Сохранение данных в SQL базу данных

В этом уроке

1. [Определить схемы и контракты](http://www.fandroid.info/sohranenie-dannyh-v-sql-bazu-dannyh/#1)
2. [Создание базы данных с помощью SQL помощника](http://www.fandroid.info/sohranenie-dannyh-v-sql-bazu-dannyh/#2)
3. [Положить информацию в базу данных](http://www.fandroid.info/sohranenie-dannyh-v-sql-bazu-dannyh/#3)
4. [Считать информацию из базы данных](http://www.fandroid.info/sohranenie-dannyh-v-sql-bazu-dannyh/#4)
5. [Удаление информации из базы данных](http://www.fandroid.info/sohranenie-dannyh-v-sql-bazu-dannyh/#5)
6. [Обновление базы данных](http://www.fandroid.info/sohranenie-dannyh-v-sql-bazu-dannyh/#6)

Вы также должны прочитать

* [Использование баз данных](http://developer.android.com/guide/topics/data/data-storage.html#db)
* [Как подключить базу данных SQLite в Android приложение](http://www.fandroid.info/kak-podklyuchit-bazu-dannyh-sqlite-v-android-prilozhenie/)

Сохранение данных в базу данных идеально подходит для повторяющихся или структурированных данных, такие как контактная информация. Этот курс предполагает, что вы знакомы с SQL базами данных в целом, и это поможет вам начать работу с SQLite базами данных на Android. API интерфейсы, которые вы будете использовать для работы с базой данных на Android доступны в [android.database.sqlite](http://developer.android.com/reference/android/database/sqlite/package-summary.html) пакете.

Определить схемы и контракты

Одним из основных принципов SQL баз данных является схема: формальное описание того, как база данных организована. Схема отражена в SQL запросах, которые вы используете для создания базы данных. Возможно, вам будет полезно создать класс компаньон, известный как *контракт* , который явно определяет макет вашей схемы систематическим и самодокументируемым образом.

Контрактный класс представляет собой контейнер для констант, определяющих имена для URI, таблиц и столбцов. Класс контракта позволяет использовать одни и те же константы во всех других классах в том же пакете. Это позволяет изменить имя столбца в одном месте и распространить изменение по всему коду.

Хороший способ организовать класс контракта это разместить определения, которые являются глобальными для всей базы данных, в корневом уровне класса. Затем создайте внутренние классы для каждой из таблиц, которые перечисляют свои столбцы.

**Примечание:** Реализуя [BaseColumns](http://developer.android.com/reference/android/provider/BaseColumns.html) интерфейс, ваш внутренний класс может унаследовать поле первичного ключа, называемое \_ID , наличие которых будут ожидать некоторые Android классы, такие как курсоры адаптеров. Это не обязательно, но это может помочь вашей работе с базой данных в гармонии с библиотеками Android.

Например, этот фрагмент определяет имя таблицы и имена столбцов для одной из таблиц:

public final class FeedReaderContract {

    // To prevent someone from accidentally instantiating the contract class,

    // give it an empty constructor.

    public FeedReaderContract() {}

    /\* Inner class that defines the table contents \*/

    public static abstract class FeedEntry implements BaseColumns {

        public static final String TABLE\_NAME = "entry";

        public static final String COLUMN\_NAME\_ENTRY\_ID = "entryid";

        public static final String COLUMN\_NAME\_TITLE = "title";

        public static final String COLUMN\_NAME\_SUBTITLE = "subtitle";

        ...

    }

}

Создание базы данных с помощью SQL помощника

После того как вы определили, как выглядит ваша база данных, вы должны реализовать методы, которые создают и поддерживают базу данных и таблицы. Вот некоторые типичные запросы, которые создают и удалять таблицы:

private static final String TEXT\_TYPE = " TEXT";

private static final String COMMA\_SEP = ",";

private static final String SQL\_CREATE\_ENTRIES =

    "CREATE TABLE " + FeedEntry.TABLE\_NAME + " (" +

    FeedEntry.\_ID + " INTEGER PRIMARY KEY," +

    FeedEntry.COLUMN\_NAME\_ENTRY\_ID + TEXT\_TYPE + COMMA\_SEP +

    FeedEntry.COLUMN\_NAME\_TITLE + TEXT\_TYPE + COMMA\_SEP +

    ... // Any other options for the CREATE command

    " )";

private static final String SQL\_DELETE\_ENTRIES =

    "DROP TABLE IF EXISTS " + FeedEntry.TABLE\_NAME;

Так же, как файлы, которые вы храните на устройстве во [внутреннем хранилище](http://developer.android.com/guide/topics/data/data-storage.html#filesInternal), Android хранит вашу базу данных в приватной части дискового пространства, ассоциированном с приложением. Ваши данные надежно защищены, потому что по умолчанию эта область не является доступной для других приложений.

