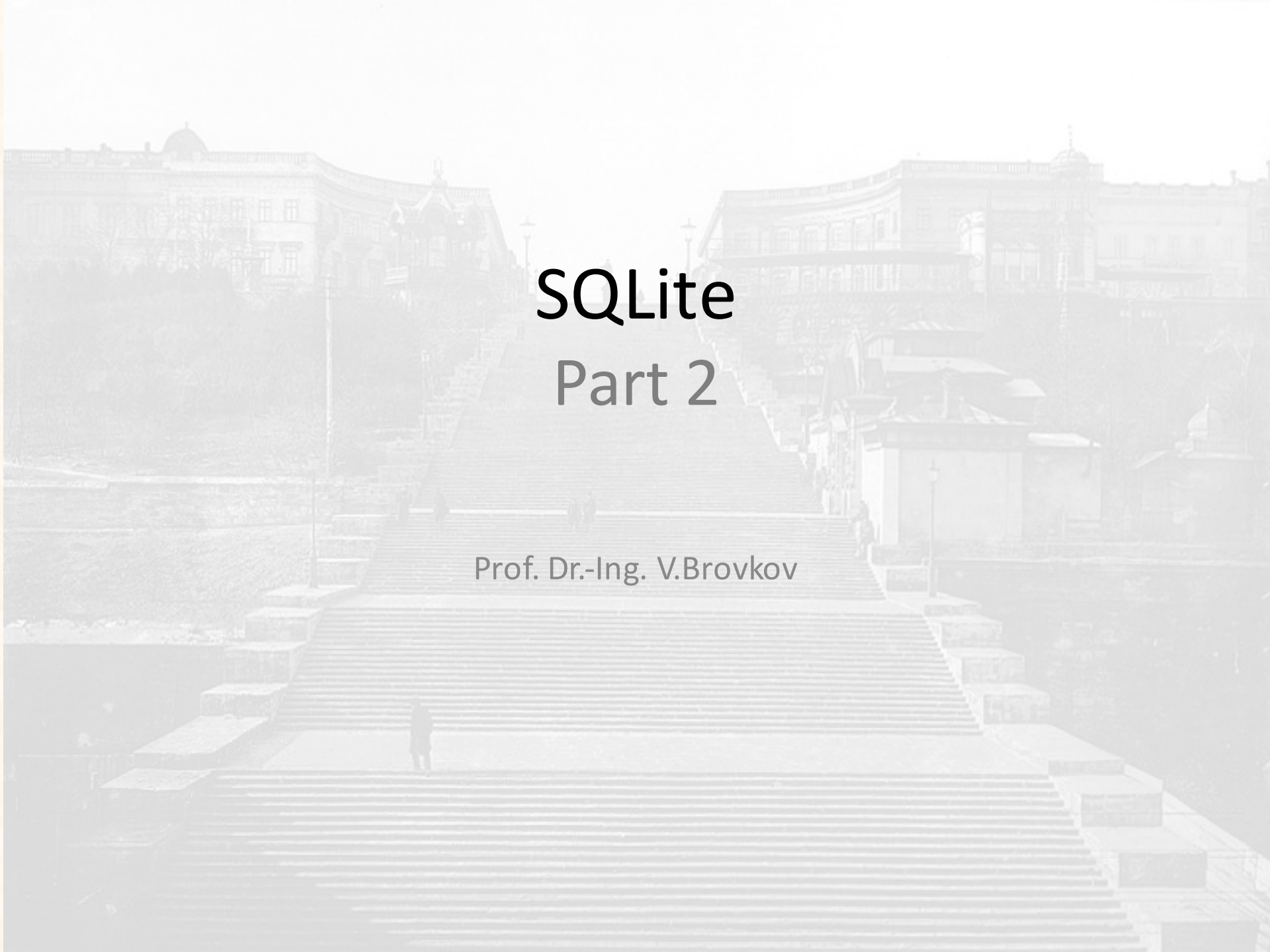


SQLite
Part 2

Prof. Dr.-Ing. V.Brovkov

innerJoin und rawQuerry in SQLite

Projekt: P0371_SQLiteInnerJoin

http://startandroid.ru/ru/uroki/vse-uroki-spiskom/77-urok-37-zaprosy-iz-svjazannyh-tablits-inner-join-v-sqlite-metod-rawquery.html

