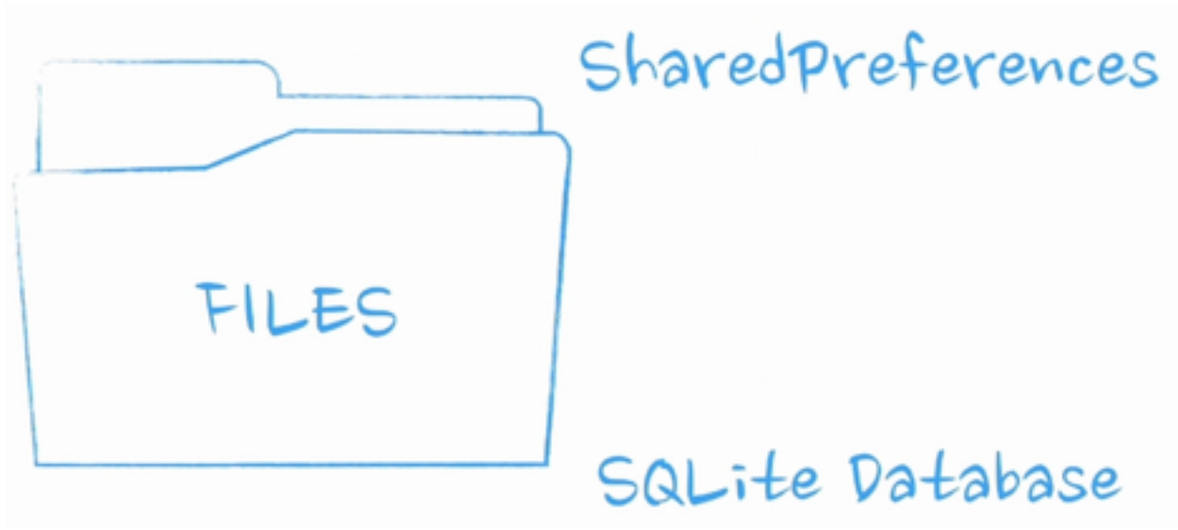
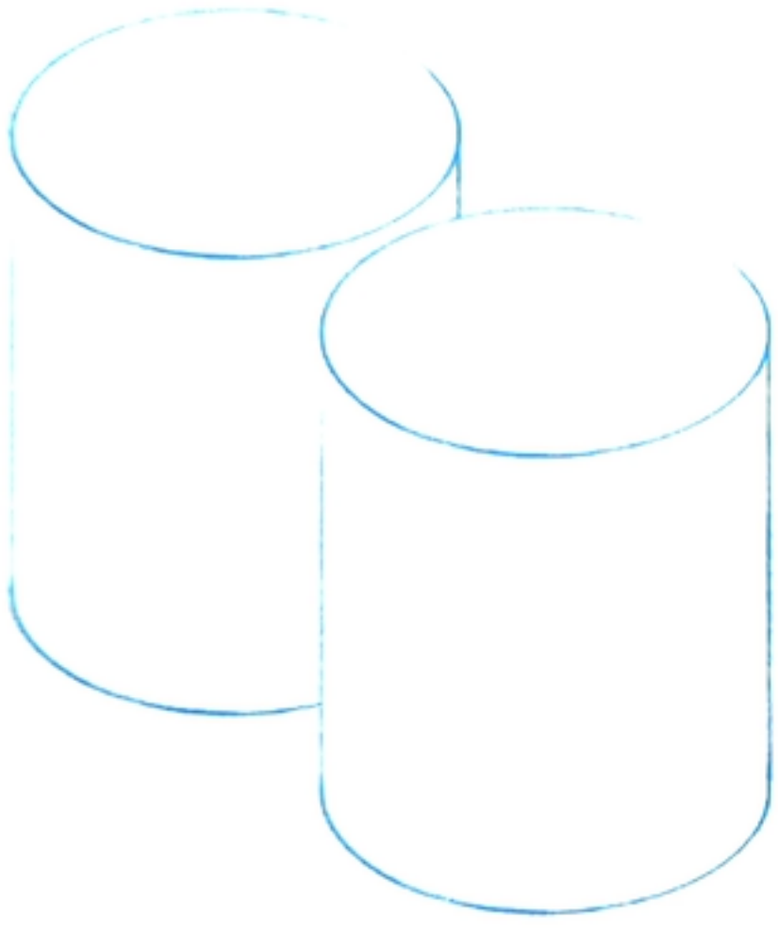
