Welcome, आप इस सप्ताह के अंत में गर्मी हराया था? स्पर्स ईथर नहीं कर सकता था, बुरा मत मानना. कम से कम वे सेल्टिक्स पसंद करने के लिए तत्पर हैं करने के लिए पुनर्निर्माण के बहुत है.

AITI IIT Bombay Class 2013
Lecture 8

Agenda

- Remaining Schedule
- Android SQL
- Android Language Support
- Checkins

Remaining Schedule

- ~3.5 weeks till showtime!
- This week: more advanced topics, revenue, connecting android and Django
- Please have a working demo draft by the end of this week

Tomorrow

- Android Multimedia / Graphics
- Multi-threading
- Location Services
- Revenue ☺: Advertising, Google Play

Android SQLite

- Define a Schema
- Create a
- Reading to an SQL Database
- Writing to an SQL Database

Defining a Schema

- A “schema” is a map or description of how your data base in organized.
- Similar to column or row headings in excell

Benefits from Web Site	YEAR	1	2	3
Direct sales		$15,000.00	$50,000.00	$75,000.00
Incremental sales resulting from enhanced promotional/salesperson effectiveness		$25,000.00	$25,000.00	$25,000.00
Incremental sales resulting from increased partner participation		$25,000.00	$25,000.00	$25,000.00
Reduced travel costs		$25,000.00	$25,000.00	$25,000.00
Reduced customer service costs		$50,000.00	$50,000.00	$50,000.00
Reduced printing and shipping costs		$5,000.00	$5,000.00	$5,000.00

Defining a Schema

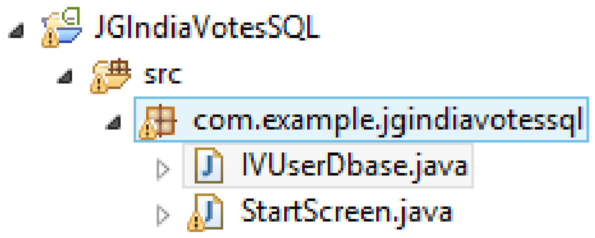
- A “schema” is a map or description of how your data base in organized.
- Similar to column or row headings in excell

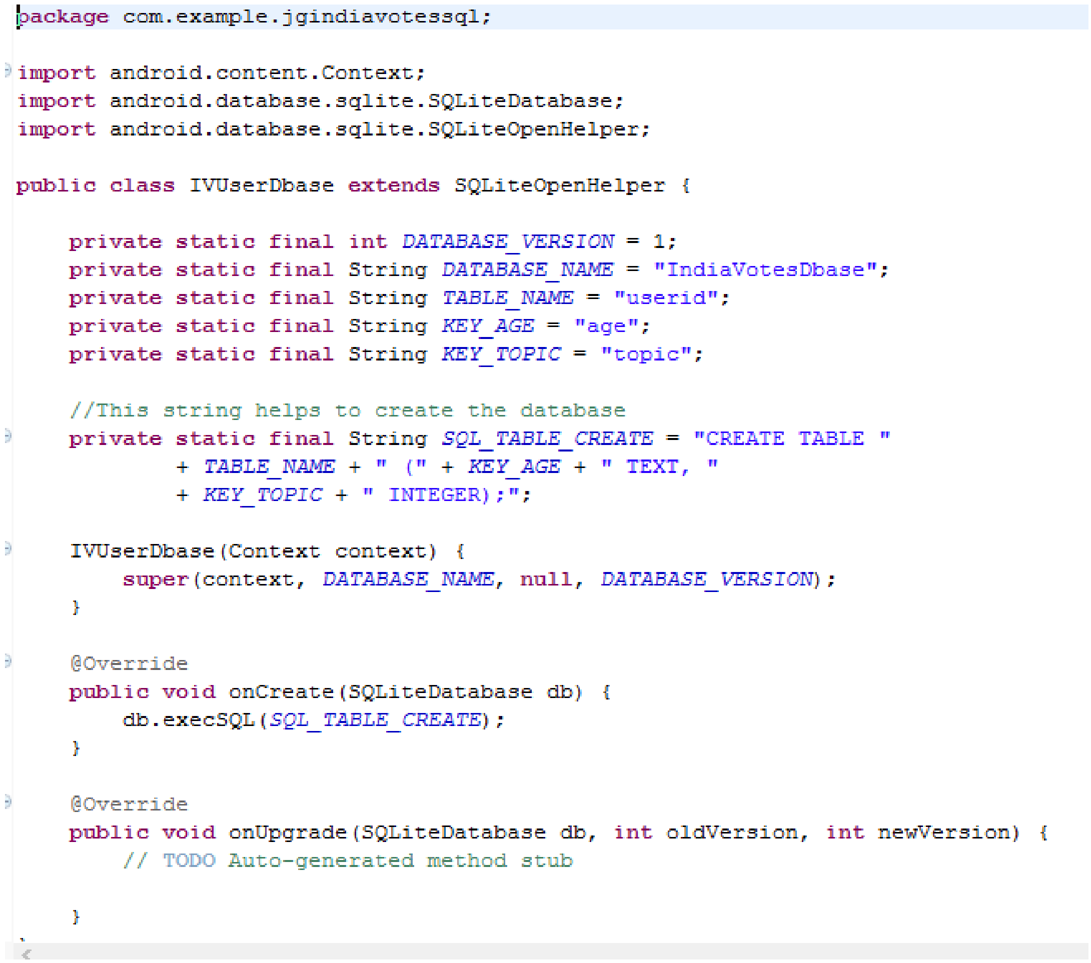
Schema:

