параметров символов текста / Н. П. Шутько, Д. М. Романенко, П. П. Урбанович // Труды БГТУ. Сер. VI: Физ.-мат. науки и информатика. – 2015. – № 6. – С. 152–156.

2. Новосельская О. А. Алгоритмы и программное средство для генерации защитных изображений печатных документов / О. А. Новосельская, Н. А. Савчук, А. Н. Щербакова, Д. М. Романенко // Труды БГТУ. Сер. 3, Физико-математические науки и информатика. – 2022. – № 1 (254). – С. 64–72.

УДК 004.78

Д.Р. Нургалиев
Казанский национальный исследовательский
технический университет им. А.Н. Туполева
Казань, Россия

REALM. ОБЪЕКТНО-ОРИЕНТИРОВАННАЯ БАЗА ДАННЫХ ДЛЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Аннотация. *В данной статье рассматривается объектно-ориентированная база данных Realm. Проведен обзор и сравнение с другими технологиям, которые используются для хранения данных в мобильной разработке. В заключении показан пример того, как данная технология используется в языке программирования Swift.*

Ключевые слова: *Realm, объектно-ориентированная база данных, мобильное приложение, iOS, язык программирования Swift*

D.R. Nurgaliev
Kazan National Research Technical University
named after A.N. Tupolev
Kazan, Russia

REALM. OBJECT-ORIENTED DATABASE FOR MOBILE APPLICATIONS

Abstract. *This article discusses the object-oriented database Realm. Technologies that are used for data storage in mobile development were reviewed and compared against each other. In conclusion, an example of how this technology is used in the Swift programming language.*

Keywords: *Realm, object-oriented database, mobile application, iOS, Swift programming language.*

Введение

База данных Realm – относительно новый вид мобильной базы данных, которая была создана с нуля для поддержки современных приложений.

Realm – полностью построен на клиентской стороне, то есть это не обертка над существующей библиотекой или технологии.

В этой статье мы рассмотрим несколько аспектов, которые отличают Realm от других технологий, и почему же приложения с миллионами пользователями доверяют Realm.

Библиотеки баз данных и объектно-реляционное отображение

Библиотеки баз данных — это просто библиотеки сторонних разработчиков, использующие определенный механизм хранения, которые обеспечивают современный и функциональный подход к доступу к существующему слою данных.

Можно использовать библиотеки на языке Swift или Java для доступа к хранилищу SQLite, используя generics (пользовательские классы, структуры и перечисления, которые могут работать с любым типом) или структуры. Но для хранения данных по-прежнему используется SQLite. Кроме того, в любой сторонней библиотеке все равно потребуется преобразовать структуры данных в некоторый промежуточный формат, выполнить закулисные SQL-запросы и преобразовать данные в строки таблицы SQL.

Более оптимальный подход – использовать ORM (объектно-реляционное отображение). ORM – это технология программирования, которая связывает базы данных с концептами ООП языков программирования, создавая “виртуальную объектную базу данных”. При использовании полноценного ORM всегда будет множество операций, выполняемых в фоновом потоке асинхронно. ORM будет непрерывно преобразовывать объекты в промежуточный формат и запускать SQL-запросы для взаимодействия с файлом SQLite или другими СУБД. Каждый раз при обращении к объекту, ORM преобразует это в SQL-запрос JOIN, далее находит соответствующие записи в таблице, где хранятся эти связанные объекты. Весь этот процесс изображен на Рис. 1

