плагіни драйверів для взаємодії з API для різних баз даних. Так як API SQL модуля не залежить від баз даних, то код, специфічний для певної бази даних, міститься в її драйверах. Деякі драйвери поставляються разом з Qt, а інші можуть бути додані певним шляхом. Перш, ніж отримати доступ до бази даних за допомогою класів QSqlQuery або QSqlQueryModel, ми повинні встановити з базою даних хоча б одне з'єднання. З'єднання з базою даних ідентифікуються за допомогою довільних рядків. QSqlDatabase також підтримує концепцію з'єднання за замовчуванням, яке використовується класом Qt SQL за замовчуванням, якщо ніяке інше з'єднання не вказано. Цей механізм дуже зручний для додатків, що використовують тільки одне з'єднання з базою даних [5].

Приклад коду, що встановлює з'єднання з базою даних SQL з використанням додаткових параметрів може бути представлений наступним чином:

```
CInfo::CInfo() :
   db(QSqlDatabase::addDatabase("QSQLITE"))
   {
      db.setHostName("localhost");
      db.setDatabaseName("myBase");
```

```
        db.setUserName("root");
        db.setPassword("pass");
        db.open();
        db.setDatabaseName("job.db");
        bIsOpened = db.open();
        if(bIsOpened)
        {
            updateDBVersion();
         }
    }
```

QSqlDatabase::addDatabase() – це ім'я драйвера. Для отримання списку драйверів, дивіться документацію addDatabase() [8]. Для ініціалізації даних з'єднання необхідно викликати ряд функцій setHostName(), setDatabaseName(), setUserName() і setPassword(). Для видалення з'єднання з базою даних спочатку слід закрити базу даних за допомогою власного метода QSqlDatabase::close (), а потім потрібно видалити її за допомогою статичного методу QSqlDatabase::removeDatabase()[6].

Для виконання запитів SQL необхідно створити об'єкт QSqlQuery і викликати метод QSqlQuery::exec(). QSqlQuery надає одноразовий доступ до результуючої бази вибіркою одного запиту. Після виклику exec() внутрішній покажчик QSqlQuery вказує на позицію перед першим записом. В цьому випадку необхідно викликати метод QSqlQuery::next() один раз, щоб перемістити покажчик на перший запис таблиці. Потім знову необхідно повторювати виклик метода next(), щоб отримувати доступ до інших записів до тих пір, поки він не поверне значення false[7].

Функція QSqlQuery::value() повертає значення поля поточного запису. Поля задаються індексами, починаючи з нуля. Функція QSqlQuery::value() повертає значення типу QVariant, який може зберігати значення різних типів C++ і ядра Qt, такі навіть як int, QString і QByteArray. Різні типи значень бази даних в Qt автоматично приводяться до найближчого еквівалента. Можна переміщатися вперед і назад по вибірці, використовуючи функції QSqlQuery::next(), QSqlQuery::previous(), QSqlQuery::first(), QSqlQuery::last() і QSqlQuery::seek(). Поточний номер рядка можна отримати за допомогою функції QSqlQuery::at(), а загальна кількість рядків у вибірці, якщо це підтримується базою даних, повертається функцією QSqlQuery::size().

У наведеному нижче прикладі вказувати з'єднання не потрібно, а тому використовується з'єднання за замовчуванням.

```
QSqlQuery query;
query.exec ("SELECT * FROM tabMain;");
while (query.next ())
{
   tableWidget-> insertRow (0);
   tableWidget-> setItem (0, 0, new QTableWidgetItem (query.value (0) .toString ()));
   tableWidget-> setItem (0, 1, new QTableWidgetItem (query.value (1) .toDateTime ().toString ()));
   tableWidget-> setItem (0, 2, new QTableWidgetItem (query.value (2) .toString ()));
   tableWidget-> setItem (0, 3, new QTableWidgetItem (query.value (3) .toString ()));
   tableWidget-> setRowHeight (0, 20);
}
```

Якщо виникає помилка, метод exec() повертає значення false. Доступ до помилки можна отримати за допомогою функції QSqlQuery::lastError().

