

try? Realm

Timur Khairullin, EPAM

Коротко о себе

Коротко о себе

- Разрабатываю > 8 лет

Коротко о себе

- Разрабатываю > 8 лет
- iOS > 4 лет

Коротко о себе

- Разрабатываю > 8 лет
- iOS > 4 лет
- ЕРАМ == 1 год

Коротко о себе

- Разрабатываю > 8 лет
- iOS > 4 лет
- ЕРАМ == 1 год
-  Power!

Коротко о себе

- Разрабатываю > 8 лет
- iOS > 4 лет
- ЕРАМ == 1 год
-  Power!
- Использовал Realm в продакшн

Содержание

Содержание

- Что такое Realm

Содержание

- Что такое Realm
- Realm на платформах Apple

Содержание

- Что такое Realm
- Realm на платформах Apple
- Типичные задачи с Realm

Содержание

- Что такое Realm
- Realm на платформах Apple
- Типичные задачи с Realm
- Больше деталей

Содержание

- Что такое Realm
- Realm на платформах Apple
- Типичные задачи с Realm
- Больше деталей
- CoreData vs Realm

Realm Database

Realm Database

- Open-source

Realm Database

- Open-source
- Java, Objective-C, JavaScript(React Native, Node-JS), Swift, Xamarin

Realm Database

- Open-source
- Java, Objective-C, JavaScript(React Native, Node-JS), Swift, Xamarin
- RealmCore(C++)

Realm Database

Realm Database

- NoSQL

Realm Database

- NoSQL
- Схема

Realm Database

- NoSQL
- Схема
- Индексы, ключи, связи

Realm Database

- NoSQL
- Схема
- Индексы, ключи, связи
- Транзакции

Realm Object Server

Realm Object Server

- Синхронизирует данные между клиентами

Realm Object Server

- Синхронизирует данные между клиентами
- Правила синхронизации:
 1. Удаление всегда в приоритете
 2. Кто последний тот главней

Realm Object Server

- Синхронизирует данные между клиентами
- Правила синхронизации:
 1. Удаление всегда в приоритете
 2. Кто последний тот главней
- Не mBaaS

Realm Object Server Prices

Realm Platform
Developer Edition



- Realtime two-way data sync
- Automatic and seamless
- Cross platform: iOS, Android, & more
- Runs on any server, Linux or Mac
- Free forever, even in production

[Free download](#)

[Read more](#) or [compare platform options](#)

Realm Platform
Professional Edition



All Developer Edition features plus:

- Server-side data access and integrations
- Business-grade support
- Starts at \$1750/month for up to 1,000 daily active users

[Start a free trial](#)

[Read more](#) or [compare platform options](#)

Realm Platform
Enterprise Edition



All Professional Edition features plus:

- Data integration API and pre-built data connectors
- Massive scalability, 1M+ concurrent users
- Enterprise-grade support
- Custom pricing based on use and scale

[Request a demo](#)

[Read more](#) or [compare platform options](#)