Benefits from Web Site	YEAR	1	2	3
Direct sales		$15,000.00	$50,000.00	$75,000.00
Incremental sales resulting from enhanced promotional/salesperson effectiveness		$25,000.00	$25,000.00	$25,000.00
Incremental sales resulting from increased partner participation		$25,000.00	$25,000.00	$25,000.00
Reduced travel costs		$25,000.00	$25,000.00	$25,000.00
Reduced customer service costs		$50,000.00	$50,000.00	$50,000.00
Reduced printing and shipping costs		$5,000.00	$5,000.00	$5,000.00

Defining a Schema

- First Create an SQLiteHelper subclass in your source folder. Make sure its part of your android package:

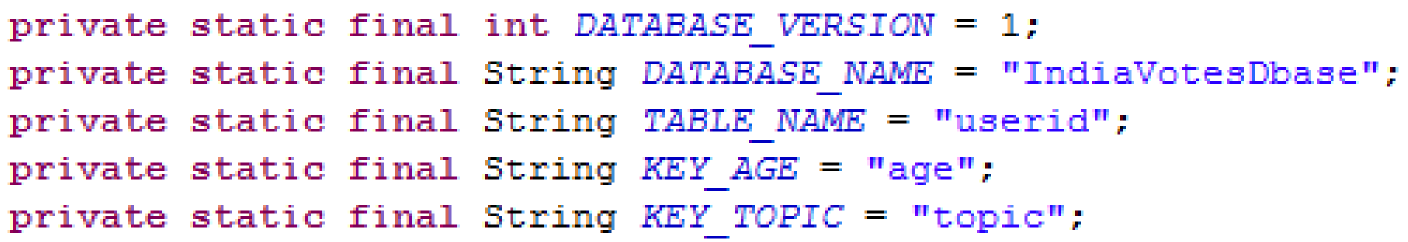




```
package com.example.jgindiavotessql;

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

public class IVUserDbase extends SQLiteOpenHelper {

    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "IndiaVotesDbase";
    private static final String TABLE_NAME = "userid";
    private static final String KEY_AGE = "age";
    private static final String KEY_TOPIC = "topic";

    //This string helps to create the database
    private static final String SQL_TABLE_CREATE = "CREATE TABLE "
            + TABLE_NAME + " (" + KEY_AGE + " TEXT, "
            + KEY_TOPIC + " INTEGER);";

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

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_TABLE_CREATE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // TODO Auto-generated method stub

    }
```

Defining a Schema

- Use static final strings to define the fields your database, or your data base "schema":