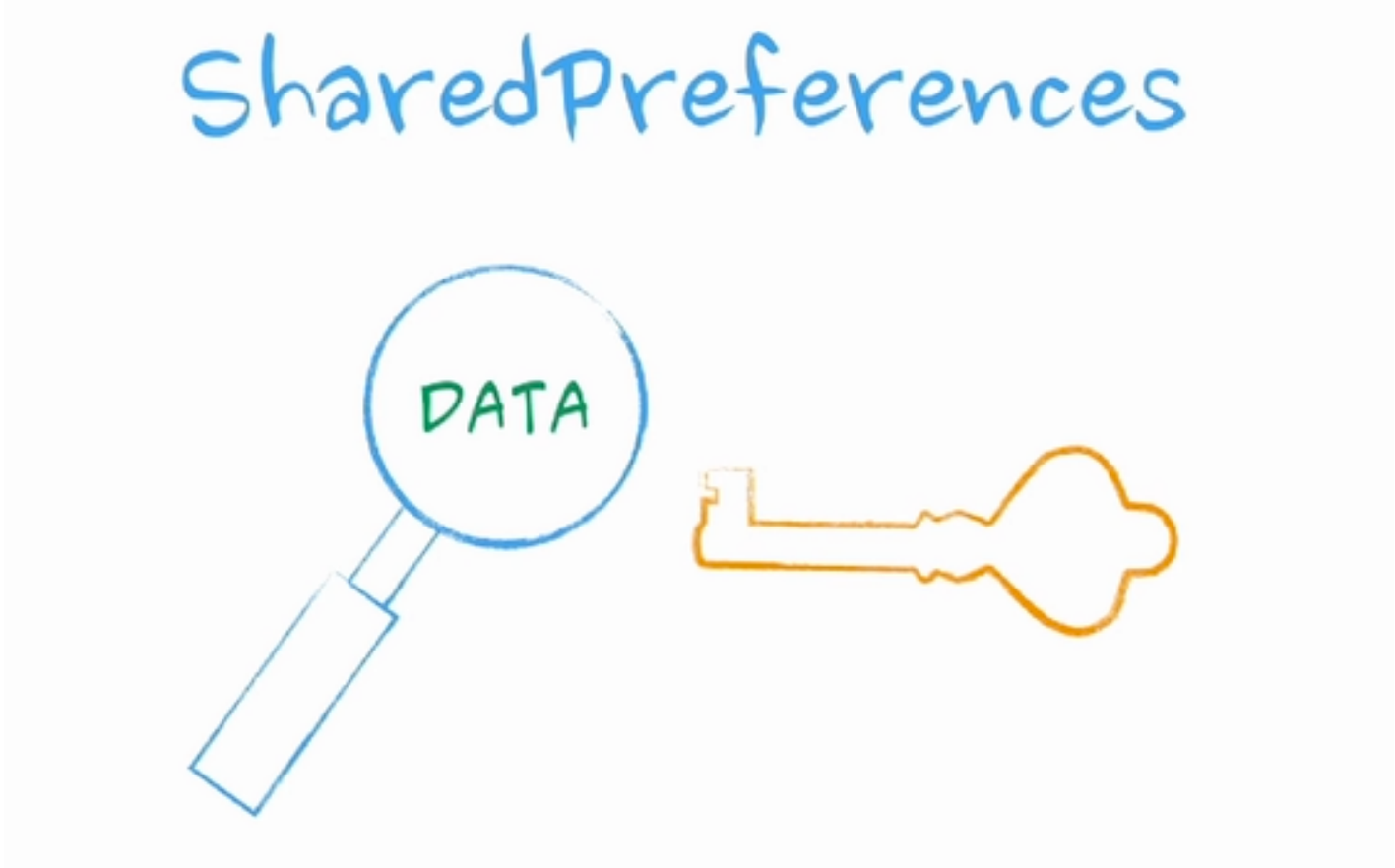
Android Persistence

