**Сохранение данных в базах данных SQL**

Содержание этого урока

1. [Определение схемы и контракта](https://developer.android.com/training/basics/data-storage/databases.html?hl=ru#DefineContract)
2. [Создание базы данных с помощью SQL Helper](https://developer.android.com/training/basics/data-storage/databases.html?hl=ru#DbHelper)
3. [Размещение информации в базе данных](https://developer.android.com/training/basics/data-storage/databases.html?hl=ru#WriteDbRow)
4. [Чтение информации из базы данных](https://developer.android.com/training/basics/data-storage/databases.html?hl=ru#ReadDbRow)
5. [Удаление информации из базы данных](https://developer.android.com/training/basics/data-storage/databases.html?hl=ru#DeleteDbRow)
6. [Обновление базы данных](https://developer.android.com/training/basics/data-storage/databases.html?hl=ru#UpdateDbRow)

См. также:

* [Использование баз данных](https://developer.android.com/guide/topics/data/data-storage.html?hl=ru#db)

Сохранение данных в базе идеально подходит для повторяющихся и структурированных данных, таких как контактная информация. В этом учебном курсе предполагается, что вы владеете общими знаниями о базах данных SQL, и он поможет вам начать работать с базами данных SQLite на платформе Android. API-интерфейсы, необходимые для использования базы данных на платформе Android, доступны в составе пакета [android.database.sqlite](https://developer.android.com/reference/android/database/sqlite/package-summary.html?hl=ru).

Определение схемы и контракта

Одним из основных элементов баз данных SQL является схема, которая представляет собой формальную декларацию способа организации базы данных. Схема отражается в выражениях SQL, используемых для создания базы данных. Для вас может оказаться полезным создать сопутствующий класс (*класс-контракт*), явно указывающий структуру схемы систематическим и самодокументирующим способом.

Класс-контракт представляет собой контейнер, определяющий имена для URI-адресов, таблиц и столбцов. Класс-контракт позволяет использовать одни и те же постоянные значения во всех других классах этого же пакета. Таким образом, при изменении имени столбца в одном месте это изменение применяется во всем коде.

Для удобства организации класс-контракта стоит поместить глобальные определения базы данных на корневой уровень класса. Затем нужно создать внутренний класс для каждой таблицы, производящей нумерацию столбцов.

**Примечание.** За счет реализации интерфейса [BaseColumns](https://developer.android.com/reference/android/provider/BaseColumns.html?hl=ru) внутренний класс может наследовать поле первичного ключа \_ID, наличия которого ожидают от него некоторые классы Android (например, адаптеры курсора). Это не является обязательным условием, однако может помочь обеспечить гармоничную работу вашей базы данных в инфраструктуре Android.

Например, в этом фрагменте кода определяются имя таблицы и имена столбцов для одной таблицы:

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 Helper

После определения внешнего вида базы данных следует реализовать методы создания и обслуживания базы данных и таблиц. Вот некоторые типичные выражения для создания и удаления таблиц:

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;

Как и при сохранении файлов во [внутренней памяти](https://developer.android.com/guide/topics/data/data-storage.html?hl=ru#filesInternal) устройства, Android сохраняет вашу базу данных в закрытой области диска, связанной с приложением Эти данные защищены, потому что эта область по умолчанию недоступна другим приложениям.

Полезный набор API-интерфейсов содержится в классе [SQLiteOpenHelper](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=ru). Если вы используете этот класс для получения ссылок на свою базу данных, система выполняет потенциально долговременные операции создания и обновления базы данных только тогда, когда это необходимо, а *не при запуске приложения*. Для этого нужно использовать вызов [getWritableDatabase()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=ru" \l "getWritableDatabase()) или [getReadableDatabase()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=ru" \l "getReadableDatabase()).

**Примечание.** Поскольку операции могут выполняться длительное время, вызывайте [getWritableDatabase()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=ru" \l "getWritableDatabase()) или [getReadableDatabase()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=ru" \l "getReadableDatabase()) в фоновом потоке, например с [AsyncTask](https://developer.android.com/reference/android/os/AsyncTask.html?hl=ru) или [IntentService](https://developer.android.com/reference/android/app/IntentService.html?hl=ru).

