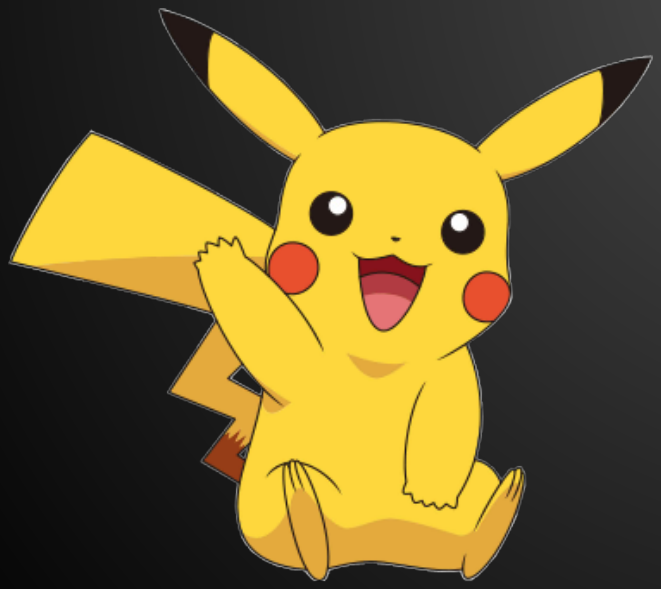
Gotta Persist ‘Em All: Realm as Replacement for SQLite

Siena Aguayo
Indiegogo
@dotheastro

Overview

- What Realm Is
- Pros and cons
- Compare and contrast with SQLite
- What Realm Isn’t

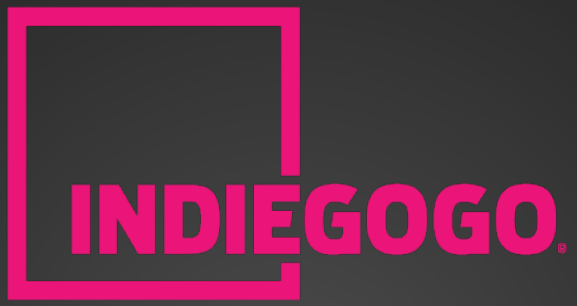
But First, Hello!