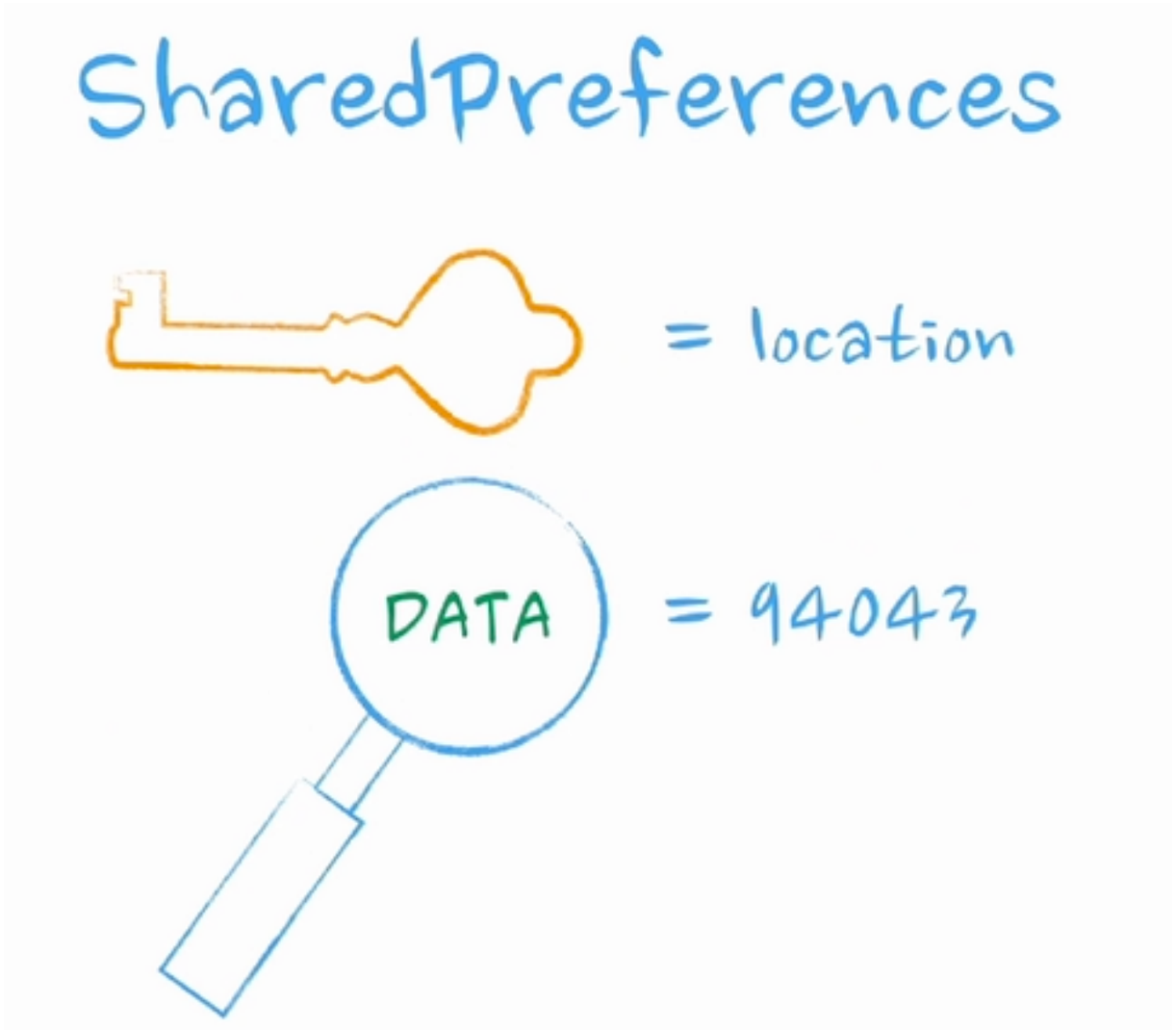
Shared Preferences, Files and Databases





STORING DATA IN ANDROID


SharedPreferences
DATA


SharedPreferences
= location
DATA
= 94043


SQLite Database


SQLite Database

SQLite Database

_ID	DATE	MIN	MAX	HUMIDITY	PRESSURE
1	20140625	16	20	0	1029
2	20140626	17	21	0	1031
3	20140627	18	22	0	1055
4	20140628	18	21	10	1070

