УДК 004.65
Коцюба А.Ю., Саковський С.В.
Луцький національний технічний університет

ПРО ІМПОРТУВАННЯ БАЗ ДАНИХ В ANDROID-ДОДАТОК

Коцюба А.Ю., Саковський С.В. Про імпортування баз даних в Android-додаток. Проведено аналіз сучасних підходів щодо зв'язування додатків створених для ОС Android з базами даних під різними СКБД. Описано основні труднощі, які можуть при цьому виникнути. Запропоновано два основні алгоритми вирішення цієї проблеми: паралельний та послідовний. Детальніше розглянуто послідовний алгоритм зв'язування, кінцева мета якого це імпортування БД, створеної, наприклад, на базі СКБД SQLite, в Android-додаток. Запропоновано та проілюстровано за допомогою відповідних лістингів коду два різних способи вирішення цієї проблеми.
Ключові слова: імпортування баз даних, Android-додаток, СКБД SQLite.

Коцюба А.Ю., Саковский С.В. Об импортировании баз данных в android-приложение. Проведен анализ современных подходов по связыванию приложений созданных для ОС Android с базами данных под различными СУБД. Описаны основные трудности, которые могут при этом возникнуть. Предложено два основных алгоритма решения этой проблемы: параллельный и последовательный. Подробнее рассмотрено последовательный алгоритм связывания, конечная цель которого это импорт БД, созданной, например, на базе СУБД SQLite, в Android-приложение. Предложены и проиллюстрированы с помощью соответствующих листингов кода два различных способа решения этой проблемы.
Ключевые слова: импортирование баз данных, Android-приложение, СУБД SQLite.

Kotsyuba A.Yu, Sakovskyy S.V. About importing databases into the android-application. The analysis of modern approaches for linking applications created for the Android OS with databases under different DBMSs is carried out. The main difficulties that can arise in this case are described. Two main algorithms for solving this problem are proposed: parallel and sequential. A sequential binding algorithm whose ultimate goal is database import created, for example, based on DBMS SQLite, into the Android application, is considered in detail. Two different ways of solving this problem are proposed and illustrated with the help of the corresponding code listings.
Keywords: importing database, Android-application, DBMS SQLite.

Вступ. Однією із головних проблем програмного забезпечення високо рівня була і досі залишається реалізація взаємодії з незалежним програмним забезпеченням обробки інформації для роботи з великими об'ємами інформації. Рішення цієї проблеми беруть на себе системи керування базами даних (СКБД). Вони реалізують інтерфейс між апаратними засобами накопичення інформації та програмним забезпеченням обробки інформації. При цьому в останні роки широкої популярності набула проблема розробки додатків для мобільних пристроїв (наприклад таких, які працюють на платформі ОС Android). У процесі створення Android-додатків досить часто виникає питання про можливість коректного імпортування баз даних великого об'єму зі складною, наприклад, схемою зв'язків.

Аналіз останніх досліджень та публікацій. Більше дізнатися про створення Android-додатків можна з електронного ресурсу [4]. У джерелах [1, 2, 5] наведено найвідоміші прийоми створення додатків під платформу ОС Android. Використання баз даних та їх співпрацю з такими додатками описано в електронних ресурсах [3, 5]. Для дослідження методик вирішення поставленої в роботі проблеми можна використати джерела [2, 4, 5]. Роботу з базами даних за допомогою СКБД висвітлено працях [3, 5].

Мета роботи – розгляд методик перенесення (або імпортування) бази даних в Android-додатки та визначення найефективнішої з них. Слід зауважити, що мету можна розділити на початкову (яка і є основною) та кінцеву (єдина проблема, яка стоїть на заваді її практичної реалізації – це знаходження найбільш оптимальних та простих способів налаштування ефективної взаємодії Android-додатків з відповідними БД). Кінцевою метою роботи є написання хоча б двох можливих способів такої методики імпортування та підключення готових баз даних до Android-додатків, що є відносно не складними в реалізації та одночасно оптимально ефективними.

Постановка наукової проблеми та обґрунтування вибору способу її рішення. Одним із найпоширеніших методів налаштування взаємодії Android-додатку з коректно створеною БД є паралельне її створення під час написання інтерфейсу та коду додатка. Зазвичай сучасні середовища, в яких реалізовано можливість створення такого додатка дозволяють це робити. Але при цьому є обмеження на СКБД (є ряд систем керування, які запропоновані розробнику); і в процесі ускладнення БД виникає багато проблем, вирішення яких пов'язано зі значними труднощами (при цьому не завжди вдається налаштувати оптимально коректну взаємодію).

