

GitHub

This repository Search

Explore Features Enterprise Blog

Sign up

Sign in



realm / realm-cocoa

★ Star

2,377

🍴 Fork

161

Realm is a mobile database: a replacement for Core Data & SQLite <http://realm.io>

1,566 commits

43 branches

60 releases

Contributors

Master

realm-cocoa / +

Merge pull request #1078 from

3 days ago

leproj

amic fr

Object no retain its object

g and

examples

Added Group, view example entries in examples/README.md

5 days ago

plugin

Switch the file name back to rlm_lldb.py

3 days ago

scripts

Move all of the fat framework building logic to build.sh

25 days ago

tools/RealmBrowser

Package the lldb script with the plugin

3 days ago

.dir-locals.el

Project specific Emacs settings updated: Always show trailing white-s...

a year ago

.gitignore

The core folder is now a symlink to a versioned folder

4 months ago

CHANGELOG.md

Changelog and setup instructions

3 days ago

CONTRIBUTING.md

small modifications from PR feedback

2 months ago

<> Code

Issues

46

Pull requests

12

Discussions

Packages

Graphs

HTTPS clone URL

<https://github.com/realm/>

You can clone with [HTTPS](#) or [Subversion](#). [?](#)

Clone in Desktop

Download ZIP

What is Realm?

- **Fast, zero-copy, embedded database**
- **Used in apps with *millions* of users**
- **NoSQL**
- **Full ACID transactions**
- **Well defined threading model**
- **Cross-platform C++ core with many language bindings**
(currently Objective-C, Swift & Android)

Open Source*



github.com/realm/realm-cocoa

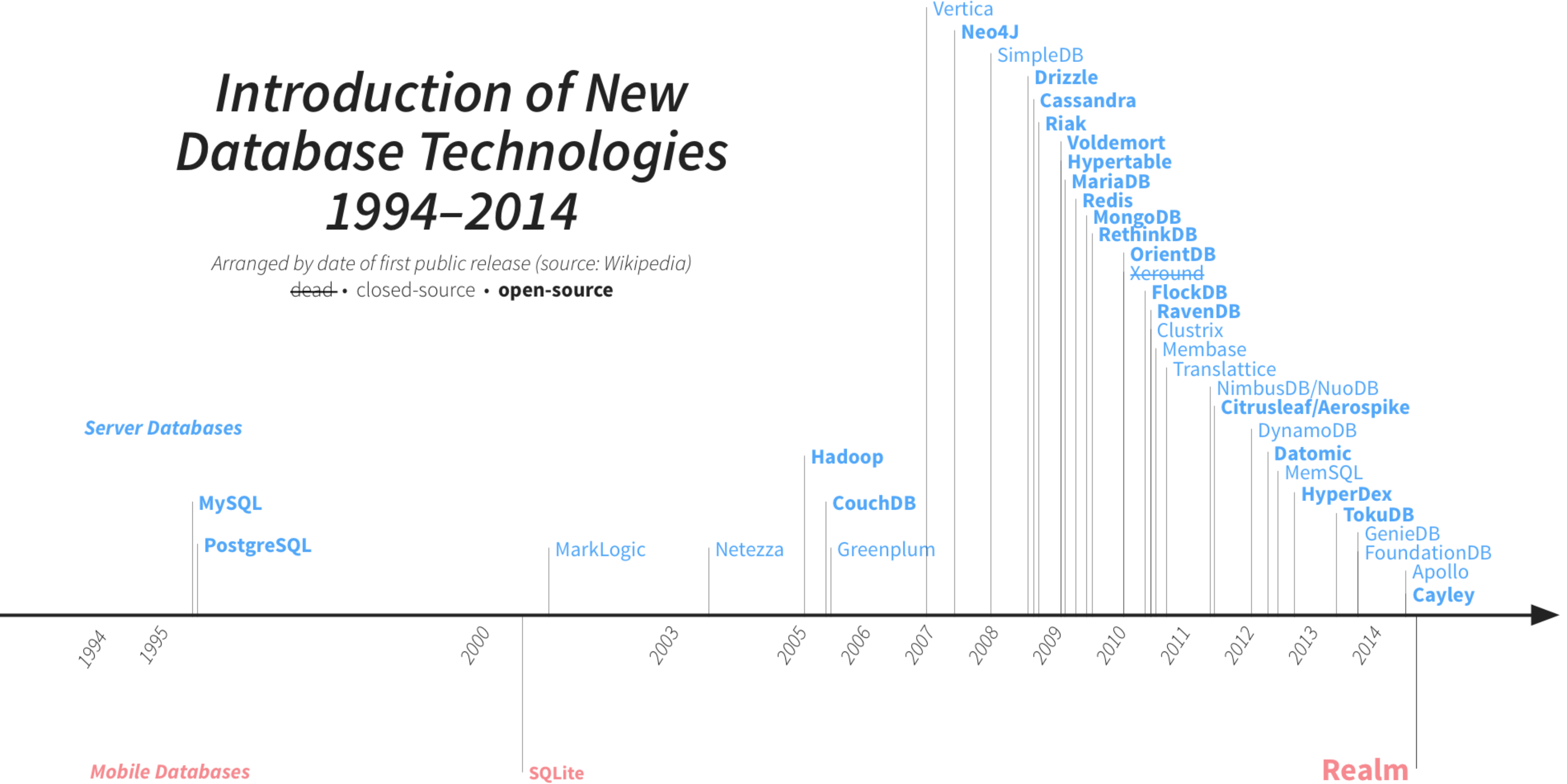
*** Bindings 100% open source, C++ core launching as
Apache 2**