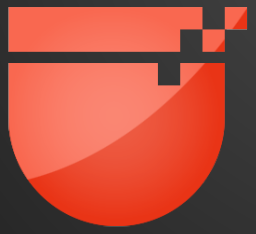


Pokemon Master

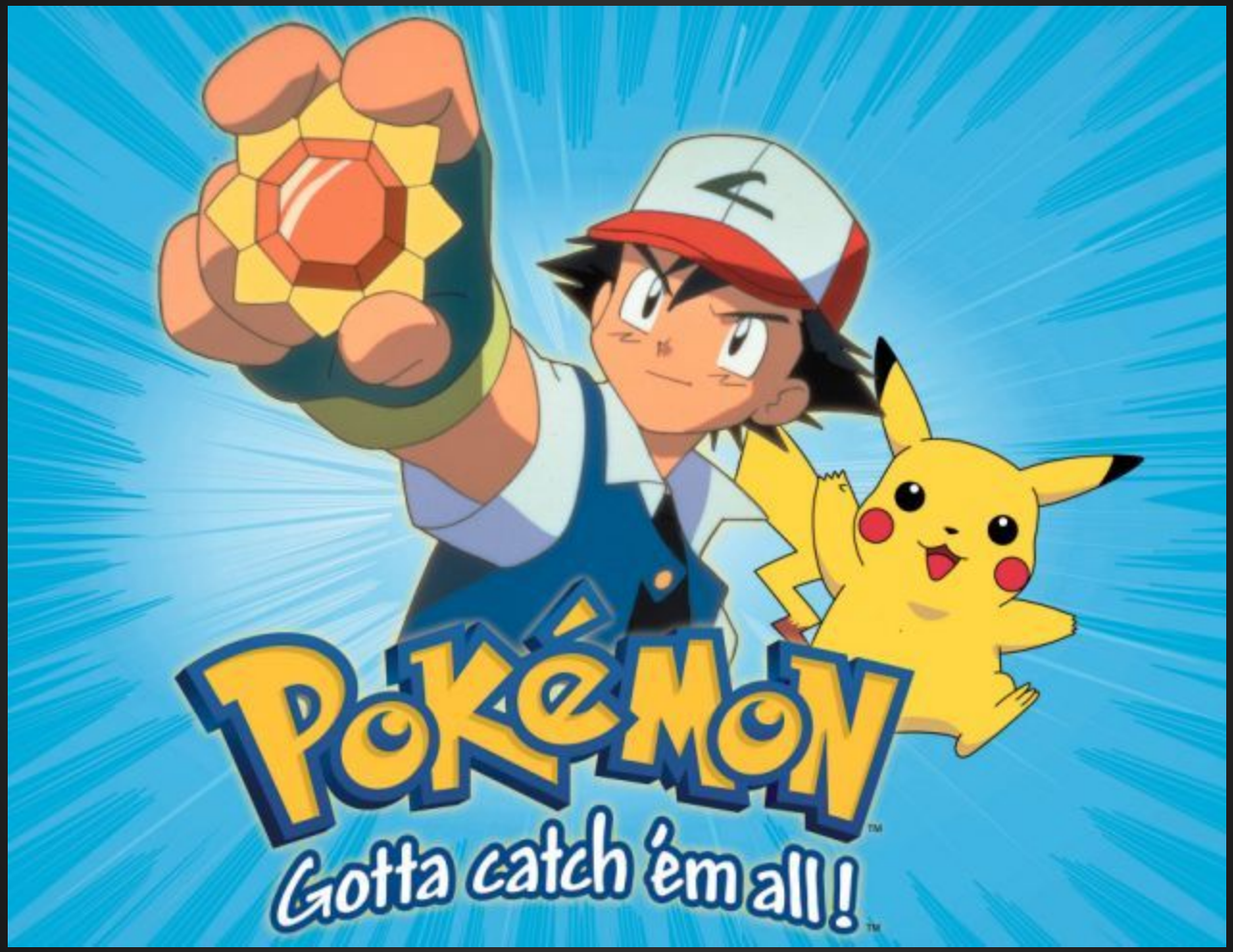
Siena Aguayo

droidcon Montreal 2015

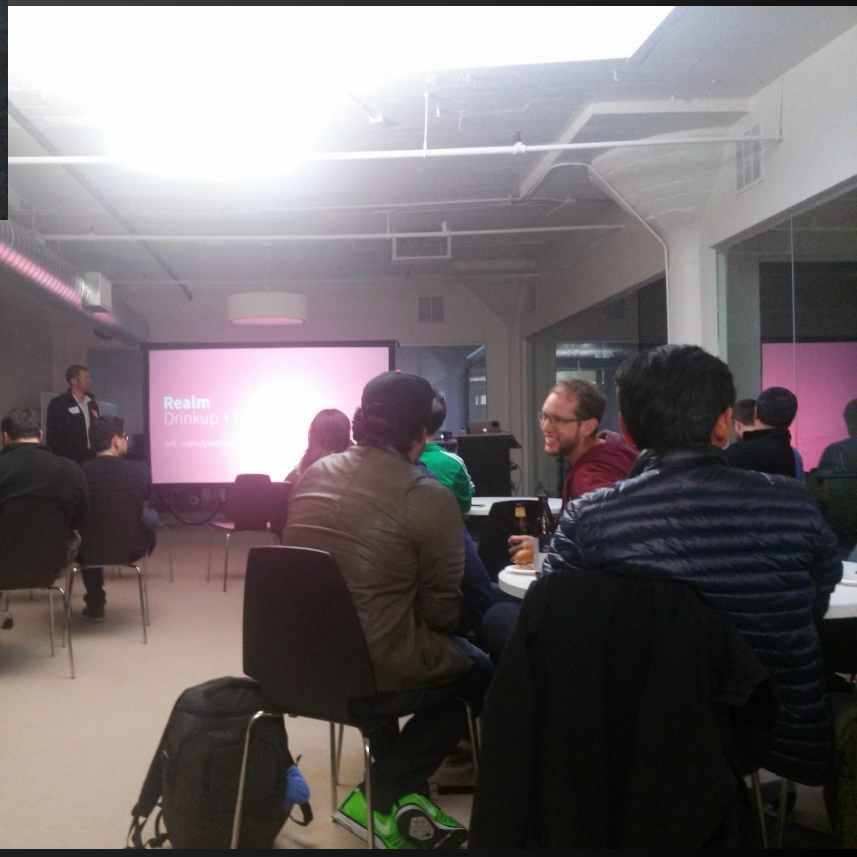




Hackbright
Academy


Pokémon
Gotta catch 'em all!


Player
Siena
We hereby certify that this Trainer
has achieved the admirable feat of
completing the National Pokédex
with grace and much poise.
GAME FREAK
Staff