Вставляючи запис в таблицю, тобто використовуючи INSERT, можна одночасно вставити багато записів. Але найчастіше ефективніше відокремити запит від реально вставлених значень. Це можна зробити за допомогою вставки значень через параметри[8]. У наступному прикладі показані вставки записів за допомогою проіменованого параметра:

```
QSqlQuery query;
query.prepare("INSERT INTO tabMain VALUES (null,: datetime,: string,: int);");
query.bindValue(": datetime", QDateTime::currentDateTime ());
query.bindValue(": string", lineEdit-> text ());
query.bindValue(": int", spinBox-> value ());
query.exec ();
```

Крім зручності виконання, вставка через параметри має ще й ту перевагу, що розробник позбавлений від необхідності піклуватися про перетворення спеціальних символів. Зміна записів дуже схожа на вставку в таблицю:

```
QSqlQuery query;
query.prepare("UPDATE tabMain SET String = ': string', Int =: int WHERE ID =: id;");
query.bindValue(": string", lineEdit-> text ());
query.bindValue(": int", spinBox-> value ());
query.bindValue(": id", iID);
query.exec ();
```

Також наведемо приклад виразу REMOVE:

```
QSqlQuery query;
query.prepare ( "DELETE FROM tabMain WHERE ID =: id;");
query.bindValue(": id", tableWidget-> item (tableWidget-> currentIndex (). row (), 0) -> text());
query.exec ();
```

Повна функція зчитування усіх даних з бази виглядатиме так:

```
bool CInfo::getLastRowData(int &id, QDateTime &dt)
{
   QString query("SELECT max(id) FROM jobs;");
   QSqlQuery q = db.exec(query);
   bool result = q.lastError().type() == QSqlError::NoError;
   if(result)
   {
      id = q.record().value(0).toInt();
      query = "SELECT date_created FROM jobs WHERE id = " + QString::number(id) + ";";
      q = db.exec(query);
      result = q.lastError().type() == QSqlError::NoError;
      if(result)
      {
         dt = QDateTime::fromTime_t(q.record().value(0).toUInt());
      }
   }
   return result;
}
```

Особливості функціонування додатку

Результат роботи додатка Organizer демонструє єдине значення (рис. 1). При натисненні кнопки "Edit" можна побачити усю інформацію про запис з можливістю її редагування. У даному випадку це дані рядкового типу та типу дата/час, які при необхідності можна змінювати. Усі наведені вище програмні коди знаходяться у файлі-ресурсі mainwindow.cpp

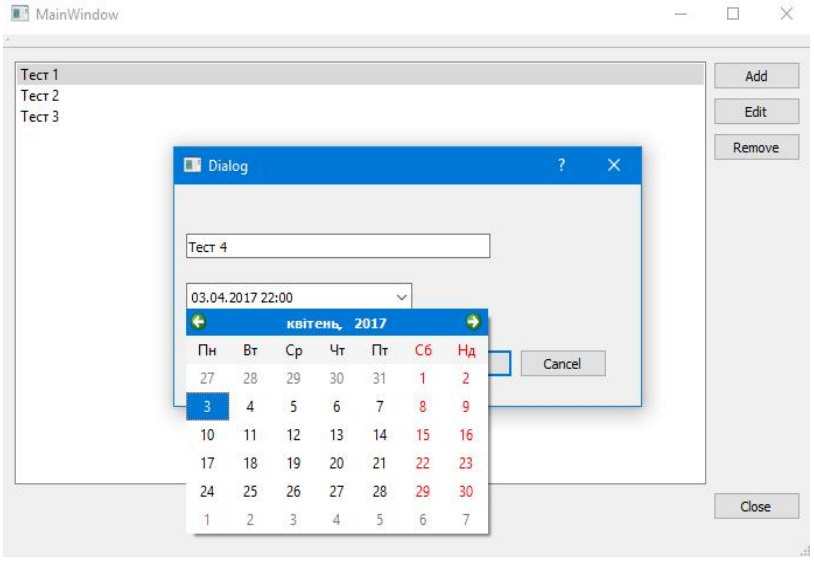

Рис. 1 – Результат роботи додатка

У диспетчері задач можна переглянути ресурсозатратність для операційної системи Windows 10, яка для прикладу наведена на рисунку 2.