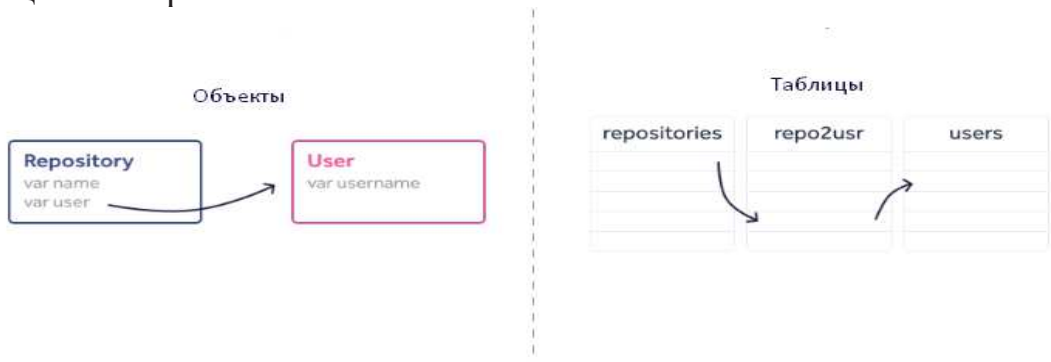

Рис. 1 - Процесс поиска объекта

Это очень ресурсоемкий процесс для простого доступа к данным. Чтобы убрать всю эту работу с основного потока, нужно внедрять правильные методы многопоточности, что делает ваш код объемным и сложно читаемым.

Уникальность Realm

Учитывая недостатки других технологий, Realm имеет собственный высокопроизводительный механизм хранения данных.

База данных Realm не связана с SQLite или другой базой данных SQL, вместо этого она направлена на решение многих проблем, существующих в этой области. Realm – это не хранилище в виде ключ – значение, так как разработчики хотят иметь возможность напрямую работать с объектами в коде. Также Realm не является ORM. ORM преобразует данные, такие как база данных SQL, в объектные графы, чтобы их можно было использовать в коде. Механизм хранения Realm не требует преобразования данных, поэтому необходимости в ORM нет.

База данных Realm сохраняет объекты непосредственно на диске с наименьшим преобразованием типов. Поскольку нет преобразования или другого сложного распутывания объектов, данная технология способна быстро загружать объекты из памяти на диск. Это позволяет записывать и читать в основном потоке, при этом не блокирую пользовательский интерфейс. На Рис. 2 представлено сравнение с другими хранилищами данных.

Ядро Realm написан на языке программирования C++, и он отображает объекты C++ на диск. Поскольку он разрабатывается вместе с SDK (комплект для разработки программного обеспечения), его API(программный интерфейс) и поведение тесно связаны с тем, как мобильные приложения фактически используют данные.

Рис. 2 - Графическое сравнение баз данных

Такое согласование напоминает iOS: Apple создает аппаратное обеспечение для iPhone, а также предоставляет разработчикам один из лучших SDK для использования этого оборудования. Именно поэтому при прокрутке длинного списка контактов на iPhone задержек не возникает, даже если аппаратное обеспечение не такое мощное, как у других устройств.

Использование надежного и мультиплатформенного ядра C++, Realm имеет потенциал работы практически на любой платформе.

Ядро, которое приспособлено к нескольким языкам программирования

База данных Realm поддерживает 5 SDK. Удобные интерфейсы, которые дают возможность пользоваться с низкоуровневым API через мобильные языки программирования. Взаимодействия ядра Realm Core с языками программирования представлено на Рис. 3.

