

# Core Data — Поваренная книга

Антон Давыдов

Кошелёк & Swoo

# Как это по-русски?

- Core Data
- SQL
- Faults
- Realm

# О чём поговорим

- 1) Когда нужна DB, почему CoreData
- 2) Инструментарий
- 3) Устройство и архитектура
- 4) Работа с данными
- 5) Перформанс, тестирование
- 6) Сложности и ограничения

## Страх использования:

- Долго запрягать, тяжелая настройка стэка
- Страшные названия
- Много промежуточных частей
- Непонятности с многопоточкой
- Objective-C наследие 

# Что улучшилось за последние года

# Что улучшилось за последние годы

## Безопасное форматирование

```
request.predicate = NSPredicate(format: "(\(BaseItem.type)) = 0")
```

# Что улучшилось за последние годы

## Поддержка Swift generics

```
let request: NSFetchedRequest<BaseItem> = NSFetchedRequest(entityName: BaseItem.self.description())
let items: [BaseItem] = try viewContext.fetch(request)
```

# Что улучшилось за последние годы

## NSPersistentStoreContainer

```
@available(iOS 10.0, *)
open class NSPersistentContainer : NSObject {

    open class func defaultDirectoryURL() -> URL

    open var name: String { get }

    open var viewContext: NSManagedObjectContext { get }

    open var managedObjectModel: NSManagedObjectModel { get }

    open var persistentStoreCoordinator: NSPersistentStoreCoordinator { get }

    open var persistentStoreDescriptions: [NSPersistentStoreDescription]

    // Creates a container using the model named `name` in the main bundle
    public convenience init(name: String)

    public init(name: String, managedObjectModel model: NSManagedObjectModel)

    // Load stores from the storeDescriptions property that have not already been successfully added to the container. The completion handler is called once for each store that succeeds or fails.
    open func loadPersistentStores(completionHandler block: @escaping (NSPersistentStoreDescription, Error?) -> Void)

    open func newBackgroundContext() -> NSManagedObjectContext

    open func performBackgroundTask(_ block: @escaping (NSManagedObjectContext) -> Void)
}
```

# Что улучшилось за последние годы

## Поддержка SwiftUI

```
@FetchRequest(sortDescriptors: [NSSortDescriptor(keyPath: \Item.timestamp, ascending: true)])
private var items: FetchedResults<Item>
```

## Чего сейчас не хватает

- Поддержка Combine для работы со SwiftUI
- HR бренд

# Нужна ли вам DB (Core Data) в проекте?

С большой долей вероятности база данных не нужна  
вообще

# Пример

Личный кабинет пользователя

Приложение - дополнение к веб-сайту

## Когда вам DB (Core Data) нужна

- Нужен оффлайн
- Есть коллекции объектов
- Развесистый граф
- Нужна фильтрация и поиск

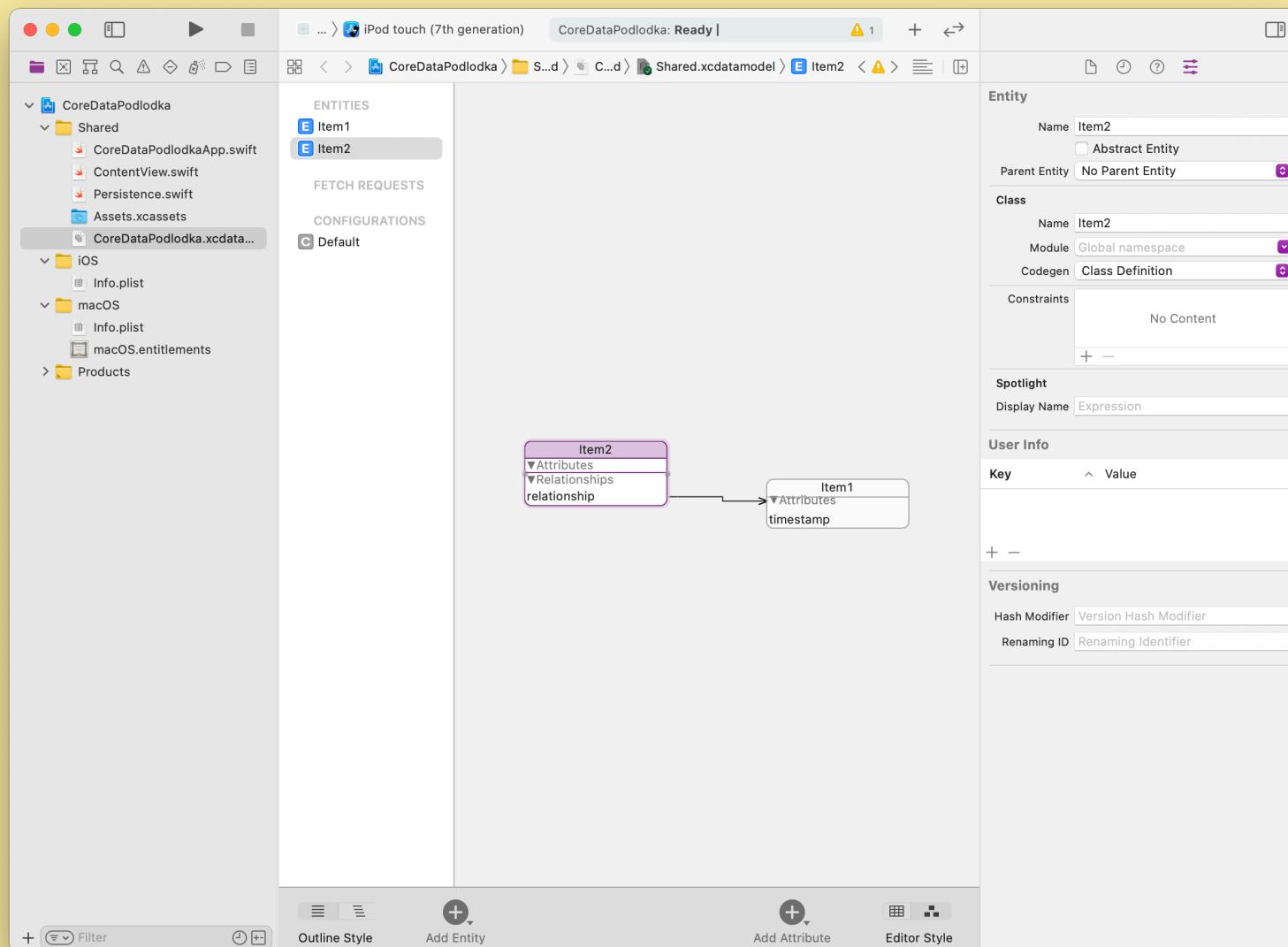
# Почему CoreData

- Нативно
- Больше чем просто персистенция данных
- Модель, версионирование и миграция

