Сохранение данных в SQL базах данных

[Предыдущая](https://developer.android.com/training/basics/data-storage/files.html)[Следующая](https://developer.android.com/training/basics/data-storage/databases.html)

Этот урок научит вас

1. [Определение схемы и договора](https://developer.android.com/training/basics/data-storage/databases.html#DefineContract)
2. [Создание базы данных с помощью SQL-Helper](https://developer.android.com/training/basics/data-storage/databases.html#DbHelper)
3. [Поместить информацию в базу данных](https://developer.android.com/training/basics/data-storage/databases.html#WriteDbRow)
4. [Читайте информацию из базы данных](https://developer.android.com/training/basics/data-storage/databases.html#ReadDbRow)
5. [Удаление информации из базы данных](https://developer.android.com/training/basics/data-storage/databases.html#DeleteDbRow)
6. [Обновление базы данных](https://developer.android.com/training/basics/data-storage/databases.html#UpdateDbRow)

Вы также должны прочитать

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

Сохранение данных в базу данных , идеально подходит для повторяющихся или структурированных данных, таких как контактная информация. Этот класс предполагает , что вы знакомы с базами данных SQL в целом и помогает вам начать работу с базами данных SQLite на Android. Эти API вам нужно использовать базу данных на Android доступны в [android.database.sqlite](https://developer.android.com/reference/android/database/sqlite/package-summary.html)пакете.

Определение схемы и договора

Одним из основных принципов баз данных SQL является схема: официальное заявление о том ,как организована базы данных. Схема находит свое отражение в заявлениях SQL , которые используются для создания базы данных. Вы можете найти полезным создать класс -компаньон, известный как *контракт* класса, который явно указывает расположение вашей схемы на систематической и самодокументирующимся образом.

Класс контракта является контейнером для констант, которые определяют имена для URIs, таблиц и столбцов. Класс контракта позволяет использовать одни и те же константы во всех других классов в том же пакете. Это позволяет изменить имя столбца в одном месте и он будет распространяться по всему коду.

Хороший способ организовать класс контракта должен поместить определения, которые являются глобальными для всей базы данных в корневом уровне класса. Затем создайте внутренний класс для каждой таблицы, которая перечисляет ее столбцы.

**Примечание:** При реализации [BaseColumns](https://developer.android.com/reference/android/provider/BaseColumns.html)интерфейса, ваш внутренний класс может наследовать поле первичного ключа с именем ,\_IDчто некоторые Android классы , такие как курсор адаптеров будет ожидать , что она есть. Это не обязательно, но это может помочь ввашей базе данных работать гармонично с Android рамки.

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

общественности окончательный класс FeedReaderContract { // Чтобы предотвратить кто - то от случайного инстанцирования класса контракта, // сделать конструктор частные. частный FeedReaderContract () {} / \* Внутренний класс , который определяет содержание таблицы \* / общественный статический класс FeedEntry реализует BaseColumns { общественное статическое окончательный Строка TABLE\_NAME = "запись" ; общественная статическая конечная строка COLUMN\_NAME\_TITLE = "название" , публичное статическое окончательное Строка COLUMN\_NAME\_SUBTITLE = "подзаголовок" ; } }   
      
      
      
  
      
      
          
          
       

Создание базы данных с помощью SQL-Helper

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 высказывания, которые создают и удалить таблицу:

частный статический конечная строка TEXT\_TYPE = "TEXT" ; частное статическое окончательное Строка COMMA\_SEP = "" ; частные статические окончательный строк SQL\_CREATE\_ENTRIES = "CREATE TABLE" + FeedEntry . TABLE\_NAME + "(" + FeedEntry . \_ID + "INTEGER PRIMARY KEY," + FeedEntry . COLUMN\_NAME\_TITLE + TEXT\_TYPE + COMMA\_SEP + FeedEntry . COLUMN\_NAME\_SUBTITLE + TEXT\_TYPE + ")" ; частные статические финальные струнных SQL\_DELETE\_ENTRIES = "DROP TABLE IF EXISTS" + FeedEntry . TABLE\_NAME ;   
   
   
      
      
      
   

Так же , как файлы , которые сохранены на устройства [внутренней памяти](https://developer.android.com/guide/topics/data/data-storage.html#filesInternal) , Android хранит вашу базу данных в частном секторе дискового пространства, ассоциированную приложение. Ваши данные в безопасности, поскольку по умолчанию эта область не доступна для других приложений.