Realm на платформах Apple

Realm на платформах Apple

- iOS 8+

Realm на платформах Apple

- iOS 8+
- macOS 10.9+

Realm на платформах Apple

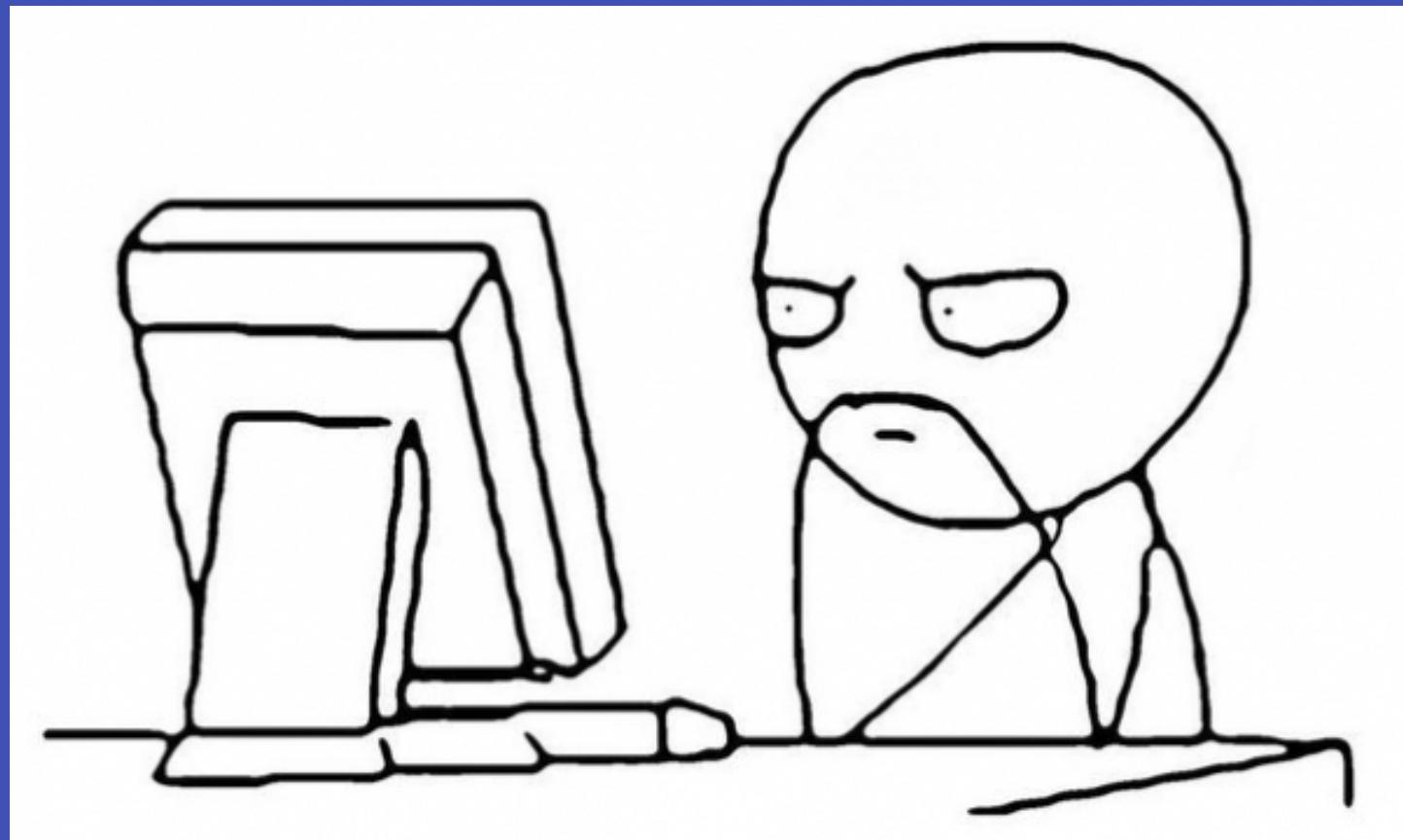
- iOS 8+
- macOS 10.9+
- tvOS

Realm на платформах Apple

- iOS 8+
- macOS 10.9+
- tvOS
- watchOS

Realm на платформах Apple

- iOS 8+
- macOS 10.9+
- tvOS
- watchOS
- RealmSwift / Realm Objective-C



Инстанс Базы Данных

```
pod 'RealmSwift', '~> 2.10.0'
```

...

```
let realm = try? Realm()
```

Модели

```
class EPAMer: Object {  
    @objc dynamic var name: String = ""  
    @objc dynamic var birthday: Date?  
    @objc dynamic var avatarURL: String?  
}
```

Индексы, ключи

```
class EPAMer: Object {  
    ...  
    override static func indexedProperties() -> [String] {  
        return [#keyPath(EPAMer.birthday)]  
    }  
  
    override class func primaryKey() -> String? {  
        return #keyPath(EPAMer.name)  
    }  
}
```