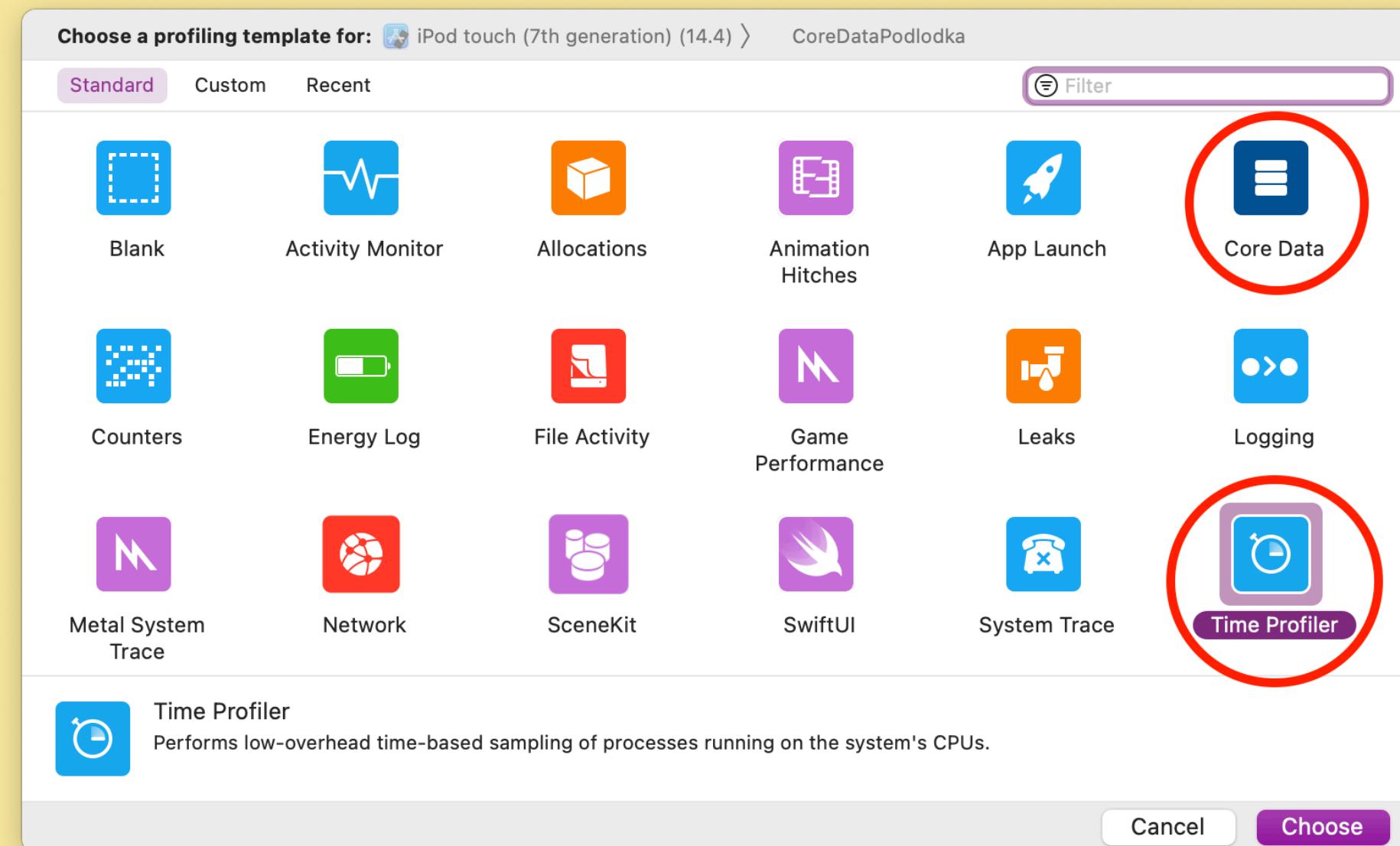


<https://www.objc.io/books/core-data/>

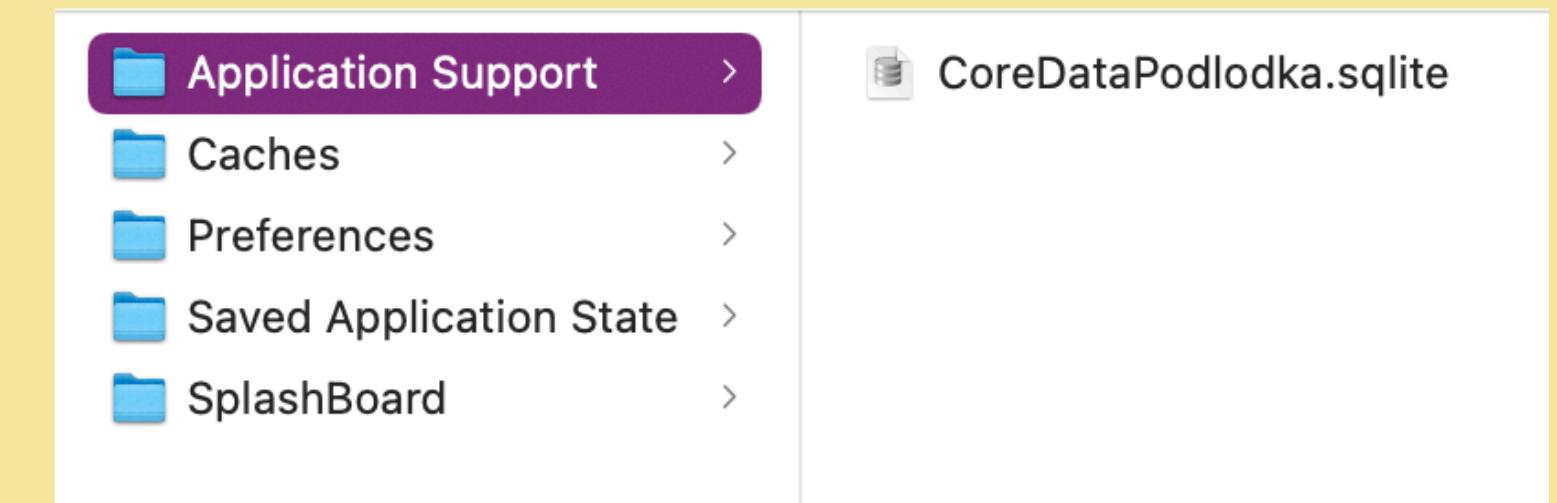
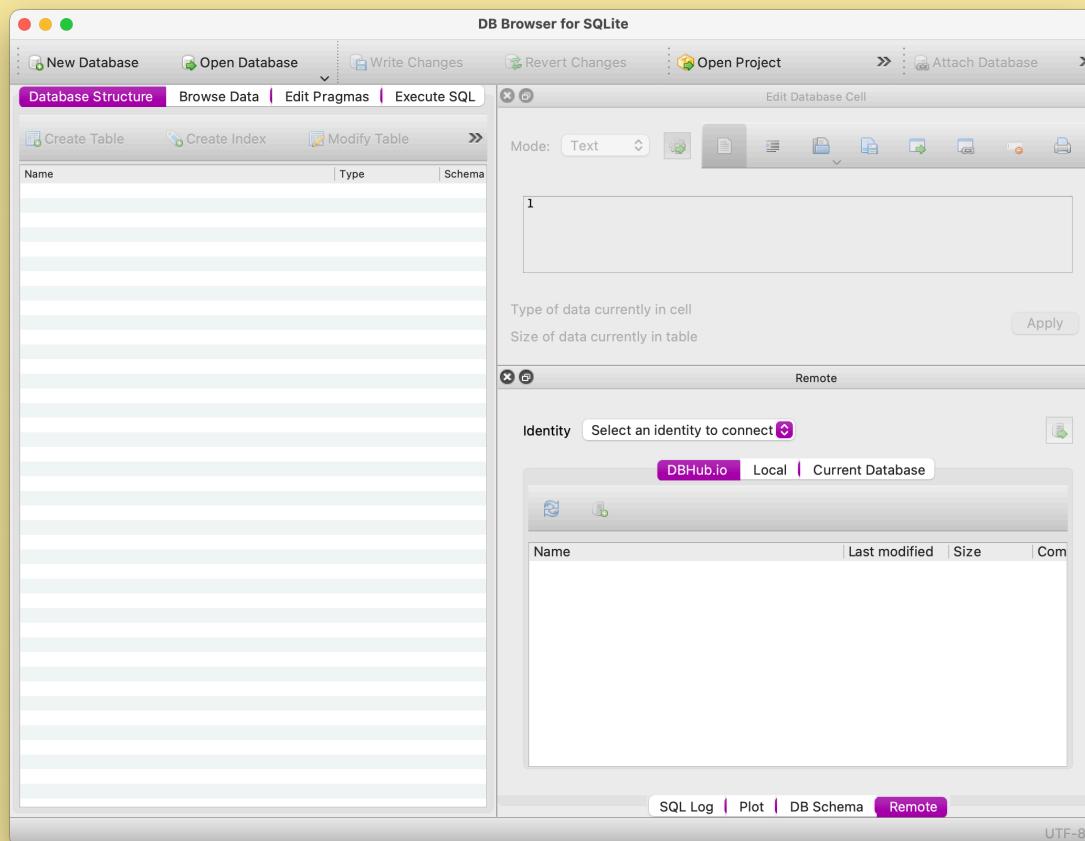
# Инструменты #1. Визуальный редактор моделей



