
GitHub, Inc.
GitHub
This repository
Search or type a command
Explore
Features
Enterprise
Blog
Sign up
Sign in
realm / realm-cocoa
Star 1,540
Fork 85
Realm
An insider's perspective
Realm-Xcode6.xcodeproj
Realm.xcodeproj
add support for querying nil objects
Realm
Fix sign comparison warnings
16 minutes ago
docs
- update docs readme
12 days ago
examples
9 days ago
plugin
scripts
Don't add a duplicate copy of libtightdb-ios.a to the framework
14 days ago
tools
Version bump
a day ago
.dir-locals.el
Project specific Emacs settings updated: Always show trailing white-s…
11 months ago
.gitignore
The core folder is now a symlink to a versioned folder
24 days ago
Pull
17
Wiki
Pulse
Graphs
HTTPS clone URL
https://github.com/realm/
Clone in Desktop
Download ZIP

Who am I?

- JP Simard
- Work at Realm
- **@simjp**
- **realm.io**
- Recently moved to San Francisco
- BUT...


From Ottawa
MOOSEHEAD
CANADIAN

This talk

- Overview of Realm
- Share perspective
- Share lessons learnt
- Gather feedback
- Not a pitch

What is Realm?

What is Realm?

- **Embedded database**
- **NoSQL**
- **Full ACID transactions**
- **C++ core**
- **Many language bindings** (only Objective-C & Swift released)

Open Source

github.com/realm/realm-cocoa

What Realm is *not*

What Realm is *not*

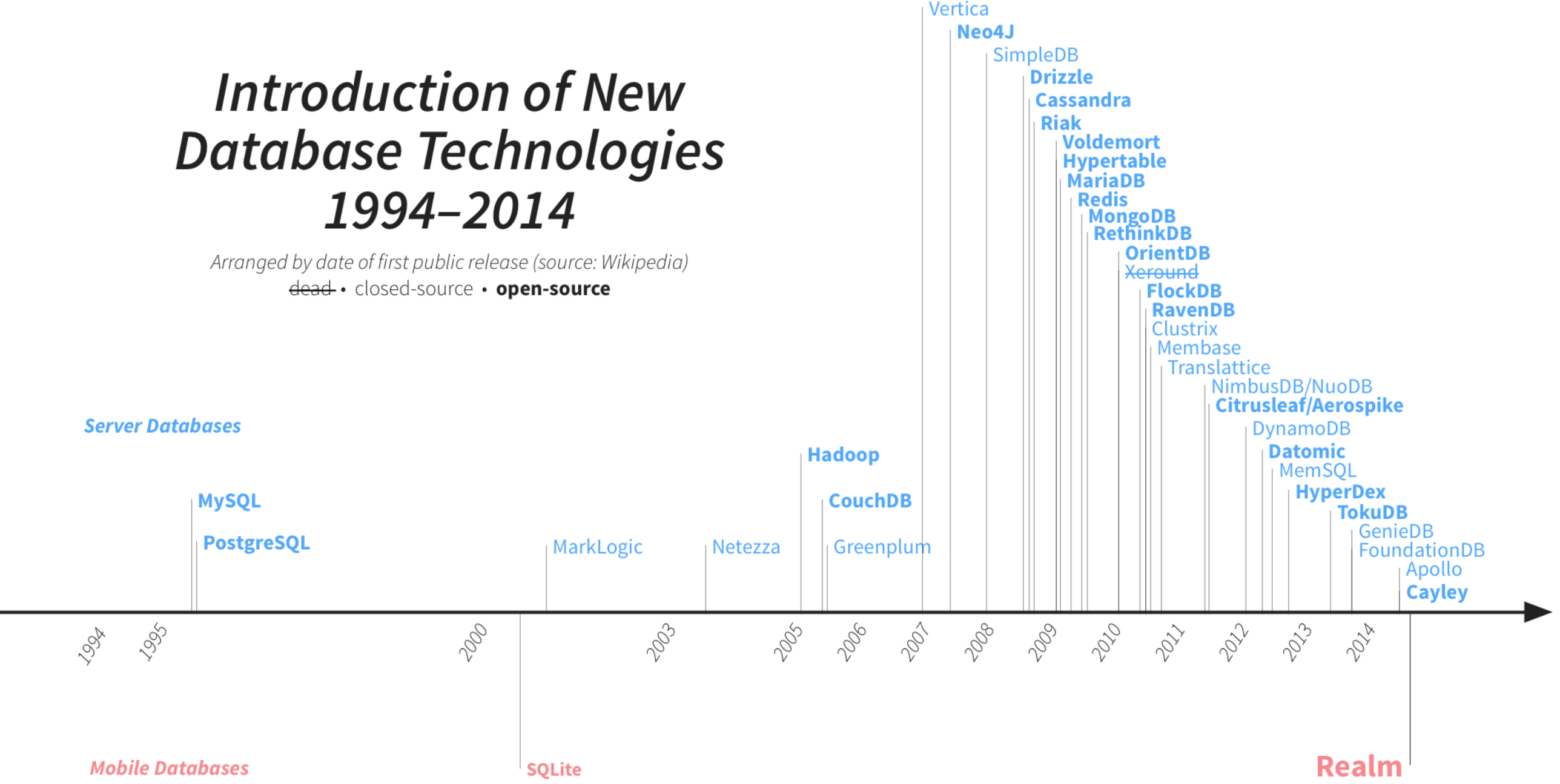
- Not an ORM
- Not API-compatible with Core Data
- Not API-stable (constantly improving)