СВЯЗИ

```
class EPAMer: Object {
```

```
... .
```

```
    var RM: EPAMer?
```

```
    let projects = List<Project>()
```

```
}
```

```
class Project: Object {
```

```
... .
```

```
    let epamers = LinkingObjects(fromType: EPAMer.self,  
                                property: "projects")
```

```
}
```

Редактирование объектов

Создание

```
let ерамер = ЕРАMer()  
ерамер.name = "Timur"
```

```
try? realm?.write {  
    realm?.add(ерамер)  
}
```

Редактирование объектов

Редактирование

```
try? realm?.write {  
    ерамер.name = "Ricardo"  
    ...  
    realm?.add(ерамер, update: true)  
}
```

Редактирование объектов

Удаление

```
try? realm?.write {  
    realm?.delete(ерамер)  
}
```

Запросы

```
realm?.objects(EPAMer.self)
    .filter("name == %@", "Timur")
    .filter({ $0.name == "Timur" })
    .filter(NSPredicate(format: "name == %@", "Timur"))
    .sorted(byKeyPath: "name")
    .sorted(by: { $0.name < $1.name })
    .sorted(by: [SortDescriptor(keyPath: "name"),
                SortDescriptor(keyPath: "birthday")])
```

Нотификации

База

```
let token = realm?.observe { notification, realm in  
    ...  
}  
...  
token?.invalidate()
```

Нотификации

Коллекция

```
let token = realm?.objects(EPAMer.self).observe { (changes) in
    ...
}

...
token?.invalidate()
```

Нотификации

Объект

```
let token = eramer.observe { (change) in  
    ...  
}  
...  
token.invalidate()
```



Немного инициализации

```
public final class Realm {  
  
    public convenience init() throws  
    public convenience init(configuration: Configuration) throws  
  
    public struct Configuration {  
  
        public static var defaultConfiguration: Configuration  
        public init(fileURL: URL? = ... , // "default.realm"  
                    inMemoryIdentifier: String? = nil,  
                    syncConfiguration: SyncConfiguration? = nil,  
                    encryptionKey: Data? = nil,  
                    readOnly: Bool = false,  
                    schemaVersion: UInt64 = 0,  
                    migrationBlock: MigrationBlock? = nil,  
                    deleteRealmIfMigrationNeeded: Bool = false,  
                    shouldCompactOnLaunch: ((Int, Int) -> Bool)? = nil,  
                    objectTypes: [Object.Type]? = nil)  
    }  
}
```

Немного инициализации

Разделяем базы по:

Немного инициализации

Разделяем базы по:

- пользователям

Немного инициализации

Разделяем базы по:

- пользователям
- объектам

Немного инициализации

Разделяем базы по:

- пользователям
- объектам
- командам разработчиков

Немного инициализации

Разделяем базы по:

- пользователям
- объектам
- командам разработчиков
- типу доступа

Live данные

Live данные

- Нет копий данных

Live данные

- Нет копий данных
- Нет refresh / refetch

Live данные

- Нет копий данных
- Нет refresh / refetch
- Объекты
 - нет didSet / willSet



Live данные

- Нет копий данных
- Нет refresh / refetch
- Объекты
 - нет didSet / willSet
 - invalidated

Lazy коллекции

Lazy коллекции

- нет batchSize / Limits/ PropertiesToFetch

Lazy коллекции

- нет batchSize / Limits/ PropertiesToFetch
- цепочки запросов

Properties

Type	Non-optional	Optional
Bool	<code>@objc dynamic var value = false</code>	<code>let value = RealmOptional<Bool>()</code>
Int	<code>@objc dynamic var value = 0</code>	<code>let value = RealmOptional<Int>()</code>
Float	<code>@objc dynamic var value: Float = 0.0</code>	<code>let value = RealmOptional<Float>()</code>
Double	<code>@objc dynamic var value: Double = 0.0</code>	<code>let value = RealmOptional<Double>()</code>
String	<code>@objc dynamic var value = ""</code>	<code>@objc dynamic var value: String? = nil</code>
Data	<code>@objc dynamic var value = Data()</code>	<code>@objc dynamic var value: Data? = nil</code>
Date	<code>@objc dynamic var value = Date()</code>	<code>@objc dynamic var value: Date? = nil</code>
Object	n/a: must be optional	<code>@objc dynamic var value: Class?</code>
List	<code>let value = List<Type>()</code>	n/a: must be non-optional
LinkingObjects	<code>let value = LinkingObjects(fromType: Class.self, property: "property")</code>	n/a: must be non-optional