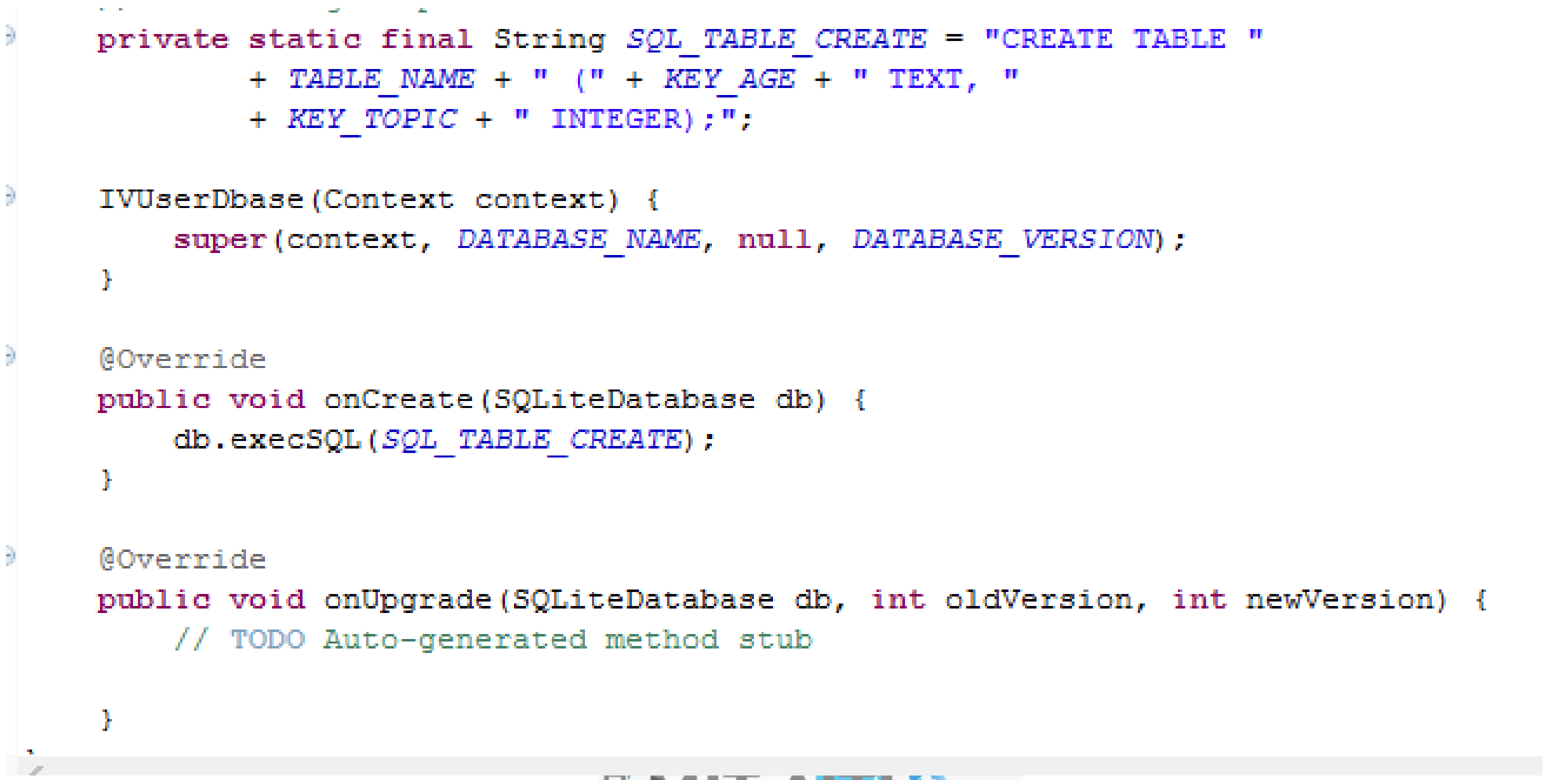


```
private static final int DATABASE_VERSION = 1;
private static final String DATABASE_NAME = "IndiaVotesDbase";
private static final String TABLE_NAME = "userid";
private static final String KEY_AGE = "age";
private static final String KEY_TOPIC = "topic";
```

Other parts of your Database class

- Add a string that will create your data base, it needs to have enclosing parentheses in your value fields. This string will be given to the execSQL method on when the create method is called

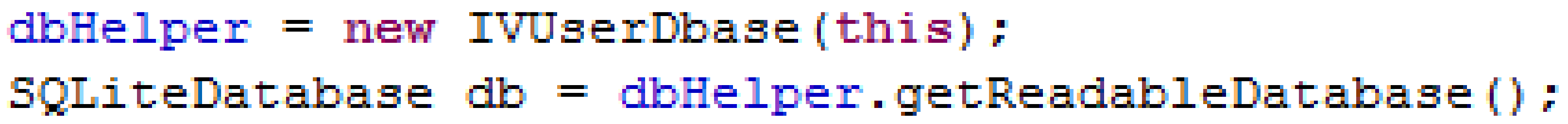
- See the Android tutorial or piazza for an example of an SQLite class



```
private static final String SQL_TABLE_CREATE = "CREATE TABLE "
        + TABLE_NAME + " (" + KEY_AGE + " TEXT, "
        + KEY_TOPIC + " INTEGER);";

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

@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(SQL_TABLE_CREATE);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // TODO Auto-generated method stub

}
```

