Mobile and Smart Phone App Development

Dr. Nilesh Patel, PhD

Week 11.1

SQLite

- A Light Weight Database
- Supported by both iPhone and Android
- Implements most of SQL-92 standard
- Supports complex queries with exception of **outer joins**

Whose turn is it?

Simple explanation (Lets consider)

TableA

id	name
1	Pirate
2	Monkey
3	Ninja
4	Spaghetti

TableB

id	name
1	Rutabaga
2	Pirate
3	Darth Vader
4	Ninja

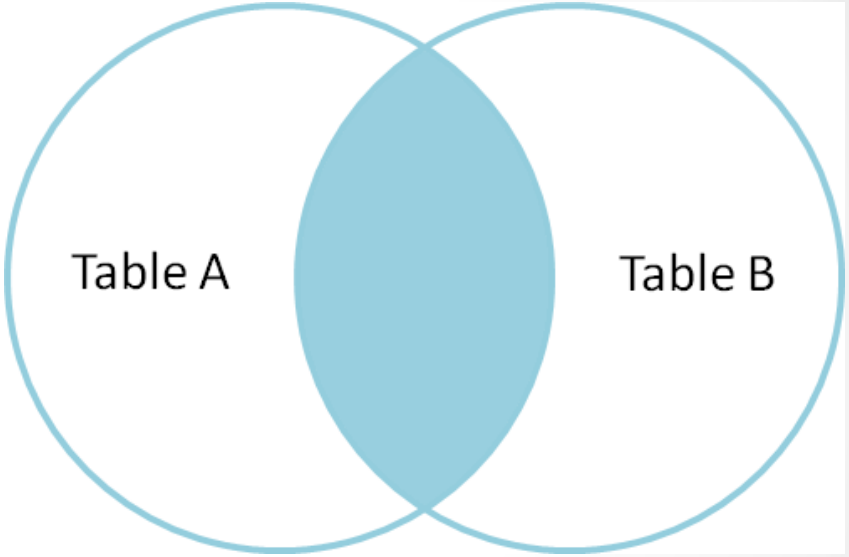
Whose turn is it?

http://www.codinghorror.com/blog/2007/10/a-visual-explanation-of-sql-joins.html

Inner join (A ∩ B)

```
SELECT * FROM TableA
INNER JOIN TableB
ON TableA.name = TableB.name
```

id	name	id	name
1	Pirate	2	Pirate
3	Ninja	4	Ninja



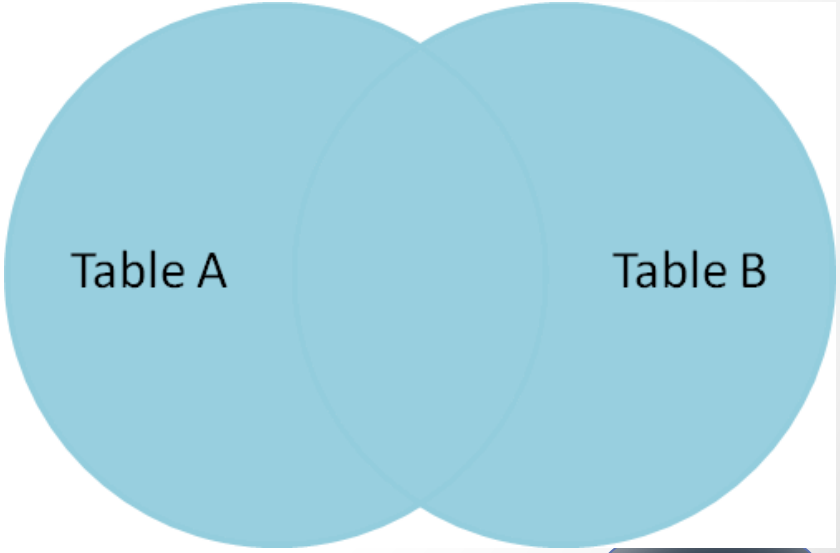
Whose turn is it?

http://www.codinghorror.com/blog/2007/10/a-visual-explanation-of-sql-joins.html

Full Outer Join (A U B)

```
SELECT * FROM TableA
FULL OUTER JOIN TableB
ON TableA.name = TableB.name
```

id	name	id	name
1	Pirate	2	Pirate
2	Monkey	null	null
3	Ninja	4	Ninja
4	Spaghetti	null	null
null	null	1	Rutabaga
null	null	3	Darth Vader



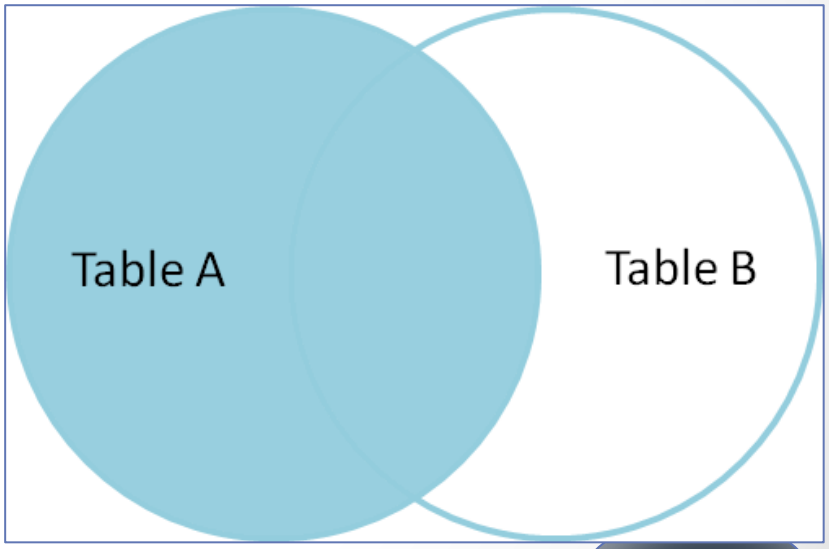
Whose turn is it?

http://www.codinghorror.com/blog/2007/10/a-visual-explanation-of-sql-joins.html

Left Outer Join (A)

```
SELECT * FROM TableA
LEFT OUTER JOIN TableB
ON TableA.name = TableB.name
```

id	name	id	name
1	Pirate	2	Pirate
2	Monkey	null	null
3	Ninja	4	Ninja
4	Spaghetti	null	null