SQLiteInnerJoin

- Untersuchen wir die Erstellung eines Result Sets aus mehreren Tabellen.
- Erzeugen wir einen DB mit zwei Tabellen. Die Daten sind als Arrays als folgend vorbereitet:

```
16  // Positions
17  int[] position_id = { 1, 2, 3, 4 };
18  String[] position_name = { "Chef", "Informatiker", "Ökonom", "Security" };
19  int[] position_salary = { 15000, 13000, 10000, 8000 };
20
21  // Persons
22  String[] people_name = { "Julian", "Samira", "Elias", "Markus",
23                           "Laura", "Alexander", "Daniel", "Lukas" };
24  int[] people_posid = { 2, 3, 2, 2, 3, 1, 2, 4 };
```

SQLiteInnerJoin: onCreate()

```
109     public void onCreate(SQLiteDatabase db) {
110       Log.d(LOG_TAG, "--- onCreate database ---");
111       ContentValues cv = new ContentValues();
112       // Create a table position
113       db.execSQL("create table position ("
114           + "id integer primary key,"
115           + "name text," + "salary integer"
116           + ");");
117       // fill a table position
118       for (int i = 0; i < position_id.length; i++) {
119         cv.clear();
120         cv.put("id", position_id[i]);
121         cv.put("name", position_name[i]);
122         cv.put("salary", position_salary[i]);
123         db.insert("position", null, cv);
124       }
125       // create a table people
126       db.execSQL("create table people ("
127           + "id integer primary key autoincrement,"
128           + "name text,"
129           + "posid integer"
130           + ");");
131       // fill a table people
132       for (int i = 0; i < people_name.length; i++) {
133         cv.clear();
134         cv.put("name", people_name[i]);
135         cv.put("posid", people_posid[i]);
136         db.insert("people", null, cv);
137       }
138     }
```

inner join with rawQuerry

```
51      // Log INNER JOIN with rawQuery
52      Log.d(LOG_TAG, "--- INNER JOIN with rawQuery ---");
53      String sqlQuery = "select PL.name as Name, "
54          + "PS.name as Position, salary as Salary "
55          + "from people as PL "
56          + "inner join position as PS "
57          + "on PL.posid = PS.id "
58          + "where salary < ? ";
59          //+ "where salary < ? or PL.name = ? ";
60      c = db.rawQuery(sqlQuery, new String[] {"12000"});
61      //c = db.rawQuery(sqlQuery, new String[] {"12000", "Alexander"});
62
63      logCursor(c);
64      c.close();
65      Log.d(LOG_TAG, "--- ---");
```

inner join with querry

```
67      // Log INNER JOIN with query
68      Log.d(LOG_TAG, "--- INNER JOIN with query ---");
69      String table = "people as PL inner join position as PS on PL.posid = PS.id";
70      String columns[] = { "PL.name as Name",
71                           "PS.name as Position",
72                           "salary as Salary" };
73      String selection = "salary > ?";
74      //String selection = "salary < ? or PL.name = ?";
75      String[] selectionArgs = {"12000"};
76      //String[] selectionArgs = {"12000", "Lukas"};
77      c = db.query(table, columns, selection, selectionArgs, null, null, null);
78      logCursor(c);
79      c.close();
80      Log.d(LOG_TAG, "--- ---");
```

Cursor Logging

```
86  // Log cursor data
87  void logCursor(Cursor c) {
88    if (c != null) {
89      if (c.moveToFirst()) {
90        String str;
91        do {
92          str = "";
93          for (String cn : c.getColumnNames()) {
94            str = str.concat(cn + " = " + c.getString(c.getColumnIndex(cn)) + "; ");
95          }
96          Log.d(LOG_TAG, str);
97        } while (c.moveToNext());
98      }
99    } else
100     Log.d(LOG_TAG, "Cursor is null");
101 }
```

Transactions in SQLite

Projekt P0381_Transaction

http://startandroid.ru/ru/uroki/vse-uroki-spiskom/78-urok-38-tranzaktsii-v-sqlite.html