Для випадків, коли БД є об'ємною і деякі структурні її характеристики в процесі розробки будуть весь час оптимізуватися, вищеописаний метод паралельного створення зазвичай не дасть

таких результатів як метод послідовного створення. Він полягає у тому, що спочатку БД і Android-додаток створюються незалежно (при написанні додатку, очевидно, потрібно враховувати подальше його зв'язування з БД); а далі сформована БД з вибраної розробником СКБД імпортується в додаток, який "готовий" до такої взаємодії. Виникає питання при виборі останнього підходу: з якими СКБД в такому випадку найефективніше співпрацювати? Дати однозначну відповідь, яка б задовольнила всіх компетентних розробників таких додатків, на нашу думку неможливо, оскільки технології в цьому напрямку весь час змінюються і майже все залежить від навиків та вмінь розробника. Тому нами запропоновано огляд деяких найвідоміших СКБД в контексті проблеми імпортування розробленої БД в Android-додаток.

MySQL – вільна (відкрита) система керування базами даних. MySQL є власністю компанії Oracle Corporation. Цю систему управління базами даних з відкритим кодом було створено як альтернативу комерційним системам. MySQL із самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL – одна з найпоширеніших таких систем. Вона використовується, у першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL є рішенням для малих і середніх додатків. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або віддалені клієнти, проте до дистрибутиву входить бібліотека внутрішнього сервера, що дозволяє включати MySQL до автономних програм. Вихідні коди сервера компілюються на багатьох платформах. Для некомерційного використання MySQL є безкоштовною. Можливості сервера MySQL можна сказати наступне:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн.;
- висока швидкість виконання команд;
- наявність простої та ефективної системи безпеки;
- можливість імпортування БД у різні середовища, зокрема і у андроїд подібні програми.

Але незважаючи на всі плюси даного середовища існують і кращі, з більш розвиненими функціями у вибраній нами сфері використання.

СКБД **Oracle** – це найпотужніший програмний комплекс, що дозволяє створювати додатки будь-якої складності. Ядром цього комплексу є база даних, що зберігає інформацію, кількість якої за рахунок наданих засобів масштабування практично безмежна. З високою ефективністю працювати з цією інформацією одночасно може практично будь-яка кількість користувачів (за наявності достатніх апаратних ресурсів), не проявляючи тенденції до зниження продуктивності системи при різкому збільшенні їхньої кількості.

Однією складовою успіху СКБД Oracle є те, що вона поставляється практично для всіх існуючих на сьогодні операційних систем. Працюючи під ОС Sun Solaris, Linux, Windows або під іншими платформами з продуктами Oracle не буде виникати ніяких проблем у роботі. Мінусом даного середовища є те, що воно платне, і потребує ліцензії на використання.

SQLite – це відкрита вбудована реляційна система керування базами даних. Основними характеристиками SQLite є:

- високий рівень мобільності (незалежності від платформи);
- простота у використанні;
- ефективність та надійність;
- має досить невеликий розмір (близько 300 кілобайт), що дозволяє використовувати цю СКБД на малопотужних пристроях (наприклад iPhone).

Крім цього SQLite доступний на будь-якому Android-пристрої, його не потрібно встановлювати окремо. SQLite підтримує типи TEXT (аналог String в Java), INTEGER (аналог long в Java) і REAL (аналог double в Java). Решта типів слід конвертувати, перш ніж зберігати в базі даних. SQLite сама по собі не перевіряє типи даних, тому ви можете записати ціле число в колонку, призначену для рядків, і навпаки.

На нашу думку саме СКБД SQLite є найбільш раціональним варіантом вибору середовища для імпортування баз даних в Android-додаток. Крім цього по функціональних можливостях вона не уступає вищеописаним системам керування. Звичайно, існують і інші не менш ефективні рішення даної проблеми. Але **метою** даної роботи є не лише вибір СКБД такої, що дозволила б задовільнити потреби будь-якого мобільного додатка в контексті описаної проблеми, а і опис процесу імпортування, при якому можна налагодити ефективну взаємодію з розробленим під БД Android-додатком.

Імпортування за допомогою СКБД SQLite. Зазначимо, що при використанні системи керування SQLite користувачу не потрібно застосовувати можливості віддаленого сервера, база даних після перенесення в додаток зберігається уже на самому мобільному пристрої.

Опишемо два способи такого процесу перенесення в додаток (імпортування).