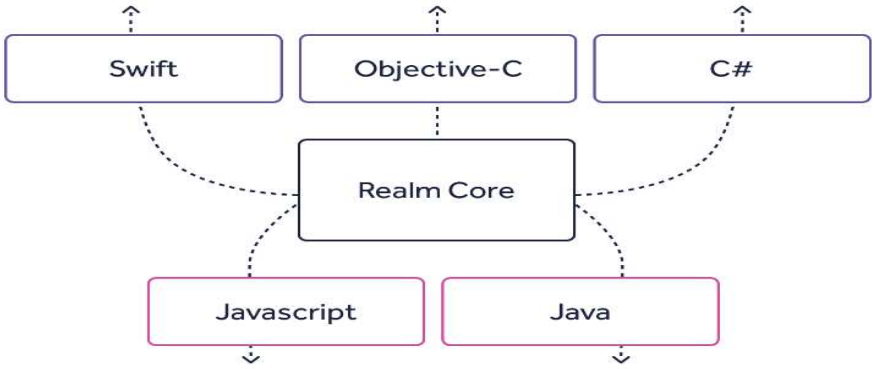

Рис. 3 - Взаимодействия ядра Realm Core с ЯП

Каждый из SDK предназначен для использования языка и возможностей платформы, к которым привыкли разработчики. Пример создания модели Realm на языке программирования Swift изображен на Рис. 4.

```
8  import RealmSwift
9
10 class Person {
11     let name: String
12     let surname: String
13     let age: Int
14
15     init(name: String, surname: String, age: Int) {
16         self.name = name
17         self.surname = surname
18         self.age = age
19     }
20 }
21
22 @objcMembers
23 class PersonRealm: Object {
24     dynamic var name: String = ""
25     dynamic var surname: String = ""
26     dynamic var age: Int = 0
27 }
28
```

Рис. 4 - Фрагмент создание модели Realm на языке Swift

В первом случае это обычный класс, второй класс – модель, для работы с базой данных Realm. Различий минимально. Ключевое слово dynamic перед переменными позволяет Realm получать уведомления об изменениях переменных модели и, следовательно, отображать их в базе данных.

На Рис. 5 показан небольшой класс-менеджер, который позволяет работать с базой данных на языке Swift. По изображению видно, что синтаксис не отличается от того, которому привыкли iOS разработчики, это позволяет быстро и надежно добавить базу данных Realm в проект.

Также существуют инструменты для визуального просмотра, редактирования и проектирования файлов базы данных Realm. Одним из них является Realm Studio (Рис. 6).

С помощью него разработчики могут:

- Запрашивать объекты в базе данных
- Просматривать объекты в режиме реального времени во время запуска приложения
- Создавать, изменять и удалять объекты
- Добавлять классы и свойства в схему
- Экспортировать схемы в виде определений классов в C#, Java, JavaScript, Kotlin, Swift
- Сохранять и импортировать изменения в клиентское приложение.

```
88  class RealmManager {
89      // создание главное объекта Realm
90      private lazy var mainRealm: Realm? = {
91          do {
92              return try Realm(configuration: .defaultConfiguration)
93          } catch {}
94          return nil
95      }()
96
97      // метод для сохранения объекта в базу данных
98      func save(object: Object) {
99          do {
100             try mainRealm?.write {
101                 mainRealm?.add(object)
102             }
103         } catch {
104             print(error.localizedDescription)
105         }
106     }
107
108     // сортировка объектов по имени
109     func sortBy(name: String) -> PersonRealm? {
110         if let models = mainRealm?.objects(PersonRealm.self) {
111             let filtered = models.first { $0.name == name }
112             return filtered
113         }
114
115         return nil
116     }
117 }
118
```