Transactions

```
1   void myActions() {
2     db = dbh.getWritableDatabase();
3     db.beginTransaction();
4     try{
5         delete(db, "mytable");
6         insert(db, "mytable", "val1");
7         insert(db, "mytable", "val2");
8         db.setTransactionSuccessful();
9     } finally {
10        db.endTransaction();
11    }
12    read(db, "mytable");
13    dbh.close();
14  }
```

- Transaction:
  - das Prinzip „Alles oder nichts“ bei Datenänderungen in DB
  - Begonnene Transaction muss unbedingt abgeschlossen werden. Das „finally“ ist empfohlen!
  - blockiert die DB von weiteren Anschlüssen. Das Ausdruck
    db = dbh.getWritableDatabase();
    innerhalb einer Transaction wird ein Fehler erzeugen
  - Die Funktionen delete(), insert(), read() sind weiter gezeigt.

Transactions

```
16    void insert(SQLiteDatabase db, String table, String value) {
17      Log.d(LOG_TAG, "Insert in table " + table + " value = " + value);
18      ContentValues cv = new ContentValues();
19      cv.put("val", value);
20      db.insert(table, null, cv);
21    }
22
23    void read(SQLiteDatabase db, String table) {
24      Log.d(LOG_TAG, "Read table " + table);
25      Cursor c = db.query(table, null, null, null, null, null, null);
26      if (c != null) {
27        Log.d(LOG_TAG, "Records count = " + c.getCount());
28        if (c.moveToFirst()) {
29          do {
30            Log.d(LOG_TAG, c.getString(c.getColumnIndex("val")));
31          } while (c.moveToNext());
32        }
33        c.close();
34      }
35    }
36
37    void delete(SQLiteDatabase db, String table) {
38      Log.d(LOG_TAG, "Delete all from table " + table);
39      db.delete(table, null, null);
40    }
```

Transactions

- Eine Aktion ohne Transaktion.
  - Alle Aktivitäten reiherfolgenach

- Was passiert?

```
void myActions() {    //1
    db = dbh.getWritableDatabase();
    delete(db, "mytable");
    insert(db, "mytable", "val1");
    read(db, "mytable");
    dbh.close();
}
```

```
D/myLogs: --- onCreate Activity ---
D/myLogs: Delete all from table mytable
D/myLogs: Insert in table mytable value = val1
D/myLogs: Read table mytable
D/myLogs: Records count = 1
D/myLogs: val1
```

Transactions

- Transaktion gestartet und nicht abgeschlossen
- Was passiert?

```
void myActions() {          //2
    db = dbh.getWritableDatabase();
    delete(db, "mytable");
    db.beginTransaction();
    insert(db, "mytable", "val1");
    db.endTransaction();
    insert(db, "mytable", "val2");
    read(db, "mytable");
    dbh.close();
}
```

```
D/myLogs: --- onCreate Activity ---
D/myLogs: Delete all from table mytable
D/myLogs: Insert in table mytable value = val1
D/myLogs: Insert in table mytable value = val2
D/myLogs: Read table mytable
D/myLogs: Records count = 1
D/myLogs: val2
```

Transactions

- Transaktion gestartet und abgeschlossen
  - Weitere Aktivitäten danach
  - Was passiert?

```
void myActions() {          //3
    db = dbh.getWritableDatabase();
    delete(db, "mytable");
    db.beginTransaction();
    insert(db, "mytable", "val1");
    db.setTransactionSuccessful();
    insert(db, "mytable", "val2");
    db.endTransaction();
    insert(db, "mytable", "val3");
    read(db, "mytable");
    dbh.close();
}
```

```
D/myLogs: --- onCreate Activity ---
D/myLogs: Delete all from table mytable
D/myLogs: Insert in table mytable value = val1
D/myLogs: Insert in table mytable value = val2
D/myLogs: Insert in table mytable value = val3
D/myLogs: Read table mytable
D/myLogs: Records count = 3
D/myLogs: val1
D/myLogs: val2
D/myLogs: val3
```