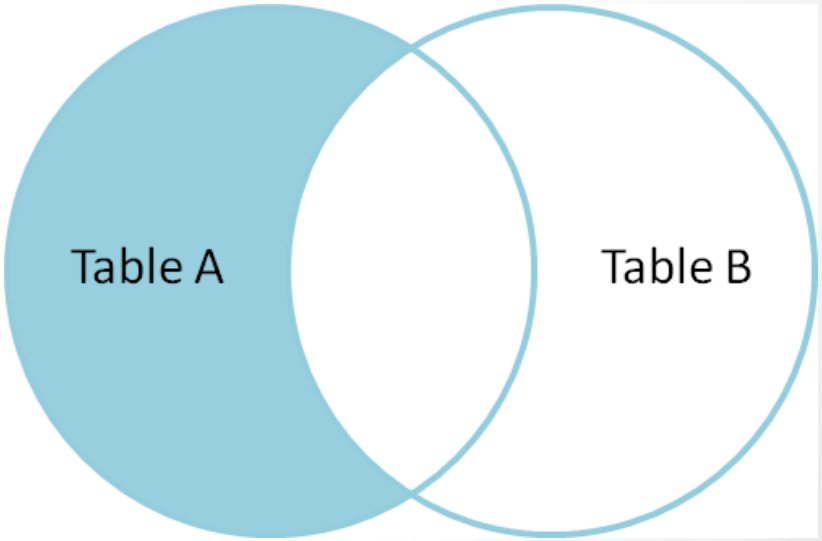
Whose turn is it?

http://www.codinghorror.com/blog/2007/10/a-visual-explanation-of-sql-joins.html

Left Outer Join (A – B)

```
SELECT * FROM TableA
LEFT OUTER JOIN TableB
ON TableA.name = TableB.name
WHERE TableB.id Is null

id  name          id    name
--  ----          --    ----
2   Monkey        null  null
4   Spaghetti     null  null
```



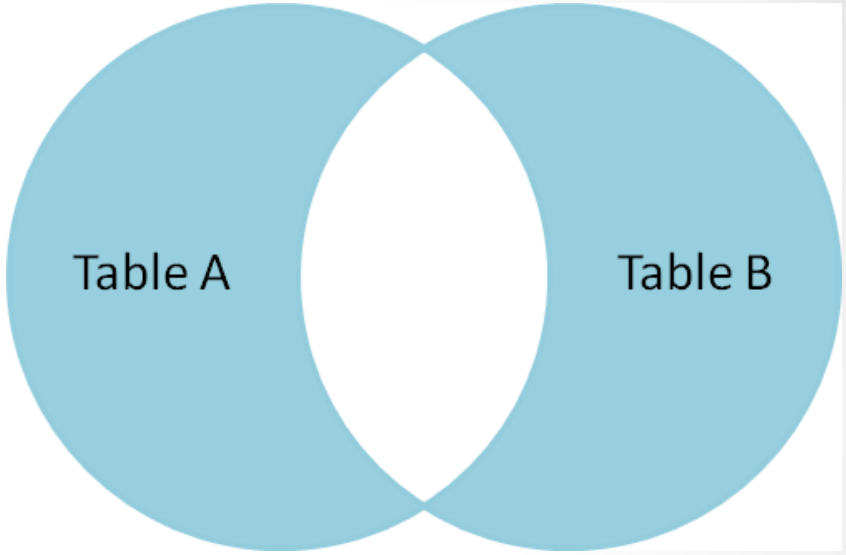
Whose turn is it?

http://www.codinghorror.com/blog/2007/10/a-visual-explanation-of-sql-joins.html

Outer Join (A ∪ B) – (A ∩ B)

```
SELECT * FROM TableA
FULL OUTER JOIN TableB
ON TableA.name = TableB.name
WHERE TableA.id IS null
OR TableB.id IS null
```

id	name	id	name
2	Monkey	null	null
4	Spaghetti	null	null
null	null	1	Rutabaga
null	null	3	Darth Vader



http://www.codinghorror.com/blog/2007/10/a-visual-explanation-of-sql-joins.html

Whose turn is it?

What can we do?

- Create a database
- Create a table
- Insert rows
- Delete rows
- Create indices
- Perform queries (raw or action)
- Administer the database

Whose turn is it?

Creating Database

- Helper class: **SQLiteDatabase**
- Operation:

```
SQLiteDatabase.openDatabase(
        String path,
        SQLiteDatabase.CursorFactory f,
        int flags);
```

- Flags: OPEN_READWRITE,
  OPEN_READONLY,
  CREATE_IF_NECESSORY

Create Application and database

```
//   filePath is a absolue path – i.e. "/data/data/<namespace>/<databaseName>" or "/sdcard/<databasename>"
try {
        SQLiteDatabase db = SQLiteDatabase.openDatabase(
                "/sdcard/edu.oakland.cse495/BaseballStat",
                null,
                SQLiteDatabase.CREATE_IF_NECESSARY);
        db.close();
}
catch (SQLiteException e) {
        Toast.makeText(this, e.getMessage(), 1).show();
}
```

Check Success of creation (DDMS)

Permissions

- Warning
  - Ensure your application seeks permission to write on external storage card
  - Update Manifest file for
    - Android.permission.WRITE_EXTERNAL_STORAGE

Second Method

- Create in local data space of the application

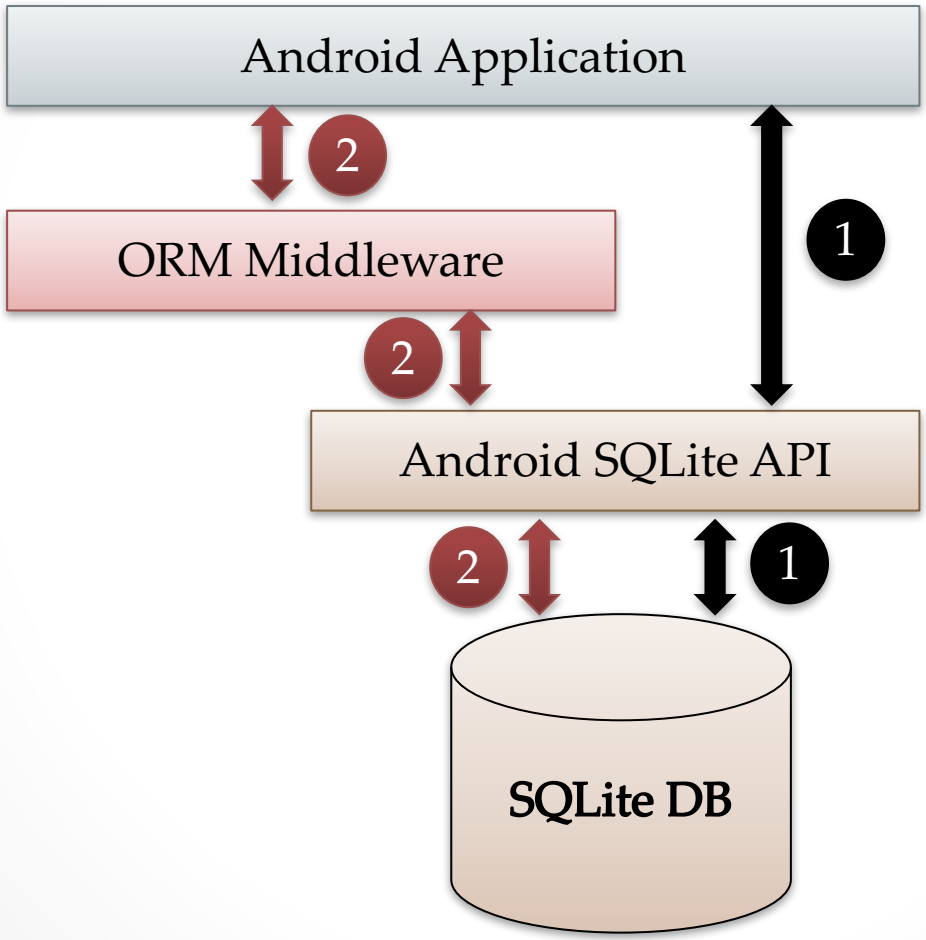
- **SQLiteDatabase db = this.openOrCreate(String path,**
  int mode,
  SQLiteDatabase.CursorFactory c);

- Always created in application data space

- Mode: MODE_PRIVATE, MODE_WORLD_READABLE
  MODE_WORLD_WRITABLE

Whose turn is it?