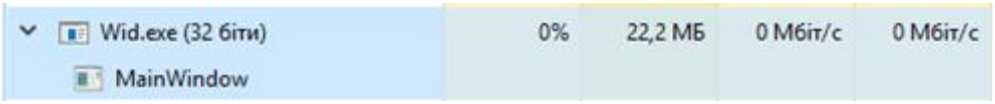

Рис. 2 – Використання ресурсів комп'ютера розробленим додатком

Споріднені роботи

Застосування SQLite разом із середовищем Qt Cretor охоплює досить широке коло додатків завдяки вільному його використанню. Проте науково-дослідних робіт, які б досліджували конкретно цю СУБД, порівняно мало. Її використовують разом із готовими програмними додатками, де важлива загальна продуктивність усієї системи. Дана стаття наочно продемонструвала ефективність вищеназваних засобів. У науковій статті «Захист SQL-бази в операційній системі Android» за авторством О. М. Гайдаєнко вказано за якими стандартами шифрувати дані у базі для їх криптостійкості. [9] Така інформація може доповнювати поточний додаток, розроблений у даній статті потрібно лише розробити необхідний алгоритм шифрування.

Інша робота «Розробка бази даних інформаційної системи діагностики глаукоми» [10]. База даних там із великою кількістю таблиць та зв'язків, проте не вказано за яким принципом буде надаватись доступ до цих даних, відповідно швидкодія може бути вкрай низькою, якщо не оптимізувати код.

Висновки. На основі проведених досліджень, які передбачались науково-дослідною роботою, проведений аналіз характеристик використання СУБД SQLite та відзначено ряд переваг для, які дозволяють зрозуміти її потенціальні переваги і в подальшому використовувати їх у програмних додатках, створених для операційної системи Windows. Наведений варіант розробленої програми-додатка, яка може застосовуватись як Organizer для нагадування про певну подію, використовуючи при цьому мінімум системних вимог та ресурсів. Сформований алгоритм може послужити як приклад застосування СУБД SQLite при створенні та управлінні найрізноманітніших баз даних для Windows-додатків з урахуванням зауважених переваг, що значно спрощуватиме роботу для розробників програмного забезпечення.

1. С.Д. Кузнєцов. Стандарты языка реляционных баз данных SQL: краткий обзор. / SIT Forum. Новости мира IT. http://citforum.ru/database/articles/art_2.shtml
2. А.Д. Хомоненко. База даних: учебное пособие. / Хоменко А.Д., Цыганков В.М., Мальцев М.Г. – ООО «КОРОНА-Век» 190005, Санкт-Петербург. – 736 с.
3. Саак А.Э. Информационные технологии управления: учебник для вузов / А.Э. Саак, Е.В. Пахомов, В.Н. Тюшняков. – СПб.: Питер, 3-е изд., 2011. – 640 с.
4. Д. Елджер. Бібіліотека програміста C++, – К.: Грамота.
5. Qt Downloads. https://www.qt.io/
6. SQLite Home Page. http://www.sqlite.org/
7. С. Кузнецов. Понятие модели данных. Обзор разновидностей моделей данных. / SIT Forum. Новости мира IT. http://citforum.ru/database/advanced_intro/6.shtml.
8. SqlQueryModel Class. QT Documentation. http://doc.qt.io/qt-5/qsqlquerymodel.html.
9. О. М. Гайдаєнко. - «Захист SQL-бази в операційній системі Android». – Вінницький Національний Технічний Університет, Вінниця.
10. О. В. Висоцька, І.Ю. Панфьорова, Г.М. Страшненко, С.О. Синенко, Ю.А. Дьомін. – «Розробка бази даних інформаційної системи діагностики глаукоми». – Харківський Національний Університет, Харків.