why?

Introduction of New Database Technologies 1994–2014

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

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



Why a new database?

- SQLite is over 14 years old
- SQLite designed to run on military aircraft carriers
- Core Data is 10 years old, but most of its design if from 1994

Lots has changed in last decade

- Smartphone Revolution
- NoSQL
- Need for Sync

Demo

Relationships

```
@interface Person : RLMObject
@property NSString        *name;
@property NSData          *picture;
@property RLMArray<Dog> *dogs;
@end

RLMPerson *person = [[RLMPerson alloc] init];
person.name = @"Tim";
[person.dogs addObject:mydog];
```

Swift

```
class Person : RLMObject {
    dynamic var name = ""
    dynamic var picture = NSData()
    dynamic var dogs = RLMArray(objectClassName: "Dog")
}

let person = Person()
person.name = "Tim"
person.dogs.addObject(mydog)
```

Interesting Lesson #1

Swift Introspection

```
let valueType = mirror.valueType
var t = ""
switch valueType { // Detect basic types (including optional versions)
case is Bool.Type, is Bool?.Type:
    t = "c"
case is NSDate.Type, is NSDate?.Type:
    t = "@\"NSDate\""
case let objectType as RLMObject.Type:
    let mangledClassName = NSStringFromClass(objectType.self)
    let objectClassName = demangleClassName(mangledClassName)
    t = "@\"\(mangledClassName))\""
    // Construct RLMProperty with objectClassName
}

// create objc property
let attr = objc_property_attribute_t(name: "T", value: t)
class_addProperty(aClass, name, [attr], 1)
```

Interesting Lesson #2

```
// This compiles and runs in Swift
class 💩💩💩💩 {
    var 🚀 = 2
    func 💩💩💩(😎: Int, 😺: Int) -> Int {
        return 🚀 + 😎 + 😺
    }
}

var 🐔 = 3
var 🐸 = 🐔 + 2
var 💩 = 💩💩💩💩()
💩.💩💩💩(🐔, 😺:🐸) // => 10
```

How does Objective-C generate setter names?

OR

What is a valid Objective-C or Swift property name?

How Objective-C generates setter names

```
// Objective-C setters only capitalize the first letter
// of the property name if it falls between 'a' and 'z'
int asciiCode = [_name characterAtIndex:0];
BOOL shouldUppercase = asciiCode > 'a' && asciiCode < 'z';
NSString *firstChar = [_name substringToIndex:1];
firstChar = shouldUppercase ? firstChar.uppercaseString : firstChar;
// Setter Name:
// @"set%@%@:", firstChar, [_name substringFromIndex:1]
```

Resources

- Realm docs: **realm.io/docs**
- This talk: **github.com/jpsim/talks**
- Wikipedia NoSQL: **wikipedia.org/wiki/NoSQL**
- Deckset: **decksetapp.com**
- DeckRocket: **github.com/jpsim/DeckRocket**

Thanks!

@simjp, jp@realm.io