Architecture



Whose turn is it?

Application

Modify code to use second method

Whose turn is it?

Ready to perform operations

- **Retrieval Query:**
  - is typically a SQL-Select command in which a table holding a number of fields and rows is produced as an answer to a data request.
- **Action Query:**
  - usually performs maintenance and administrative tasks such as manipulating tables, users, environment, etc.
  - **Action queries** inside shall be implemented within a protective frame of a **database transaction** where the policy of **Atomicity** is transparently enforced.

Atomic Transaction Implementation

```
db.beginTransaction();
try {
        //perform your database operations here ...
        db.setTransactionSuccessful();     //commit your changes
}
catch (SQLiteException e) {
        //report problem
}
finally {
        db.endTransaction();
}
```

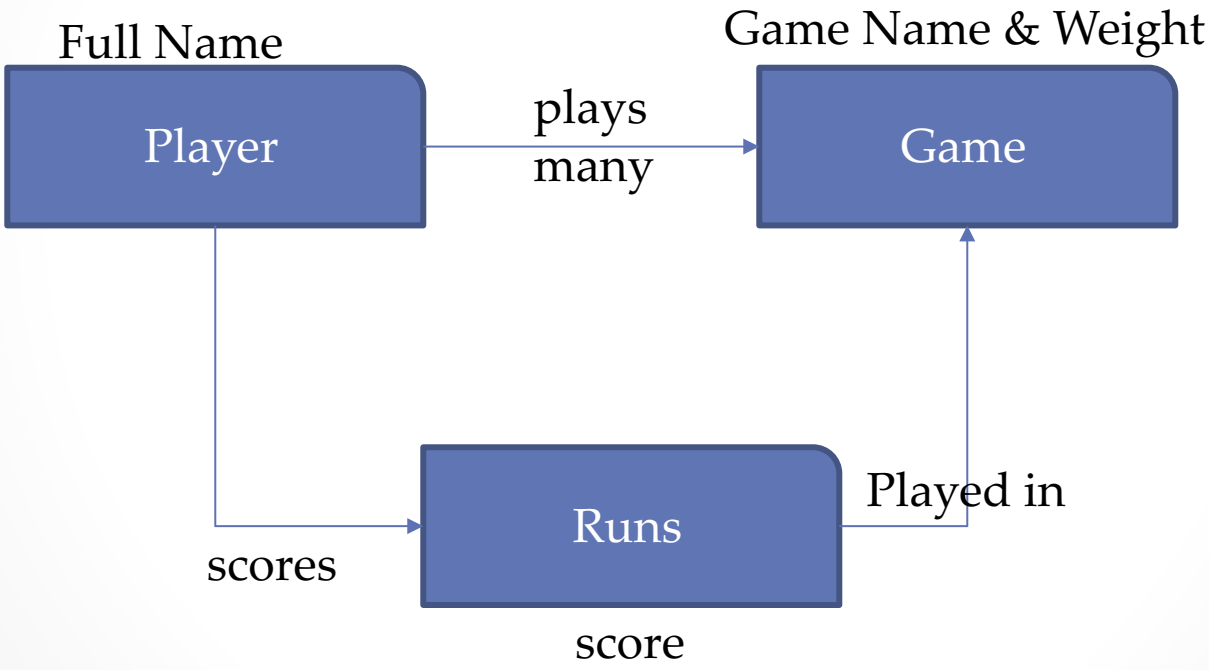
Whose turn is it?

Understanding SQL Basics

- Lets understand it through an application requirement
- You have been asked to design a mobile application that keep track of Baseball player scores in various games. Each game is marked with its weight, showing importance of the game between 0.0 and 1.0. The default weight for any game, if not changed is 0.1. The application allows user to enter Player full name (last and first), their participation in Game and the home runs they hit. The application is also responsible for generating stats on who is the winning player at any given time.

Whose turn is it?

Analysis



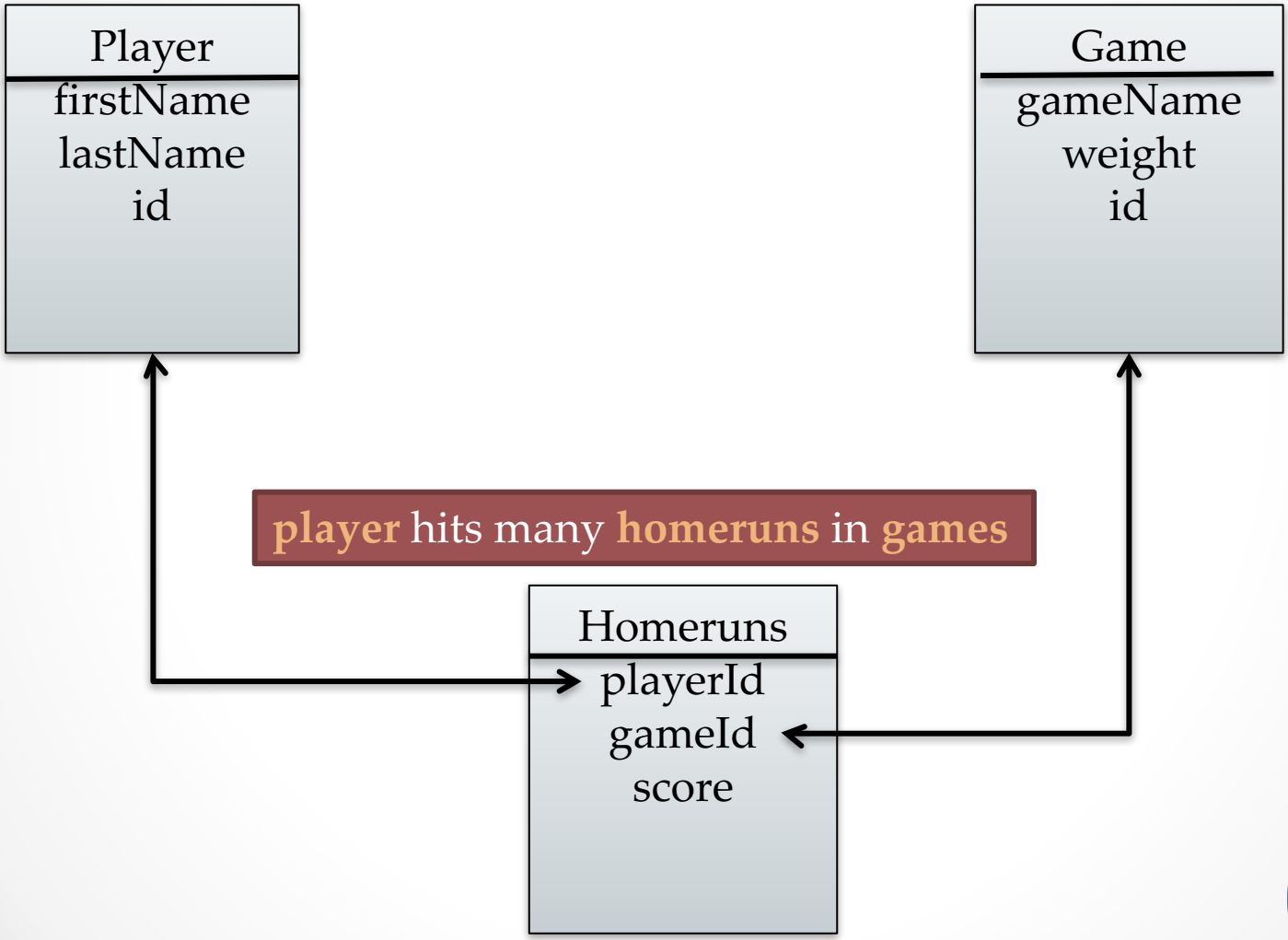
Whose turn is it?