Why?

Introduction of New Database Technologies 1994–2014


Arranged by date of first public release (source: Wikipedia)

~~dead~~ • closed-source • **open-source**



Current State of iOS Persistence

Core Data

- Full-featured & mature
- 10+ years old ORM for SQLite
- Slow
- Complex and difficult to learn/debug
-  only

SQLite

- Faster than Core Data
- Optimized for iOS
- Cross Platform
- 14+ years old
- Bad user experience
- Manual mapping and queries
- Lack of thread safety

Other Options

- FMDB
- YapDatabase
- CouchbaseLite
- LevelDB

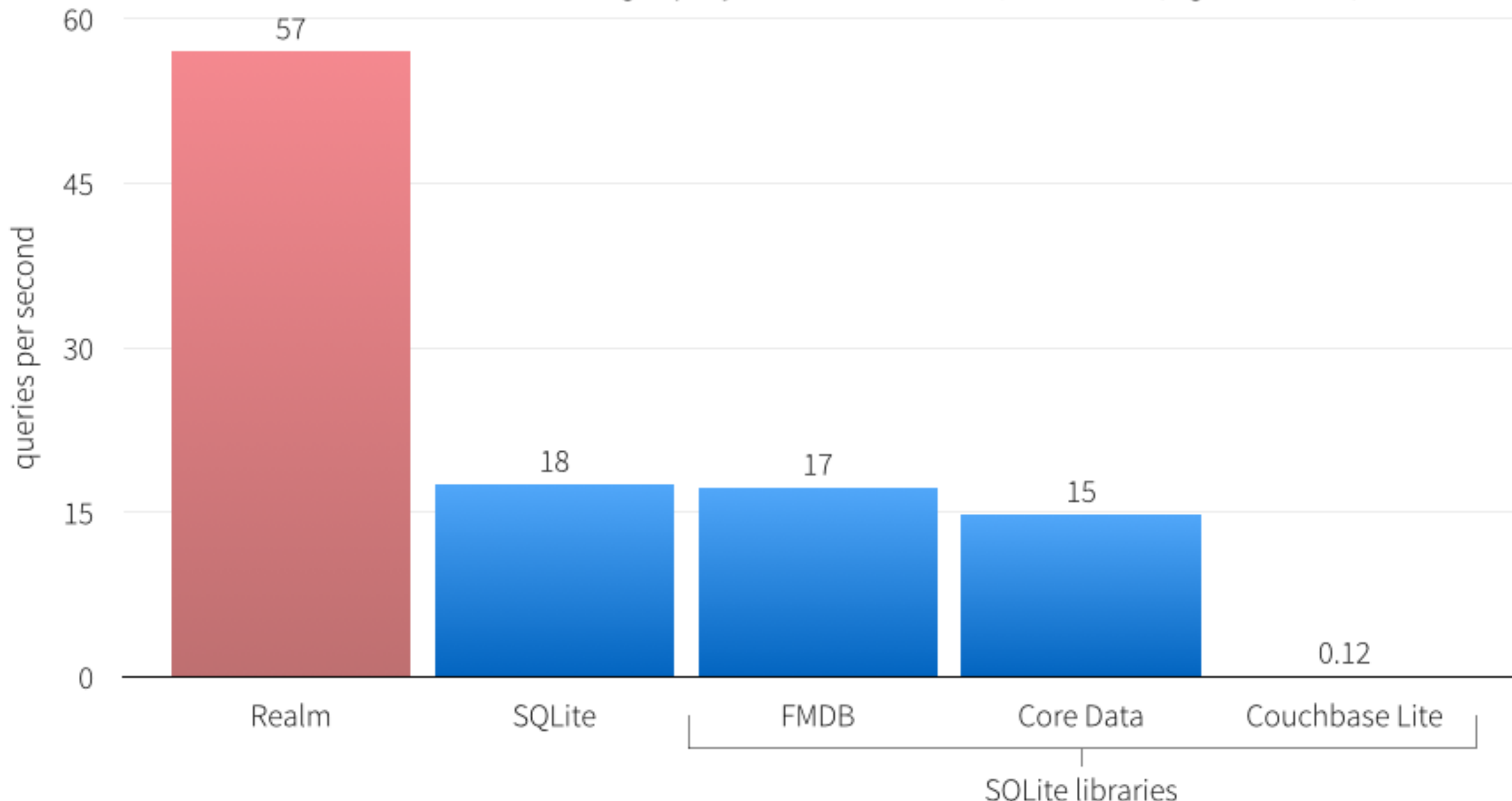
Lots has changed in last decade

- Smartphone Revolution
- NoSQL
- Need for Sync

Benchmarks

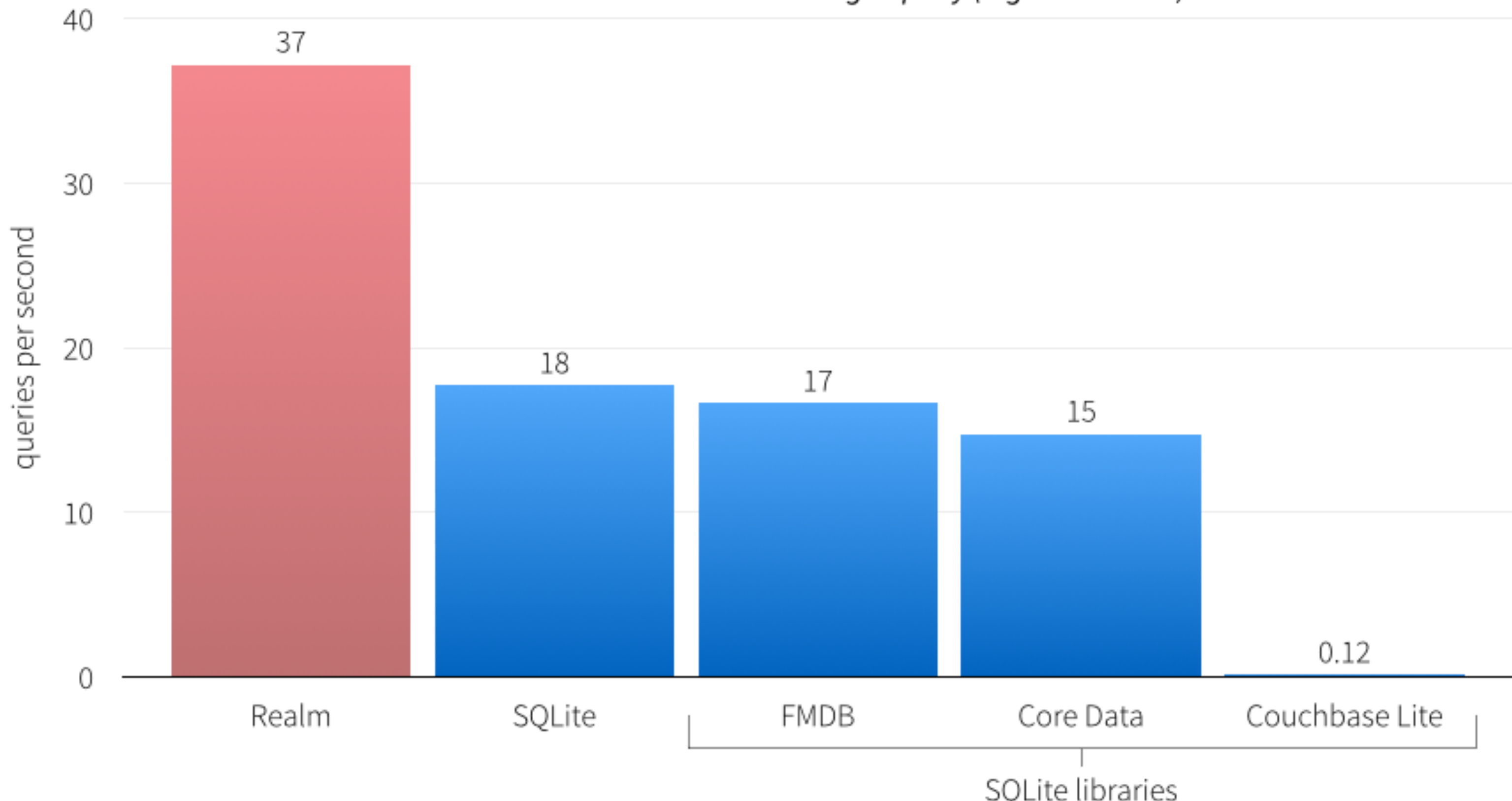
Counts

Get count of records matching a query on a database of 150,000 records (higher is better)



