8
Bases de datos SQLite

Android utiliza bases de datos SQLite

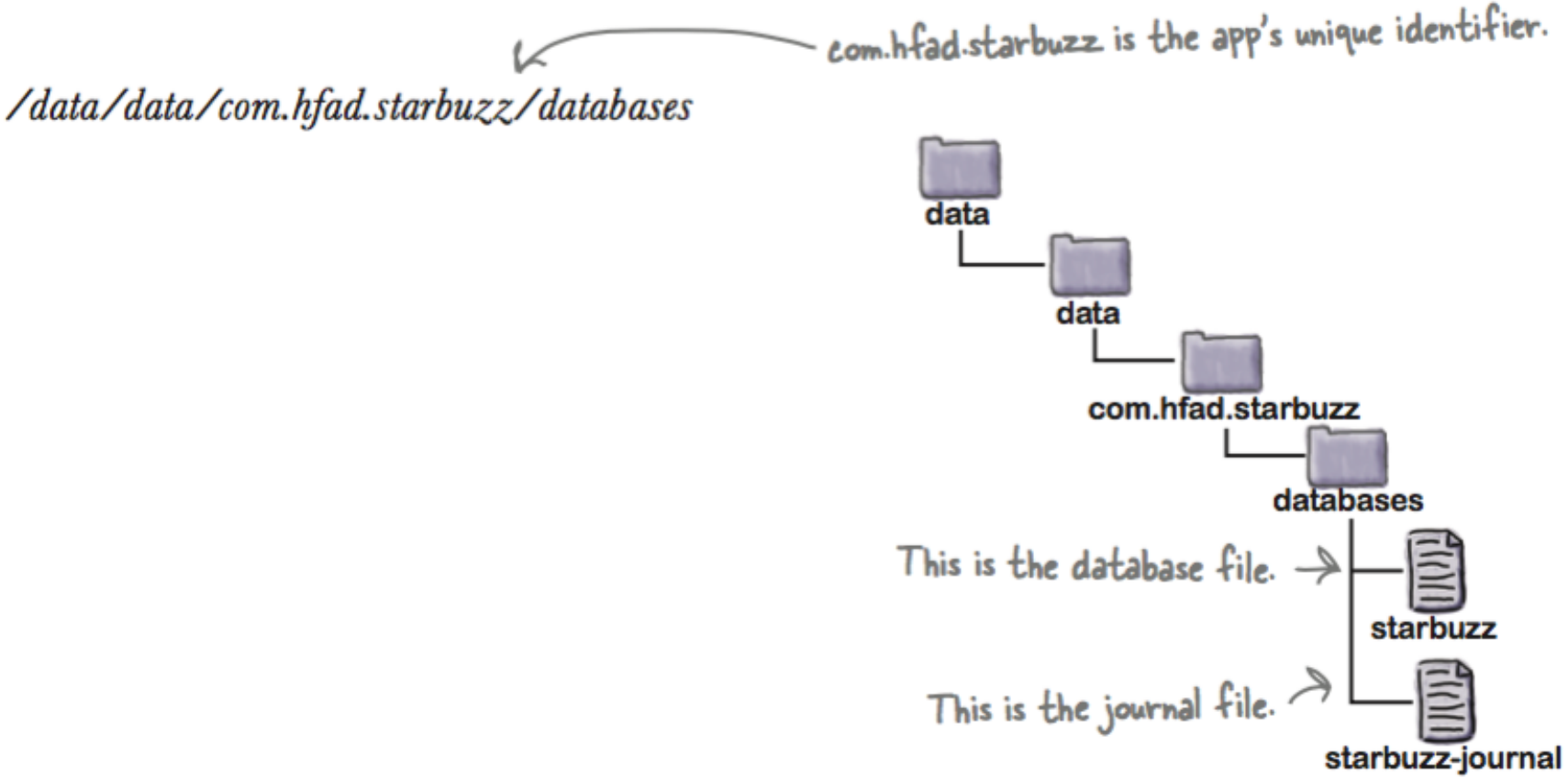
- **It's lightweight.**
  Most database systems need a special database server process in order to work. SQLite doesn't, a SQLite database is just a file. When you're not using the database, it doesn't use up any processor time. That's important on a mobile device, because we don't want to drain the battery.

- **It's optimized for a single user.**
  Our app is the only thing that will talk to the database, so we shouldn't have to identify ourselves with a username and password.

- **It's stable and fast.**
  SQLite databases are amazingly stable. They can handle database transactions, which means if you're updating several pieces of data and screw up, SQLite can roll the data back. Also, the code that reads and writes the data is written in optimized C code. Not only is it fast, but it also reduces the amount of processor power it needs.

If you plan on doing a lot of database heavy lifting in your apps, we suggest you do more background reading on SQLite and SQL.

Donde se almacena la base de datos ?



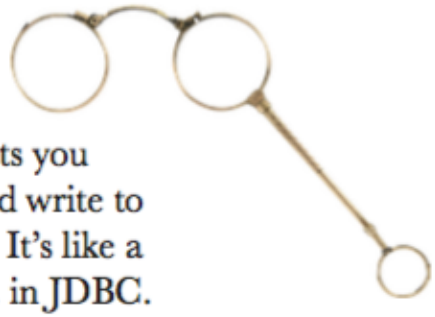
Android incluye clases SQLite

The SQLite Helper

You create a SQLite helper by extending the `SQLiteOpenHelper` class. This enables you to create and manage databases.

The SQLite Database

The `SQLiteDatabase` class gives you access to the database. It's like a `SQLConnection` in JDBC.