Lets understand SQL Way First

- Understanding Schema
  - There are Players with first and last name
  - There are Games with name and weight of the game
  - There are home runs hit by players in each game
- In a nutshell- we have three tables
  - Players
  - Games
  - Homeruns

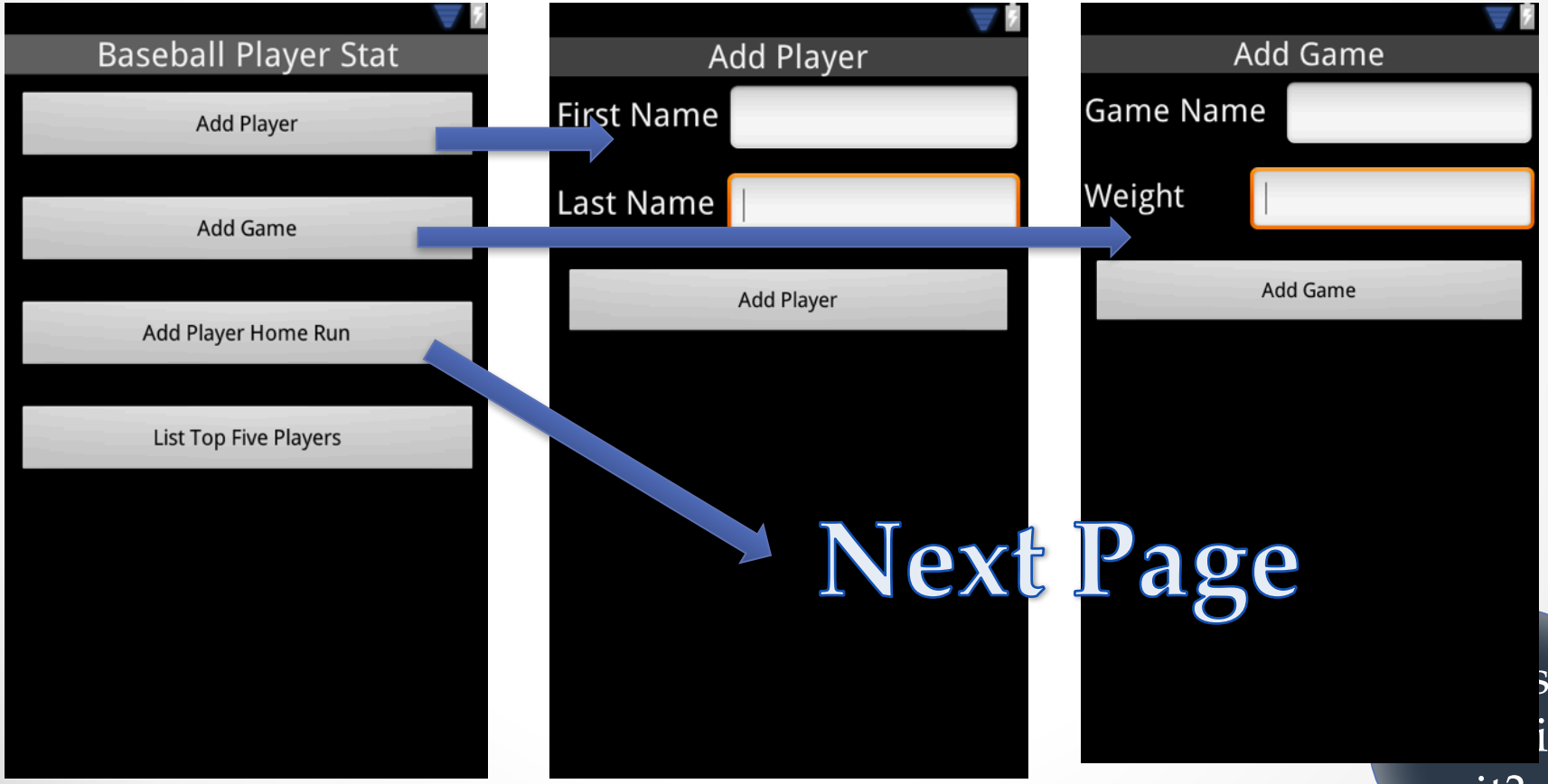
Whose turn is it?