Спосіб №1. Є багато програм різних програм для роботи з SQLite. Запропонуємо SQLite Database Browser, яка є відкритою і легко інсталюється в ПК розробника під ОС Windows.

Спочатку необхідно виконати копіювання заздалегідь підготовленого файлу бази даних в папку **assets**, а далі програмно можна скопіювати даний файл в потрібну системну папку бази даних мобільного додатка. При цьому слід зважати на те, що крім таблиці створеної розробником для додатка, Android також створює для своїх цілей нову таблицю **android_metadata** у вашій базі. Тому при ручному створенні бази вам необхідно створити як мінімум дві таблиці: системну і свою робочу. Для цього відкриємо створену базу і додамо нову таблицю під назвою "android_metadata":

```
CREATE TABLE "android_metadata" ("locale" TEXT DEFAULT 'en_US')
```

Додамо в таблицю один рядок:

```
INSERT INTO "android_metadata" VALUES ('en_US')
```

Далі створимо свої таблиці для роботи. При необхідно перейменувати поле "id" на "_id", як вимагає Android. У SQLite Database Browser це можна зробити, натиснувши на кнопку *Редагувати*, а потім вибрати таблицю, яку потрібно змінити, та поле для перейменування. Після виконання зазначених операцій база даних готова для використання у вашому додатку. Помістимо файл бази даних в папку **assets** мобільного проекту і створимо новий клас, що успадковується від SQLiteOpenHelper. Детальніше код додатку, що відповідає маніпуляціям наведено на наступному лістингу коду для переносу БД в Android-додатки:

```
public class DataBaseHelper extends SQLiteOpenHelper {

// Шлях до бази даних вашої програми
  private static String DB_PATH="/ data / data / YOUR_PACKAGE / databases /";
  private static String DB_NAME="myDBName";
  private SQLiteDatabase myDataBase;
  private final Context mContext;

// Конструктор
// Приймаємо і зберігаємо посилання на переданий контекст для доступу до ресурсів дод.
// @param context
  public DataBaseHelper (Context context) {
    super (context, DB_NAME, null, 1); this.mContext=context; }

// Створюємо порожню базу даних і перезаписує її нашої власною базою
  public void createDataBase() throws IOException {
    boolean dbExist=checkDataBase ();
    if (dbExist) { /* Нічого не робимо - база вже є */ }
    else {
// Викликаючи цей метод створюємо порожню базу, пізніше вона буде переписана
        this.getReadableDatabase ();
        try {
          copyDataBase (); }
        Catch (IOException e) {
          throw new Error ( "Error copying database"); }}}

// Перевіряємо, чи існує вже ця база, щоб не копіювати її кожен раз при запуску прогр.
// @return true якщо існує, false якщо не існує
  private boolean checkDataBase() {
    SQLiteDatabase checkDB=null;
    try {
      String myPath=DB_PATH+DB_NAME;
    checkDB=SQLiteDatabase.openDatabase (myPath,null,SQLiteDatabase.OPEN_READONLY); }
    Catch (SQLiteException e) { /* База ще не існує */ }
    if (checkDB!=null) { checkDB.close(); }
    return checkDB!=null?true:false; }

// Копіюємо базу з папки assets замість створеної локальної БД
// Виконуємо шляхом копіювання потоку байтів.
  private void copyDataBase() throws IOException {
// Відкриваємо локальну БД як вхідний потік
```

```
    InputStream myInput=mContext.getAssets().Open(DB_NAME);
// Шлях до новоствореної БД
    String outFileName=DB_PATH+DB_NAME;
// Відкриваємо порожню базу даних як вихідний потік
    OutputStream myOutput=new FileOutputStream(outFileName);
// Переміщаємо байти з вхідного файлу в вихідний
    byte [] buffer=new byte[1024];
    int length;
    while ((length=myInput.read (buffer))>0) { myOutput.write(buffer,0,length); }
// Закриваємо потоки
    myOutput.flush();
    myOutput.close();
    myInput.close(); }

  public void openDataBase () throws SQLException {
// Відкриваємо БД
    String myPath=DB_PATH+DB_NAME;
    myDataBase=SQLiteDatabase.openDatabase(myPath,null,SQLiteDatabase.OPEN_READONLY);}

  @Override

  public synchronized void close() {
    if (myDataBase!=null) myDataBase.close();
    super.close (); }

  @Override

  public void onCreate(SQLiteDatabase db) { ... }

  @Override

  public void onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion) { ...
// Тут можна додати допоміжні методи для доступу і отримання даних з БД
// Тут можна повертати курсори через "return myDataBase.query (....)", це полегшить їх
// використання в створенні адаптерів для ваших view
 ... }

// Тепер можна створити новий екземпляр класу DataBaseHelper і викликати методи
// createDataBase () і openDataBase (). Не забудьте змінити YOUR_PACKAGE на ім'я
// пакета в вашому додатку в рядку DB_PATH.

  DataBaseHelper myDbHelper=new DataBaseHelper (this);
  myDbHelper=new DataBaseHelper (this);

  try {
    myDbHelper.createDataBase (); }
  Catch (IOException ioe) {
    throw new Error ("Unable to create database"); }

  try {
    myDbHelper.openDataBase (); }
  Catch (SQLException sqle) {
    throw sqle; }
  ..........................................
```