Transactions

- Transaktion gestartet
  - Ein zweiter Anschluss
- Was passiert?

```
60      void myActions() {
61          try {           //4
62              db = dbh.getWritableDatabase();
63              delete(db, "mytable");
64              db.beginTransaction();
65              insert(db, "mytable", "val1");
66              Log.d(LOG_TAG, "create DBHelper");
67              DBHelper dbh2 = new DBHelper(this);
68              Log.d(LOG_TAG, "get db");
69              SQLiteDatabase db2 = dbh2.getWritableDatabase();
70              read(db2, "mytable");
71              dbh2.close();
72              db.setTransactionSuccessful();
73              db.endTransaction();
74              read(db, "mytable");
75              dbh.close();
76          } catch (Exception ex) {
77              Log.d(LOG_TAG, ex.getClass()
78                      + " error: " + ex.getMessage());
79          }
80      }
```

```
D/myLogs: --- onCreate Activity ---
D/myLogs: Delete all from table mytable
D/myLogs: Insert in table mytable value = val1
D/myLogs: create DBHelper
D/myLogs: get db
D/myLogs: class android.database.sqlite.SQLiteDatabaseLockedException
          database is locked (code 5): , while compiling: PRAGMA journal_mode
```

DB Upgrade in SQLite

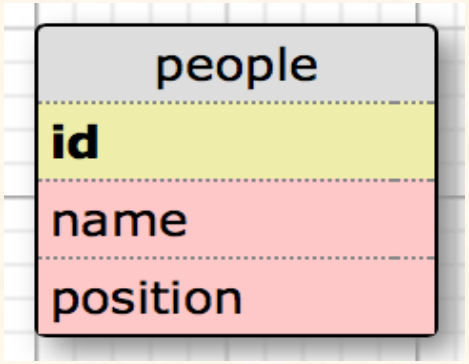
http://startandroid.ru/ru/uroki/vse-uroki-spiskom/79-urok-39-onupgrade-obnovljaem-bd-v-sqlite.html

Projekt: P0391_SQLiteOnUpgradeDB

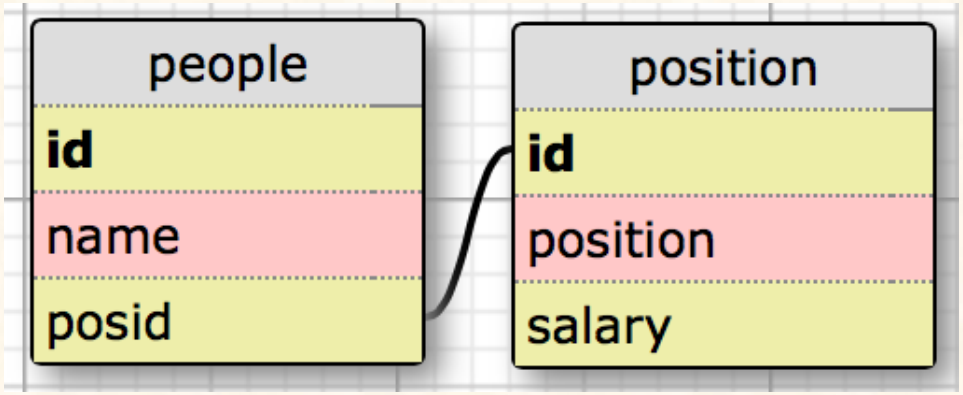
DB upgrade

- Es gab die 1. App-Version mit einer Datenbank
  - nur eine Tabelle mit 3 Spalten
  - die Methode **onCreate()** erstellt und füllt die Tabelle **people**
- Es muss die 2. Version implementiert werden. DB mit zwei verbundene Tabellen
  - die Methode **onCreate()** erstellt und füllt die Tabellen **position, people**
  - die Methode **onUpgrade()**
    - erstellt und füllt die Tabelle **position**;
    - einführt in die Tabelle **people** eine neue Spalte und einfügt die passende Daten
    - erstellt die Tabelle **people_tmp** und übernimmt die Daten aus der Tabelle **people**
    - löscht die Tabelle **people** und erzeigt neu mit anderer Struktur
    - übernimmt die Daten aus der Tabelle **people_tmp** in die neue Tabelle **people**