# Инструменты #2. Xcode Instruments

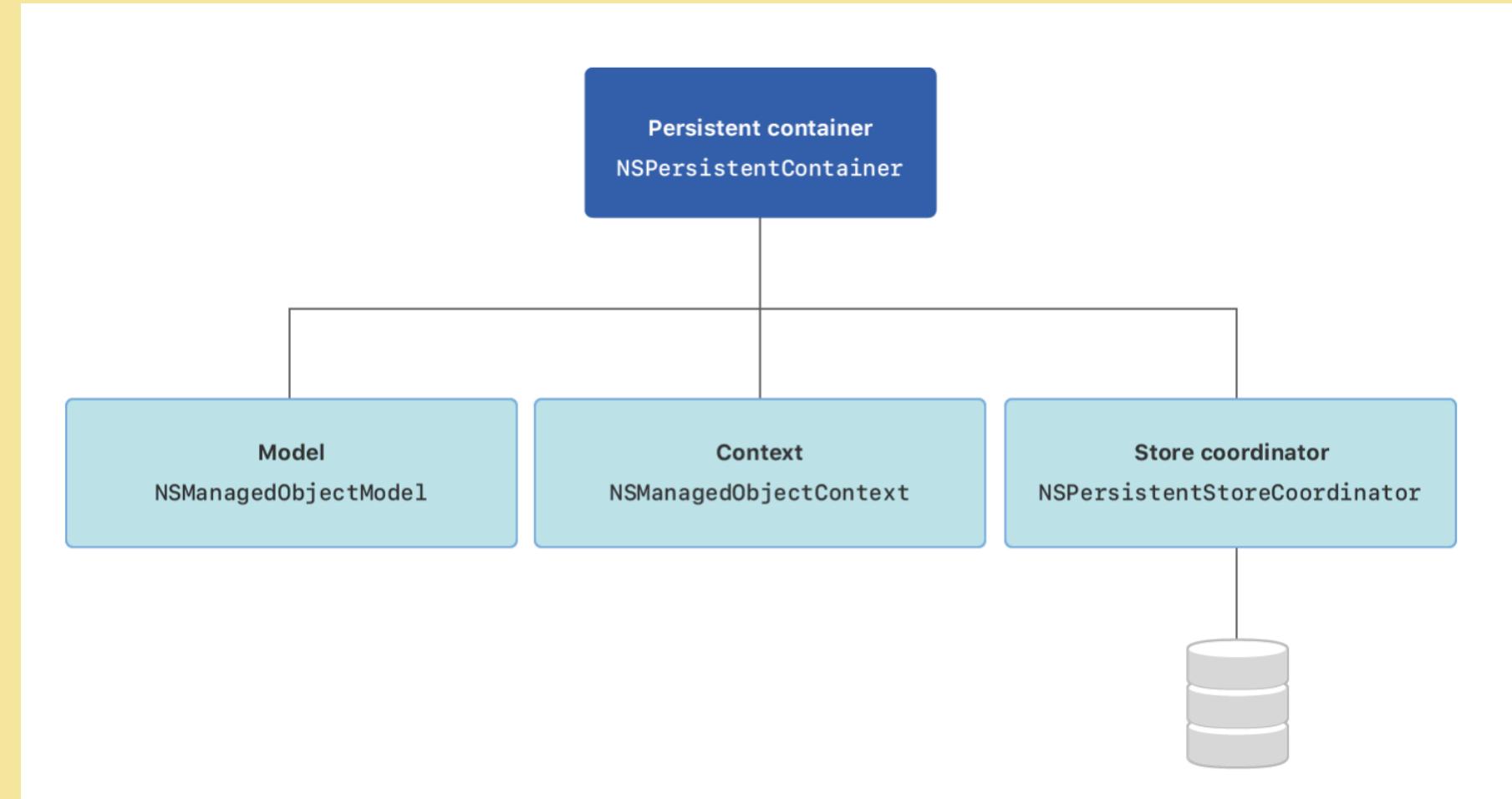


# Инструменты #3. SQLite DB Reader



<https://sqlitebrowser.org>

# Устройство #1. Основные сущности



<https://developer.apple.com/documentation/coredata>

## Типичный пример сохранения данных

```
let context = container.newBackgroundContext()  
let newItem = Item(context: context)  
newItem.timestamp = Date()  
if context.hasChanges {  
    try? context.save()  
}
```

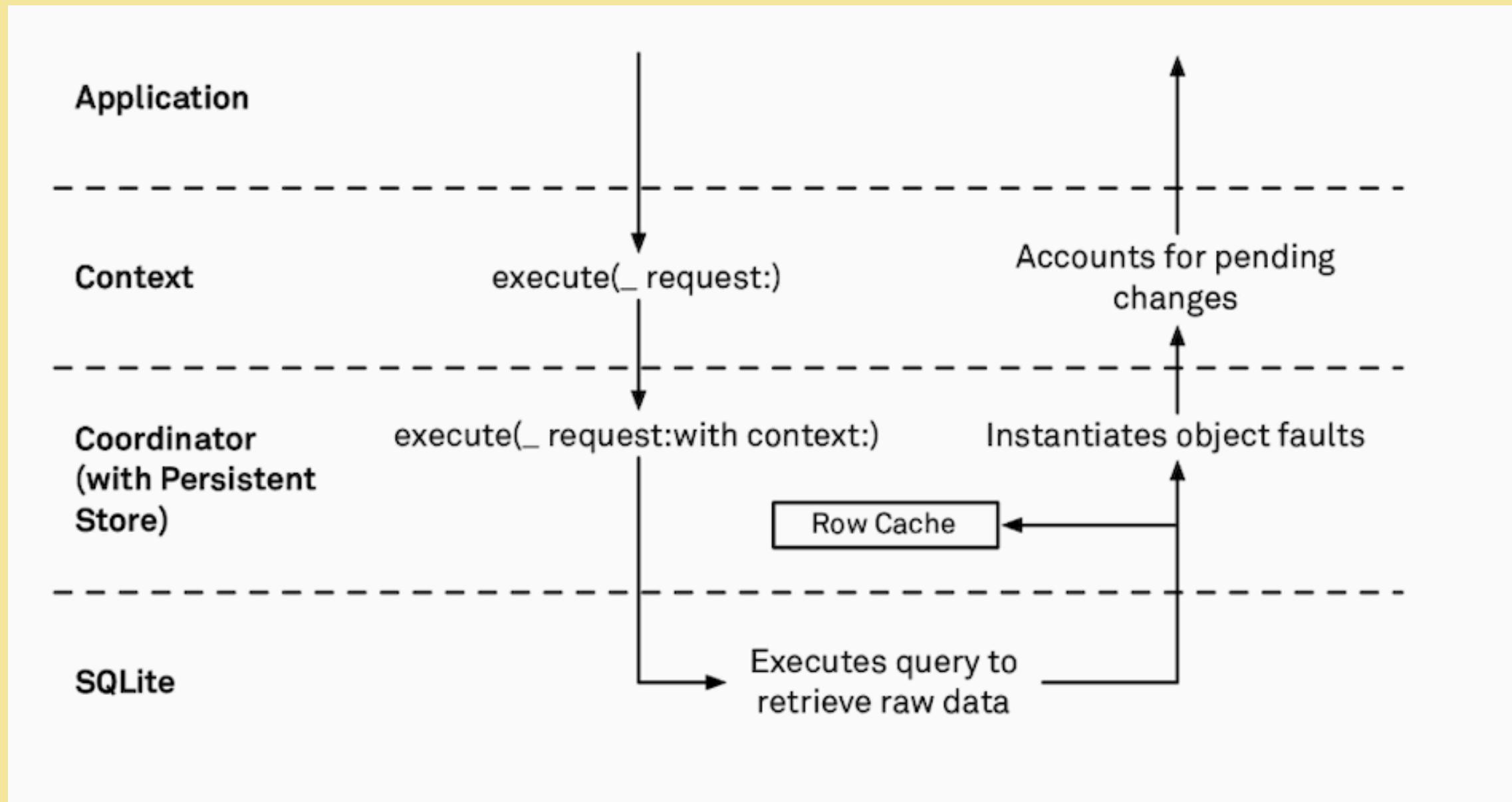
# Типичный пример получения данных

```
let request: NSFetchedResultsController<Item> = NSFetchedResultsController(entityName: Item.self.description())
request.predicate = NSPredicate(format: "(\(Item.timestamp)) ≠ NULL")
let items = try? container.viewContext.fetch(request)
```