Schema



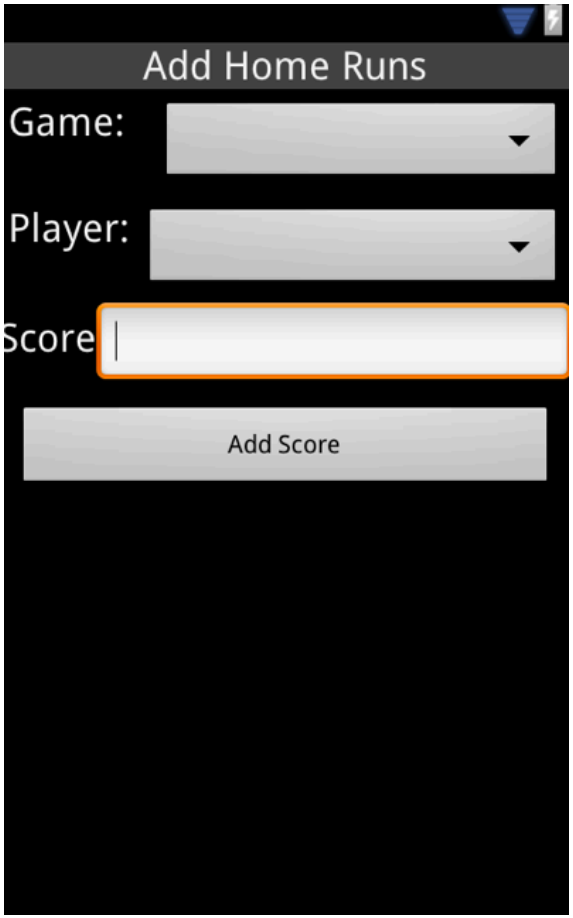
Whose turn is it?

Application Design



se
is
it?

Controlling when to execute



Whose turn is it?

Players Tables

- It has three fields
  - First name of type -> text
  - Last name of type -> text
  - Id of type -> int (represents unique ID)
- How do we create it?
- Many choices
  - SQLite Administrator
  - Command line interface

Whose turn is it?

Players Table

- SQL Command:
  - **create table Players(**

    id INTEGER PRIAMRY KEY AUTOINCREAMENT,

    firstName text NOT NULL,

    lastName text NOT NULL);

Whose turn is it?

Games Table

- SQL Command:
  - **create table Games(**

    id integer primary key,

    gameName text not null,

    weight real default 0.1 check(weight < 1.0));

Whose turn is it?

Homeruns Table

- SQL Command:
  - **create table Homeruns(**

    playerId integer references Players(id),

    gameId integer references Games(id),

    homeRuns integer not null,

    primary key (playerId, gameId));

Whose turn is it?

Helper To Manage DB Operations

```
public class DBHelper extends SQLiteOpenHelper {
        public DBHelper(){
                super(context,DATABASE_NAME,null,1);
        }
        public void onCreate(SQLiteDatabase db) {}
        public void onUpgrade(SQLiteDatabase database,
                int oldVersion, int newVersion) {}
}
```

Whose turn is it?

Simple Case

Whose turn is it?

Create Activity

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    firstName = findViewById(R.id.editTextFirst);
    lastName = findViewById(R.id.editTextLast);
    addButton = findViewById(R.id.buttonAdd);
    listView = findViewById(R.id.listView);

    addButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String first_name = firstName.getText().toString();
            String last_name = lastName.getText().toString();

            db.createPlayer(first_name, last_name);
            players = db.getAllPlayers();
            adapter = new ArrayAdapter<String>(view.getContext(), android.R.layout.simple_list_item_1,
players);
            listView.setAdapter(adapter);
        }
    });
    db = new AppDatabaseHelper(this);
    players = db.getAllPlayers();

    adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, players);
    listView.setAdapter(adapter);

}
```

Whose turn is it?

Helper To Manage DB Operations

```
public class AppDatabaseHelper extends SQLiteOpenHelper {
    Context context = null;
    public static final String DATABASE_NAME = "game.db";
    public static final String PLAYERS_TABLE_NAME = "players";

    public AppDatabaseHelper(Context context) {
        super(context, DATABASE_NAME , null, 1);
    }

    public AppDatabaseHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int
version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        sqLiteDatabase.execSQL( "create table players " +
                    "(id integer primary key autoincreament, firstName text not null, lastName text not null)"
      );
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS contacts");
        onCreate(sqLiteDatabase);
    }

    // Create CRUD
    public ArrayList<String> getAllPlayers(){}
    public int getCount(){}
    public boolean updatePlayer(int id, String first_name, String last_name){}
    public int deletePlayer(int id){}

}
```

Insert value

- insert into Players(firstName, lastName) values ('x', 'y');
- Insert into Games(gameName, weight) values ('Z', 0.25);
- Insert into Homeruns(playerId, gameId, homeRuns)

  values (1, 1, 2);

Whose turn is it?

Altering or Updating database

- update Players set lastName='Junk' where playerid = 2;
- delete from Players where playerid=1;
- delete from Players;

Whose turn is it?

Selecting items from database

- select * from Players;
- select firstName || ' ' || lastName as PlayerName, id from Players;
- select playerId, gameId, homeRuns from Homeruns;

Whose turn is it?

Complex Queries

```
select Player.firstName | | ' ' | |
Player.lastName as PlayerName,
Games.gameName, Homeruns.homeruns,

from Homeruns

Join Players on(Homeruns.playerId =
Players.id)

Join Games on(Homeruns.gameId =
Games.id)

Where gameId = 5;
```

Whose turn is it?

Complex Queries

```
select Player.firstName|| ‘ ‘ || Player.lastName
as PlayerName,
        sum(Games.weight*Homeruns.score) as
TotalScore,
from Homeruns
join Players on(Homeruns.playerId = Players.id)
join Games on(Homeruns.gameId = Games.id)
group by Homeruns.playerId
order by TotalScore desc;
```

Whose turn is it?

SQL Syntax

select $field_1, field_2, \ldots, field_n$

from $table_1, table_2, \ldots, table_n$

where (restriction-join-condition)

order by $field_{1n}, field_{2n}, \ldots, field_{nn}$

group by $field_{1m}, field_{2m}, \ldots, field_{nm}$

having (group-condition)

Whose turn is it?

Programmatic Query

- Many ways to query database
  - Look at SQLiteDatabase class methods
- Simple applroach
  - SQLiteDatabase.rawQuery(String query, String[] args);

Whose turn is it?

Examples

```
Cursor c = db.rawQuery( "select count(*) as Count from Player”, null);
```

- **What does it do?**
  - Query the database to find total number of players in Player table
  - Create a new temporary table called Count (1 raw and 1 column)
  - Adds the value in that table
  - Returns the cursor that points to the first raw of that table

Whose turn is it?

Example (Advanced)

```
String sql = "insert into Players (firstName, lastName) values (?, ?)"
String args[] = {"Nilesh", "Patel"}

Cursor c = db.rawQuery( sql, args);
```

- What is happening here?
  - Inserts "Nilesh" as firstName and "Patel" as lastName in database table Players

Whose turn is it?

Cursor

- Primarily provides access to one raw at a time of resulting Table.
- Provides traversal and data access mechanisms:
- **Positional awareness:**
  - *isFirst(), isLast(), isBeforeFirst(), isAfterLast()),*
- **Navigation:**
  - *moveToFirst(), moveToLast(), moveToNext(), moveToPrevious(), move(n))*
- **Data Access:**
  - *getInt, getString, getFloat, getBlob, getDate, etc.*
- **Schema inspection:**
  - getColumnName, getColumnNames, getColumnIndex, getColumnCount, getCount)

Whose turn is it?

Cursor Example