Creating a database in your Activity

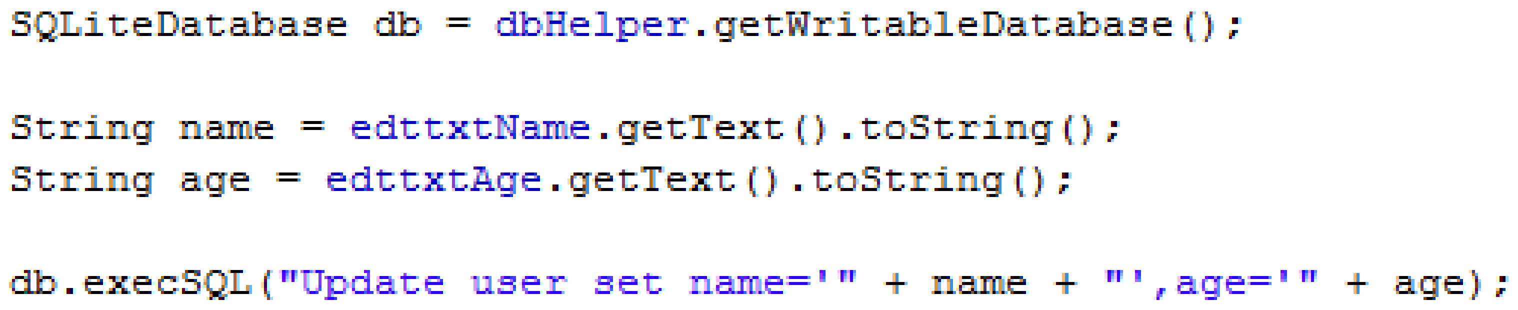
- Create the class as a private object in your starting activity
- Initialize it in the onCreate() method



```
dbHelper = new IVUserDbase(this);
SQLiteDatabase db = dbHelper.getReadableDatabase();
```

Write to your Database

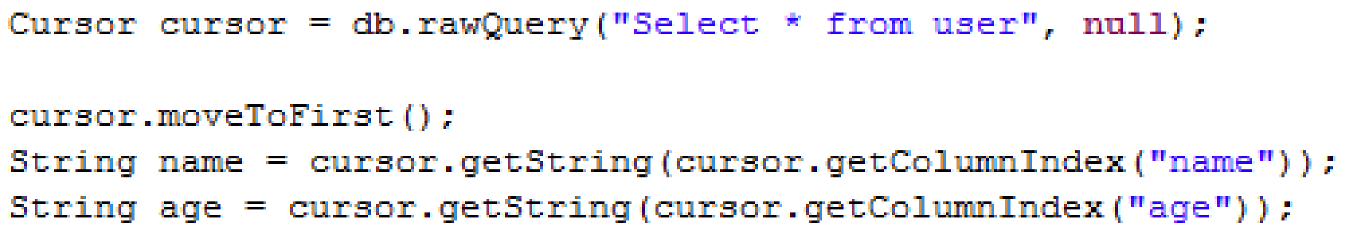
- To write to your database, call execSQL with your updated strings



```
SQLiteDatabase db = dbHelper.getWritableDatabase();

String name = edttxtName.getText().toString();
String age = edttxtAge.getText().toString();

db.execSQL("Update user set name='" + name + "',age='" + age);
```