Полезный набор функций API доступен в [SQLiteOpenHelper](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html) классе. Когда вы используете данный класс для получения ссылки на базу данных, система выполняет потенциально длительные операции создания и обновления базы данных только при необходимости и *не во время запуска приложения*. Все что вам нужно сделать это вызвать [getWritableDatabase()](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "getWritableDatabase())или[getReadableDatabase()](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "getReadableDatabase()).

**Примечание:** Т.к. они могут быть длительными, убедитесь, что вы вызываете[getWritableDatabase()](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#getWritableDatabase()) или [getReadableDatabase()](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "getReadableDatabase()) в фоновом потоке, например, используя[AsyncTask](http://developer.android.com/reference/android/os/AsyncTask.html) или [IntentService](http://developer.android.com/reference/android/app/IntentService.html).

Чтобы использовать [SQLiteOpenHelper](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html), создайте подкласс, который перегружает [onCreate()](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "onCreate(android.database.sqlite.SQLiteDatabase)),[onUpgrade()](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "onUpgrade(android.database.sqlite.SQLiteDatabase, int, int)) и [onOpen()](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "onOpen(android.database.sqlite.SQLiteDatabase)) методы обратного вызова. Вы также можете реализовать [onDowngrade()](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "onDowngrade(android.database.sqlite.SQLiteDatabase, int, int)), но это не обязательно.

Например, вот реализация [SQLiteOpenHelper](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html) , который использует некоторые из команд, показанных выше:

public class FeedReaderDbHelper extends SQLiteOpenHelper {

    // If you change the database schema, you must increment the database version.

    public static final int DATABASE\_VERSION = 1;

    public static final String DATABASE\_NAME = "FeedReader.db";

    public FeedReaderDbHelper(Context context) {

        super(context, DATABASE\_NAME, null, DATABASE\_VERSION);

    }

    public void onCreate(SQLiteDatabase db) {

        db.execSQL(SQL\_CREATE\_ENTRIES);

    }

    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

        // This database is only a cache for online data, so its upgrade policy is

        // to simply to discard the data and start over

        db.execSQL(SQL\_DELETE\_ENTRIES);

        onCreate(db);

    }

    public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {

        onUpgrade(db, oldVersion, newVersion);

    }

}

Для доступа к базе данных, создайте экземпляр подкласса [SQLiteOpenHelper](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html):

FeedReaderDbHelper mDbHelper = new FeedReaderDbHelper(getContext());

Положить информацию в базу данных

Вставьте данные в базу данных, передав [ContentValues](http://developer.android.com/reference/android/content/ContentValues.html) объект в [insert()](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "insert(java.lang.String, java.lang.String, android.content.ContentValues)) метод:

// Gets the data repository in write mode

SQLiteDatabase db = mDbHelper.getWritableDatabase();

// Create a new map of values, where column names are the keys

ContentValues values = new ContentValues();

values.put(FeedEntry.COLUMN\_NAME\_ENTRY\_ID, id);

values.put(FeedEntry.COLUMN\_NAME\_TITLE, title);

values.put(FeedEntry.COLUMN\_NAME\_CONTENT, content);

// Insert the new row, returning the primary key value of the new row

long newRowId;

newRowId = db.insert(

         FeedEntry.TABLE\_NAME,

         FeedEntry.COLUMN\_NAME\_NULLABLE,

         values);

Первый аргумент для [insert()](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "insert(java.lang.String, java.lang.String, android.content.ContentValues)) это просто имя таблицы. Второй аргумент содержит имя столбца, в который библиотека может вставить значение NULL в случае, если [ContentValues](http://developer.android.com/reference/android/content/ContentValues.html) пуст (если вы вместо этого установите его "null", то библиотека не сможет вставить строку, когда нет ни одного значения).