Properties

Workaround для [Int], [String]

```
class StringObject: Object {  
    @objc dynamic var value: String = ""  
}
```

```
class EPAMer: Object {  
    ...  
    let names = List<StringObject>()  
}
```

Каскадное удаление



Каскадное удаление

Workaround

```
protocol CascadeDeletable {
    func deleteCascade(realm: Realm)
}

extension Realm {
    func delete(_ object: Object & CascadeDeletable, cascade: Bool) {
        if cascade { object.deleteCascade(realm: self) }
        delete(object)
    }
}

class EPAMer: Object, CascadeDeletable {
    ...
    let names = List<StringObject>()
    func deleteCascade(realm: Realm) {
        realm.delete(names)
        // realm.delete(some, cascade: true)
    }
}
```

Наследование



Наследование

Наследование

- Нет casting'a между типами

Наследование

- Нет casting'a между типами
- Нет multi-class запросов

Наследование

- Нет casting'a между типами
- Нет multi-class запросов
- Нет multi-class контейнеров

Наследование

Workaround для запросов

```
class A: Object {
    @objc dynamic var intProp = 0
}
class B: A {}
class C: A {}
extension Realm {
    func filter<ParentType: Object>(_ parentType: ParentType.Type,
                                    _ subclasses: [ParentType.Type],
                                    _ predicate: NSPredicate) -> [ParentType] {
        return ([parentType] + subclasses).flatMap { classType in
            return Array(self.objects(classType).filter(predicate))
        }
    }
}

// Usage

let realm = try? Realm()
let allAClassesGreaterThanZero = realm?.filter(A.self, [B.self, C.self], NSPredicate(format: "intProp > 0"))
```

Наследование

Workaround для контейнеров

```
class A: Object {  
    @objc dynamic var intProp = 0  
}  
class B: A {}  
class C: A {}  
class AClasses: Object {  
    @objc dynamic var a: A?  
    @objc dynamic var b: B?  
    @objc dynamic var c: C?  
}  
class D: Object {  
    @objc dynamic var polymorphicA: AClasses?  
}
```

Наследование

Убрать наследование



Наследование

Убрать наследование

```
class Animal: Object {  
    @objc dynamic var age = 0  
}  
class Duck : Object {  
    @objc dynamic var animal: Animal?  
    @objc dynamic var name = ""  
}  
class Frog : Object {  
    @objc dynamic var animal: Animal?  
    @objc dynamic var dateProp = NSDate()  
}
```