Realm Meetup

Drinkup + Roadmap
March 24th, 2015
San Francisco

What Realm Is

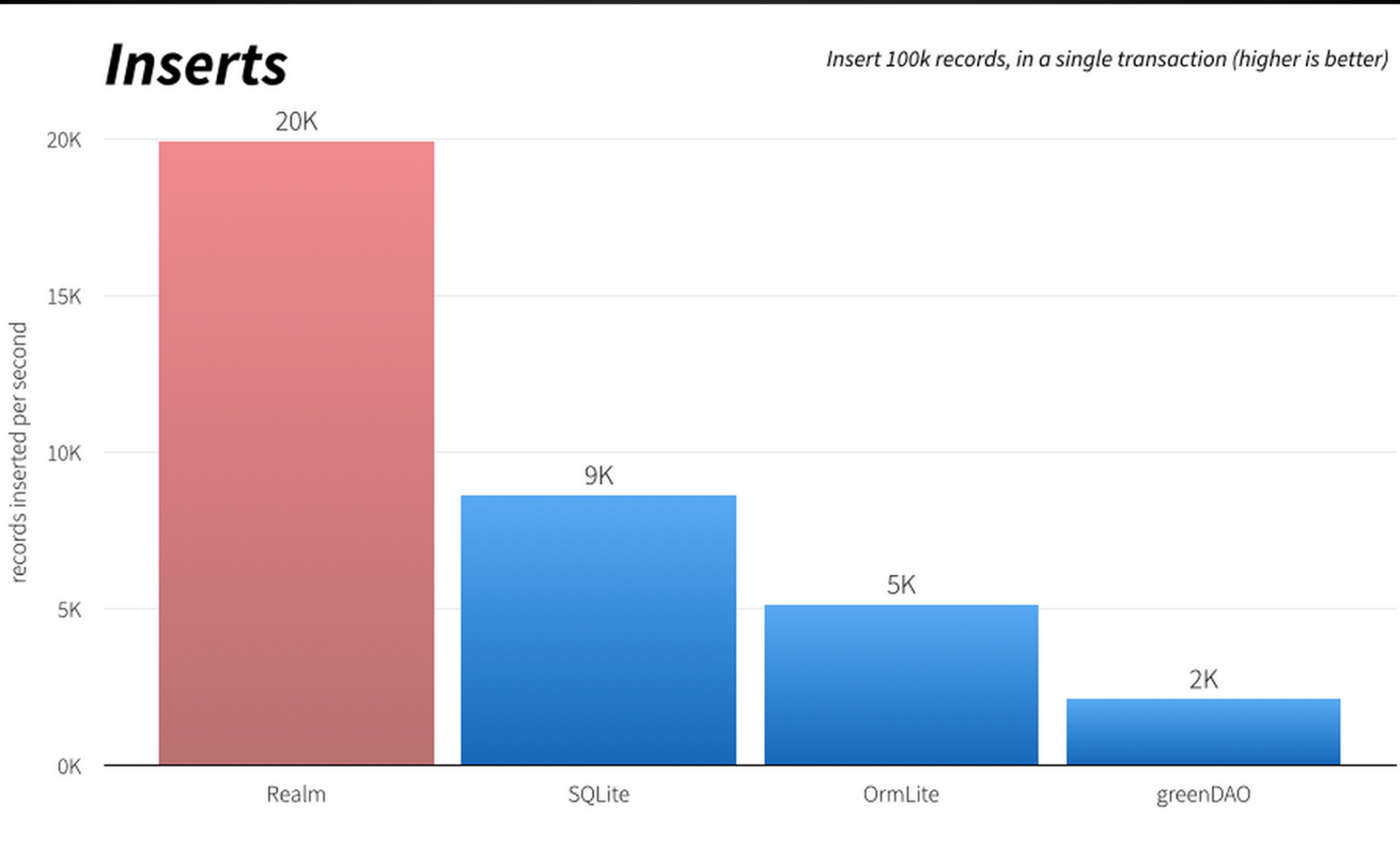
Realm

- “embedded mobile database”
- Core is written in C++
- Available for Android and iOS
- Second most-deployed mobile database in the world

Pros

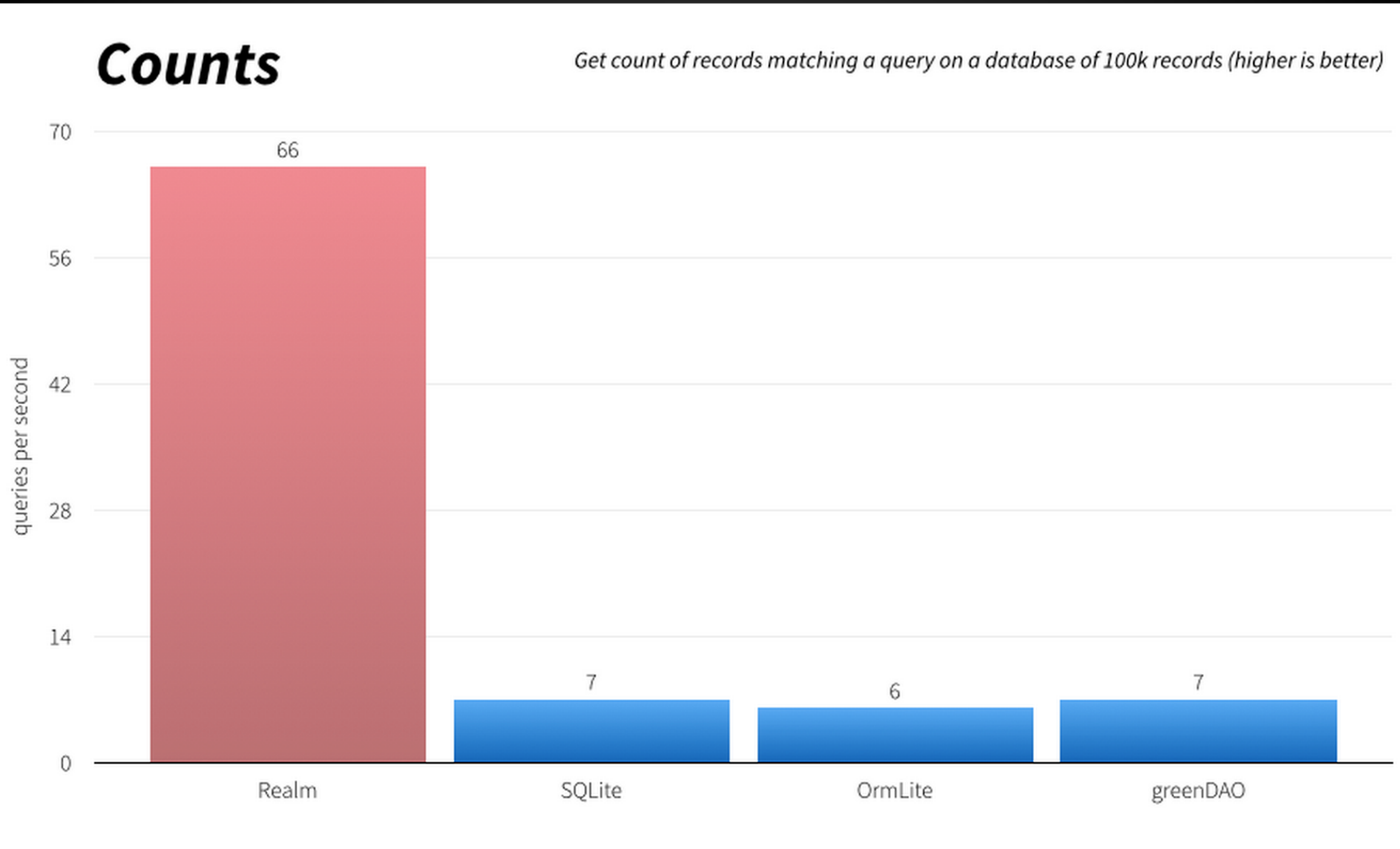
Pros for Realm

- Easy to use
- Object conversion handled for you
- Convenient for creating and storing data on the fly
- Faster than SQLite
- Very responsive team