# Типичный пример использования в SwiftUI

```
@FetchRequest(  
    sortDescriptors: [NSSortDescriptor(keyPath: \Item.timestamp, ascending: true)],  
    animation: .default  
)  
private var items: FetchedResults<Item>
```

# Устройство #2. Алгоритм получения данных



# Что в SQLite базе хранится

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Create Table | Create Index | Print

Name	Type	Schema
Tables (4)		
ZBASEITEM	CREATE TABLE ZBASEITEM ( Z_PK INTEGER PRIMARY KEY,Z_ENT INTEGER,Z_OPT INTEGER,ZITEMFIELD INTEGER,ZITEM2FIELD INTEGER,ZRELATIONSHIP INTEGER,ZTIMESTAMP TIMESTAMP,ZCOMMONFIELD VARCHAR)	
Z_PK	INTEGER	"Z_PK" INTEGER
Z_ENT	INTEGER	"Z_ENT" INTEGER
Z_OPT	INTEGER	"Z_OPT" INTEGER
ZITEMFIELD	INTEGER	"ZITEMFIELD" INTEGER
ZITEM2FIELD	INTEGER	"ZITEM2FIELD" INTEGER
ZRELATIONSHIP	INTEGER	"ZRELATIONSHIP" INTEGER
ZTIMESTAMP	TIMESTAMP	"ZTIMESTAMP" TIMESTAMP
ZCOMMONFIELD	VARCHAR	"ZCOMMONFIELD" VARCHAR
Z_METADATA	CREATE TABLE Z_METADATA (Z_VERSION INTEGER PRIMARY KEY,Z_UUID VARCHAR(255),Z_PLIST BLOB)	
Z_VERSION	INTEGER	"Z_VERSION" INTEGER
Z_UUID	VARCHAR(255)	"Z_UUID" VARCHAR(255)
Z_PLIST	BLOB	"Z_PLIST" BLOB
Z_MODELCACHE	CREATE TABLE Z_MODELCACHE (Z_CONTENT BLOB)	
Z_CONTENT	BLOB	"Z_CONTENT" BLOB
Z_PRIMARYKEY	CREATE TABLE Z_PRIMARYKEY (Z_ENT INTEGER PRIMARY KEY,Z_NAME VARCHAR,Z_SUPER INTEGER,Z_MAX INTEGER)	
Z_ENT	INTEGER	"Z_ENT" INTEGER
Z_NAME	VARCHAR	"Z_NAME" VARCHAR
Z_SUPER	INTEGER	"Z_SUPER" INTEGER
Z_MAX	INTEGER	"Z_MAX" INTEGER
Indices (2)		
ZBASEITEM_ZRELATIONSHIP_INDEX	CREATE INDEX ZBASEITEM_ZRELATIONSHIP_INDEX ON ZBASEITEM (ZRELATIONSHIP)	
ZBASEITEM_Z_ENT_INDEX	CREATE INDEX ZBASEITEM_Z_ENT_INDEX ON ZBASEITEM (Z_ENT)	
Views (0)		
Triggers (0)		

# Возможности визуальной настройки в редакторе моделей

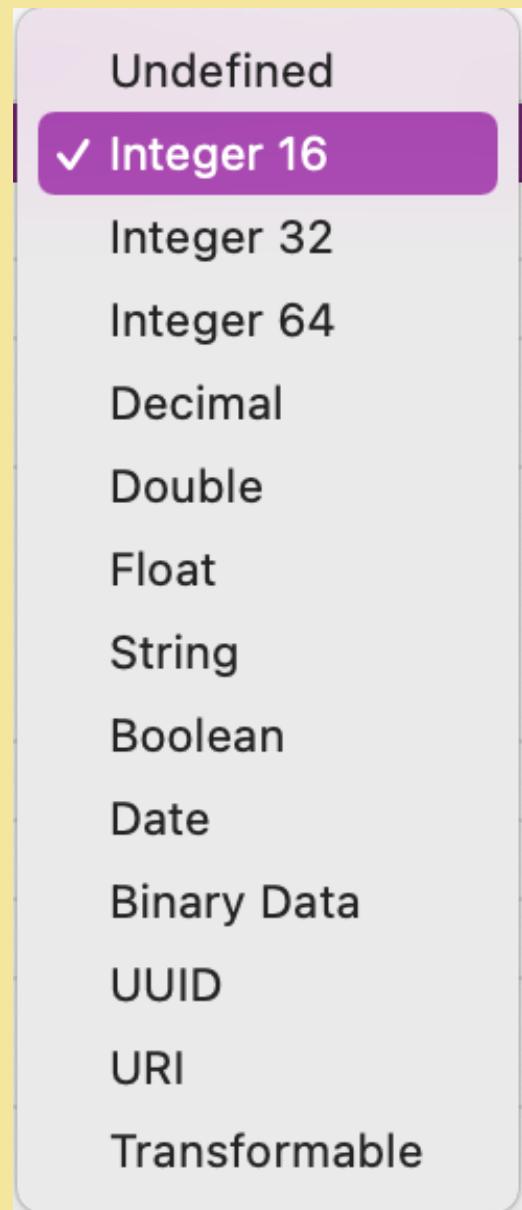
# Кодогенерация

## Code Generation

Language **Swift**



# Поддерживаемые типы



# Настройка связей

Relationship

Name

Properties  Transient  Optional

Destination

Inverse

Delete Rule

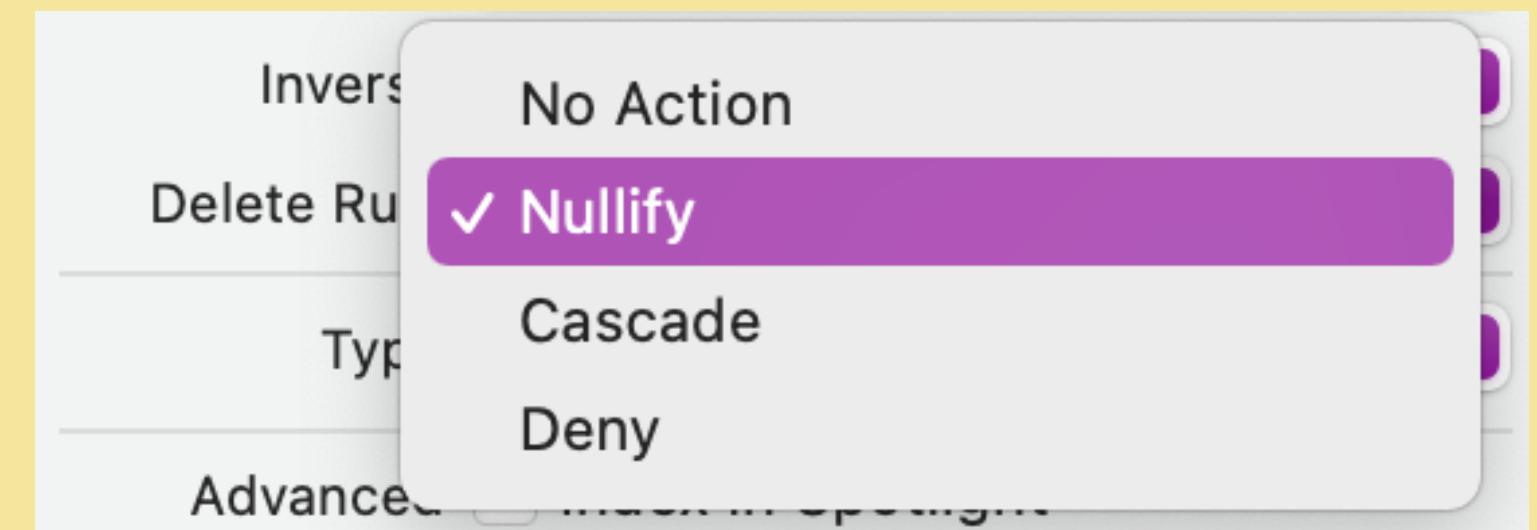
Type

Advanced  Index in Spotlight

Deprecated

Spotlight  Store in External Record File

User Info



# Контроль времени жизни

- Cascade (Композиция)
- Nullify (Агрегация)