```
Class Player{int id; String fullName};
ArrayList<Player> player = new ArrayList<Player>();

String sql="select id, firstName, lastName from Players";
Cursor cur= db.rawQuery(sql, null);

int idCol = cur.getColumnIndex("id");
int firstNameCol = cur.getColumnIndex("firstName");
Int lastNameCol = cur.getColumnIndex("lastName");

while (cur.moveToNext()) {
    Player p = new Player();
    p.Id = cur.getInt(idCol);
    p.fullName = cur.getString(firstNameCol) + " " + cur.getString(lastNameCol);
    players.add(p);
}
```

Whose turn is it?

getPlayers()

```
public ArrayList<String> getAllPlayers() {
        ArrayList<String> array_list = new ArrayList<String>();
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor res = db.rawQuery( "select * from players", null );
        res.moveToFirst();
        while(res.isAfterLast() == false){
                String id = res.getString(0);
                String firstName = res.getString(1);
                String lastName = res.getString(2);
                array_list.add(id + ": " + firstName + ", " + lastName);
                res.moveToNext();
        }
        return array_list;
}
```

Whose turn is it?

Implements Various Methods

```
public boolean createPlayer(String first_name, String last_name) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(”firstName", first_name);
        contentValues.put(”lastName", last_name);
        db.insert(”players", null, contentValues);
        return true;
}
```

Whose turn is it?

Implements Various Methods

```
public int getPlayersCount() {
        SQLiteDatabase db = this.getReadableDatabase();
        int numRows = (int) DatabaseUtils.queryNumEntries(db,
                        CONTACTS_TABLE_NAME);
        return numRows;
}
```

Whose turn is it?

Implements Various Methods

```
public Integer deletePlayer (Integer id) {
        SQLiteDatabase db =
                this.getWritableDatabase();
        return db.delete("players", "id = ? ",
                new String[] { Integer.toString(id) });
}
```

Whose turn is it?

Implements Various Methods

```
public boolean updatePlayer (Integer id, String first_name, String
last_name) {
        SQLiteDatabase db =  this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put("firstName", first_name);
        contentValues.put("lastName", last_name);
        db.update("players", contentValues, "id = ? ",
                new String[] { Integer.toString(id) } );
        return true;
}
```

Whose turn is it?

Persistent Storage ORM Way

Object **R**elational **M**apping

Whose turn is it?

What is ActiveAndroid?

Lite weight **M**apper

Maps **O**bjects and

their **R**elationships to

Database

Whose turn is it?

Different Solutions

- ActiveAndroid
- ORMLite
- SugarORM
- GreenDAO
- Realm

Whose turn is it?

Make following Changes in Gradle

Build.gradle

```
repositories {
    mavenCentral()
    maven { url "https://oss.sonatype.org/content/repositories/snapshots/" }
}

dependencies {
    compile 'com.michaelpardo:activeandroid:3.1.0-SNAPSHOT'
}
```

Build.gradle(app)

Make a clean build

Whose turn is it?

Make changes in Manifest

```
<application …
        <meta-data android:name="AA_DB_NAME"
                android:value="game_score.db" />
        <meta-data android:name="AA_DB_VERSION"
                android:value="3" />
        <provider
           android:name="com.activeandroid.content.ContentProvider"
           android:authorities="edu.cse523.npatel.learningorm"
           android:enabled="true"
           android:exported="false">
        </provider>

</application>
```

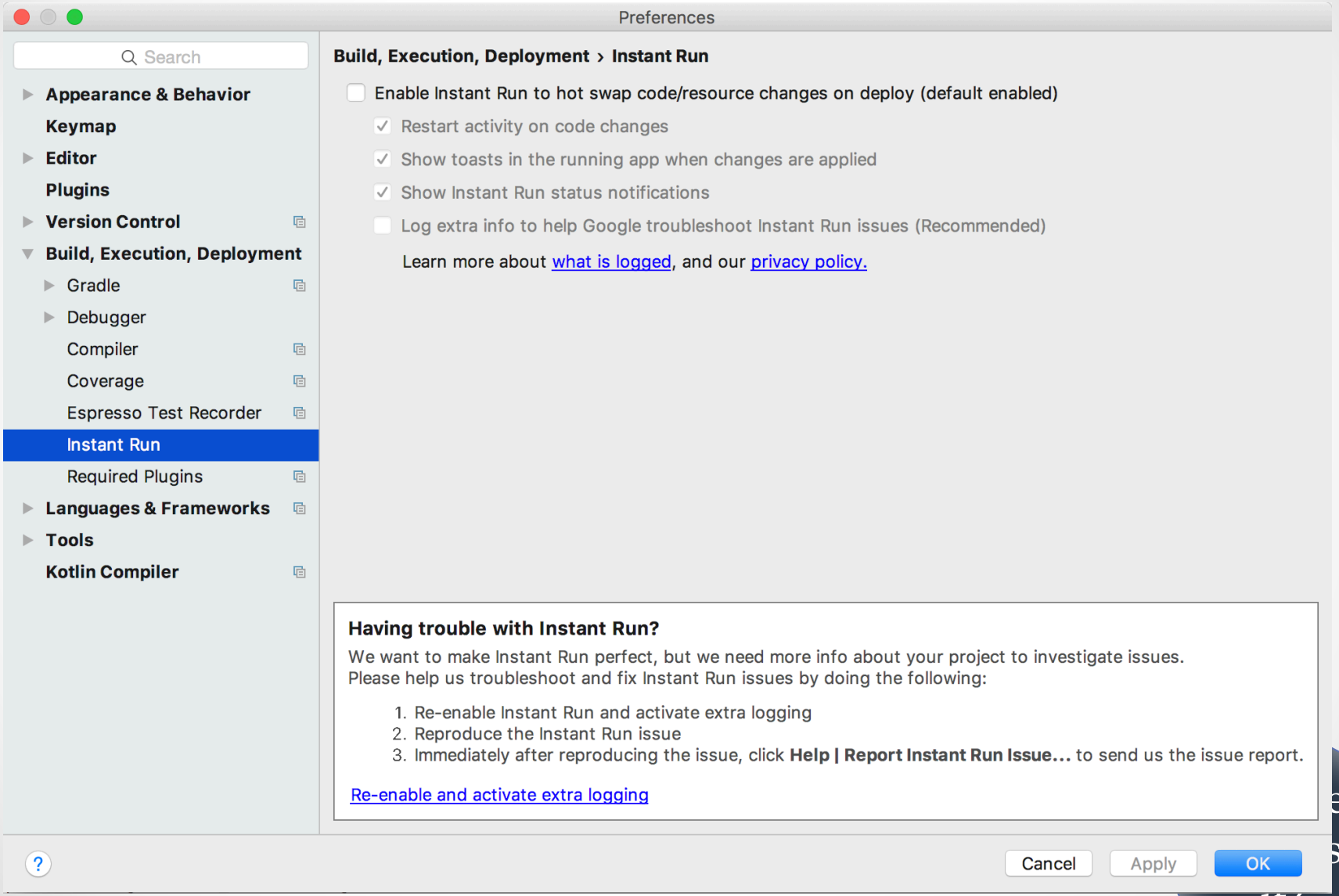
Name of database

Database version

Content Provider

Whose turn is it?

We may require Disabling Instant Run



Initialize Database

Extend Application Class