Многопоточность

Многопоточность

- Модель транзакций

Многопоточность

- Модель транзакций
- Транзакции синхронные и блокирующие

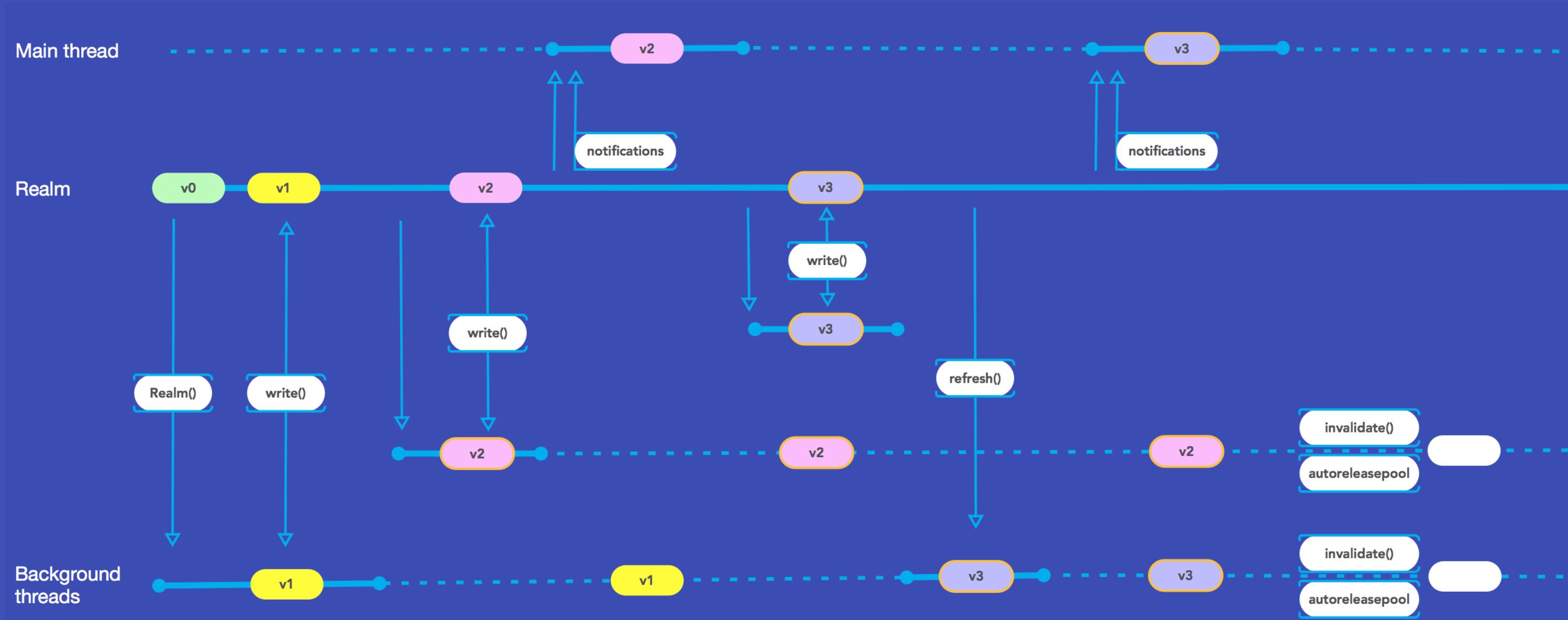
Многопоточность

- Модель транзакций
- Транзакции синхронные и блокирующие
- Нельзя использовать объект на разных потоках

Многопоточность

- Модель транзакций
- Транзакции синхронные и блокирующие
- Нельзя использовать объект на разных потоках
- Каждый поток ссылается на транзакционную версию

МНОГОПОТОЧНОСТЬ





Многопоточность

Рекомендации. Main thread

Многопоточность

Рекомендации. Main thread

- Autorefresh = true

Многопоточность

Рекомендации. Main thread

- Autorefresh = true
- Только чтение

Многопоточность

Рекомендации. Background thread

Многопоточность

Рекомендации. Background thread

- Все транзакции

Многопоточность

Рекомендации. Background thread

- Все транзакции
- Много объектов в 1 транзакции

Многопоточность

Рекомендации. Background thread

- Все транзакции
- Много объектов в 1 транзакции
- Всегда Realm()