Спосіб №2. Покладемо файл бази даних в zip-архів і перемістимо його в папку res/raw. У методі onCreate() перевіримо – чи запускається додаток перший раз. Розпакуємо БД і перемістимо її на SD-карту. Встановимо з'єднання з базою. Перевіримо, чи існує файл БД, якщо не існує, то потрібно це файл розгорнути.

Даний метод помістимо в onCreate().

```
public void createDataBase() throws IOException {
// Отримуємо шлях до SD-карти.
  File DB_PATH=myContext.getExternalCacheDir();
// Створюємо каталог для нашої бази даних
  DB_PATH.mkdirs ();
// Перевіряємо чи є вже файл БД на карті
  File db=new File (DB_PATH, DB_NAME);
  if (!db.exists()) {
// Якщо файлу немає, то спробуємо його створити
  db.createNewFile(); }
  try { copyFromZipFile(); }
```

```
  Catch (IOException e) { throw new Error("Error copying database",e); }}
```

Цей метод копіює файл БД з res/raw на SD-карту

```
private void copyFromZipFile() throws IOException {
  InputStream is=myContext.getResources().OpenRawResource(R.raw.database);
  File outFile=new File(DB_PATH,DB_NAME);
  OutputStream myOutput=new FileOutputStream(outFile.getAbsolutePath());
  ZipInputStream zis=new ZipInputStream(new BufferedInputStream(is));
  try {
    ZipEntry ze;
    while ((ze=zis.getNextEntry())!=null) {
      ByteArrayOutputStream baos=new ByteArrayOutputStream();
      byte[] buffer=new byte[1024];
      int count;
      while ((count=zis.read (buffer))!=-1) { baos.write(buffer,0,count); }
      baos.writeTo(myOutput); }}
  Finally {
    zis.close();
    myOutput.flush();
    myOutput.close();
    is.close(); }}
```

Встановлюємо з'єднання з БД.

```
public SQLiteDatabase openDataBase() throws SQLException {
  File DB_PATH=myContext.getExternalCacheDir();
  File dbFile=new File(DB_PATH,DB_NAME);
  myDataBase=SQLiteDatabase.openDatabase(dbFile.getAbsolutePath(),null,
                                                    SQLiteDatabase.OPEN_READWRITE);
  return myDataBase; }
```

Висновки. Після проведеного аналізу існуючих систем керування базами даних було обрано для використання систему SQLite. У ній спрощено підключення баз даних до Android-додатків. Програма є безкоштовною для користувачів і має відкритий код, та можливість роботи з БД, що створені за допомогою інших СКБД. Також за допомогою даного середовища можна принаймні 2-ма способами робити перенесення БД в Android-додатки. Перший спосіб полягає у звичайному копіюванні усієї бази даних у сам додаток, інший же полягає у створенні і записуванні на флеш пам'ять, яка є практично у всіх сучасних мобільних пристроях всієї бази даних а потім через вказування шляху у Android-додатку відбувається підключення до неї.

1. Дейтел П. Android для программистов: создаём приложения / П. Дейтел, Х. Дейтел, Э. Дейтел, М. Моргано. – СПб. : Питер, 2013. – 560 с.: ил.
2. Порменюк М.С. Порівняння засобів перевірки обмежень у реляційних СУБД / М.С. Порменюк, О.Ю. Безносик // Комп'ютерно-інформаційні технології в багаторівневій вищій освіті. – К., 2014. – С. 282-287.
3. Неполное Руководство по SQLite для пользователей Windows / Перевод: А.Г Пискунов. – 9 октября 2015 г. – 118 с.
4. Офіційний сайт розробників Android [Електронний ресурс]. – https://developer.android.com.
5. Пособие пользователя. Робота в среде SQLite [Електронний ресурс]. – http://ukrhosting.ua/posibnik_koristu-vacha-p-1.html.