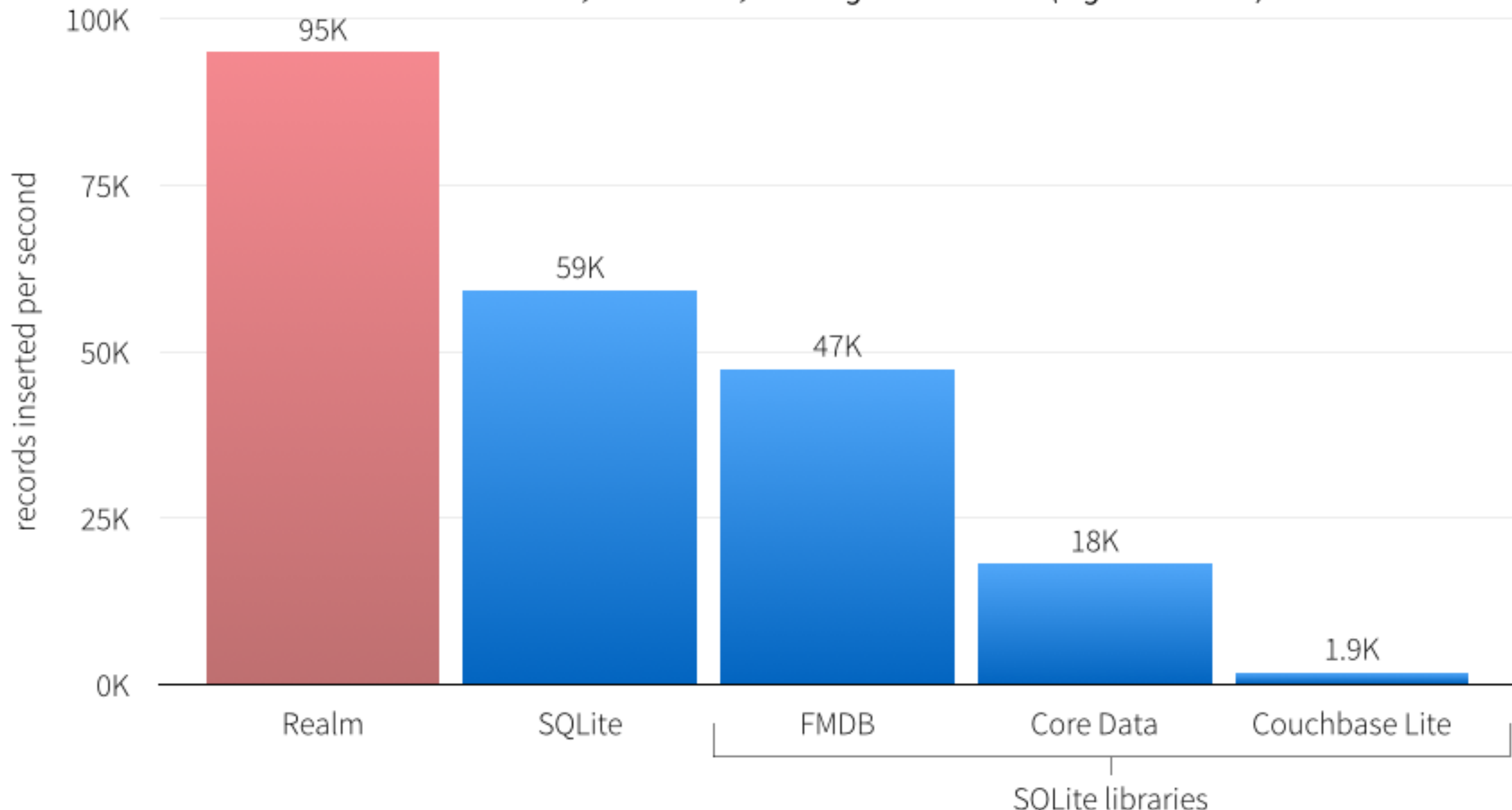
Queries

Iterate over all records matching a query (higher is better)



Inserts

Insert 150,000 records, in a single transaction (higher is better)



Realm Models

```
@interface Employee : RLMObject
@property NSString *name;
@property NSDate *startDate;
@property float salary;
@property BOOL fullTime;
@end
```

```
RLM_ARRAY_TYPE(Employee)
```

```
@interface Company : RLMObject
@property NSString *name;
@property Employee *ceo;
@property RLMArray<Employee> *employees;
@end
```

Realm Models (Swift)

```
class Employee: Object {
    dynamic var name = "" // you can specify defaults
    dynamic var startDate = NSDate()
    dynamic var salary = 0.0
    dynamic var fullTime = true
}

class Company: Object {
    dynamic var name = ""
    dynamic var ceo: Employee? // optional. who needs CEO's?!
    let employees = List<Employee>()
}
```


Using Realm

```
// Using Realm Objects
```

```
Company *company = [[Company alloc] init];  
company.name = @"Realm"; // etc...
```

```
// Transactions
```

```
RLMRealm *realm = [RLMRealm defaultRealm];  
[realm transactionWithBlock:^(  
    [realm addObject:company];  
)];
```

```
// Querying objects
```

```
RLMArray *companies = [Company allObjects];  
RLMArray *FTEmployees = [Employee objectsWhere:@"fullTime == YES"];
```

Using Realm (Swift)

```
let company = Company() // Using Realm Objects  
company.name = "Realm" // etc...
```

```
defaultRealm().write { // Transactions  
    defaultRealm().add(company)  