```
public class ORMApplication extends
Application {
    @Override
    public void onCreate() {
        super.onCreate();
        ActiveAndroid.initialize(this);
    }
}
```

Initialize

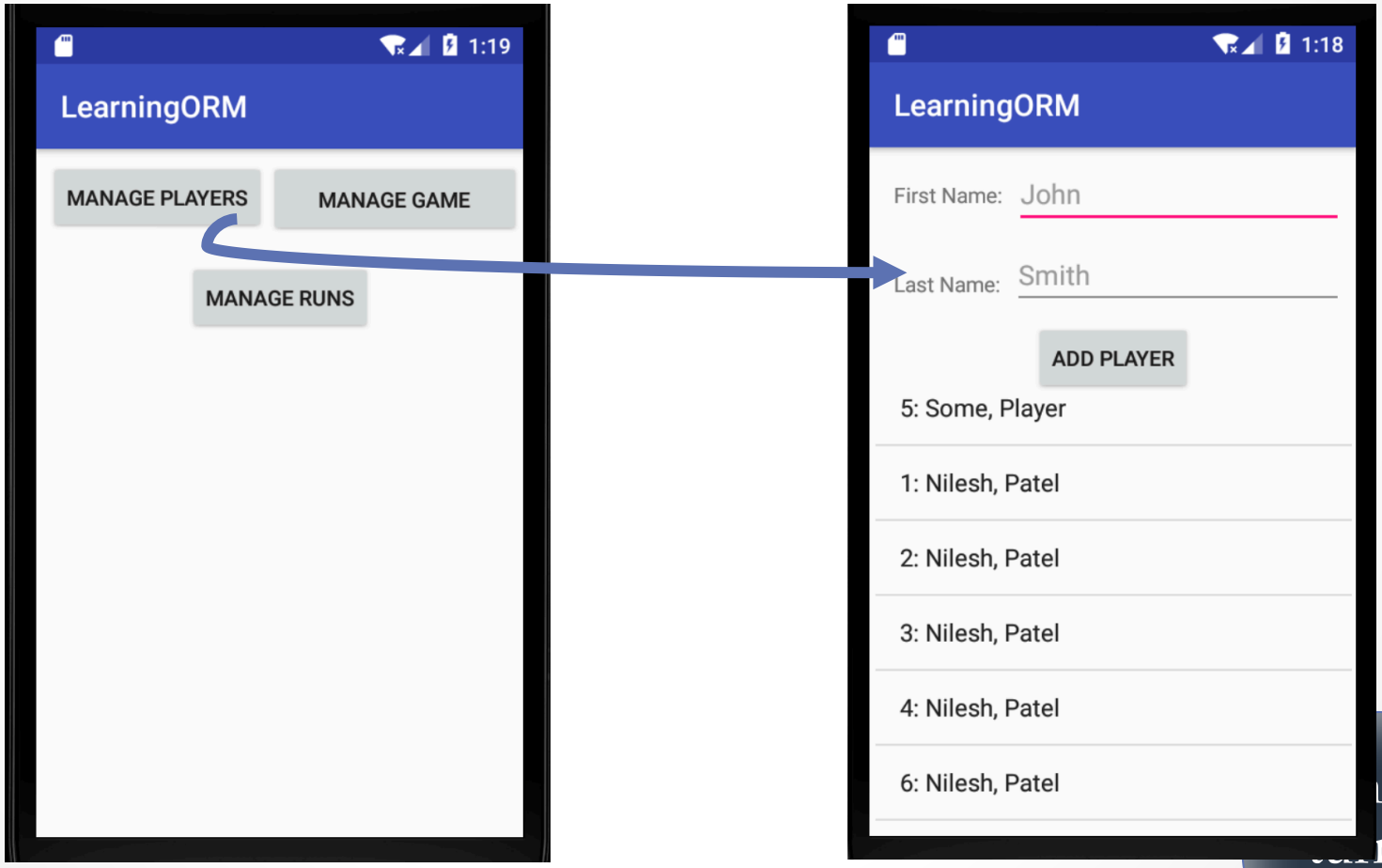
Whose turn is it?

Put it in Manifest

```
<application android:name=".ORMApplication”
    :
    :
```

Whose turn is it?

Create App



ose
n is
it?

Implement Model Class for Table Player

```
@Table(name = "player")
public class Player extends Model {

    @Column(name = "first_name")
    public String firstName;

    @Column(name = "last_name")
    public String lastName;

    public Player(){super();}
    public Player(String firstName, String lastName) {
        super();
        this.firstName = firstName;
        this.lastName = lastName;
    }

    @Override
    public String toString(){
        return getId() + ": " + firstName + ", " + lastName;
    }

    public static List<Player> getAllPlayers(){
        Select query = new Select();
        return query.from(Player.class).orderBy("first_name DESC").execute();
    }
}
```

Table Name

Base class

Attributes as Column

Query Builder

Whose turn is it?

What do you do with it?

- Helps with all C(Create) R(Read) U(Update) D(Delete) operations – Often referred as CRUD
- Create (saves data in database):
  - playe.save() // After initializing player object
- Read (Read data from database):
  - List<Player> players = Select().from(Player.class).execute();// Get all players

References to Player player object

Whose turn is it?

What do you do with it?

- Update (saves changed values in database):
  - player.save(); // After modifying player object
- Delete (Delete row(s) from database):
  - player.delete();

Declare in Manifest