Полезный набор API , можно найти в [SQLiteOpenHelper](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html)классе. При использовании данного класса , чтобы получить ссылки на базы данных, система выполняет потенциально Продолжительные операции создания и обновления базы данных только при необходимости и *не во время запуска приложения* . Все , что вам нужно сделать , это позвонить [getWritableDatabase()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "getWritableDatabase())или [getReadableDatabase()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "getReadableDatabase()).

**Примечание:** Потому что они могут быть длинными работает, убедитесь , что вы звоните [getWritableDatabase()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "getWritableDatabase())или[getReadableDatabase()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#getReadableDatabase())в фоновом потоке, например, с помощью [AsyncTask](https://developer.android.com/reference/android/os/AsyncTask.html)или [IntentService](https://developer.android.com/reference/android/app/IntentService.html).

Для использования [SQLiteOpenHelper](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html), создать подкласс , который переопределяет [onCreate()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "onCreate(android.database.sqlite.SQLiteDatabase)), [onUpgrade()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "onUpgrade(android.database.sqlite.SQLiteDatabase, int, int))и [onOpen()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "onOpen(android.database.sqlite.SQLiteDatabase))методы обратного вызова. Вы также можете реализовать [onDowngrade()](https://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](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html)которая использует некоторые из команд , показанных выше:

общественный класс FeedReaderDbHelper расширяет SQLiteOpenHelper { // Если изменить схему базы данных, необходимо увеличить версию базы данных. публичного статического конечная INT DATABASE\_VERSION = 1 ; общественная статическая конечная строка DATABASE\_NAME = "FeedReader.db" ; общественный FeedReaderDbHelper ( Context контекст ) { супер ( контекст , DATABASE\_NAME , нуль , DATABASE\_VERSION ); } общественного недействительными OnCreate ( SQLiteDatabase дБ ) {         дБ . ExecSQL ( SQL\_CREATE\_ENTRIES ); } общественного недействительными onUpgrade ( SQLiteDatabase дб , INT oldVersion , INT newVersion ) { // Эта база данных является только кэш онлайн данных, поэтому его политика обновление // просто отбросить данные и начать заново         дб . ExecSQL ( SQL\_DELETE\_ENTRIES );         OnCreate ( дБ ); } общественного недействительными onDowngrade ( SQLiteDatabase дб , INT oldVersion , INT newVersion ) {         onUpgrade ( дБ , oldVersion , newVersion ); } }   
      
      
      
  
      
          
      
      
  
      
      
          
          
  
  
      
   

Чтобы получить доступ к базе данных, создать экземпляр подкласса [SQLiteOpenHelper](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html):

FeedReaderDbHelper mDbHelper = новый FeedReaderDbHelper ( getContext ());

Поместить информацию в базу данных

Вставка данных в базу данных, передавая [ContentValues](https://developer.android.com/reference/android/content/ContentValues.html) объект к [insert()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "insert(java.lang.String, java.lang.String, android.content.ContentValues))методу:

// Получает хранилище данных в режиме записи SQLiteDatabase дБ = mDbHelper . GetWritableDatabase (); // Создать новую карту значений, где имена столбцов являются ключами ContentValues значения = новые ContentValues (); значения . Ставить ( FeedEntry . COLUMN\_NAME\_TITLE , название ); значения . ставить ( FeedEntry . COLUMN\_NAME\_SUBTITLE , субтитров ); // Вставьте новую строку, возвращая значение первичного ключа новой строки длиной newRowId = дб . вставки ( FeedEntry . tABLE\_NAME , нулевые , значения );

Первый аргумент [insert()](https://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 указывает структуру , что делать в том случае, если [ContentValues](https://developer.android.com/reference/android/content/ContentValues.html)пусто (то есть, вы не делали [put](https://developer.android.com/reference/android/content/ContentValues.html" \l "put(java.lang.String, byte[]))каких - либо значений).Если указать имя столбца, рамки вставляет строку и устанавливает значение этого столбца NULL. Если вы укажете null, как в этом примере кода, каркас не вставить строку , когда нет значения.

Читайте информацию из базы данных

Для чтения из базы данных, используйте [query()](https://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()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "insert(java.lang.String, java.lang.String, android.content.ContentValues)) и [update()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "update(java.lang.String, android.content.ContentValues, java.lang.String, java.lang.String[])), кроме списка столбец определяет данные , которые вы хотите получать, а не данных для вставки. Результаты запроса возвращаются к вам в [Cursor](https://developer.android.com/reference/android/database/Cursor.html)объекте.