SQLite Database

```
SELECT * FROM weather
WHERE date = 20140626
```

SQLite Database

_ID	DATE	MIN	MAX	HUMIDITY	PRESSURE
2	20140626	17	21	0	1031

SELECT * FROM weather WHERE date = 20140626

SQLite Database

```
SELECT * FROM weather
WHERE date > 20140625
AND date < 20140628
```

SQLite Database

_ID	DATE	MIN	MAX	HUMIDITY	PRESSURE
2	20140626	17	21	0	1031
3	20140627	18	22	0	1055

```
SELECT * FROM weather WHERE date > 20140625
AND date < 20140628
```

Android Helper Functions

CursorLoader

CursorAdapter

CursorJoiner

ContentProvider

ContentValues

DatabaseUtils

SQLiteCursor

SQLiteDatabase

SQLiteOpenHelper

SQLiteQueryBuilder

SQLiteQuery

SQLiteStatement

Play

Database Contract

Vertrag zwischen Datenmodell und View, wie die Daten gespeichert werden. Beinhaltet die anzuzeigenden Spalten

```java
package android.provider;

public interface BaseColumns
{
    /**
     * The unique ID for a row.
     * <P>Type: INTEGER (long)</P>
     */
    public static final String _ID = "_id";

    /**
     * The count of rows in a directory.
     * <P>Type: INTEGER</P>
     */
    public static final String _COUNT = "_count";
}
```

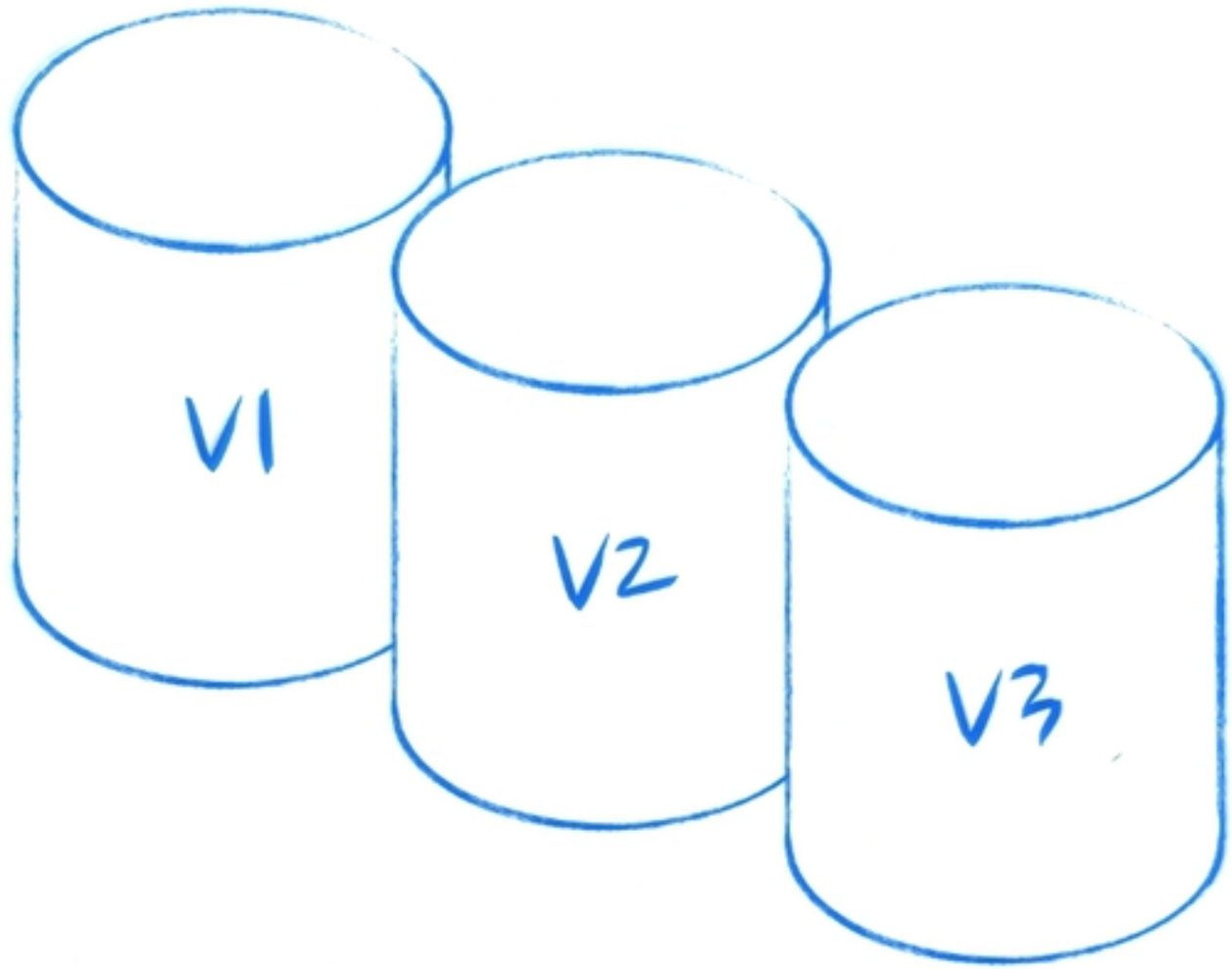
Mit dem BaseColumns - Interface kann einer Tabelle eine PrimaryKey-Spalte zugewiesen werden