Cursors

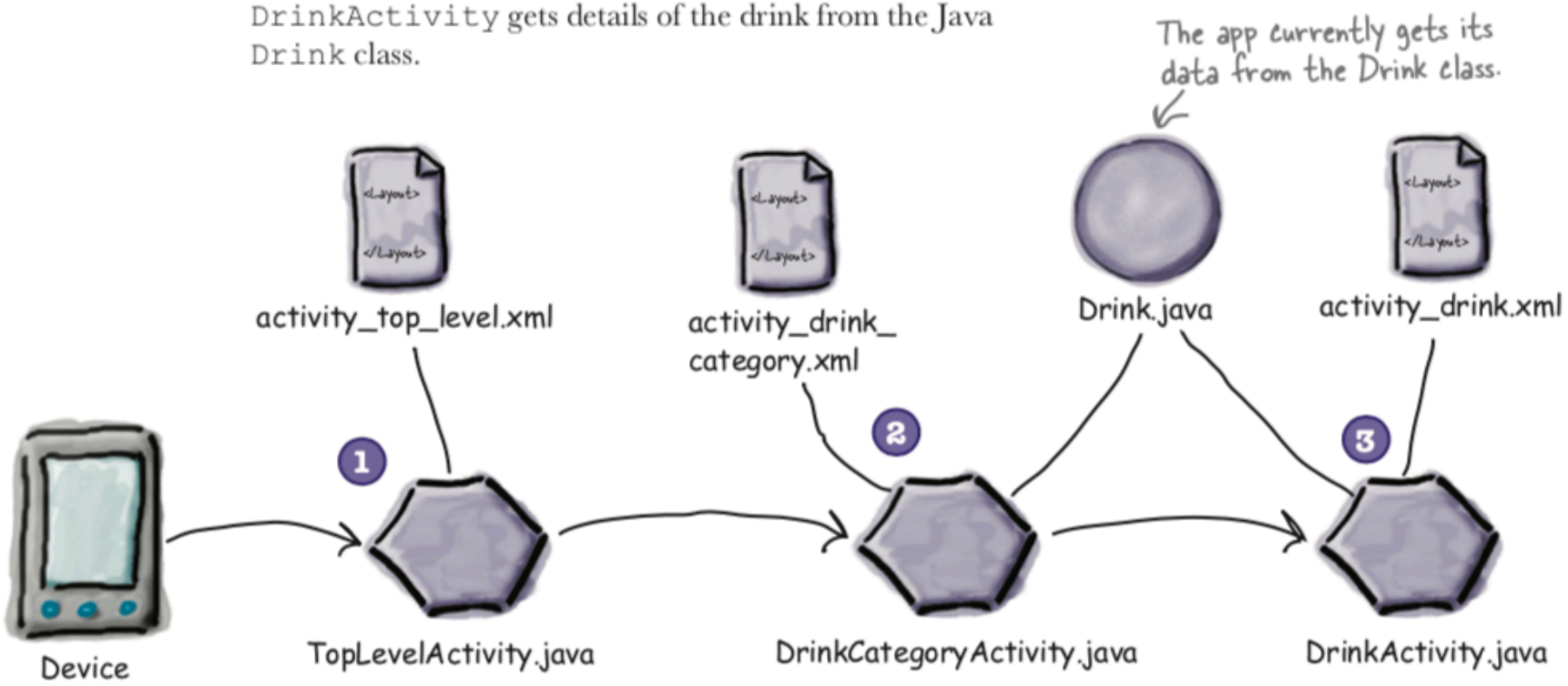
A `Cursor` lets you read from and write to the database. It's like a `ResultSet` in JDBC.

Estructura actual de la aplicación Starbuzz

1. **TopLevelActivity displays a list of options: Drinks, Food, and Stores.**

2. **When the user clicks on the Drinks option, it launches DrinkCategoryActivity.**
This activity displays a list of drinks that it gets from the Java `Drink` class.

