# 在 SQL 数据库中保存数据

本课程将向您展示如何

- 定义架构和契约
- 使用 SQL 辅助工具创建数据库
- 将信息输入到数据库
- 从数据库读取信息
- 从数据库删除信息
- 更新数据库

---

您也应该阅读

- 使用数据库

将数据保存到数据库对于重复或结构化数据（比如契约信息）而言是理想之选。 本课程假定您基本熟悉 SQL 数据库并且可帮助您开始在 Android 中使用 SQLite 数据库。 您在 Android 中使用数据库所需的 API 在 `android.database.sqlite` (https://developer.android.com/reference/android/database/sqlite/package-summary.html?hl=zh-cn) 软件包中提供。

## 定义架构和契约

SQL 数据库的主要原则之一是架构：数据库如何组织的正式声明。 架构体现于您用于创建数据库的 SQL 语句。您会发现它有助于创建伴随类，即*契约*类，其以一种系统性、自记录的方式明确指定您的架构布局。

契约类是用于定义 URI、表格和列名称的常数的容器。 契约类允许您跨同一软件包中的所有其他类使用相同的常数。您可以在一个位置更改列名称并使其在您整个代码中传播。

组织契约类的一种良好方法是将对于您的整个数据库而言是全局性的定义放入类的根级别。 然后为枚举其列的每个表格创建内部类。

> 注：通过实现 `BaseColumns` (https://developer.android.com/reference/android/provider/BaseColumns.html?hl=zh-cn) 接口，您的内部类可继承名为 `_ID` 的主键字段，某些 Android 类（比如光标适配器）将需要内部类拥有该字段。 这并非必需项，但可帮助您的数据库与 Android 框架协调工作。

例如，该代码段定义了单个表格的表格名称和列名称：

```
public final class FeedReaderContract {
    // To prevent someone from accidentally instantiating the contract class,
    // make the constructor private.
    private FeedReaderContract() {}

    /* Inner class that defines the table contents */
    public static class FeedEntry implements BaseColumns {
        public static final String TABLE_NAME = "entry";
        public static final String COLUMN_NAME_TITLE = "title";
        public static final String COLUMN_NAME_SUBTITLE = "subtitle";
    }
}
```

# 使用 SQL 辅助工具创建数据库

在您定义了数据库的外观后，您应实现创建和维护数据库和表格的方法。 这里有一些典型的表格创建和删除语句：

```
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_TITLE + TEXT_TYPE + COMMA_SEP +
    FeedEntry.COLUMN_NAME_SUBTITLE + TEXT_TYPE + " )";

private static final String SQL_DELETE_ENTRIES =
    "DROP TABLE IF EXISTS " + FeedEntry.TABLE_NAME;
```

就像您在设备的内部存储 (https://developer.android.com/guide/topics/data/data-storage.html?hl=zh-cn#filesInternal)中保存文件那样，Android 将您的数据库保存在私人磁盘空间，即关联的应用。 您的数据是安全的，因为在默认情况下，其他应用无法访问此区域。

`SQLiteOpenHelper` (https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=zh-cn) 类中有一组有用的 API。当您使用此类获取对您数据库的引用时，系统将只在需要之时而不是*应用启动*过程中执行可能长期运行的操作：创建和更新数据库。 您仅需调用 `getWritableDatabase()` (https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=zh-cn#getWritableDatabase()) 或 `getReadableDatabase()` (https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=zh-cn#getReadableDatabase()) 即可。

> 注：由于它们可能长期运行，因此请确保您在后台线程中调用 `getWritableDatabase()` (https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=zh-cn#getWritableDatabase()) 或 `getReadableDatabase()` (https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=zh-

> cn#getReadableDatabase())，比如使用 AsyncTask (https://developer.android.com/reference/android/os/AsyncTask.html?hl=zh-cn) 或 IntentService (https://developer.android.com/reference/android/app/IntentService.html?hl=zh-cn)。

要使用 SQLiteOpenHelper (https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=zh-cn)，请创建一个替换 onCreate() (https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=zh-cn#onCreate(android.database.sqlite.SQLiteDatabase))、onUpgrade() (https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=zh-cn#onUpgrade(android.database.sqlite.SQLiteDatabase, int, int)) 和onOpen() (https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=zh-cn#onOpen(android.database.sqlite.SQLiteDatabase)) 回调方法的子类。您可能还希望实现 onDowngrade() (https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=zh-cn#onDowngrade(android.database.sqlite.SQLiteDatabase, int, int))，但这并非必需操作。

例如，下面是一个使用如上所示一些命令的 SQLiteOpenHelper (https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=zh-cn) 的实现：

```
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);
    }
}
```

要访问您的数据库，请实例化 SQLiteOpenHelper (https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=zh-cn) 的子类：

```
FeedReaderDbHelper mDbHelper = new FeedReaderDbHelper(getContext());
```