Для использования [SQLiteOpenHelper](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=ru) создайте подкласс, заменяющий методы вызова [onCreate()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=ru" \l "onCreate(android.database.sqlite.SQLiteDatabase)), [onUpgrade()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=ru" \l "onUpgrade(android.database.sqlite.SQLiteDatabase, int, int)) и [onOpen()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=ru" \l "onOpen(android.database.sqlite.SQLiteDatabase)). Также вы можете использовать [onDowngrade()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=ru" \l "onDowngrade(android.database.sqlite.SQLiteDatabase, int, int)), но это не требуется.

Например, вот реализация [SQLiteOpenHelper](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=ru), при которой используются некоторые из приведенных выше команд:

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 получения доступа к базе данных создайте экземпляр подкласса [SQLiteOpenHelper](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=ru):

FeedReaderDbHelper mDbHelper = new FeedReaderDbHelper(getContext());

Размещение информации в базе данных

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

// 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 [insert()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=ru" \l "insert(java.lang.String, java.lang.String, android.content.ContentValues)) представляет собой просто название таблицы. Второй аргумент указывает имя столбца, в который инфраструктура вставляет значение NULL, если [ContentValues](https://developer.android.com/reference/android/content/ContentValues.html?hl=ru) является пустым (если вместо этого указать "null", то инфраструктура не будет вставлять строку в случае отсутствия значений).

Чтение информации из базы данных

Для чтения из базы данных используйте метод [query()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=ru" \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)) с передачей критериев выделения и желаемых столбцов. Метод сочетает элементы[insert()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=ru#insert(java.lang.String, java.lang.String, android.content.ContentValues)) и [update()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=ru" \l "update(java.lang.String, android.content.ContentValues, java.lang.String, java.lang.String[])), за исключением того, что список столбцов определяет данные, которые вы хотите получить, а не данные для вставки. Результаты запроса возвращаются в объекте [Cursor](https://developer.android.com/reference/android/database/Cursor.html?hl=ru).

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 строку в месте курсора, используйте один из методов перемещения [Cursor](https://developer.android.com/reference/android/database/Cursor.html?hl=ru), которые всегда нужно вызывать перед считыванием значений. Обычно следует начинать с вызова [moveToFirst()](https://developer.android.com/reference/android/database/Cursor.html?hl=ru" \l "moveToFirst()), который помещает "позицию чтения" на первую запись в результатах. Для каждой строки значение столбца можно прочитать, вызвав один из методов [Cursor](https://developer.android.com/reference/android/database/Cursor.html?hl=ru) get, например [getString()](https://developer.android.com/reference/android/database/Cursor.html?hl=ru" \l "getString(int)) или[getLong()](https://developer.android.com/reference/android/database/Cursor.html?hl=ru#getLong(int)). Для каждого из методов get вы должны передать указатель желаемого столбца, который может вызвать [getColumnIndex()](https://developer.android.com/reference/android/database/Cursor.html?hl=ru" \l "getColumnIndex(java.lang.String)) или[getColumnIndexOrThrow()](https://developer.android.com/reference/android/database/Cursor.html?hl=ru#getColumnIndexOrThrow(java.lang.String)). Например:

cursor.moveToFirst();  
long itemId = cursor.getLong(  
    cursor.getColumnIndexOrThrow(FeedEntry.\_ID)  
);

Удаление информации из базы данных

Для удаления строк из таблицы нужно указать критерии выделения, идентифицирующие строки. API-интерфейс базы данных обеспечивает механизм для создания критериев выделения, предоставляющий защиту от внедрения SQL-кода. Механизм делит спецификацию выбора на предложение выбора и аргументы выбора. Предложение определяет столбцы для рассмотрения, а также позволяет объединять операции тестирования столбцов. Аргументы представляют собой значения для тестирования, которые привязаны к пункту. Поскольку результат обрабатывается не как обычные выражения SQL, он защищен от внедрения SQL-кода.

// 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

При необходимости изменить набор значений базы данных используйте метод [update()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=ru" \l "update(java.lang.String, android.content.ContentValues, java.lang.String, java.lang.String[])).

Обновление таблицы сочетает значения синтаксиса [insert()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=ru" \l "insert(java.lang.String, java.lang.String, android.content.ContentValues)) и синтаксиса where для [delete()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=ru" \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);