SQLiteDatabase дБ = mDbHelper . GetReadableDatabase (); // Определяем *проекцию* , которая определяет , какие столбцы из базы данных // вы на самом деле использовать после этого запроса. Строка [] проекция = { FeedEntry . \_ID , FeedEntry . COLUMN\_NAME\_TITLE , FeedEntry . COLUMN\_NAME\_SUBTITLE }; // результаты фильтра , где "название" = "My Title" Строка выбора = FeedEntry . COLUMN\_NAME\_TITLE + "=?" ; Строка [] selectionArgs = { "My Title" }; // Как вы хотите , чтобы результаты , отсортированные в результате Cursor строки SortOrder = FeedEntry . COLUMN\_NAME\_SUBTITLE + "DESC" ; Курсор с = дБ . запросов ( FeedEntry . TABLE\_NAME , // таблица для запроса     проекции , // колонки для возврата     выбор , // столбцы для предложения WHERE     selectionArgs , // значения для предложения WHERE NULL , // не сгруппировать строки NULL , // не фильтровать по строкам групп     SortOrder                                 // порядок сортировки );

Чтобы посмотреть на строку в курсора, используйте один из [Cursor](https://developer.android.com/reference/android/database/Cursor.html)методов перемещения, которые вы всегда должны позвонить , прежде чем начать чтение значений. Как правило, вы должны начать с вызова [moveToFirst()](https://developer.android.com/reference/android/database/Cursor.html" \l "moveToFirst()), который помещает "чтение положения" на первой записи в результатах. Для каждой строки, вы можете прочитать значение столбца путем вызова одного из [Cursor](https://developer.android.com/reference/android/database/Cursor.html)методов GET, таких как[getString()](https://developer.android.com/reference/android/database/Cursor.html#getString(int))или [getLong()](https://developer.android.com/reference/android/database/Cursor.html" \l "getLong(int)). Для каждого из методов GET, вы должны передать позицию индекса столбца , желание, которое вы можете получить по телефону [getColumnIndex()](https://developer.android.com/reference/android/database/Cursor.html" \l "getColumnIndex(java.lang.String))или [getColumnIndexOrThrow()](https://developer.android.com/reference/android/database/Cursor.html" \l "getColumnIndexOrThrow(java.lang.String)). Например:

курсор . moveToFirst (); длинный Itemid = курсор . getLong (     курсор . getColumnIndexOrThrow ( FeedEntry . \_ID ) );

Удаление информации из базы данных

Для удаления строк из таблицы, вам необходимо предоставить критерии выбора, которые идентифицируют строки. API базы данных предоставляет механизм для создания критериев выбора, который защищает от инъекции SQL. Механизм делит спецификацию выбора в клаузулы выбора и отбора аргументов. Предложение определяет столбцы, чтобы посмотреть, а также позволяет объединять тесты столбцов. Аргументы значения для проверки против, которые связаны в п. Поскольку результат не обрабатываются так же, как обычный SQL заявление, он невосприимчив к инъекции SQL.

// Определить ' , где' часть запроса. Строка выбора = FeedEntry . COLUMN\_NAME\_TITLE + "как?" ; . // Укажите аргументы в порядке заполнителем Строка [] selectionArgs = { "MyTitle" }; . // Выпуск SQL заявление дб . Удалить ( FeedEntry . TABLE\_NAME , выбор , selectionArgs );

Обновление базы данных

Когда вам нужно изменить подмножество ваших значений базы данных используется [update()](https://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 таблицы сочетает в себе значения содержания синтаксиса [insert()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "insert(java.lang.String, java.lang.String, android.content.ContentValues)) с whereсинтаксисом [delete()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html" \l "delete(java.lang.String, java.lang.String, java.lang.String[])).

SQLiteDatabase дБ = mDbHelper . GetReadableDatabase (); // Новое значение для одного столбца ContentValues значений = новых ContentValues (); значений . Поместить ( FeedEntry . COLUMN\_NAME\_TITLE , название ); // Какая строка для обновления, основанный на названии Строка выбора = FeedEntry . COLUMN\_NAME\_TITLE + "как?" , строка [] selectionArgs = { "MyTitle" }; INT кол = дб . обновление ( FeedReaderDbHelper . FeedEntry . TABLE\_NAME ,     значения ,     выбор ,     selectionArgs );