}
```

```
// Queries
```

```
let companies = objects(Company)  
companies[0].name // => Realm (generics)  
let ftJacks = objects(Employee) // "Jack"s who work full time  
    .filter("fullTime == true && name == Jack")
```

Setting up Core Data

```
@lazy var managedObjectContext: NSManagedObjectContext = {
    let modelURL = NSBundle.mainBundle().URLForResource("SwiftTestOne", withExtension: "momd")
    let mom = NSManagedObjectModel(contentsOfURL: modelURL)
    ZAssert(mom != nil, "Error initializing mom from: \(modelURL)")

    let psc = NSPersistentStoreCoordinator(managedObjectModel: mom)

    let urls = NSFileManager.defaultManager().URLsForDirectory(.DocumentDirectory, inDomains: .UserDomainMask)
    let storeURL = (urls[urls.endIndex-1]).URLByAppendingPathComponent("SwiftTestOne.sqlite")

    var error: NSError? = nil

    var store = psc.addPersistentStoreWithType(NSSQLiteStoreType, configuration: nil, URL: storeURL, options: nil, error: &error)
    if (store == nil) {
        println("Failed to load store")
    }
    ZAssert(store != nil, "Unresolved error \(error?.localizedDescription), \(error?.userInfo)\nAttempted to create store at \(storeURL)")

    var managedObjectContext = NSManagedObjectContext()
    managedObjectContext.persistentStoreCoordinator = psc

    return managedObjectContext
}()
```

Setting up Realm

defaultRealm()

Work In Progress

- Change notifications
- Delete Rules
- Sync
- Support for more data types
- Open Source Core

Questions?

@simjp, jp@realm.io