https://gist.github.com/ThomasStuetz/b8a6560e7d761151cb12#file-personcontract-java

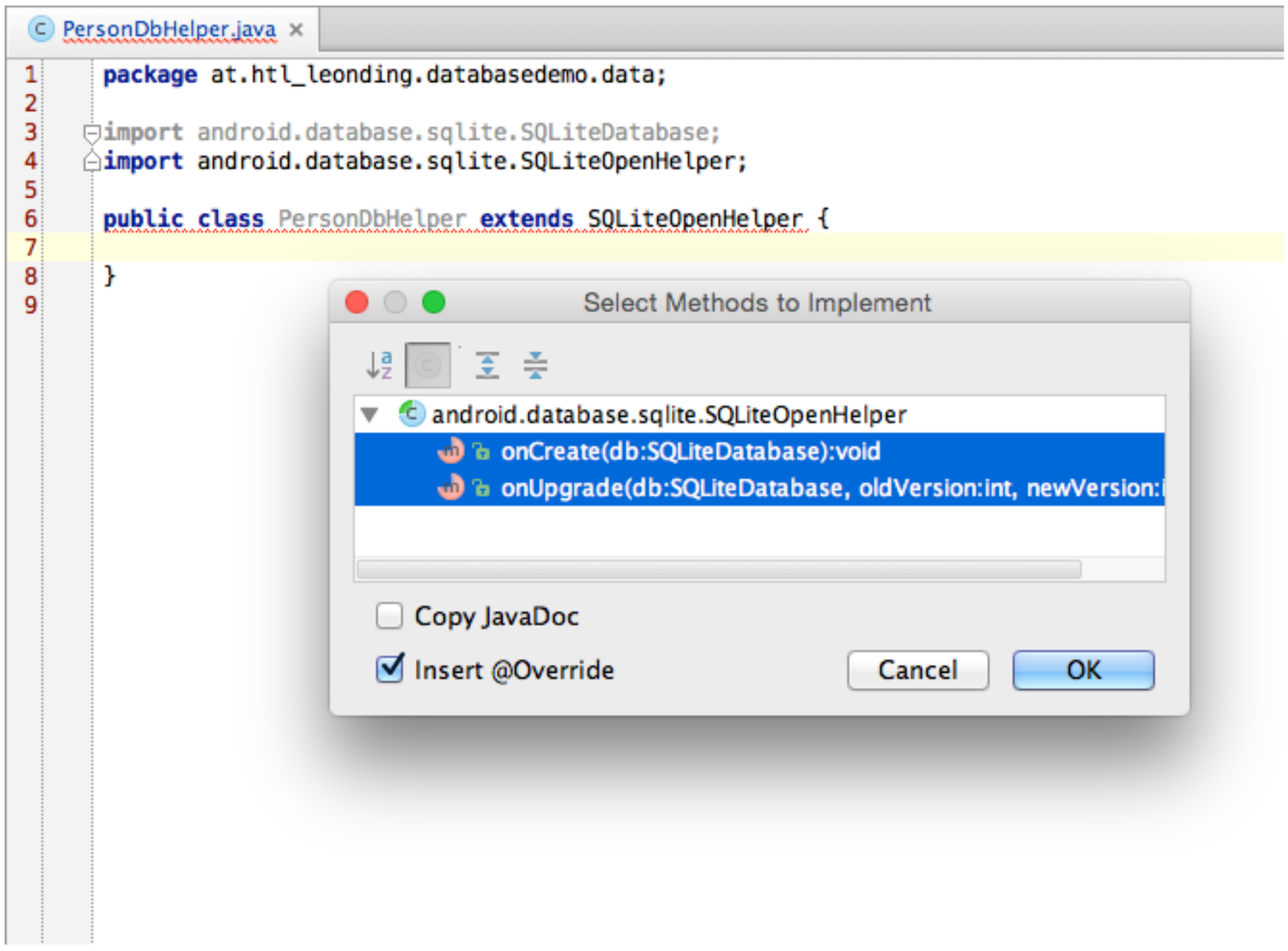
```java
public class PersonContract {

    public static final String PATH_PERSON = "person";
    public static final String PATH_LOCATION = "location";


    public static final class LocationEntry implements BaseColumns {

        public static final String TABLE_NAME = "location";

        public static final String COLUMN_CITY_NAME = "city_name";
        public static final String COLUMN_ZIP_CODE = "zip_code";
        public static final String COLUMN_OTHER_ZIP_CODES = "other_zip_codes";
    }

    public static final class PersonEntry implements BaseColumns {

        public static final String TABLE_NAME = "person";

        // foreign key to the location table
        public static final String COLUMN_LOC_KEY = "location_id";
        public static final String COLUMN_NAME = "name";
        public static final String COLUMN_DATETEXT = "dob";
    }
}
```

Dieser Klasse werden später (bei Erstellung der Content Provider) noch Infos zur Content Authority hinzugefügt.

Database Versioning



Implement Methods ^i



Override Methods ^O

PersonDbHelper.java

```
1  package at.htl_leonding.databasedemo.data;
2
3  import android.database.sqlite.SQLiteDatabase;
4  import android.database.sqlite.SQLiteOpenHelper;
5
6  public class PersonDbHelper extends SQLiteOpenHelper {
7
8
9
10     @Override
11     public void o
12
13     }
14
15     @Override
16     public void o
17
18     }
19 }
20
```

Select Methods to Override/Implement

- android.database.sqlite.SQLiteOpenHelper
  - SQLiteOpenHelper(context:Context, name:String, factory:C
  - SQLiteOpenHelper(context:Context, name:String, factory:C
  - getDatabaseName():String
  - setWriteAheadLoggingEnabled(enabled:boolean):void
  - getWritableDatabase():SQLiteDatabase
  - getReadableDatabase():SQLiteDatabase