Рис. 5 - Класс-менеджер для работы с БД Realm

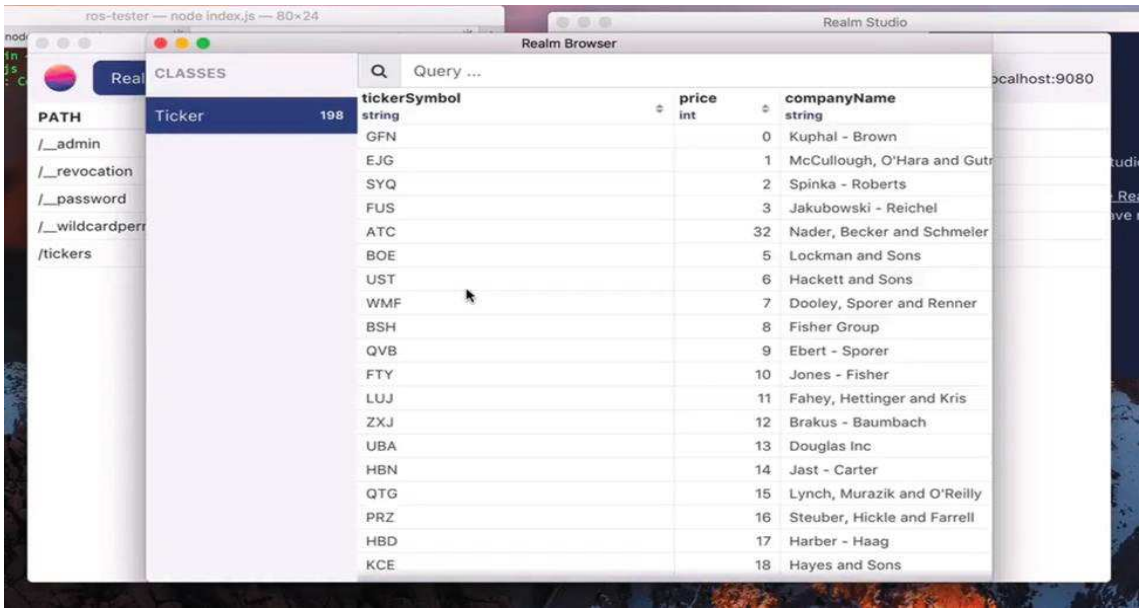

Рис. 6 - Инструмент Realm Studio для визуальной работы с объектами Realm

Заключение

В заключение отметим пункты, которые выделяют Realm перед другими технологиями:

Скорость: Использование библиотеки Realm для работы с базой данных выполняется очень быстро. Это быстрее, чем SQLite и CoreData.

Кроссплатформенность: Файлы базы данных Realm являются кроссплатформенными и могут совместно использоваться iOS и Android. Независимо от того это Java, Objective-C или Swift, можно использовать одну и ту же абстрактную модель для доступа.

Расширяемость: Если мобильное приложение должно обрабатывать большое количество записей пользовательских данных, масштабируемость базы данных очень важна. Это следует принять во внимание, прежде чем начинать разработку приложения. Realm обеспечивает хорошую масштабируемость и очень быструю работу при работе с большими объемами данных. Выбор использования Realm может повысить скорость работы приложения и повысить удобство работы с ним.

Обратная связь: Realm предоставляет удобную и подробную документацию. При необходимости также с ними можно связаться в Twitter или Github.

Доверие крупных продуктов: Realm использовалась большим количеством стартапов и крупных компаний в своих мобильных приложениях, таких как Pinterest, Dubsmash и Hipmunk.

Список используемых источников

1. “Документация базы данных Realm” [Электронный ресурс]. URL: https://realm.netlify.app/docs/swift/latest/
2. “Документация языка программирования Swift” [Электронный ресурс]. URL: https://www.swift.org/documentation/
3. “Realm tutorial: Getting started” [Электронный ресурс]. URL: https://www.raywenderlich.com/1464-realm-tutorial-getting-started
4. “Core Data tutorial: Getting started” [Электронный ресурс]. URL: https://www.raywenderlich.com/7569-getting-started-with-core-data-tutorial
5. “SQLite tutorial” [Электронный ресурс]. URL: https://www.sqlitetutorial.net/
6. “Серверная сторона Swift – объектно-реляционное отображение” [Электронный ресурс]. URL: https://medium.com/@mczachurski/server-side-swift-object-relational-mapping-orm-68879d9a1aa3

УДК 77.06:004.032.26

Н.С. Павлова
Казанский национальный исследовательский технический университет
им. А.Н. Туполева – КАИ
Казань, Россия

УЛУЧШЕНИЕ КАЧЕСТВА ФОТО С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

***Аннотация**. В данной статье приведена информация о том, как работают нейронные сети, как можно использовать нейронные сети в работе с изображениями и как в этом может помочь Google Colaboratory.*

N.S. Pavlova
Kazan National Research Technical University
named after A.N. Tupolev – KAI
Kazan, Russia

IMPROVING PHOTO QUALITY USING NEURAL NETWORKS

***Abstract.** This article provides information on how neural networks work, how neural networks can be used in working with images, and how Google Colaboratory can help with this.*