```
<meta-data
   android:name="AA_DB_NAME" android:value="Game.db" />
<meta-data
   android:name="AA_DB_VERSION" android:value="1" />
<meta-data
   android:name="AA_MODELS"
   android:value="edu.oakland.cse423.Player, edu.oakland.cse423.Game" />
```

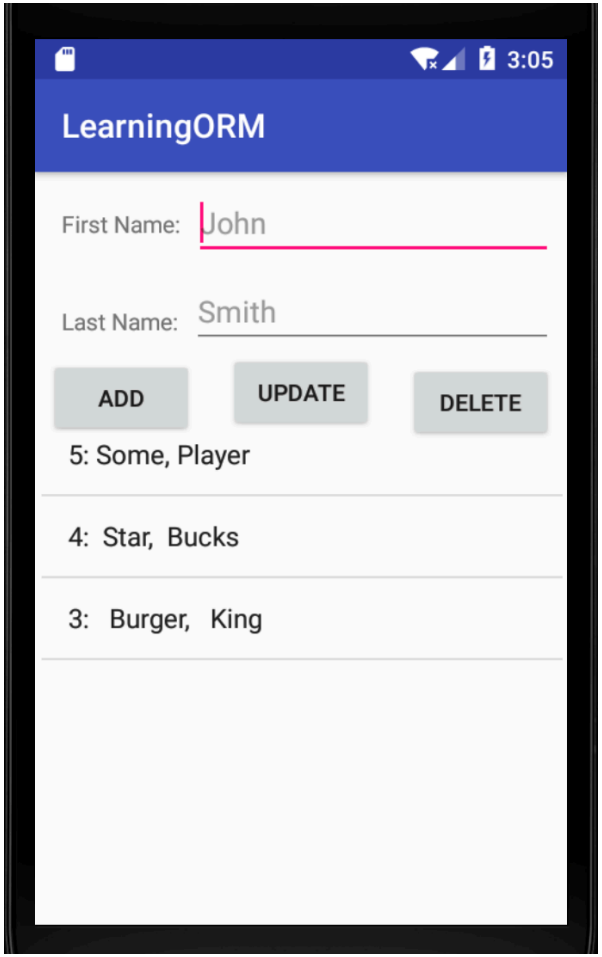
Additional settings in Manifest speeds-up the application

Whose turn is it?

Glue all pieces to gather

Whose turn is it?

Add Update / Delete



Whose turn is it?

With every action, we need

```
private void updateList()
{

    // add it to adapter
    players = Player.getAllPlayers();
    adapter = new ArrayAdapter<Player>(this, android.R.layout.simple_list_item_1, players);
    listView.setAdapter(adapter);
}

private void clearForm()
{
    firstName.setTag(-1);
    firstName.setText("");
    lastName.setText("");
    deleteButton.setEnabled(false);
    updateButton.setEnabled(false);
}
```

Whose turn is it?

Add Player

```
addButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String first_name = firstName.getText().toString();
        String last_name = lastName.getText().toString();

        Player player = new Player(first_name, last_name);
        player.save();
        updateList();
        clearForm();
    }
});
```

Whose turn is it?

Select Player

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
    @Override
    public void onItemClick(AdapterView<?> adapterView,
                            View view, int i, long l) {
        String item = ((TextView) view).getText().toString();
        String[] split = item.split(":");
        int id = Integer.parseInt(split[0]);
        split = split[1].split(",");
        split[0].trim();
        split[1].trim();
        firstName.setText(split[0]);
        lastName.setText(split[1]);
        firstName.setTag(id);
        deleteButton.setEnabled(true);
        updateButton.setEnabled(true);
    }
});
```

Whose turn is it?

Update Player

```
updateButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String first_name = firstName.getText().toString();
        String last_name = lastName.getText().toString();
        int id = (int) firstName.getTag();

        // Update record
        new Update(Player.class)
                .set("first_name = ?, last_name = ?", first_name, last_name)
                .where("id = ?", id)
                .execute();

        updateList();
        clearForm();
    }
});
```

Whose turn is it?

Delete Player

```
deleteButton.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View view) {
        int id = (int) firstName.getTag();

        // delete the record
        Select query = new Select();
        Player player = query.from(Player.class)
                .where("id = ?", id)
                .executeSingle();
        player.delete();
        updateList();
        clearForm();
    }
});
```

Whose turn is it?

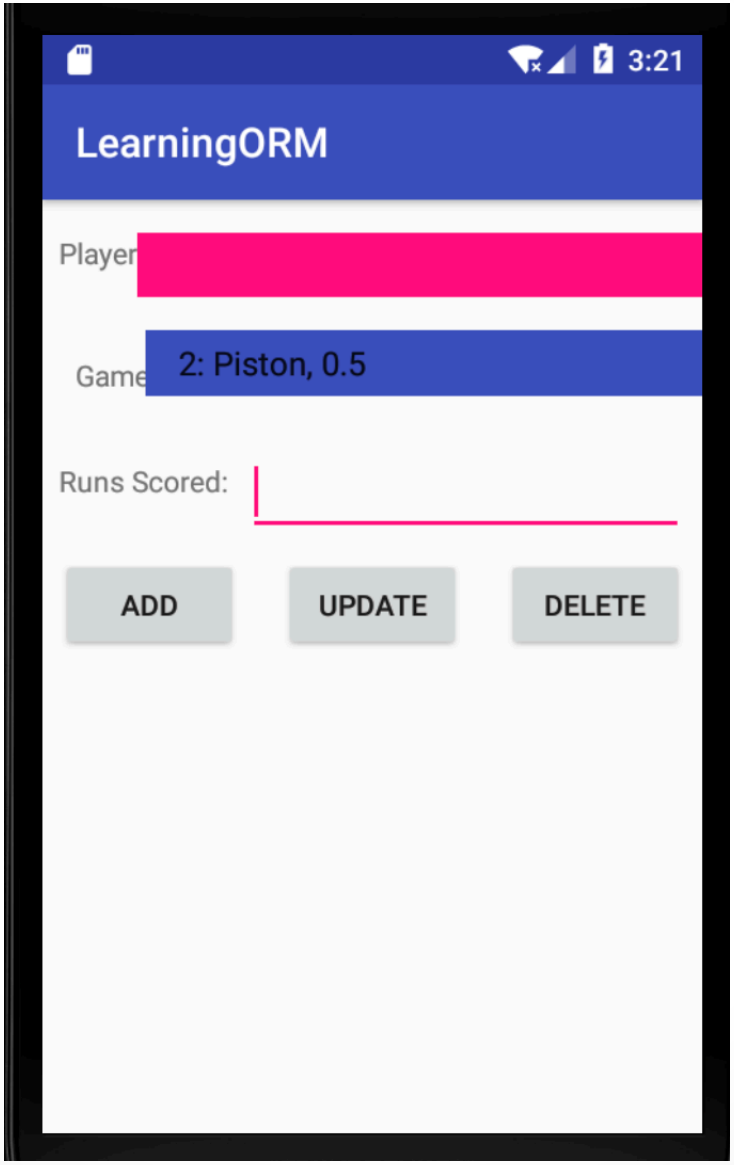
Create Manage Game

Whose turn is it?

Create Manage Runs

Whose turn is it?

ManageRuns



Whose turn is it?

Populate Spinner

```
public class ManageRunsActivity extends AppCompatActivity {
    Spinner gameSpinner = null, playerSpinner = null;
    List<Game> games = null;
    List<Player> players = null;
    ArrayAdapter<Game> gameSpinnerAdapter = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_manage_runs);

        gameSpinner = findViewById(R.id.spinnerGame);
        playerSpinner = findViewById(R.id.spinnerPlayer);

        games = Game.getAllGames();
        gameSpinnerAdapter = new ArrayAdapter<Game>(this,
                android.R.layout.simple_list_item_1, games);
        gameSpinner.setAdapter(gameSpinnerAdapter);

    }
}
```

Whose turn is it?

Lets stop here…

Confused? Ask me