# Общий принцип работы Коредаты



# Почему Коредата быстрая #1

Уровни кеширования (Object, Context, Store)

## Почему Коредата быстрая #2



CoreDataPodlodka.sqlite



CoreDataPodlodka.sqlite-shm



CoreDataPodlodka.sqlite-wal

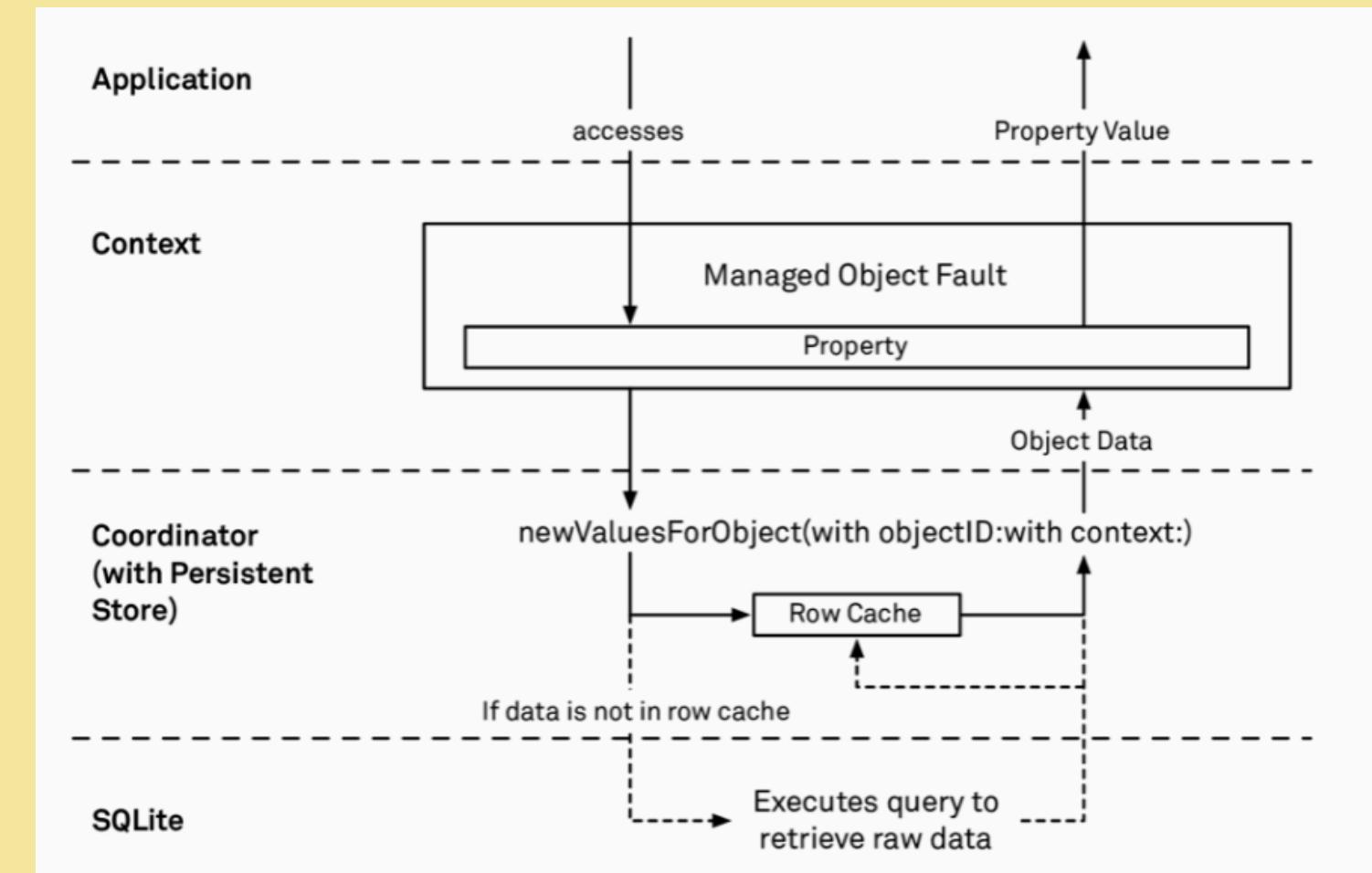
Режим журнала WAL (Write-Ahead Logging)

<https://sqlite.org/wal.html>

## Почему Коредата быстрая #3

Возможность создания нескольких контекстов и  
одновременная работа с ними

# Почему Коредата быстрая и эффективная по памяти #4



Механизм *faults*/фолтов/пустышек

## Особенности фолтов в связях

- По objectId
- One-to-many в две стадии

## Как управлять фолтами

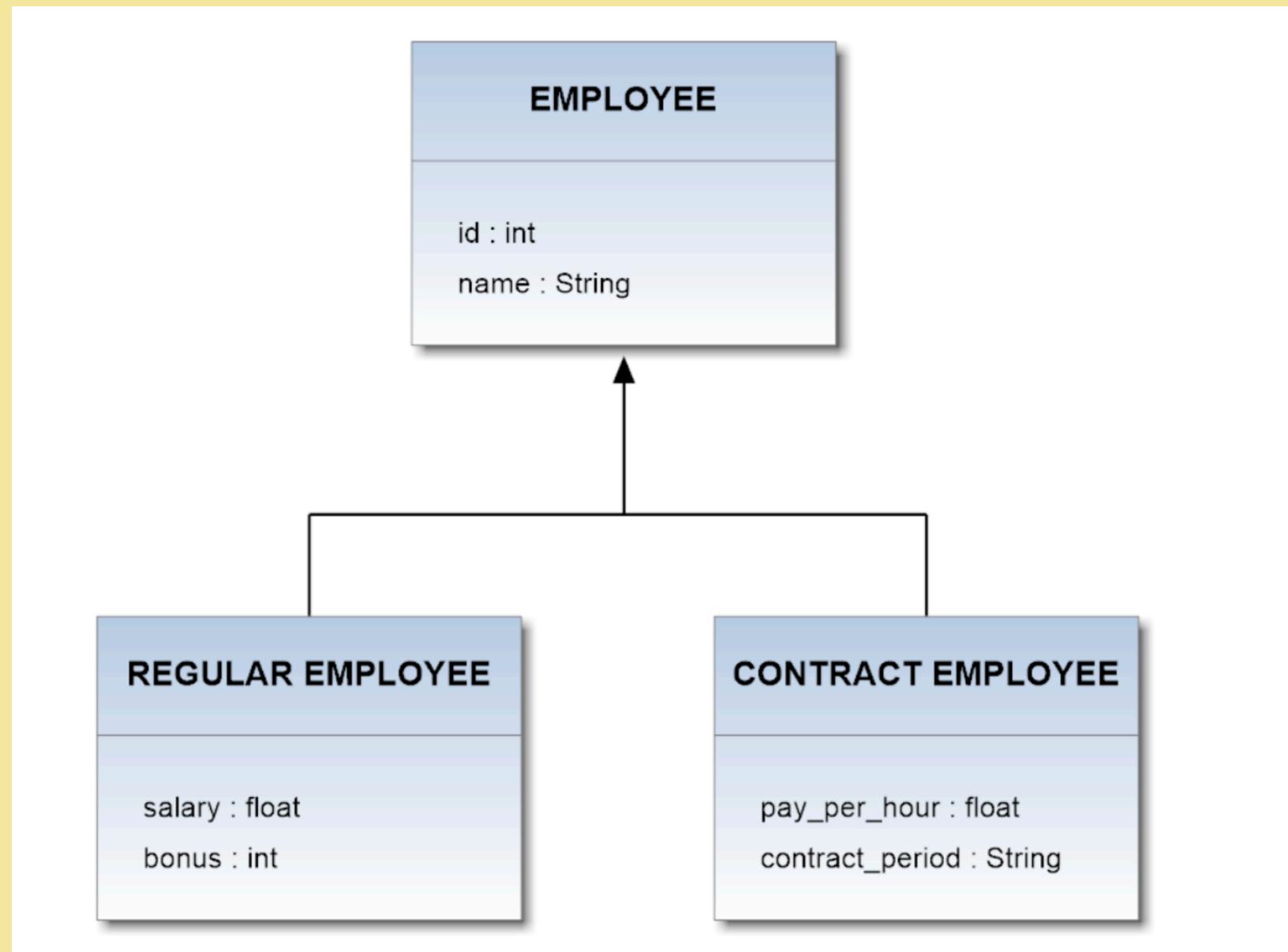
- `request.ReturnsObjectsAsFaults = false`
- `item.isFault`