Считать информацию из базы данных

Для чтения из базы данных, используйте [query()](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "query(boolean, java.lang.String, java.lang.String[], java.lang.String, java.lang.String[], java.lang.String, java.lang.String, java.lang.String, java.lang.String)) метод, передавая ему свои критерии отбора и нужные столбцы. Метод сочетает в себе элементы [insert()](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "insert(java.lang.String, java.lang.String, android.content.ContentValues)) и [update()](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "update(java.lang.String, android.content.ContentValues, java.lang.String, java.lang.String[])), но список столбцов определяет данные, которые вы хотите получить, а не вставить. Результаты запроса возвращаются в [Cursor](http://developer.android.com/reference/android/database/Cursor.html) объекте.

SQLiteDatabase db = mDbHelper.getReadableDatabase();

// Define a projection that specifies which columns from the database

// you will actually use after this query.

String[] projection = {

    FeedEntry.\_ID,

    FeedEntry.COLUMN\_NAME\_TITLE,

    FeedEntry.COLUMN\_NAME\_UPDATED,

    ...

    };

// How you want the results sorted in the resulting Cursor

String sortOrder =

    FeedEntry.COLUMN\_NAME\_UPDATED + " DESC";

Cursor c = db.query(

    FeedEntry.TABLE\_NAME,  // The table to query

    projection,                               // The columns to return

    selection,                                // The columns for the WHERE clause

    selectionArgs,                            // The values for the WHERE clause

    null,                                     // don't group the rows

    null,                                     // don't filter by row groups

    sortOrder                                 // The sort order

    );

Чтобы посмотреть на значения строки данных в курсоре, используйте один из методов перемещения [Cursor](http://developer.android.com/reference/android/database/Cursor.html) , которые вы всегда должны вызывать, прежде чем начать вычитывать значения. Как правило, вы должны начать с вызова [moveToFirst()](http://developer.android.com/reference/android/database/Cursor.html" \l "moveToFirst()), которая ставит «позицию чтения» на первую запись результата. Для каждой строки, вы можете прочитать значение столбца вызвав один из методов [Cursor](http://developer.android.com/reference/android/database/Cursor.html) для получения значений, таких как [getString()](http://developer.android.com/reference/android/database/Cursor.html" \l "getString(int)) или [getLong()](http://developer.android.com/reference/android/database/Cursor.html" \l "getLong(int)). Для каждого из методов получения, вы должны передать индекс желаемой позиции столбца, который вы можете получить, вызвав [getColumnIndex()](http://developer.android.com/reference/android/database/Cursor.html" \l "getColumnIndex(java.lang.String)) или [getColumnIndexOrThrow()](http://developer.android.com/reference/android/database/Cursor.html" \l "getColumnIndexOrThrow(java.lang.String)). Например:

cursor.moveToFirst();

long itemId = cursor.getLong(

    cursor.getColumnIndexOrThrow(FeedEntry.\_ID)

);

Удаление информации из базы данных

Для удаления строки из таблицы, необходимо предоставить критерии отбора, которые идентифицируют строки. API базы данных предоставляет механизм для создания критериев отбора, который защищает от SQL инъекций. Механизм разделяет спецификацию отбора на запрос отбора и его аргументы. Запрос определяет столбцы, по которым будут отбираться записи, а также позволяет задать значения для проверки по полям. Значения аргументов для проверки связаны с запросом. Т.к. результат не обрабатываются так же, как обычный SQL запрос, это защищает от SQL инъекций.

// Define 'where' part of query.

String selection = FeedEntry.COLUMN\_NAME\_ENTRY\_ID + " LIKE ?";

// Specify arguments in placeholder order.

String[] selectionArgs = { String.valueOf(rowId) };

// Issue SQL statement.

db.delete(table\_name, selection, selectionArgs);

Обновление базы данных

Когда вам необходимо изменить подмножество значений в базе данных, используйте [update()](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "update(java.lang.String, android.content.ContentValues, java.lang.String, java.lang.String[]))метод.

Обновление значений таблицы сочетает в себе синтаксис [insert()](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "insert(java.lang.String, java.lang.String, android.content.ContentValues)) с where синтаксисом [delete()](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "delete(java.lang.String, java.lang.String, java.lang.String[])).

SQLiteDatabase db = mDbHelper.getReadableDatabase();

// New value for one column

ContentValues values = new ContentValues();

values.put(FeedEntry.COLUMN\_NAME\_TITLE, title);

// Which row to update, based on the ID

String selection = FeedEntry.COLUMN\_NAME\_ENTRY\_ID + " LIKE ?";

String[] selectionArgs = { String.valueOf(rowId) };

int count = db.update(

    FeedReaderDbHelper.FeedEntry.TABLE\_NAME,

    values,

    selection,

    selectionArgs);