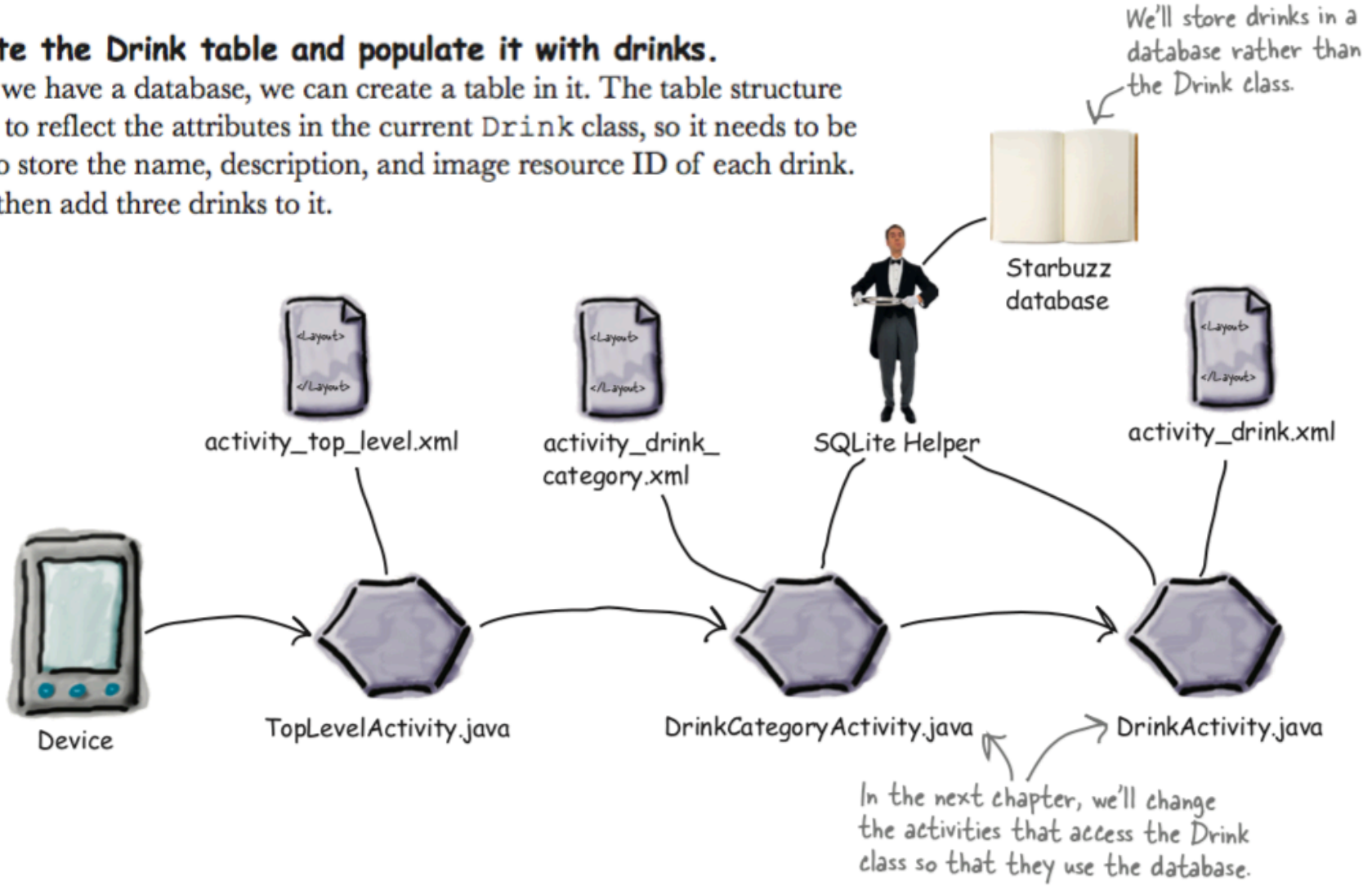
3. **When the user clicks on a drink, its details get displayed in DrinkActivity.**
`DrinkActivity` gets details of the drink from the Java `Drink` class.



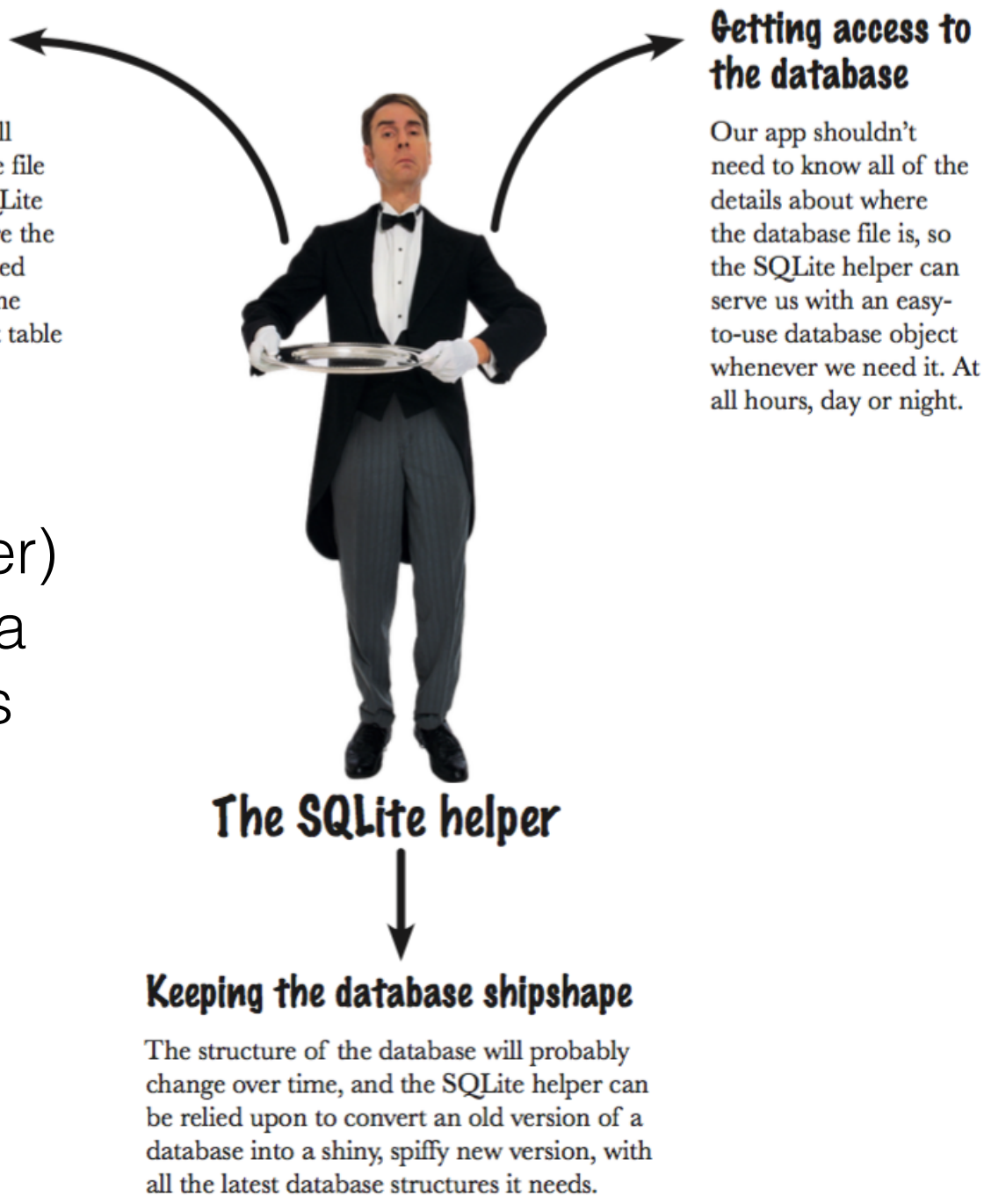
Modificando la aplicación para utilizar una base de datos

1 **Create the database.**
Before we can do anything else, we need to get the SQLite helper to create version 1 (the first version) of our Starbuzz database.