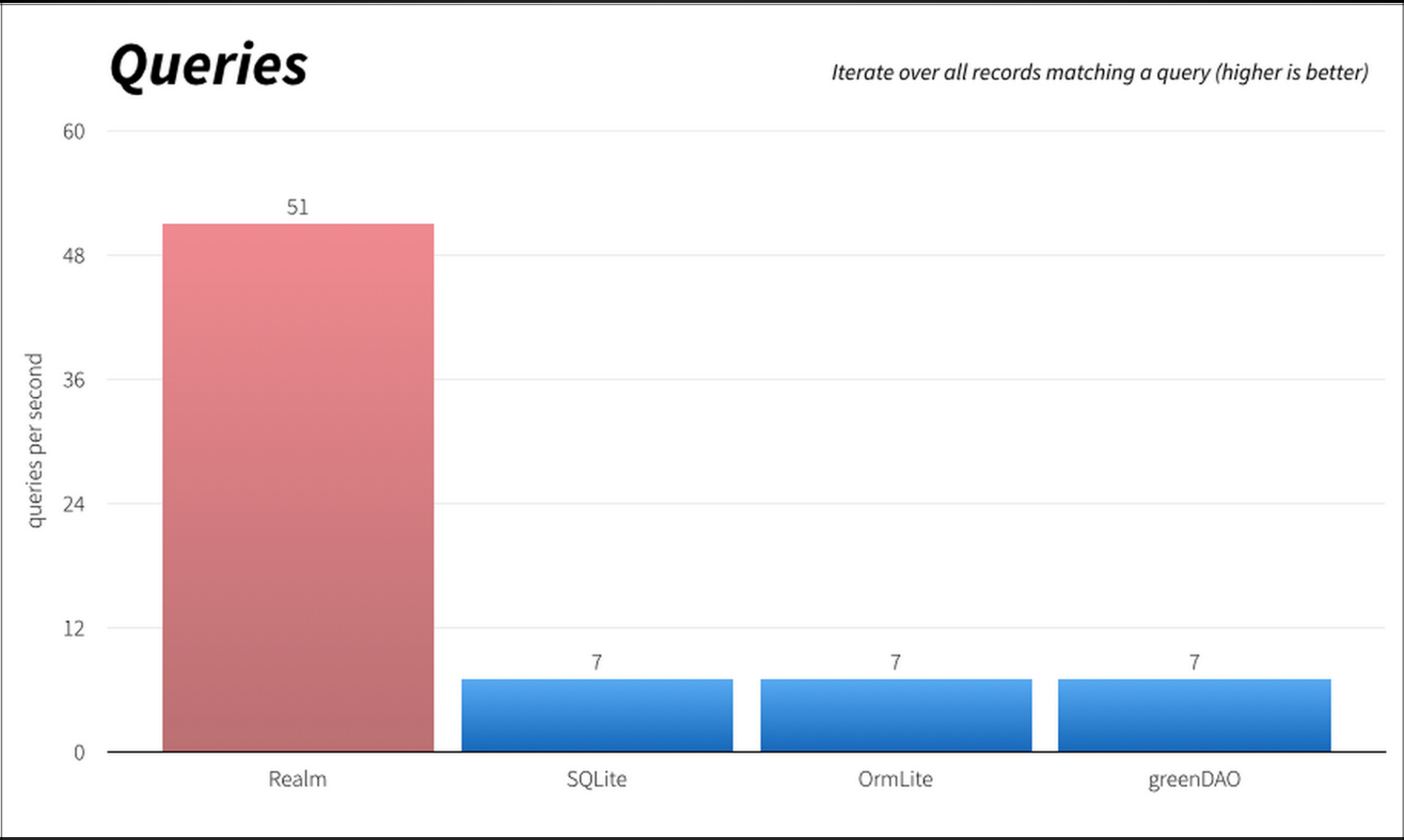
Tests run on an Galaxy S3, using the latest available version of each library as of Sept 28, 2014.

Source: https://realm.io/news/realm-for-android/#realm-for-android



Tests run on an Galaxy S3, using the latest available version of each library as of Sept 28, 2014.

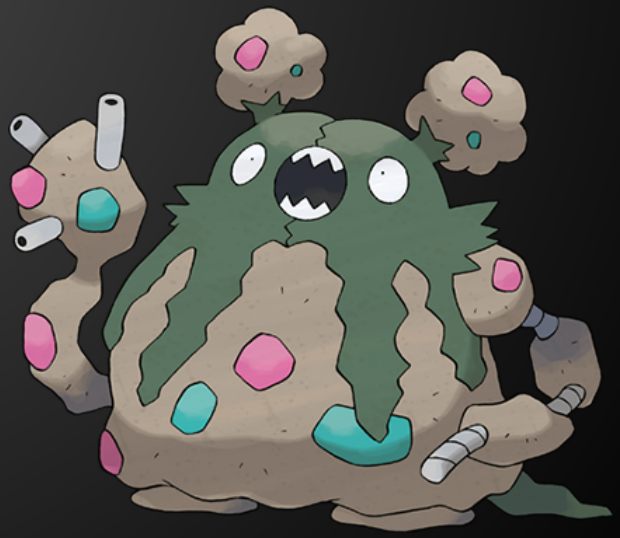
Source: https://realm.io/news/realm-for-android/#realm-for-android



Tests run on an Galaxy S3, using the latest available version of each library as of Sept 28, 2014.