1. DB Version



2. DB Version

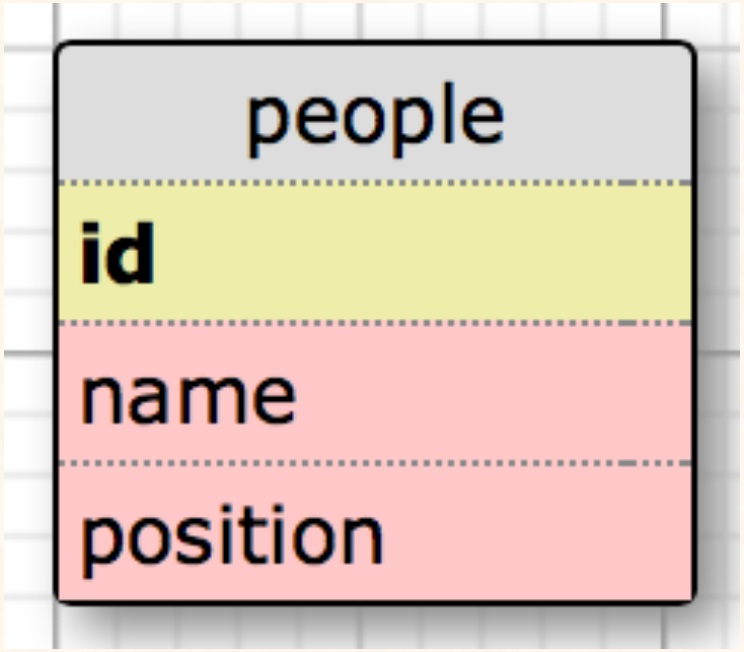


onCreate() DB_VERSION = 1

```
66      public void onCreate(SQLiteDatabase db) {
67          Log.d(LOG_TAG, " --- onCreate database --- ");
68          String[] people_name = { "Julian", "Samira", "Elias", "Markus",
69                  "Laura", "Alexander", "Daniel", "Lukas" };
70          String[] people_positions = { "Informatiker", "Ökonom",
71                  "Informatiker", "Informatiker", "Ökonom", "Chef",
72                  "Informatiker", "Security" };
73          ContentValues cv = new ContentValues();
74          db.execSQL("create table people ("
75                  + "id integer primary key autoincrement,"
76                  + "name text, position text);");
77          for (int i = 0; i < people_name.length; i++) {
78              cv.clear();
79              cv.put("name", people_name[i]);
80              cv.put("position", people_positions[i]);
81              db.insert("people", null, cv);
82          }
83      }
```

DB Data Set: DB_VERSION = 1

```
private void writeStaff(SQLiteDatabase db) {
   Cursor c = db.rawQuery("select * from people", null);
   logCursor(c, "Table people");
   c.close();
}
```



onCreate() DB_VERSION = 2

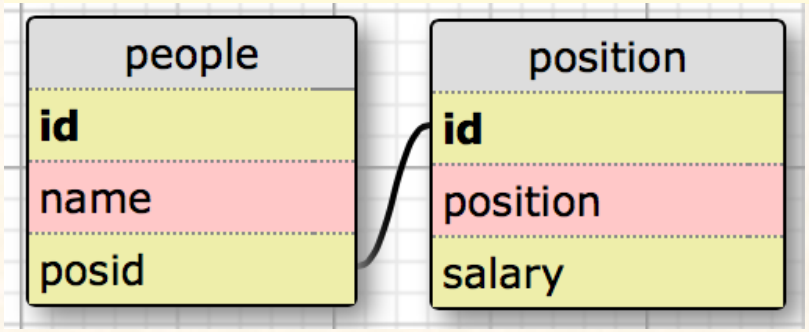
```
85      public void onCreate1(SQLiteDatabase db) {
86          Log.d(LOG_TAG, " --- onCreate database --- ");
87          // Data for a people table
88          String[] people_name = { "Julian", "Samira", "Elias", "Markus",
89                  "Laura", "Alexander", "Daniel", "Lukas" };
90          int[] people_posid = { 2, 3, 2, 2, 3, 1, 2, 4 };
91          // Data for a position table
92          int[] position_id = { 1, 2, 3, 4 };
93          String[] position_name = { "Chef", "Informatiker", "Ökonom",
94                  "Security" };
95          int[] position_salary = { 15000, 13000, 10000, 8000 };
96          ContentValues cv = new ContentValues();
97          // create and fill a position table
98          db.execSQL("create table position (" + "id integer primary key,"
99                  + "name text, salary integer" + ");");
100         for (int i = 0; i < position_id.length; i++) {
101             cv.clear();
102             cv.put("id", position_id[i]);
103             cv.put("name", position_name[i]);
104             cv.put("salary", position_salary[i]);
105             db.insert("position", null, cv);
106         }
107         // create and fill a people table
108         db.execSQL("create table people ("
109                 + "id integer primary key autoincrement,"
110                 + "name text, posid integer);");
111         for (int i = 0; i < people_name.length; i++) {
112             cv.clear();
113             cv.put("name", people_name[i]);
114             cv.put("posid", people_posid[i]);
115             db.insert("people", null, cv);
116         }
117     }
```