Copy JavaDoc

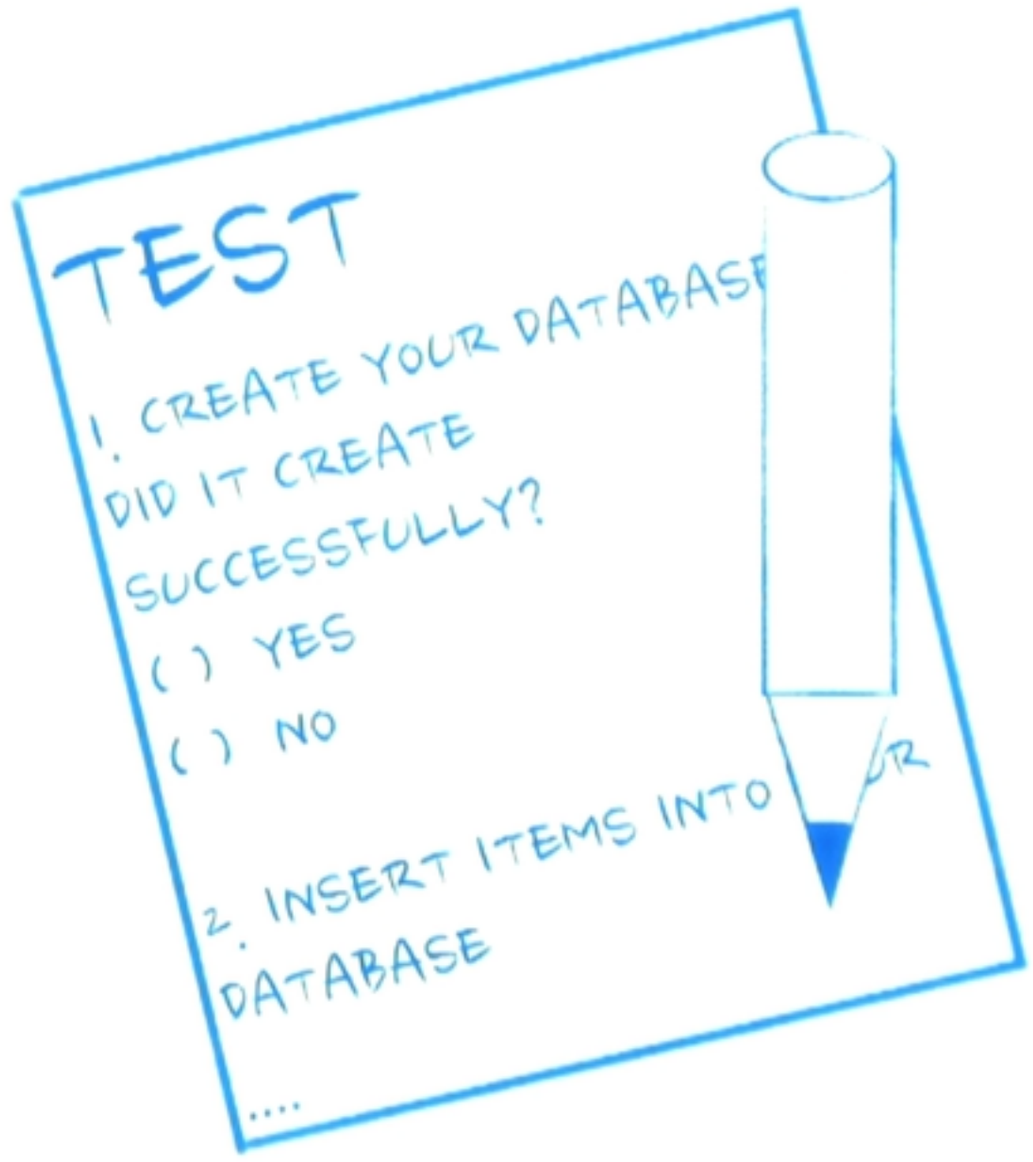
Insert @Override

Cancel OK

Eigentlich überschreiben wir hier den Konstruktor

```java
public class PersonDbHelper extends SQLiteOpenHelper {

    private static final int DATABASE_VERSION = 1;
    // DATABASE_NAME ist public, da er in Unit-Tests verwendet wird
    public static final String DATABASE_NAME = "person.db";

    public PersonDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {

        final String SQL_CREATE_LOCATION_TABLE =
                "CREATE TABLE " + LocationEntry.TABLE_NAME + " (" +
                        LocationEntry._ID + " INTEGER PRIMARY KEY, " +
                        LocationEntry.COLUMN_CITY_NAME + " TEXT NOT NULL, " +
                        LocationEntry.COLUMN_ZIP_CODE + " INTEGER NOT NULL, " +
                        LocationEntry.COLUMN_OTHER_ZIP_CODES + "TEXT" +
                        " );";

        final String SQL_CREATE_PERSON_TABLE =
                "CREATE TABLE " + PersonEntry.TABLE_NAME + " (" +
                        PersonEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                        PersonEntry.COLUMN_LOC_KEY + " INTEGER NOT NULL, " +
                        PersonEntry.COLUMN_NAME + " TEXT NOT NULL, " +
                        PersonEntry.COLUMN_DATETEXT + " TEXT NOT NULL, " +
                        " FOREIGN KEY (" + PersonEntry.COLUMN_LOC_KEY + ") REFERENCES " +
                        LocationEntry.TABLE_NAME + " (" + LocationEntry._ID + "));";

        sqLiteDatabase.execSQL(SQL_CREATE_LOCATION_TABLE);
        sqLiteDatabase.execSQL(SQL_CREATE_PERSON_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int newVersion) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + LocationEntry.TABLE_NAME);
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + PersonEntry.TABLE_NAME);
        onCreate(sqLiteDatabase);
    }
}
```

Database Testing



TestSuite

```java
package at.htl_leonding.databasedemo;

import android.test.suitebuilder.TestSuiteBuilder;

import junit.framework.Test;
import junit.framework.TestSuite;

public class FullTestSuite extends TestSuite {
    public static Test suite() {
        return new TestSuiteBuilder(FullTestSuite.class)
                .includeAllPackagesUnderHere().build();
    }

    public FullTestSuite() {
