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   utors

<> Code

ster   realm-cocoa / +   Issues   46

Merge   078 from

3 days a

eproj   amic f   25 day

Object n tain its obje   3 day

de   g and   27 day

examples   Added Gr w example entries in amples/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

Puls

Gra

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 <u>ACID</u> transactions**

- **Well defined threading model**

- **Cross-platform C++ core with many language bindings**
(currently Objective-C, Swift & Android)
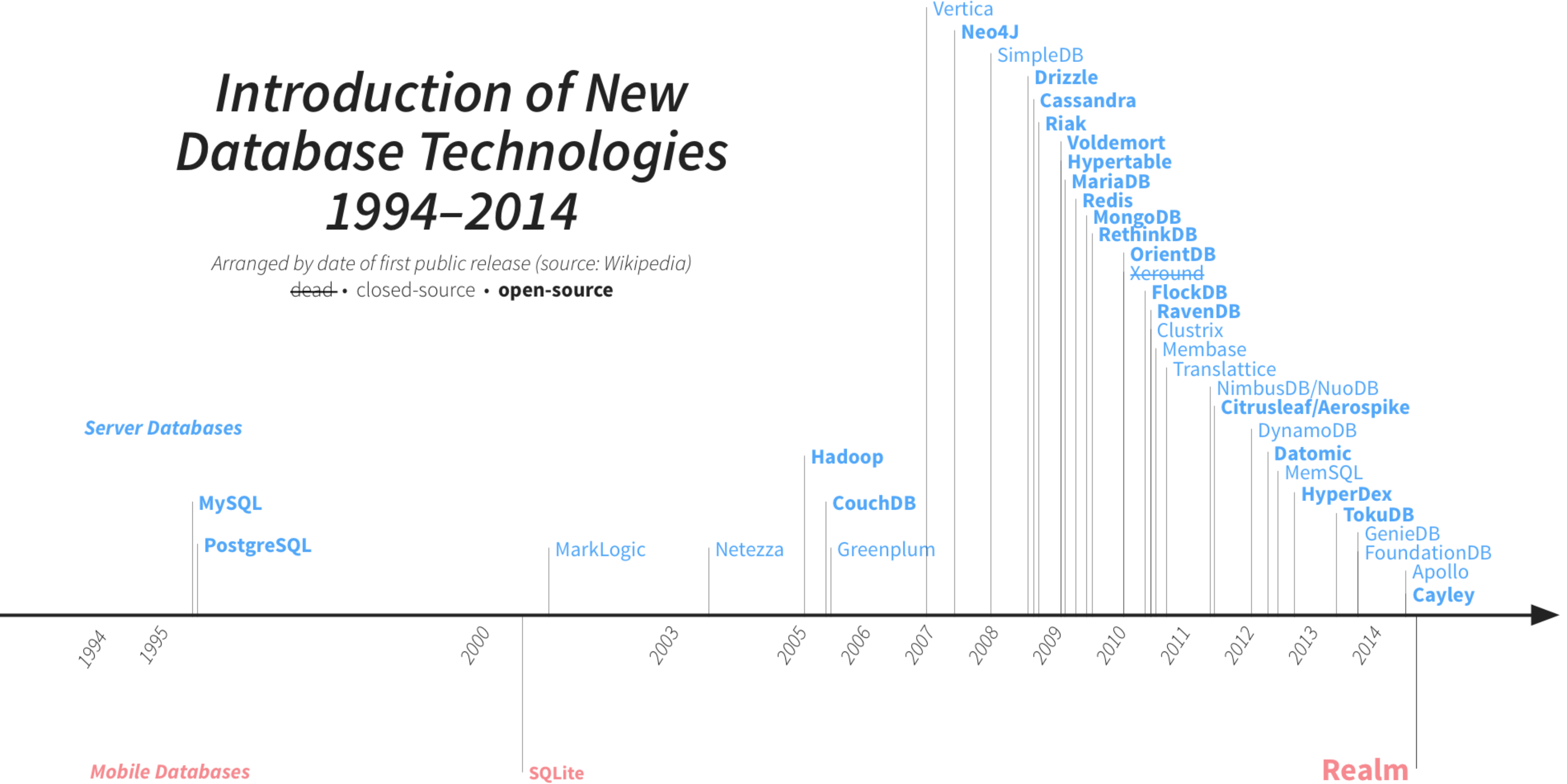
Open Source*