DB Data Set: DB_VERSION = 2

```
private void writeStaff(SQLiteDatabase db) {
  Cursor c = db.rawQuery("select * from people", null);
  logCursor(c, "Table people");
  c.close();

  c = db.rawQuery("select * from position", null);
  logCursor(c, "Table position");
  c.close();

  String sqlQuery = "select PL.name as Name, PS.name as Position, salary as Salary "
      + "from people as PL "
      + "inner join position as PS "
      + "on PL.posid = PS.id ";
  c = db.rawQuery(sqlQuery, null);
  logCursor(c, "inner join");
  c.close();
}
```



Data Set Logging: logCursor()

```
void logCursor(Cursor c, String title) {
  if (c != null) {
    if (c.moveToFirst()) {
      Log.d(LOG_TAG, title + ". " + c.getCount() + " rows");
      StringBuilder sb = new StringBuilder();
      do {
        sb.setLength(0);
        for (String cn : c.getColumnNames()) {
          sb.append(cn + " = "
              + c.getString(c.getColumnIndex(cn)) + "; ");
        }
        Log.d(LOG_TAG, sb.toString());
      } while (c.moveToNext());
    }
  } else
    Log.d(LOG_TAG, title + ". Cursor is null");
}
```

onUpgrade()

```
119  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
120      if (oldVersion == 1 && newVersion == 2) {
121          ContentValues cv = new ContentValues();
122          // Data for a position table
123          int[] position_id = { 1, 2, 3, 4 };
124          String[] position_name = { "Chef", "Informatiker", "Ökonom",
125          "Security" };
126          int[] position_salary = { 15000, 13000, 10000, 8000 };
127          db.beginTransaction();
128          try {
129            // create and fill a position table
130            db.execSQL("create table position ("
131                + "id integer primary key,"
132                + "name text, salary integer);");
133            for (int i = 0; i < position_id.length; i++) {
134              cv.clear();
135              cv.put("id", position_id[i]);
136              cv.put("name", position_name[i]);
137              cv.put("salary", position_salary[i]);
138              db.insert("position", null, cv);
139            }
```

Testen wir die DB-Versionen

Beginnen wir eine Transaction

Erstellen wir und füllten wir die Tabelle **position**

onUpgrade()

```
140             db.execSQL("alter table people add column posid integer;");
141             for (int i = 0; i < position_id.length; i++) {
142                 cv.clear();
143                 cv.put("posid", position_id[i]);
144                 db.update("people", cv, "position = ?",
145                         new String[] { position_name[i] });
146             }
147             db.execSQL("create temporary table people_tmp ("
148                     + "id integer, name text, position text, posid integer);");
149             db.execSQL("insert into people_tmp select id, name, position, posid from people;");
150             db.execSQL("drop table people;");
151             db.execSQL("create table people ("
152                     + "id integer primary key autoincrement,"
153                     + "name text, posid integer);");
154             db.execSQL("insert into people select id, name, posid from people_tmp;");
155             db.execSQL("drop table people_tmp;");
156             db.setTransactionSuccessful();
157         } finally {
158             db.endTransaction();
159         }
160     }
161 }
```

Führen wir eine neue Spalte in die Tabelle **people** ein und geben wir die passende Daten zu

Ersetzen wir die Tabelle people mit einer neuen

Beenden wir dieTransaction

Datenaufbewahrung
in Android Apps

Fragen?