2 **Create the Drink table and populate it with drinks.**
Once we have a database, we can create a table in it. The table structure needs to reflect the attributes in the current `Drink` class, so it needs to be able to store the name, description, and image resource ID of each drink. We'll then add three drinks to it.



El asistente (helper) SQLite administra la base de datos

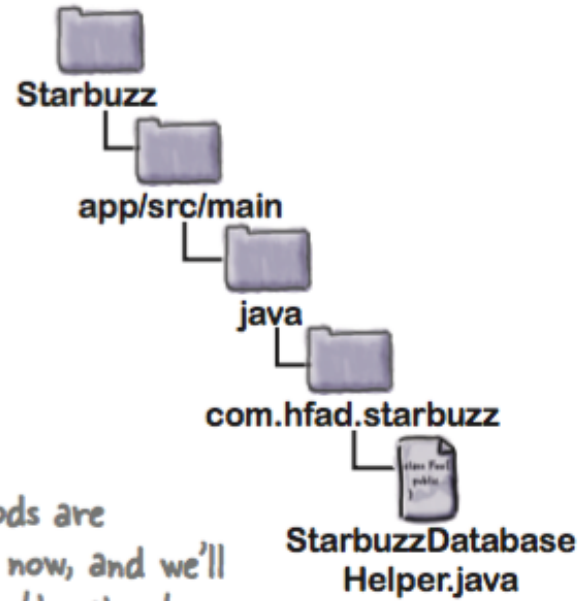


Creación del asistente SQLite

```
package com.hfad.starbuzz;

import android.database.sqlite.SQLiteOpenHelper;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;

class StarbuzzDatabaseHelper extends SQLiteOpenHelper {

    StarbuzzDatabaseHelper(Context context) {
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
}
```

SQLite helpers must extend the SQLiteOpenHelper class.

The onCreate() and onUpgrade() methods are mandatory. We've left them empty for now, and we'll look at them in more detail throughout the chapter.



Especificación de la base de datos

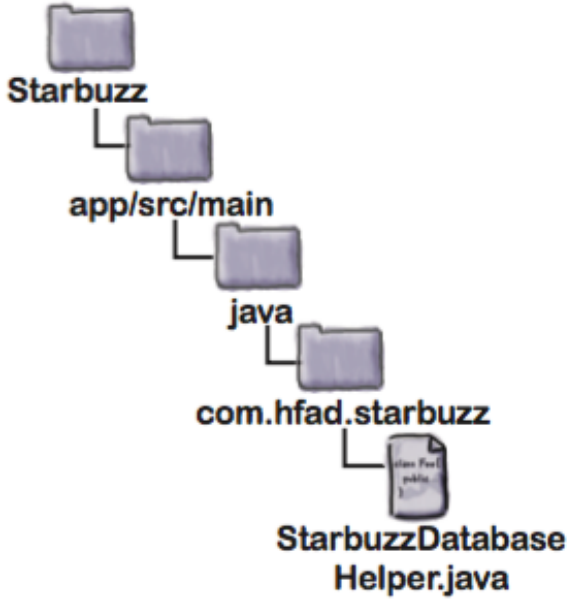
```
...
class StarbuzzDatabaseHelper extends SQLiteOpenHelper {

    private static final String DB_NAME = "starbuzz"; // the name of our database
    private static final int DB_VERSION = 1; // the version of the database

    StarbuzzDatabaseHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }
...
}
```

We're calling the constructor of the SQLiteOpenHelper superclass, and passing it the database name and version.

This parameter is an advanced feature relating to cursors. We're covering cursors in the next chapter.

SQLite database

Name: "starbuzz"
Version: 1



Adentro de una base de datos SQLite

The columns in the table are _id, NAME, DESCRIPTION, and IMAGE_RESOURCE_ID. The Drink class contained similarly named attributes.

_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID
1	“Latte”	"Espresso and steamed milk"	54543543
2	“Cappuccino”	"Espresso, hot milk and steamed-milk foam"	654334453
3	“Filter”	"Our best drip coffee"	44324234

INTEGER	Any integer type
TEXT	Any character type
REAL	Any floating-point number
NUMERIC	Booleans, dates, and date-times
BLOB	Binary Large Object

Las tablas se crean utilizando SQL

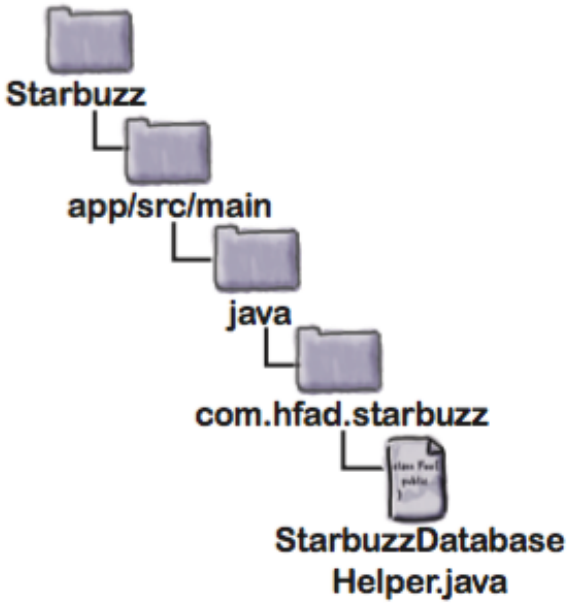
The _id column is the primary key.

```
CREATE TABLE DRINK (_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    NAME TEXT,
                    DESCRIPTION TEXT,
                    IMAGE_RESOURCE_ID INTEGER)
```