Source: https://realm.io/news/realm-for-android/#realm-for-android

Cons

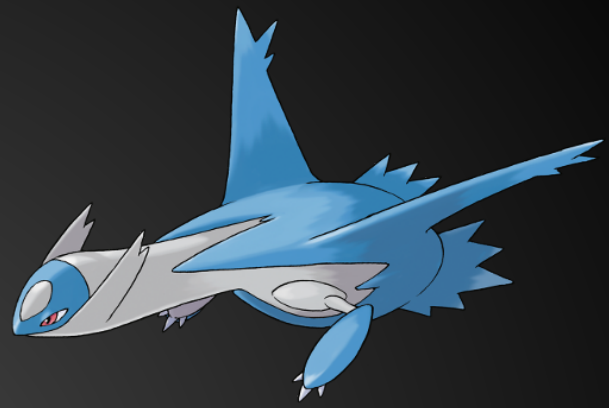


Cons for Realm

- No importing
- Still under active development
- Have to create with Java on the device
- Not a lot of content online
- Can’t access objects across threads

Missing Features

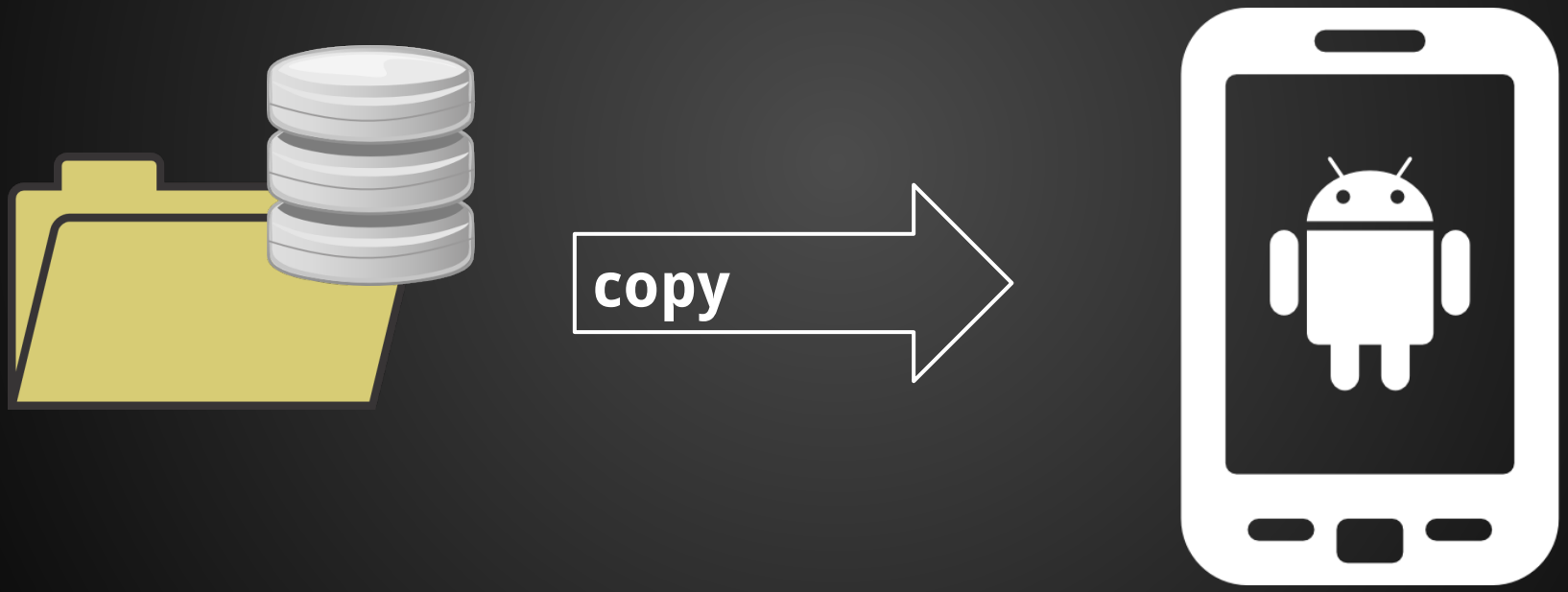
- Null support
- Auto-incrementing ids
- Map<K, V> support
- Easy migrations (exist, but are painful)
- Notifications on specific data changed
- Compound primary keys
- Testing with Robolectric



Compare and Contrast with SQLite

Database Connection and Setup

Shipping with Existing Database



SQLite: SQLiteAssetHelper
Realm: Sample migration app

Shipping with Existing Database

SQLite:

`dbHelper.getReadableDatabase()` or
`dbHelper.getWritableDatabase()`

Realm:

`Realm.getInstance(context)`

Creating Schema

SQLite

```
CREATE TABLE `pokemon` (
  `id`  INTEGER NOT NULL,
  `identifier`  VARCHAR(79) NOT NULL,
  `species_id`  INTEGER,
  `height`  INTEGER NOT NULL,
  `weight`  INTEGER NOT NULL,
  `base_experience` INTEGER NOT NULL,
  `order` INTEGER NOT NULL,
  `is_default`  BOOLEAN NOT NULL,
  PRIMARY KEY(id),
  FOREIGN KEY(`species_id`) REFERENCES pokemon_species ( id )
);
```

Creating Schema

Realm

```
public class Pokemon extends RealmObject {
  @PrimaryKey private int id;
  private String identifier;
  private int speciesId, height, weight, baseExperience, order;
  private boolean isDefault;
  private RealmList<PokemonType> types;
  private RealmList<Encounter> encounters;

  // constructors, getters, setters
}
```

```java
public class Pokemon extends RealmObject {
  @PrimaryKey private int id;
  private String identifier;
  private int speciesId, height, weight, baseExperience, order;
  private boolean isDefault;
  private RealmList<PokemonType> types;
  private RealmList<Encounter> encounters;

  public Pokemon() { }

  public void setId(int id) {
    this.id = id;
  }

  public String getIdentifier() {
    return identifier;
  }

  public void setIdentifier(String identifier) {
    this.identifier = identifier;
  }

  public int getSpeciesId() {
    return speciesId;
  }

  public void setSpeciesId(int speciesId) {
    this.speciesId = speciesId;
  }

  public int getHeight() {
    return height;
  }

  public void setHeight(int height) {
    this.height = height;
  }
```

No args constructor

Completely vanilla
getters and setters

NO custom logic in your models

In Realm, your models ARE your schema.

Reading Data

Java with SQLite

```
public List<Integer> getPokemonTypeData(int id) {
  String intString = Integer.toString(id);

  Cursor cursor = getData(
    "SELECT type_id FROM pokemon_types WHERE pokemon_id = " + intString +
    " ORDER BY slot");
  cursor.moveToFirst();
  ArrayList<Integer> types = new ArrayList<Integer>();
  while (!cursor.isAfterLast()) {
    types.add(cursor.getInt(0)); // this is the real killer
    cursor.moveToNext();
  }
  cursor.close();

  return types;
}
```

Java with Realm

Find All

```
RealmResults<PokemonType> types =
  realm.where(PokemonType.class)
    .equalTo("pokemonId", pokemon.getId())
    .findAll();
```

Find First

```
PokemonType pokemonType =
  realm.where(PokemonType.class)
    .equalTo("pokemonId", pokemon.getId())
    .findFirst();
```

Writing Data

Java with SQLite

```
public void addBulbasaur() {
  SQLiteDatabase db = dbHelper.getWriteableDatabase();

  ContentValues values = new ContentValues();
  values.put("id", 1);
  values.put("identifier", 'bulbasaur');
  values.put("height", 7);
  values.put("weight", 69);

  db.insert("pokemon", null, values);
  db.close();
}
```

Java with Realm

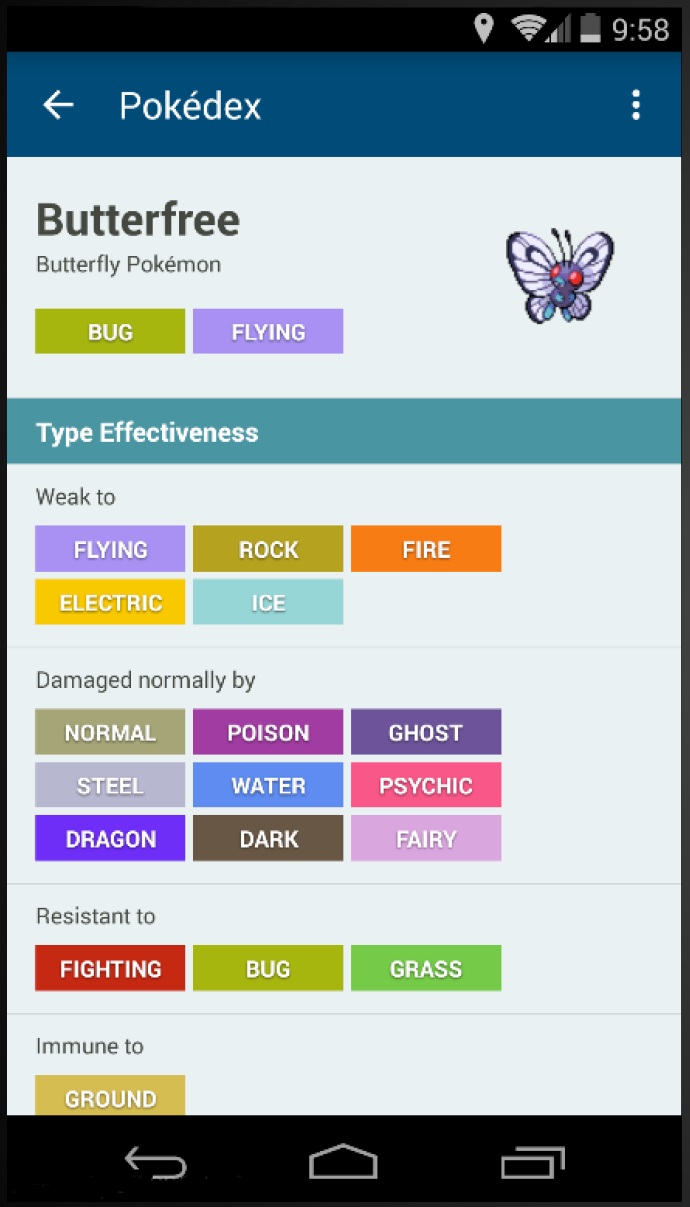
```
public void addBulbasaur() {
  Realm realm = Realm.getInstance();
  realm.beginTransaction();

  Pokemon bulbasaur = realm.createObject(Pokemon.class);
  bulbasaur.setId(1);
  bulbasaur.setIdentifier('bulbasaur');
  bulbasaur.setHeight(7);
  bulbasaur.setWeight(69);

  realm.commitTransaction();
  realm.close();
}
```