# Рекомендации по увеличению производительности

- Включать для работы на UI
- Выключать при обходе всей коллекции данных в бэкграунде

# Наследование и Core Data

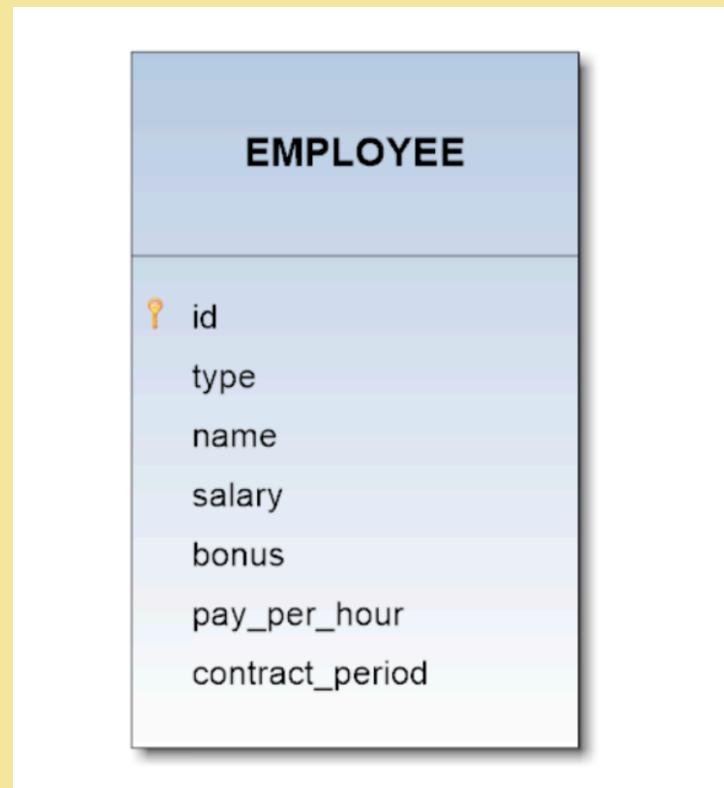
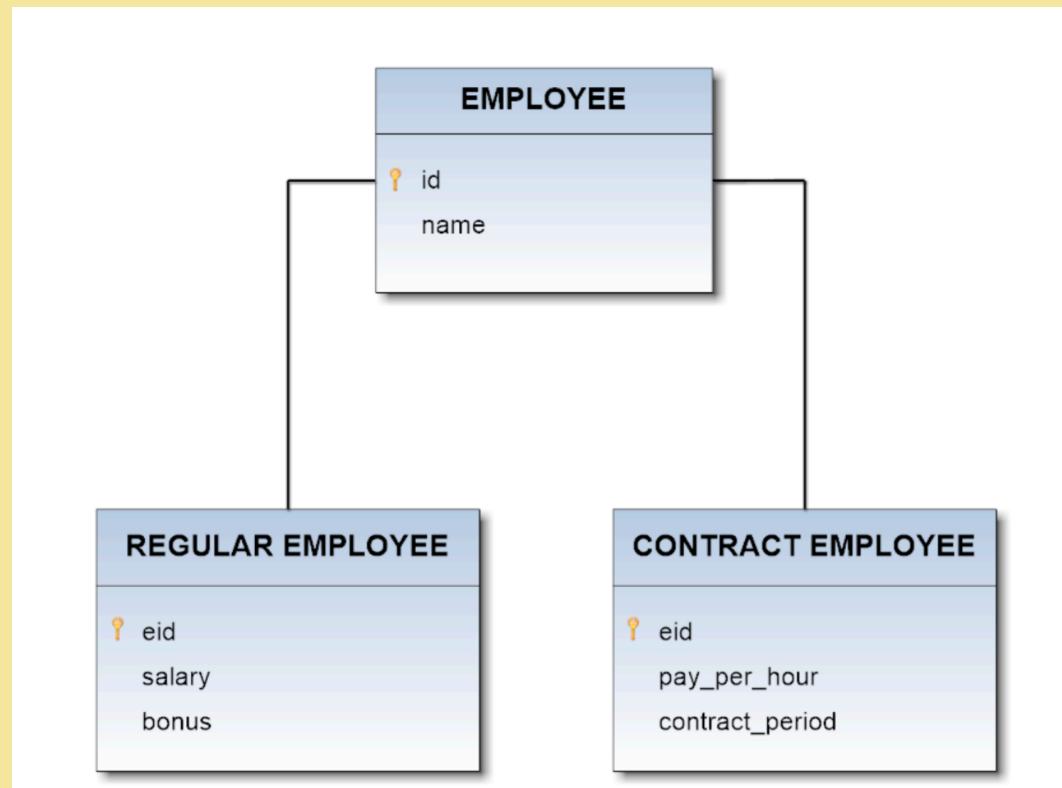
# Паттерны наследования



# Паттерны наследования. Варианты

1. Table Per Class Inheritance

2. Single Table Inheritance



# Наследование в Коредате

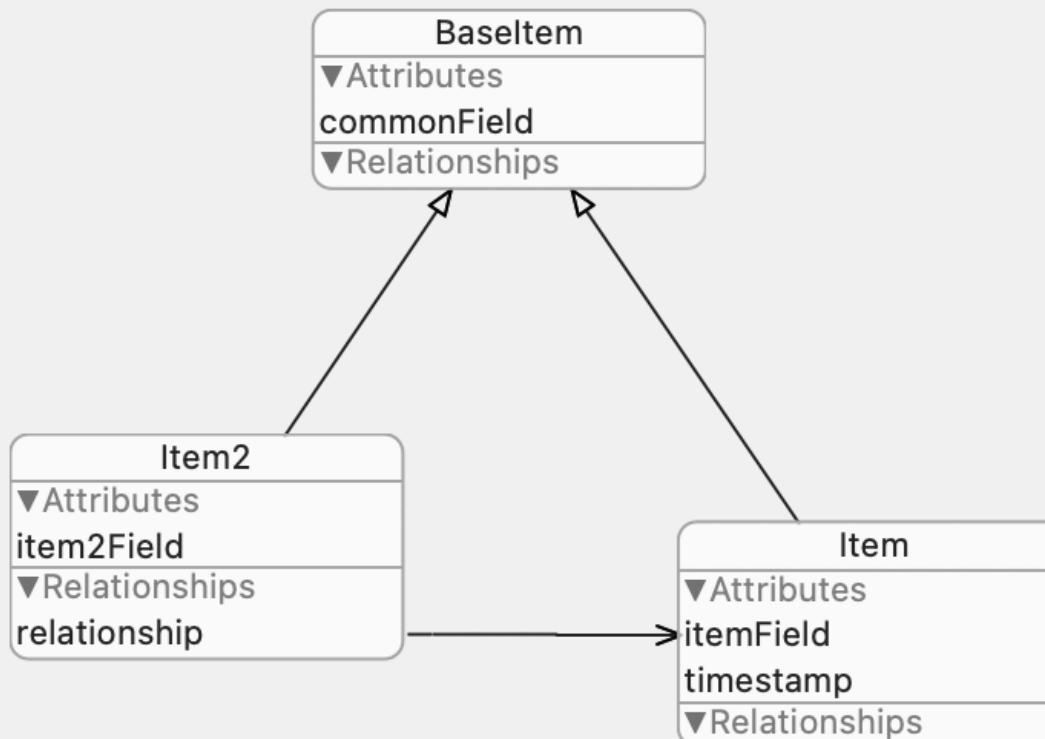


Table: ZBASEITEM

Z_PK	Z_ENT	Z_OPT	ZITEMFIELD	ZITEM2FIELD	ZRELATIONSHIP	ZTIMESTAMP	ZCOMMONFIELD
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	2	1	0	NULL	NULL	644065447.244468
2	2	2	1	0	NULL	NULL	644065448.606751
3	3	2	1	0	NULL	NULL	644065449.303292
4	4	2	1	100	NULL	NULL	644135784.414603
5	5	2	1	100	NULL	NULL	644135785.205802
6	6	3	1	NULL	1	NULL	NULL
7	7	3	1	NULL	1	NULL	NULL
8	8	3	1	NULL	1	NULL	NULL