The table name

These are the table columns.

```
@Override
public void onCreate(SQLiteDatabase db){
    db.execSQL("CREATE TABLE DRINK ("
               + "_id INTEGER PRIMARY KEY AUTOINCREMENT, "
               + "NAME TEXT, "
               + "DESCRIPTION TEXT, "
               + "IMAGE_RESOURCE_ID INTEGER);");
}
```



Insertando datos

```
ContentValues drinkValues = new ContentValues();
drinkValues.put("NAME", "Latte");
drinkValues.put("DESCRIPTION", "Espresso and steamed milk");
drinkValues.put("IMAGE_RESOURCE_ID", R.drawable.latte);
db.insert("DRINK", null, drinkValues);
```

This will put the value "Espresso and steamed milk" in the DESCRIPTION column.

You need a separate call to the put() method for each value you want to enter.

_id	NAME	DESCRIPTION	IMAGE_RESOURCE_ID
1	"Latte"	"Espresso and steamed milk"	54543543

A shiny new record gets inserted into the table.

Insertando registros múltiples

```
private static void insertDrink(SQLiteDatabase db,
                                String name,
                                String description,
                                int resourceId) {
    ContentValues drinkValues = new ContentValues();
    drinkValues.put("NAME", name);
    drinkValues.put("DESCRIPTION", description);
    drinkValues.put("IMAGE_RESOURCE_ID", resourceId);
    db.insert("DRINK", null, drinkValues);
}
```

This is the database we want to add records to.

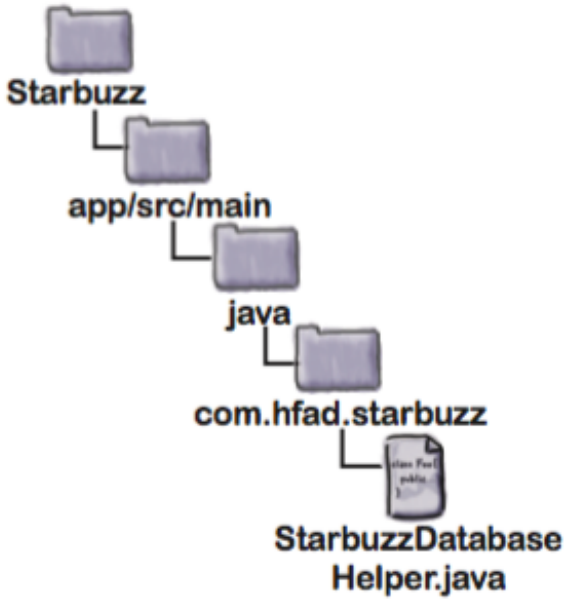
We're passing the data to the method as parameters.

Construct a ContentValues object with the data.

Then insert the data.

```
insertDrink(db, "Latte", "Espresso and steamed milk", R.drawable.latte);
insertDrink(db, "Cappuccino", "Espresso, hot milk and steamed-milk foam",
            R.drawable.cappuccino);
insertDrink(db, "Filter", "Our best drip coffee", R.drawable.filter);
```

El código de StarbuzzDatabaseHelper




```
package com.hfad.starbuzz;

import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

class StarbuzzDatabaseHelper extends SQLiteOpenHelper{

    private static final String DB_NAME = "starbuzz"; // the name of our database
    private static final int DB_VERSION = 1; // the version of the database

    StarbuzzDatabaseHelper(Context context){
        super(context, DB_NAME, null, DB_VERSION);
    }
```

Say what the database name and version is. It's the first version of the database, so the version should be 1.

El código de StarbuzzDatabaseHelper

onCreate() gets called when the database first gets created, so we're using it to create the table and insert data.

```
@Override
public void onCreate(SQLiteDatabase db){
    db.execSQL("CREATE TABLE DRINK (_id INTEGER PRIMARY KEY AUTOINCREMENT, "
               + "NAME TEXT, "
               + "DESCRIPTION TEXT, "
               + "IMAGE_RESOURCE_ID INTEGER);");
    insertDrink(db, "Latte", "Espresso and steamed milk", R.drawable.latte);
    insertDrink(db, "Cappuccino", "Espresso, hot milk and steamed-milk foam",
                R.drawable.cappuccino);
    insertDrink(db, "Filter", "Our best drip coffee", R.drawable.filter);
}
```

Create the DRINK table.

Insert each drink in a separate row.

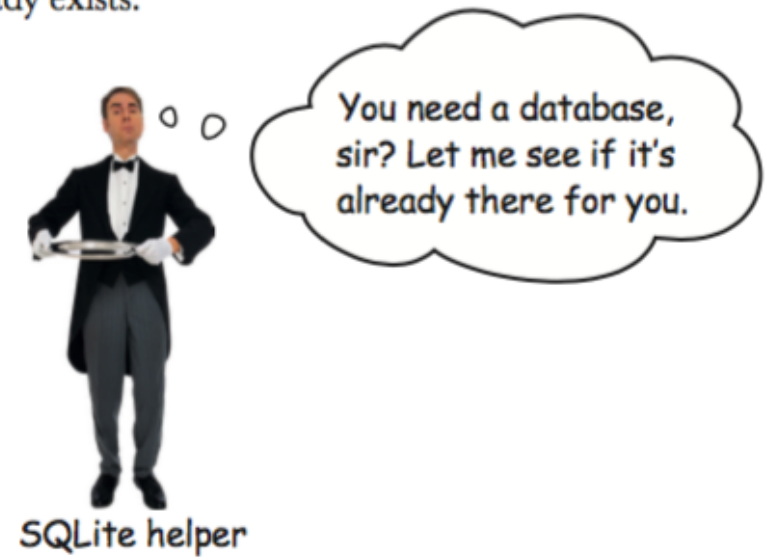
onUpgrade() gets called when the database needs to be upgraded. We'll look at this next.