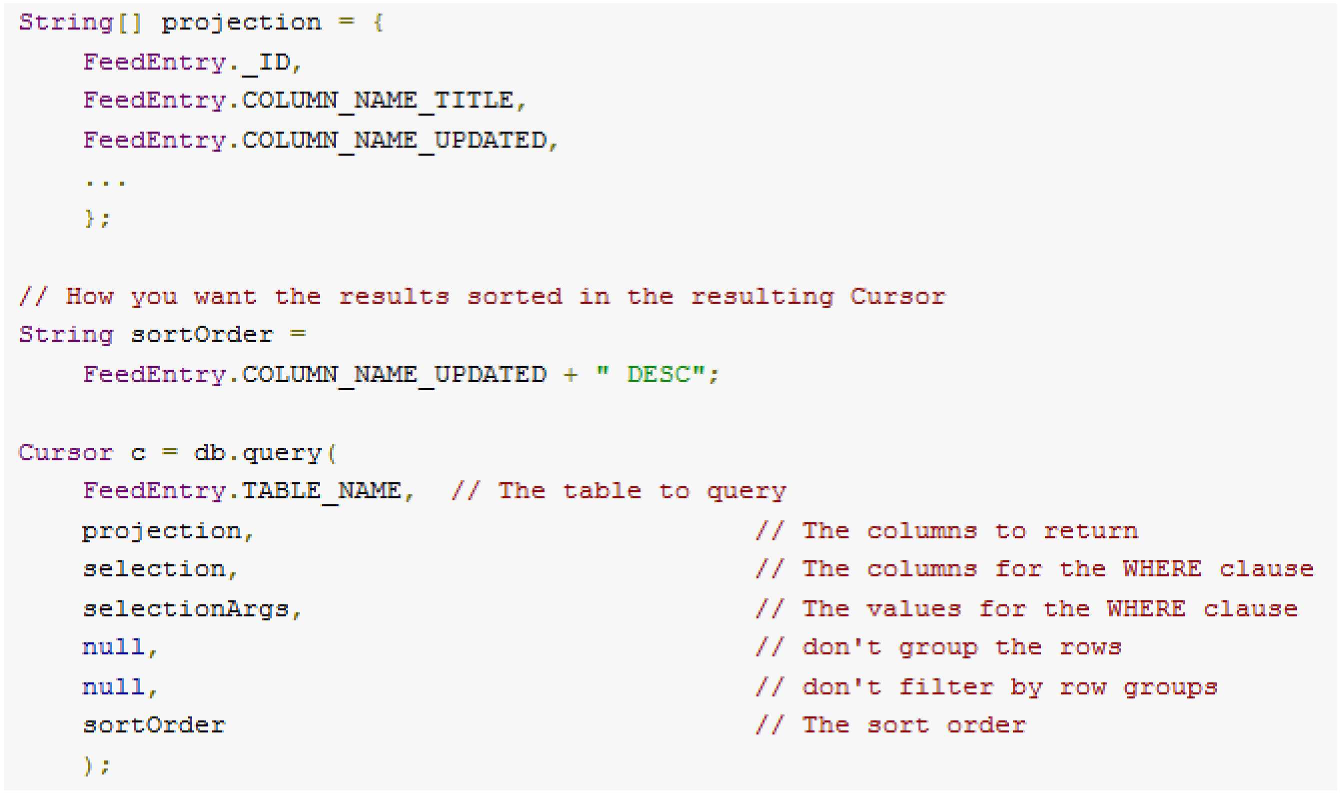
Read from your database

- Use a cursor object and the query() method to perform read functions:



```
Cursor cursor = db.rawQuery("Select * from user", null);

cursor.moveToFirst();
String name = cursor.getString(cursor.getColumnIndex("name"));
String age = cursor.getString(cursor.getColumnIndex("age"));
```

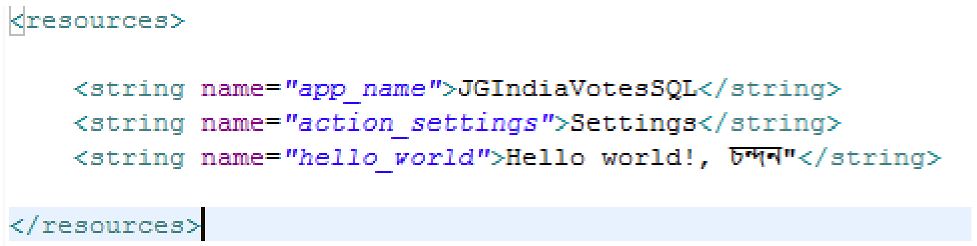
Read from your database (Android tutorial example)



```
String[] projection = {
    FeedEntry._ID,
    FeedEntry.COLUMN_NAME_TITLE,
    FeedEntry.COLUMN_NAME_UPDATED,
    ...
    };

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

Android Language Support

- There should be Unicode support for Devanagari script in Android 4.2 and beyond
- You should be able to paste Hindi script in to your code (xml and java):



```
<resources>

    <string name="app_name">JGIndiaVotesSQL</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!, चन्दन"</string>

</resources>
```

- Users will need a special keyboard to input Hindi. Make sure your app works correctly with it (I think this inputs Hindi with English)

- https://play.google.com/store/apps/details?id=com.google.android.apps.inputmethod.hindi&hl=en

Additional Language Resources

- http://developer.android.com/training/basics/supporting-devices/languages.html (use different value folders to store strings in different languages)

- Tips on handling multiple locations when publishing

http://developer.android.com/distribute/googleplay/publish/localizing.html

- Settings -> Language & Input, under "KEYBOARD & INPUT METHODS" section, check Google Hindi Input, then click Default and select "Hindi transliteration" in the "Choose input method" dialog.