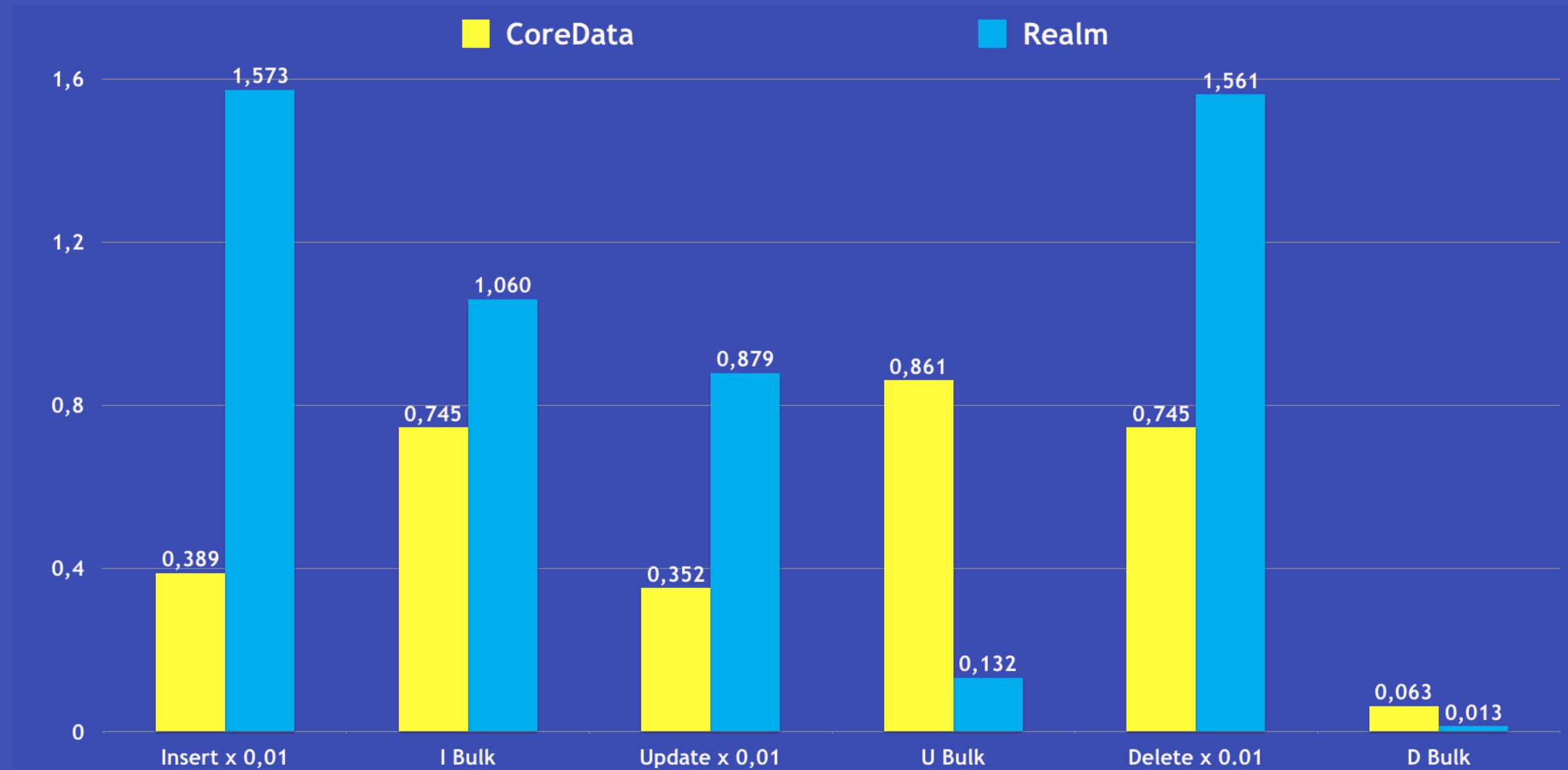
Многопоточность

Рекомендации. Background thread

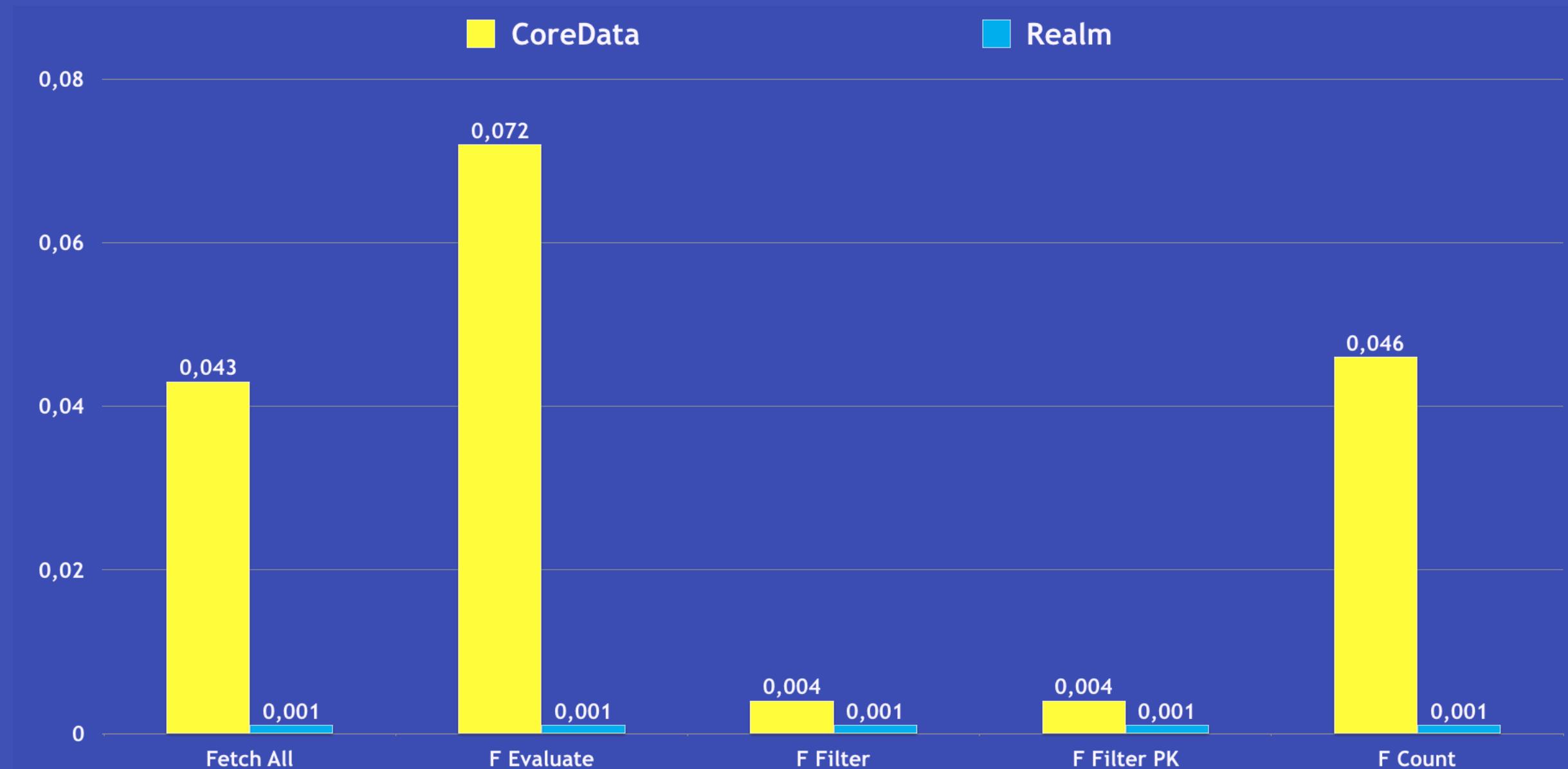
- Все транзакции
- Много объектов в 1 транзакции
- Всегда Realm()
- Всегда autoreleasepool {}

Размер файла
Смотри предыдущие слайды

Производительность



Производительность



CoreData vs Realm

	CoreData	Realm
Простота	😡	👍
Кросс-платформенность	😡	👍
iOS7	👍	😡
Индексы	👍	🙏
Ключи	😡	👍
Уникальность связей	👍	😡
Нотификация по изменении связи	😡	👍
UndoManager	👍	😡
Каскадное удаление	👍	🙏
Наследование	👍	🙏
Скорость записи	👍	😡
Скорость чтения	😡	👍

try? Realm() ?? CoreDataManager()