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
}

private static void insertDrink(SQLiteDatabase db, String name,
                                String description, int resourceId) {
    ContentValues drinkValues = new ContentValues();
    drinkValues.put("NAME", name);
    drinkValues.put("DESCRIPTION", description);
    drinkValues.put("IMAGE_RESOURCE_ID", resourceId);
    db.insert("DRINK", null, drinkValues);
}
}
```

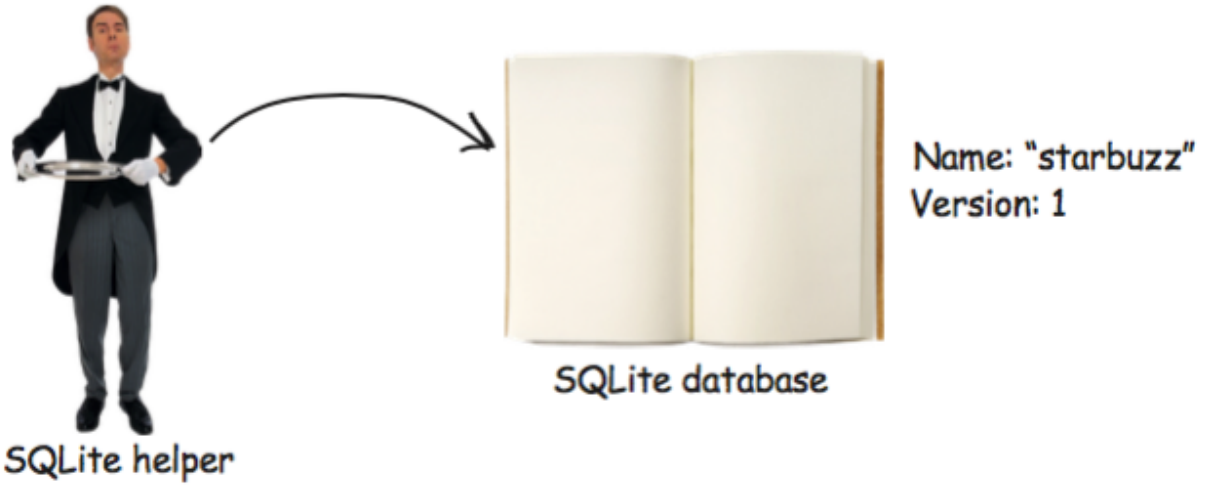
We need to insert several drinks, so we created a separate method to do this.

Lo que hace el asistente SQLite

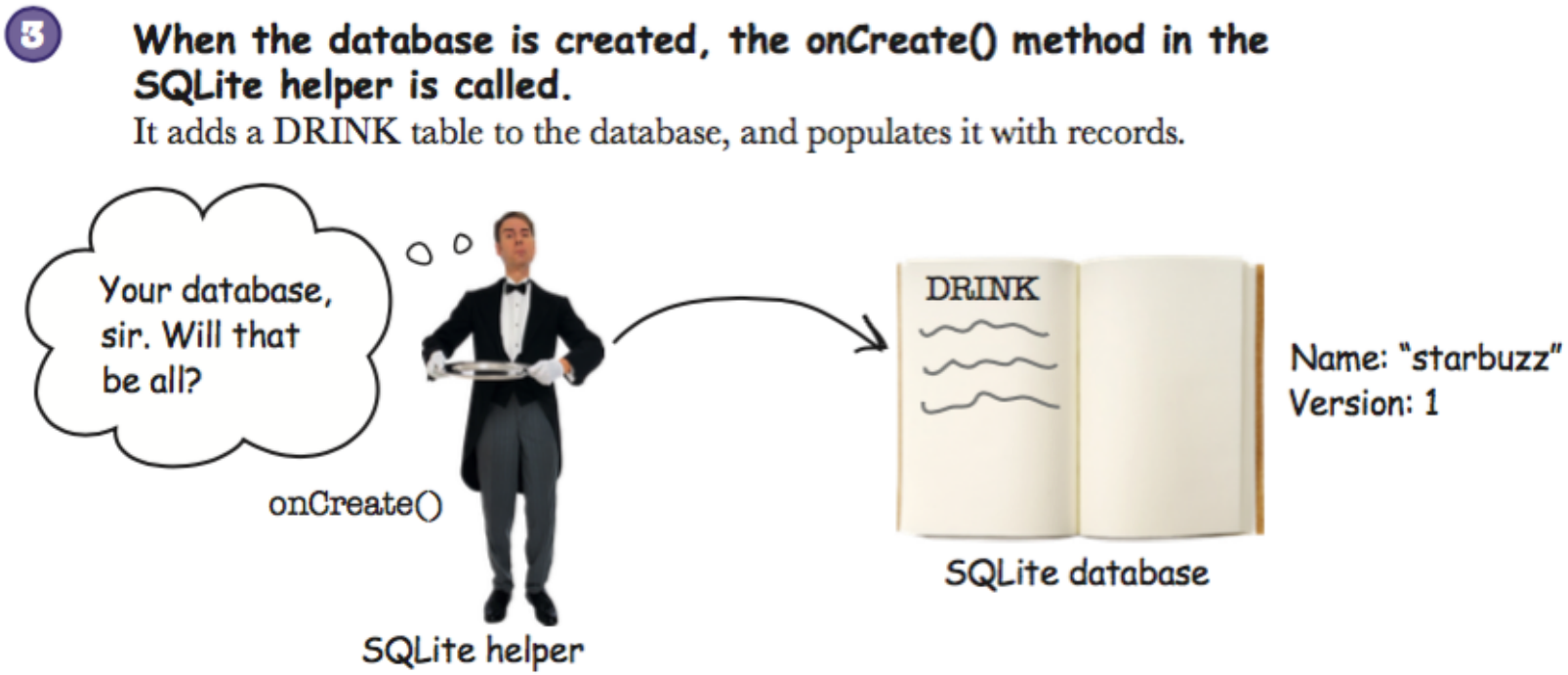
1 **The user installs the app and launches it.**
When the app needs to access the database, the SQLite helper checks to see if the database already exists.



2 **If the database doesn't exist, it gets created.**
It's given the name and version number specified in the SQLite helper.



Lo que hace el asistente SQLite



Las bases de datos SQLite tienen un número de versión que sirve para actualizarla