# 将信息输入到数据库

通过将一个 ContentValues (https://developer.android.com/reference/android/content/ContentValues.html?hl=zh-cn) 对象传递至 insert() (https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=zh-cn#insert(java.lang.String, java.lang.String, android.content.ContentValues)) 方法将数据插入数据库：

```
// 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_TITLE, title);
values.put(FeedEntry.COLUMN_NAME_SUBTITLE, subtitle);

// Insert the new row, returning the primary key value of the new row
long newRowId = db.insert(FeedEntry.TABLE_NAME, null, values);
```

insert() (https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=zh-cn#insert(java.lang.String, java.lang.String, android.content.ContentValues)) 的第一个参数即为表格名称。

第二个参数将指示框架在 ContentValues (https://developer.android.com/reference/android/content/ContentValues.html?hl=zh-cn) 为空（即，您没有 put (https://developer.android.com/reference/android/content/ContentValues.html?hl=zh-cn#put(java.lang.String, byte[])) 任何值）时执行哪些操作。如果指定列名称，则框架将插入一行并将该列的值设置为 null。如果指定 null（就像此代码示例中一样），则框架不会在没有值时插入行。

# 从数据库读取信息

要从数据库中读取信息，请使用 query() (https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=zh-cn#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)) 方法，将其传递至选择条件和所需列。该方法结合 insert() (https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=zh-cn#insert(java.lang.String, java.lang.String, android.content.ContentValues)) 和 update() (https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=zh-cn#update(java.lang.String, android.content.ContentValues, java.lang.String, java.lang.String[])) 的

元素，除非列列表定义了您希望获取的数据，而不是希望插入的数据。 查询的结果将在 Cursor (https://developer.android.com/reference/android/database/Cursor.html?hl=zh-cn) 对象中返回给您。

```
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_SUBTITLE
    };

// Filter results WHERE "title" = 'My Title'
String selection = FeedEntry.COLUMN_NAME_TITLE + " = ?";
String[] selectionArgs = { "My Title" };

// How you want the results sorted in the resulting Cursor
String sortOrder =
    FeedEntry.COLUMN_NAME_SUBTITLE + " 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
    );
```

要查看游标中的某一行，请使用 Cursor (https://developer.android.com/reference/android/database/Cursor.html?hl=zh-cn) 移动方法之一，您必须在开始读取值之前始终调用这些方法。 一般情况下，您应通过调用 moveToFirst() (https://developer.android.com/reference/android/database/Cursor.html?hl=zh-cn#moveToFirst()) 开始，其将“读取位置”置于结果中的第一个条目中。 对于每一行，您可以通过调用 Cursor (https://developer.android.com/reference/android/database/Cursor.html?hl=zh-cn) 获取方法之一读取列的值，比如 getString() (https://developer.android.com/reference/android/database/Cursor.html?hl=zh-cn#getString(int)) 或 getLong() (https://developer.android.com/reference/android/database/Cursor.html?hl=zh-cn#getLong(int))。对于每种获取方法，您必须传递所需列的索引位置，您可以通过调用 getColumnIndex() (https://developer.android.com/reference/android/database/Cursor.html?hl=zh-cn#getColumnIndex(java.lang.String)) 或 getColumnIndexOrThrow() (https://developer.android.com/reference/android/database/Cursor.html?hl=zh-cn#getColumnIndexOrThrow(java.lang.String)) 获取。例如：

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

## 从数据库删除信息

要从表格中删除行，您需要提供识别行的选择条件。 数据库 API 提供了一种机制，用于创建防止 SQL 注入的选择条件。 该机制将选择规范划分为选择子句和选择参数。 该子句定义要查看的列，还允许您合并列测试。 参数是根据捆绑到子句的项进行测试的值。由于结果并未按照与常规 SQL 语句相同的方式进行处理，它不受 SQL 注入的影响。

```
// Define 'where' part of query.
String selection = FeedEntry.COLUMN_NAME_TITLE + " LIKE ?";
// Specify arguments in placeholder order.
String[] selectionArgs = { "MyTitle" };
// Issue SQL statement.
db.delete(FeedEntry.TABLE_NAME, selection, selectionArgs);
```

## 更新数据库

当您需要修改数据库值的子集时，请使用 `update()` (https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=zh-cn#update(java.lang.String, android.content.ContentValues, java.lang.String, java.lang.String[])) 方法。

更新表可将 `insert()` (https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=zh-cn#insert(java.lang.String, java.lang.String, android.content.ContentValues)) 的内容值语法与 `delete()` (https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=zh-cn#delete(java.lang.String, java.lang.String, java.lang.String[])) 的 `where` 语法相结合。

```
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 title
String selection = FeedEntry.COLUMN_NAME_TITLE + " LIKE ?";
String[] selectionArgs = { "MyTitle" };

int count = db.update(
    FeedReaderDbHelper.FeedEntry.TABLE_NAME,
    values,
    selection,
    selectionArgs);
```