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**

Server Databases

Vertica
Neo4J
SimpleDB
Drizzle
Cassandra
Riak
Voldemort
Hypertable
MariaDB
Redis
MongoDB
RethinkDB
OrientDB
~~Xeround~~
FlockDB
RavenDB
Clustrix
Membase
Translattice
NimbusDB/NuoDB
Citrusleaf/Aerospike
DynamoDB
Datomic
MemSQL
HyperDex
TokuDB
GenieDB
FoundationDB
Apollo
Cayley

Hadoop

CouchDB

MySQL

PostgreSQL

MarkLogic   Netezza   Greenplum

1994 1995 2000 2003 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014

Mobile Databases

SQLite

Realm

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
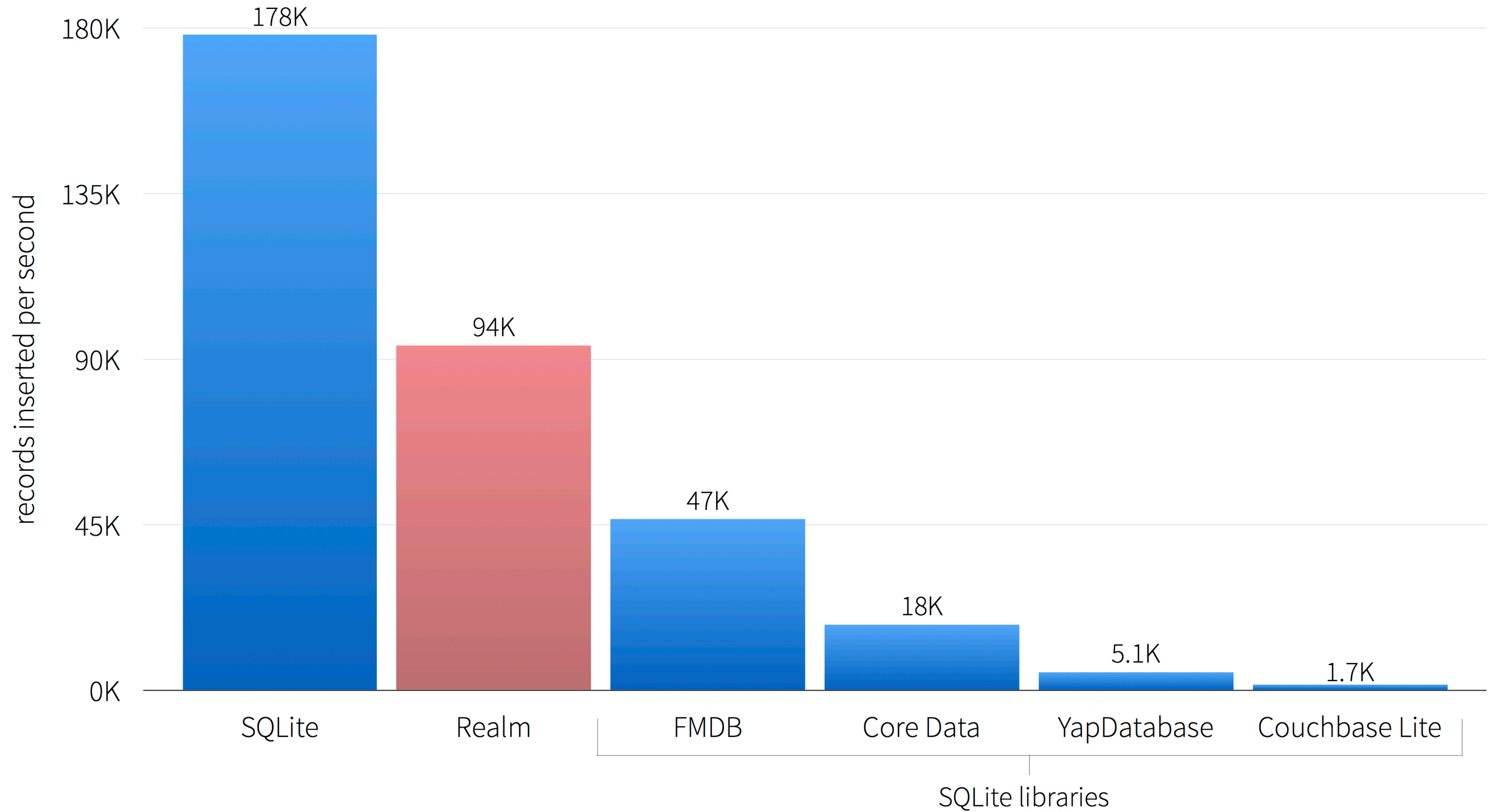
- CouchbaseLite