```
...
    private static final String DB_NAME = "starbuzz";
    private static final int DB_VERSION = 1;

    StarbuzzDatabaseHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }
...
```

```
...
    private static final int DB_VERSION = 2;
...
```

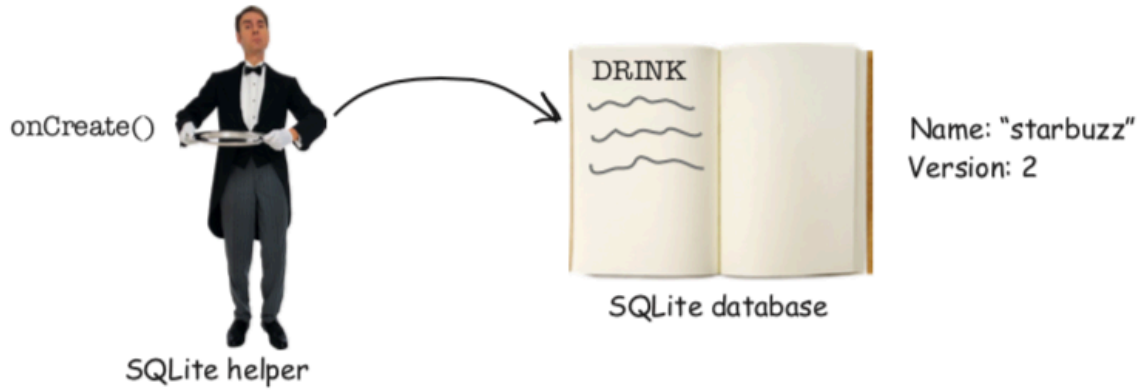
Here we're increasing the version number, so the database will get upgraded.

Actualizando la base de datos

Scenario 1: A first-time user installs the app

1. **The first time the user runs the app, the database doesn't exist, so the SQLite helper creates it.**
The SQLite helper gives the database the name and version number specified in the SQLite helper code.

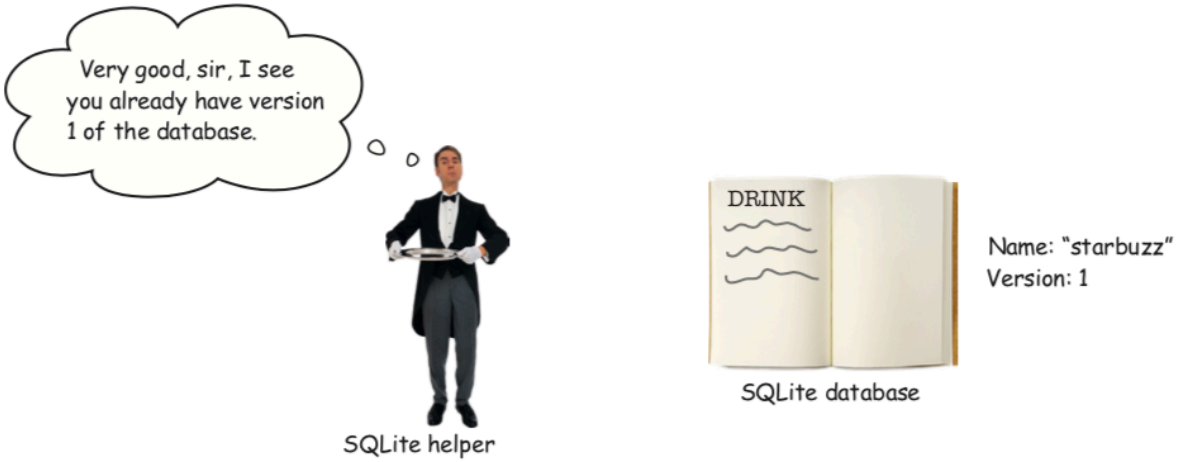
2. **When the database is created, the onCreate() method in the SQLite helper is called.**
The `onCreate()` method includes code to populate the database.



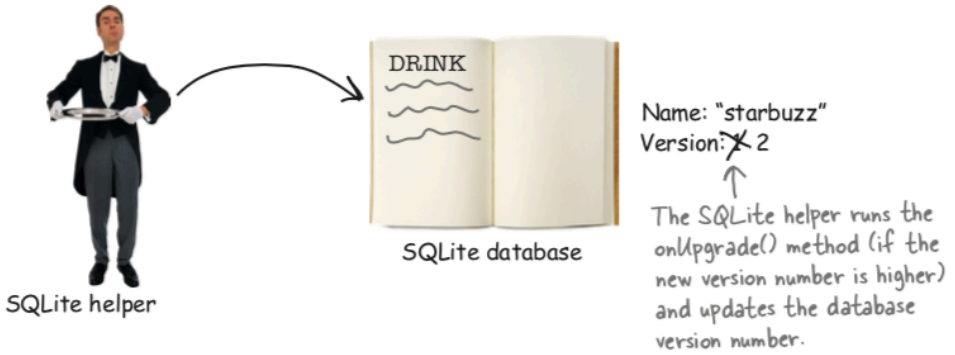
Actualizando la base de datos

Scenario 2: an existing user installs the new version

1 **When the user runs the new version of the app, the database helper checks whether the database exists.**
If the database already exists, the SQLite helper doesn't recreate it.



2 **The SQLite helper checks the version number of the existing database against the version number in the SQLite helper code.**
If the SQLite helper version number is higher than the database version, it calls the `onUpgrade()` method. If the SQLite helper version number is lower than the database version, it calls the `onDowngrade()` method. It then changes the database version number to reflect the version number in the SQLite helper code.



Actualizando una base de datos existente

- **Change the database records.**
  Earlier on in the chapter, you saw how to insert, update, or delete records in your database using the `SQLiteDatabase` `insert()`, `update()`, and `delete()` methods. You may add more records when you upgrade the database, or change or remove the records that are already there.

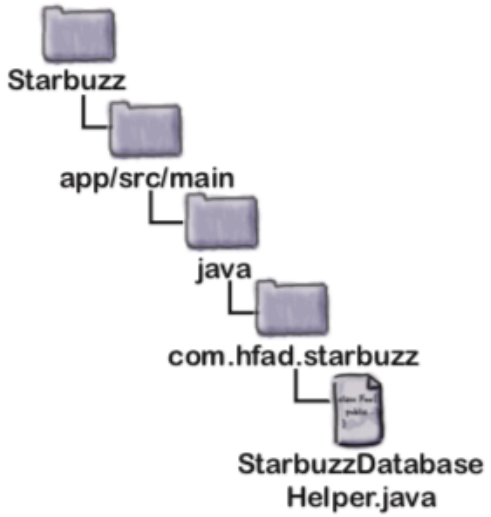
- **Change the database structure.**
  You've already seen how you can create tables in the database. You may also want to add columns to existing tables, rename tables, or remove tables completely.

Agregando columnas a la tabla