        super();
    }
}
```

https://gist.github.com/ThomasStuetz/96126aa21164ffb50424#file-fulltestsuite-java

1. Test

```java
package at.htl_leonding.databasedemo;

import android.database.sqlite.SQLiteDatabase;
import android.test.AndroidTestCase;

import at.htl_leonding.databasedemo.data.PersonDbHelper;

public class TestDb extends AndroidTestCase {

    public void testCreateDb() throws Throwable {
        mContext.deleteDatabase(PersonDbHelper.DATABASE_NAME);
        SQLiteDatabase db = new PersonDbHelper(
                this.mContext).getWritableDatabase();
        assertEquals(true, db.isOpen());
        db.close();
    }
}
```

https://gist.github.com/ThomasStuetz/a90f8f7b46279387987d#file-testdb-java

2. Test

```
public void testInsertReadDb() {
    String testCityName = "Leonding";
    int testZipCode = 4060;

    PersonDbHelper dbHelper = new PersonDbHelper(mContext);
    SQLiteDatabase db = dbHelper.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(LocationEntry.COLUMN_CITY_NAME, testCityName);
    values.put(LocationEntry.COLUMN_ZIP_CODE, testZipCode);

    long locationRowId;
    locationRowId = db.insert(LocationEntry.TABLE_NAME, null, values);

    assertTrue(locationRowId != -1);
    Log.d(LOG_TAG, "New row id: " + locationRowId);

    String[] columns = {
            LocationEntry._ID,
            LocationEntry.COLUMN_CITY_NAME,
            LocationEntry.COLUMN_ZIP_CODE
    };
```

https://gist.github.com/ThomasStuetz/a90f8f7b46279387987d#file-testdb-java

2. Test

```
Cursor cursor = db.query(
        LocationEntry.TABLE_NAME,
        columns,
        null,   // Columns for the "where" clause
        null,   // Values for the "where" clause
        null,   // Columns to group by
        null,   // Columns to filter by row groups
        null    // sort order
);

if (cursor.moveToFirst()) {
    int cityIndex = cursor.getColumnIndex(LocationEntry.COLUMN_CITY_NAME);
    String city = cursor.getString(cityIndex);

    int zipCodeIndex = cursor.getColumnIndex(LocationEntry.COLUMN_ZIP_CODE);
    int zipCode = cursor.getInt(zipCodeIndex);

    assertEquals(testCityName, city);
    assertEquals(testZipCode,zipCode);
} else {
    fail("No values returned :(");
}
```

https://gist.github.com/ThomasStuetz/a90f8f7b46279387987d#file-testdb-java