Java with Realm

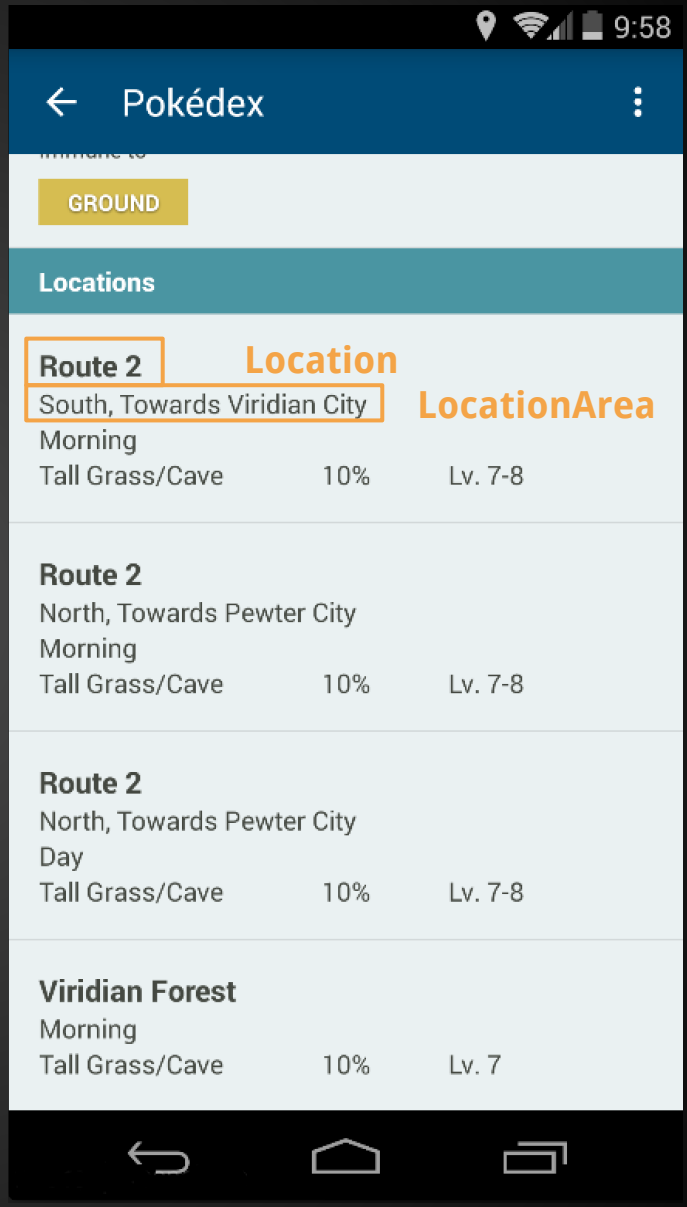
```
public void addBulbasaur() {
  Realm realm = Realm.getInstance();

  realm.executeTransaction(new Realm.Transaction() {
    @Override public void execute(Realm realm) {
      Pokemon bulbasaur = realm.createObject(Pokemon.class);
      bulbasaur.setId(1);
      bulbasaur.setIdentifier('bulbasaur');
      bulbasaur.setHeight(7);
      bulbasaur.setWeight(69);
    }
  });

  realm.close();
}
```

Adding Relationships and Complex Queries

Encounter has a LocationArea
LocationArea has a Location


9:58
Pokédex
Butterfree
Butterfly Pokémon
BUG
FLYING
Type Effectiveness
Weak to
FLYING
ROCK
FIRE
ELECTRIC
ICE
Damaged normally by
NORMAL
POISON
GHOST
STEEL
WATER
PSYCHIC
DRAGON
DARK
FAIRY
Resistant to
FIGHTING
BUG
GRASS
Immune to
GROUND

Encounter


9:58
Pokédex
GROUND
Locations
Route 2
Location
South, Towards Viridian City
LocationArea
Morning
Tall Grass/Cave 10% Lv. 7-8
Route 2
North, Towards Pewter City
Morning
Tall Grass/Cave 10% Lv. 7-8
Route 2
North, Towards Pewter City
Day
Tall Grass/Cave 10% Lv. 7-8
Viridian Forest
Morning
Tall Grass/Cave 10% Lv. 7