- LevelDB

Lots has changed in last decade

- Smartphone Revolution

- Low Connectivity

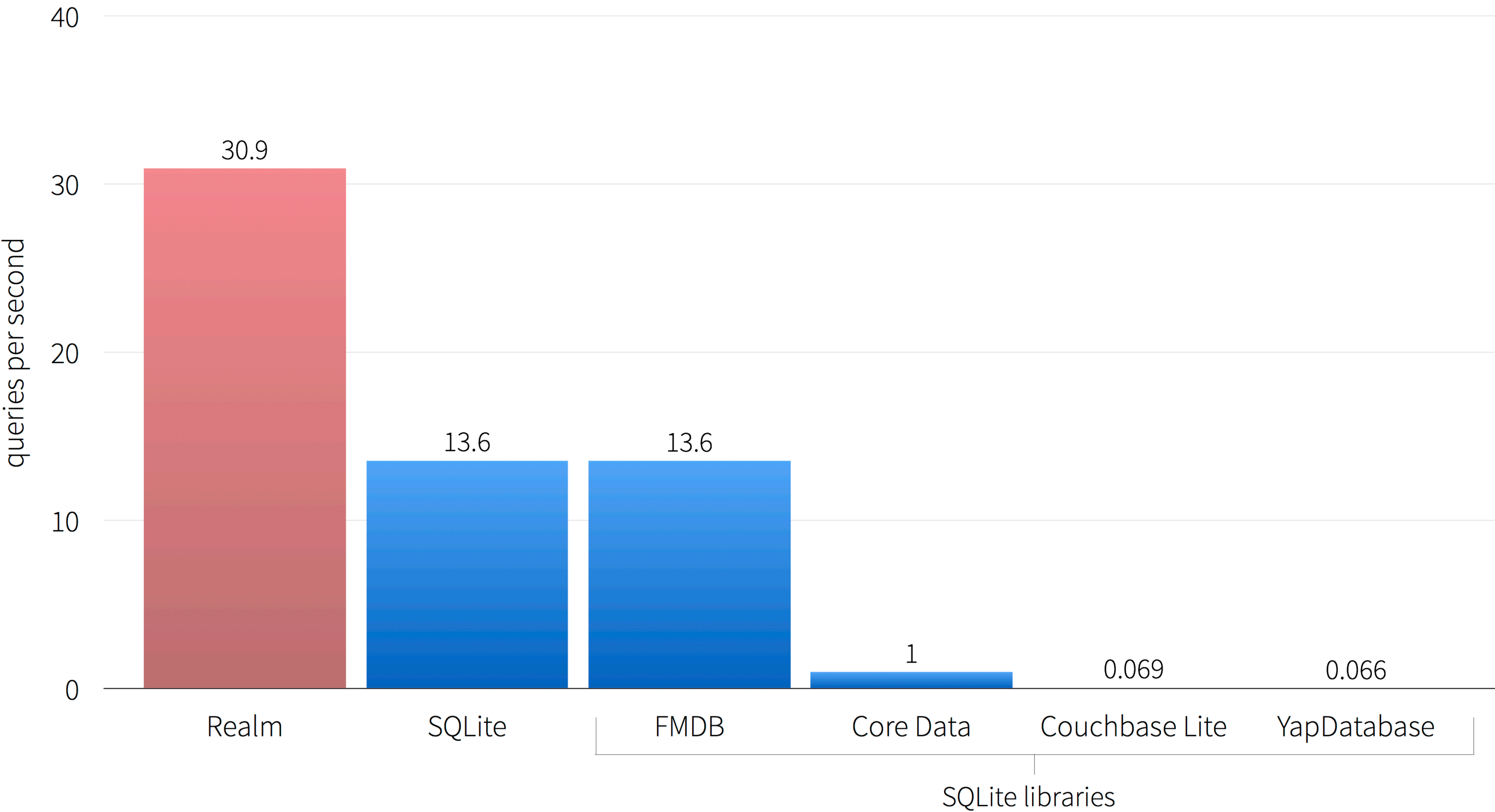- NoSQL

- Need for Sync

Benchmarks

Inserts

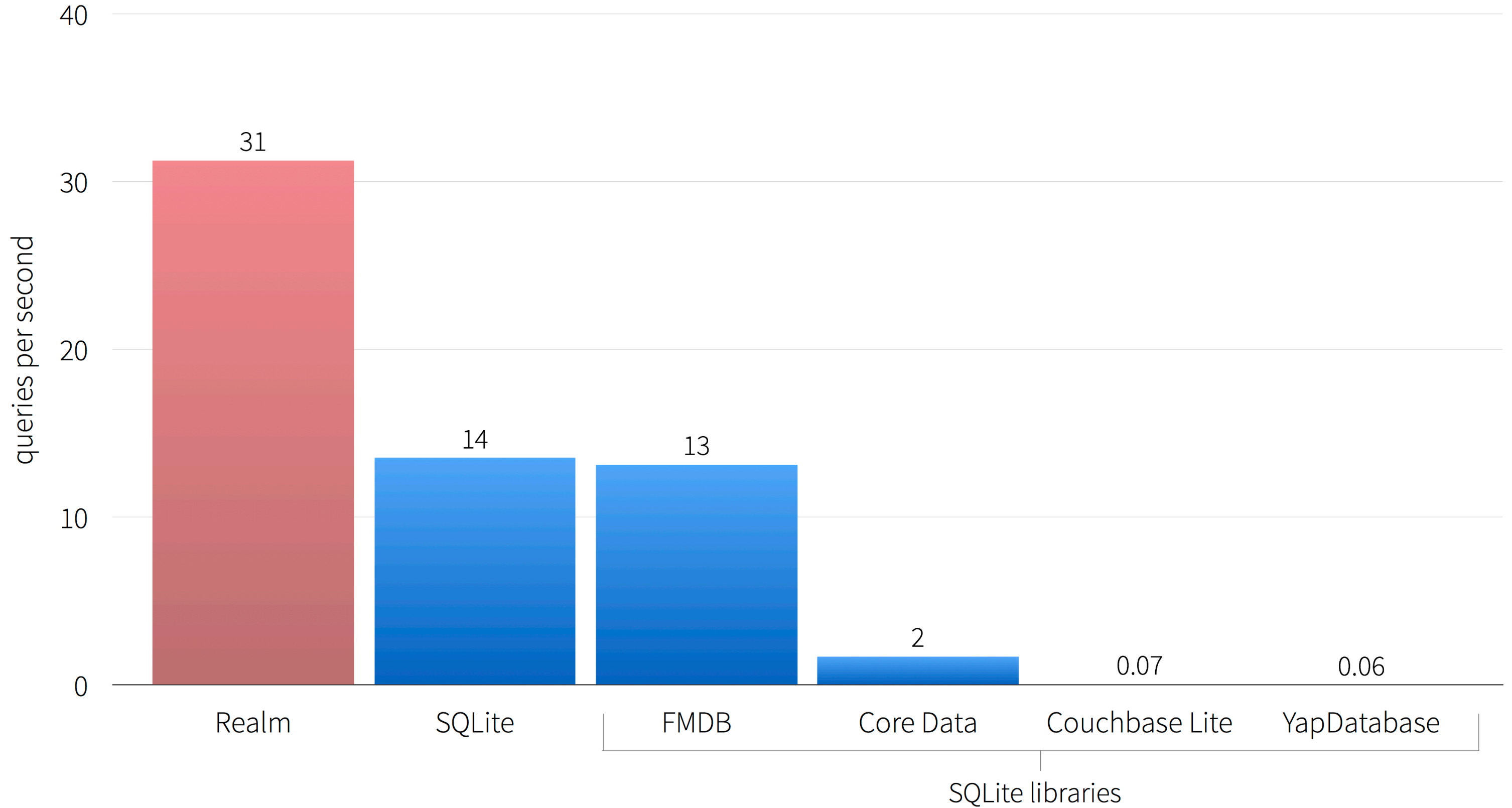*Insert 200k records, in a single transaction (higher is better)*

records inserted per second

180K	
135K	
90K	
45K	
0K	

178K — SQLite
94K — Realm
47K — FMDB
18K — Core Data
5.1K — YapDatabase
1.7K — Couchbase Lite

SQLite libraries

Counts

Get count of records matching a query on a database of 200k records (higher is better)

	queries per second	
Realm	30.9	
SQLite	13.6	
FMDB	13.6	
Core Data	1	
Couchbase Lite	0.069	
YapDatabase	0.066	

SQLite libraries

Queries

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

queries per second

Realm	31
SQLite	14
FMDB	13
Core Data	2
Couchbase Lite	0.07
YapDatabase	0.06

SQLite libraries

Realm Models

```objc
@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)

```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

```objc
// 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)

```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

```swift
@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