# Fetch по базовому классу. Пример

```
let request: NSFetchedResultsController<BaseItem> = NSFetchedResultsController(entityName: BaseItem.self.description())
request.predicate = NSPredicate(format: "(#keyPath(Item2.item2Field)) = NULL")
```

# Дополнительное поле type для фильтрации

The screenshot shows the Xcode Entity Inspector for an entity named 'BaseItem'. On the left, there are tabs for 'ENTITIES', 'FETCH REQUESTS', and 'CONFIGURATIONS'. The 'ENTITIES' tab is selected, and 'BaseItem' is highlighted with a purple bar. Under the 'Attributes' section, there are two entries:

Attribute	Type
S commonField	String
N type	Integer 16

At the bottom of the Attributes table are '+' and '-' buttons.

```
let request: NSFetchedRequest<BaseItem> = NSFetchedRequest(entityName: BaseItem.self.description())
request.predicate = NSPredicate(format: "#keyPath(BaseItem.type) = 0")
```

## Советы

- Для определения типа поля вводите дополнительное поле с типом
- Откажитесь от наследования в пользу Swift Protocol

# Работа с контекстом NSManagedObjectContext

# Контексты

- Читаем на UI для отображения (view context)
- Пишем и читаем на background (background context)
- Создание background контекстов - for free

# Спрячьте создание контекстов

```
public func readSyncMain(_ closure: @escaping (NSManagedObjectContext) → Void) {
    let context = container.viewContext
    context.performAndWait {
        closure(context)
    }
}

public func readMain(_ closure: @escaping (NSManagedObjectContext) → Void) {
    let context = container.viewContext
    context.perform {
        closure(context)
    }
}

public func readSync(_ closure: @escaping (NSManagedObjectContext) → Void) {
    let context = container.newBackgroundContext()
    context.undoManager = nil
    context.performAndWait {
        closure(context)
    }
}

public func readInBackground(_ closure: @escaping (NSManagedObjectContext) → Void) {
    let context = container.newBackgroundContext()
    context.undoManager = nil
    context.perform {
        closure(context)
    }
}
```

# Merge conflict and Merge policy

```
@available(iOS 5.0, *)
open class NSMergeConflict : NSObject {

    open var sourceObject: NSManagedObject { get }

    open var objectSnapshot: [String : Any]? { get }

    open var cachedSnapshot: [String : Any]? { get }

    open var persistedSnapshot: [String : Any]? { get }

    open var newVersionNumber: Int { get }

    open var oldVersionNumber: Int { get }
```

## Merge policy

```
context.mergePolicy =
```

- error
- rollback
- overwrite
- mergeByPropertyObjectTrump
- mergeByPropertyStoreTrump

## Настройка backgroundContext

- context.mergePolicy = NSMergePolicy.overwrite

# Избавляемся от UndoManager

+10 к производительности

```
let context = container.newBackgroundContext()  
context.undoManager = nil|
```

# Работа с запросами

## Рекомендации по настройке

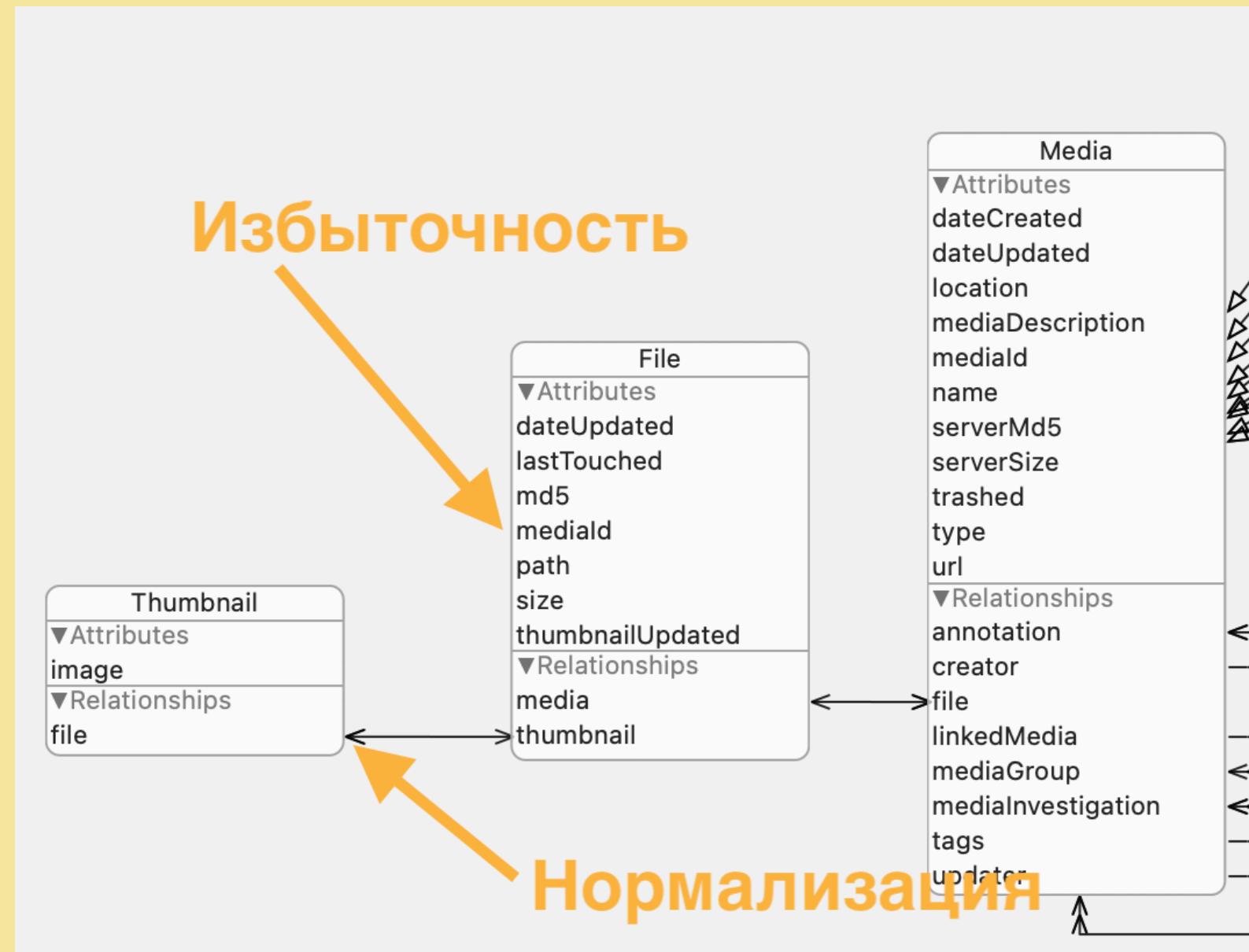
- request.fetchLimit = 1
- request.fetchBatchSize = 20

# Нормализация и избыточность для увеличения перформанса

## Советы

1. Избыточность (отказ от вложенных запросов)
2. Нормализируйте (чтобы не тянуть тяжелые данные)

# Методы рефакторинга. Пример



# NSPredicate

# Поддерживаемые операции

- NULL
- >, <, ==, !=
- AND, OR, NOT
- ANY, IN, NONE
- MATCHES, CONTAINS, like[cd]

<https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/Predicates/Articles/>

**Фильтрацию данных лучше  
делать на уровне предиката**

# Расставляйте условия в порядке увеличения вычислительной сложности

```
NSPredicate(format: "boolField == YES AND dateField  
> %@", date)
```