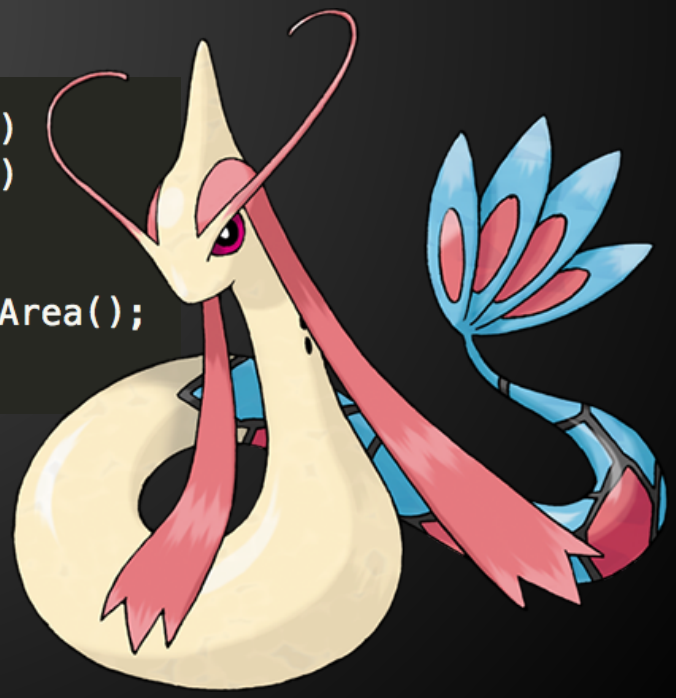
Raw SQL

```
SELECT encounters.version_id, location_names.name, location_area_prose.name,
  encounters.min_level, encounters.max_level, encounters.id FROM encounters
  INNER JOIN location_areas ON encounters.location_area_id = location_areas.id
  INNER JOIN locations ON location_areas.location_id = locations.id
  INNER JOIN location_names ON locations.id = location_names.location_id
  INNER JOIN location_area_prose ON encounters.location_area_id =
    location_area_prose.location_area_id
  WHERE encounters.pokemon_id = 322 AND location_names.local_language_id = 9
    AND location_area_prose.local_language_id = 9;
```

Java with Realm

```
RealmResults<Encounter> encounters = realm.where(Encounter.class)
                                          .equalTo("pokemonId", 322)
                                          .findAll();
```

```
Encounter encounter = realm.where(Encounter.class)
                           .equalTo("pokemonId", 322)
                           .findFirst();

LocationArea locationArea = encounter.getLocationArea();
Location location = locationArea.getLocation();
```



```java
public class Encounter extends RealmObject {
  @PrimaryKey private long id;
  private int versionId, encounterSlotId, minLevel, maxLevel, pokemonId;
  private LocationArea locationArea;
  private EncounterSlot encounterSlot;
  private int encounterConditionId;

  // constructors, getters, setters
}
```

What Realm Isn’t

Realm says they are not an ORM because your data is not copied.

An ORM or not?

Typical ORM usage

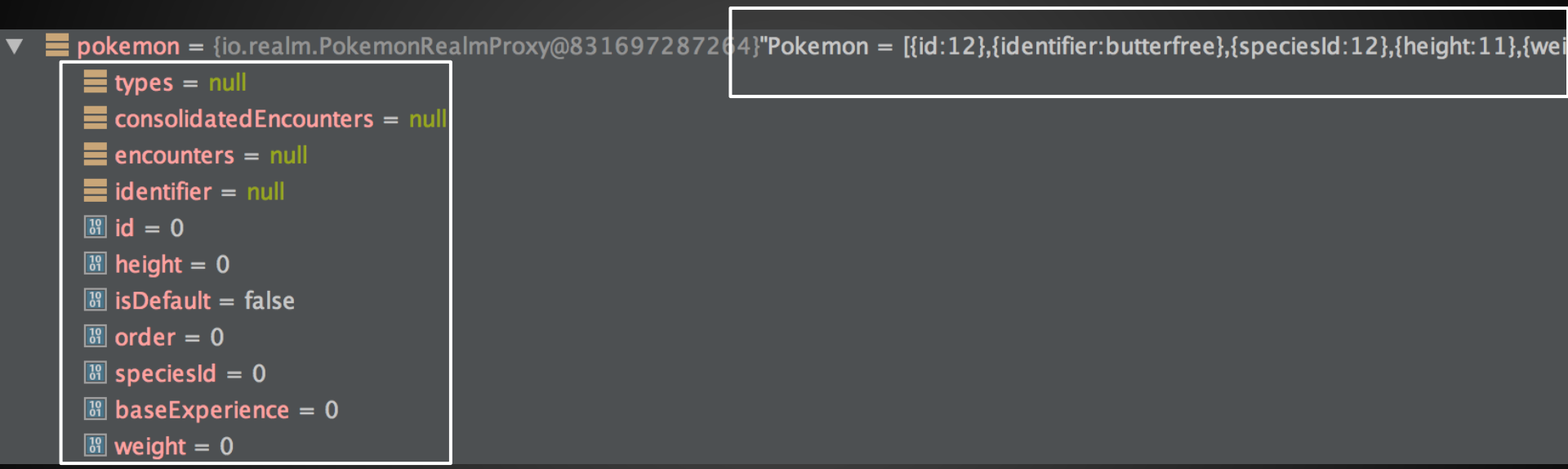
SQLite

```
>> pragma table_info(pokemon)
0|id|INTEGER|1||1
1|identifier|VARCHAR(79)|1||0
2|height|INTEGER|1||0
3|weight|INTEGER|1||0
```

Java

```
public class Pokemon {
  private int id;
  private String identifier;
  private int height;
  private int weight;
  // constructor(s),
  // getters, setters
}
```

Data from Realm in the toString()...



Member variables are all uninitialized...

Shout-Out to Other ORMs

- greenDAO
- Active Android
- Sugar ORM
- ORMLite (Java)
- Cupboard

Michael Pardo's Android Data talk

Conclusion

Acknowledgments

- Realm: http://realm.io/
- SQLite db from veekun's Pokémon repo: https://github.com/veekun/pokedex

Questions?

Twitter: @dotheastro