СИСТЕМА УПРАВЛЕНИЯ ОБЪЕКТНОЙ БАЗОЙ ДАННЫХ REALM

Вадим МОКАН

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *Данная работа посвящена базе данных Realm. Описаны особенности этой популярной базы данных, её принципы функционирования. Статья даёт ответы на множество вопросов, которыми задаются разработчики мобильных приложений при поиске и выборе оптимальной базы данных, в зависимости от поставленных задач.*

Ключевые слова: *база данных, реляционная, мобильная, React Native, объект, кроссплатформенность, ключ, синхронизация.*

Введение

Realm - это система управления объектной базой данных с открытым исходным кодом, первоначально созданная для мобильных платформ Android, iOS, а также доступна для таких платформ, как Xamarin или React Native, и др. Нативная NoSQL база даных обладает своим собственным ядром. Полностью заменяет старомодные основанные на SQLite базы данных которые являются альтернативой для разработчиков на сегодняшний день.

Стандартные ORM (Object-Relational Mapping) связанные базы данных не обходятся без копирования:

- Данные на диске;
- Данные считываются с диска;
- Копирование исходных данных в десериализованное промежуточное представление в памяти. (выделение памяти);
- Копирование промежуточного представления в языковой уровень в памяти;
- Возврат конечного объекта.

Realm обходится без копирования:

- Вычисляет смещение данных для чтения;
- Считывает из соответственного файла;
- Возврат конечного объекта.

1. Особенности Realm

Управление параллельным доступом с помощью многоверсионности - один из механизмов обеспечения параллельного доступа к БД, заключающийся в предоставлении каждому пользователю так называемого «снимка» БД, обладающего тем свойством, что вносимые пользователем изменения в БД невидимы другим пользователям до момента фиксации транзакции. Этот способ управления позволяет добиться того, что пишущие транзакции не блокируют читающих, и читающие транзакции не блокируют пишущих.

В упрощенном виде этот механизм можно представить следующим образом: все операции с данными можно условно разделить на чтение (select), вставку (insert), удаление (delete), обновление (update).

Realm поддерживает ACID транзакции:

Atomicity — транзакции атомарны, то есть либо все изменения транзакции фиксируются (commit), либо все откатываются (rollback);

Consistency — транзакции не нарушают согласованность данных, то есть они переводят базу данных из одного корректного состояния в другое. Тут можно упомянуть допустимые значения полей, внешние ключи и более сложные ограничения целостности;

Isolation — работающие одновременно транзакции не влияют друг на друга, то есть многопоточная обработка транзакций производится таким образом, чтобы результат их параллельного исполнения соответствовал результату их последовательного исполнения;

Durability — если транзакция была успешно завершена, никакое внешнее событие не должно привести к потере совершенных ей изменений.

Другими особенностями Realm являются:

- Realm - это не единственная база данных в приложении. Обычно в приложении используется одна SQL база данных, тогда как можно создать множественное количество Realm-ов для работы с данными более эффективно, для их распределения и контроля.
- Realm - не таблица! Таблицы обычно хранят только один вид информации: записи пользователя, сообщения электронной почты и т.д. Realm может содержать несколько видов объектов.
- Realm не является хранилищем документов. Поскольку свойства объектов аналогичны парам ключ: значение, легко представить Realm как хранилище, но объекты имеют определенные схемы, которые поддерживают присвоение значений по умолчанию или их маркировку по мере необходимости.
- Как только объект был добавлен в Realm, все действия, связанные с ним в коде проводятся и в самом Realm автоматически, не нужно вызывать вспомогательный метод для обновления или лишнего добавления в Realm для проведения изменений. Они будут сделаны автоматически.
- Это касается и синхронных объектов Realm, не нужно прилагать никаких усилий для проведения изменений на серверной стороне. Изменяем объект и при дальнейшем подключении устройства к сети, изменения распространятся на сервер и на другие клиенты Realm, которые синхронизируются с этими объектами.

Заключение

Realm сложнее, чем кажется на первый взгляд. Однако все недостатки с лихвой покрываются его мощностью и удобством. Live объекты, нотификации и реактивность, удобный интерфейс и множество других вещей упрощают создание приложений. Полностью же Realm раскрывает себя при построении оффлайн приложений, когда все данные мы получаем из кэша и нам необходимы сложные выборки, а также постоянное обновление данных.

Realm лучше использовать по следующим причинам:

- Быстрые запросы - Realm предоставляет результаты относительно быстрее, чем SQLite, и имеет большую производительность;
- Лёгкость в использовании - Realm гораздо проще в использовании, благодаря тому, что он является объектной базой данных. Просто создав класс, который расширяет класс RealmObject, вы настраиваете абсолютно всё;
- Кроссплатформенность - Хорошая альтернатива SQLite (Android) и CoreData (IOS). Его можно использовать на всех платформах, таких как iOS, Xamarin и React Native;
- Простое создание и хранение данных.

Библиография

1. System Properties Comparison Firebase Realtime Database vs. Realm vs. SQLite. - [Электронный ресурс]. – Режим доступа: https://db-engines.com/en/system/Firebase+Realtime+Database%3BRealm%3BSQLite
2. Realm Databese. - [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Realm_(database)
3. Реалистичный Realm. - [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/328418/
4. Realm Sync Documentation. - [Электронный ресурс]. – Режим доступа: https://docs.realm.io/sync/