```
ALTER TABLE DRINK
ADD COLUMN FAVORITE NUMERIC
```

The table name

The column you want to add




```
package com.hfad.starbuzz;

import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

class StarbuzzDatabaseHelper extends SQLiteOpenHelper{

    private static final String DB_NAME = "starbuzz"; // the name of our database
    private static final int DB_VERSION = 2; // the version of the database

    StarbuzzDatabaseHelper(Context context){
        super(context, DB_NAME, null, DB_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db){
        updateMyDatabase(db, 0, DB_VERSION);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        updateMyDatabase(db, oldVersion, newVersion);
    }
```

Changing the version number to a larger integer enables the SQLite helper to spot that you want to upgrade the database.

The code to create any database tables is in the updateMyDatabase() method.

The code to upgrade the database is in our updateMyDatabase() method.

Actualizando una base de datos existente

```
    private static void insertDrink(SQLiteDatabase db, String name,
                                    String description, int resourceId) {
        ContentValues drinkValues = new ContentValues();
        drinkValues.put("NAME", name);
        drinkValues.put("DESCRIPTION", description);
        drinkValues.put("IMAGE_RESOURCE_ID", resourceId);
        db.insert("DRINK", null, drinkValues);
    }

    private void updateMyDatabase(SQLiteDatabase db, int oldVersion, int newVersion) {
        if (oldVersion < 1) {
            db.execSQL("CREATE TABLE DRINK (_id INTEGER PRIMARY KEY AUTOINCREMENT, "
                       + "NAME TEXT, "
                       + "DESCRIPTION TEXT, "
                       + "IMAGE_RESOURCE_ID INTEGER);");
            insertDrink(db, "Latte", "Espresso and steamed milk", R.drawable.latte);
            insertDrink(db, "Cappuccino", "Espresso, hot milk and steamed-milk foam",
                        R.drawable.cappuccino);
            insertDrink(db, "Filter", "Our best drip coffee", R.drawable.filter);
        }
        if (oldVersion < 2) {
            db.execSQL("ALTER TABLE DRINK ADD COLUMN FAVORITE NUMERIC;");
        }
    }
}
```

Add a numeric FAVORITE column to the DRINK table.

Actualizando una base de datos existente

BULLET POINTS

- Android uses SQLite as its backend database to persist data.
- The `SQLiteDatabase` class gives you access to the SQLite database.
- A SQLite helper lets you create and manage SQLite databases. You create a SQLite helper by extending the `SQLiteOpenHelper` class.
- You must implement the `SQLiteOpenHelper onCreate()` and `onUpgrade()` methods.
- The database gets created the first time it needs to be accessed. You need to give the database a name and version number, starting at 1. If you don't give the database a name, it will just get created in memory.
- The `onCreate()` method gets called when the database first gets created.
- The `onUpgrade()` method gets called when the database needs to be upgraded.
- Execute SQL using the `SQLiteDatabase execSQL(String)` method.
- Use the SQL `ALTER TABLE` command to change an existing table. Use `RENAME TO` to rename the table, and `ADD COLUMN` to add a column.
- Use the SQL `DROP TABLE` command to delete a table.
- Add records to tables using the `insert()` method.
- Update records using the `update()` method.
- Remove records from tables using the `delete()` method.