# Полнотекстовый поиск

# Полнотекстовый поиск через NSPredicate

"longTextField like[cd] \*%@\*"

"longTextField MATCHES '\*%@\*' "

# Полнотекстовый поиск через NSPredicate

"longTextField like[cd] \*%@\*"

"longTextField MATCHES '\*%@\*' "

Работает очень медленно

## Полнотекстовый поиск. Решение

Core Data: <id, field1, field2>

sqlite3 + FTS: <id, longTextField>

## Полнотекстовый поиск. Алгоритм

- Пользователь вводит текст
- Получаем список id из sqlite с помощью fts
- `Predicate("id IN %@", ids)` получаем выборку из Core Data

# Полнотекстовый поиск. Подводные камни

Забыли поиндексировать

## Полнотекстовый поиск. Материалы

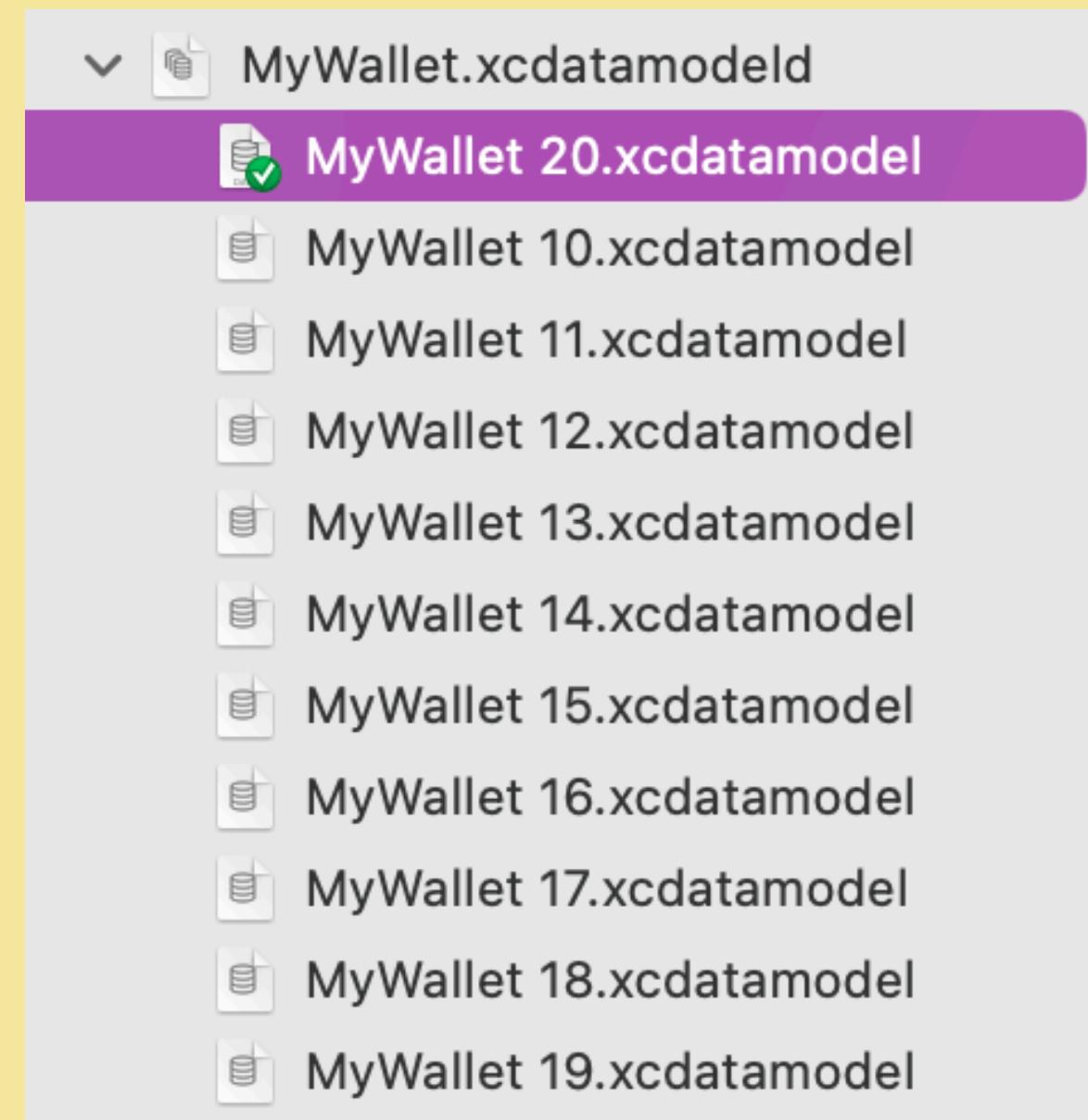
- <https://www.sqlite.org/fts3.html>
- <https://github.com/stephencelis/SQLite.swift>

## Советы

- Настройте индексы
- проверяйте `isDeleted` - показать пример с считыванием в бэкграунде и параллельным удалением

# Особенности версионирования и миграций

# Версионирование



# Миграции от X к X+1 версии

- Легковесные
- Кастомные

## Легковесные

- Добавление опционального поля или с дефолтным значением
- Добавление нового класса
- Удаление поля или класса

## Тяжеловесные

- \*.xcmappingmodel
- NSEntityMigrationPolicy

Attribute Mappings	
Destination Attribute	Value Expression
A dateCreated	◊ \$source.dateCreated
A dateUpdated	◊ \$source.dateUpdated
A location	◊ \$source.location
A mediaDescription	◊ \$source.mediaDescription
A mediald	◊ \$source.mediald
A name	◊ \$source.name
A serverMd5	◊ \$source.serverMd5
A serverSize	◊ \$source.serverSize
A trashed	◊ \$source.trashed
A type	◊ 8
A url	◊ \$source.url

Relationship Mappings	
Destination Relationship	Value Expression
O annotation	◊ FUNCTION(\$manager, "destinationInstancesForEntityMa...")
O creator	◊ FUNCTION(\$manager, "destinationInstancesForEntityMa...")
O file	◊ FUNCTION(\$manager, "destinationInstancesForEntityMa...")
M linkedMedia	◊ FUNCTION(\$manager, "destinationInstancesForSourceR...")
O mediaGroup	◊ FUNCTION(\$manager, "destinationInstancesForEntityMa...")
O medialInvestigation	◊ FUNCTION(\$manager, "destinationInstancesForEntityMa...")
O tags	◊ FUNCTION(\$manager, "destinationInstancesForEntityMa...")
O updaters	◊ FUNCTION(\$manager, "destinationInstancesForEntityMa...")

<https://www.objc.io/issues/4-core-data/core-data-migration/>

## Особенности миграций

- Миграция занимают время (продумайте UI)
- Миграция занимает память (дропайте тяжелые ненужные данные)

## Советы



- Одна версия на релиз (при активной разработке)
- Тестирование миграции
- Тестируйте мэппинги
- Используйте *in memory storage* для тестов
- Делайте перформанс тесты

# Тестирование и отладка. Флаги дебага

The screenshot shows the 'Arguments' tab of the Xcode Test Navigator. The tab bar includes 'Info', 'Arguments' (which is selected and highlighted in purple), 'Options', and 'Diagnostics'. Below the tabs, a section titled 'Arguments Passed On Launch' is expanded, showing three checked arguments:

- com.apple.CoreData.MigrationDebug 1
- com.apple.CoreData.SQLDebug 3
- com.apple.CoreData.Logging.stderr 1

At the bottom left of the list, there are '+' and '-' buttons for adding or removing arguments.

# Как получить крэш



## Как получить крэш

- Не указать обязательное поле без дефолтного значения
- Неправильно написать миграцию
- Передавать объекты за пределы потока
- Работать с объектом, который уже `isDeleted = true`

O'REILLY®

# Designing Data-Intensive Applications

THE BIG IDEAS BEHIND RELIABLE, SCALABLE,  
AND MAINTAINABLE SYSTEMS



Martin Kleppmann

## Q&A



tg [https://t.me/km\\_engineering](https://t.me/km_engineering)