Project Part 3: Realm-Java

Authors: Wen-Chia Yang, Junxian Chen, Zihua Weng

February 20, 2020

ABSTRACTION

Objective

Realm Database provides a creative database alternative for mobile app development.[1] It is an embedded database that could run directly on a variety of devices and overcomes the hardware constraint to provide a smoother user experience. realm-java is an SDK for Realm Database for Java and Android platforms. It was created to connect the Realm Database core which is written in C++ and aims to use the Java language and platform features while introducing the new database[2]. To better understand what is Realm, we need to dive deep into both realm-java (the Java wrapper project) and realm-core (the core project that implements the entire database).

Goals

To extend our understanding of realm-java, we need to gain a big picture of this project. First of all, we want to understand its stakeholders, functionalities and key developers. Additionally, we also want to explore its existing open issues on Github and try to get involved in contributing to the project.

Source Code

Solution-Accepted/realm-java: https://github.com/solution-accepted/realm-java
Branch: master

RESEARCH

Stakeholders

According to the definition from a business website, a person, group or organization that has interest or concern in an organization. Stakeholders can affect or be affected by the organization's actions, objectives and policies[3]. Here, we are listing four different types of stakeholders.

1. **Mobile App Developers**: developers who use the Realm to create personal apps.
2. **Realm Database customers**: companies who use the Realm in their products
   ex: Google, Amazon, eBay, etc[4].
3. **MongoDB Company**: who owns the Realm project[5].
4. **Realm Database contributors and developers**: Realm is an open-source project that all developers could contribute to.

Key Developers

According to Github website, there are 86 contributors in the realm-java repository[6]. In this research, we pick several people who contributed more compared to other developers. First of all, based on Christian Melchior's LinkedIn profile[7], we think he is the core maintainer. This is because he has worked at Realm since 2014 (now in MongoDB) as Android Lead and he contributes the most commits (over 1000 times). Secondly, we think Christian Melchior, Mulong Chen[8], Emanuele Zattin[9], Kenneth Geisshirt[10], Makoto Yamazaki[11], Nikolche Mihajlovski[12], Nabil Hachicha[13], Kristian Spangsege[14] and Brian Munkholm[15] are developers of a team that is mainly focusing on realm-java. For testers and documentation writers, we think Emanuele Zattin is in charge of this role according to his LinkedIn profile. As for triagers, they might be Christian Melchior, Brian Munkholm, Emanuele Zattin, and Gabor Varadi.

Realm

Realm is an embedded, NoSQL, object-oriented database built from scratch to overcome mobile hardware constraints and provide better database solution for mobile developers. The structure of Realm is composed of two parts. The core project named realm-core (written in c++), which implements the entire database features from scratch. The others are wrapper projects (realm-java, realm-cocoa, realm-js and realm-dotnet) provide high-level abstractions of the realm-core, keeping most functionalities down to several classes.

In Realm, the most essential functional aspects are as follows.

1. **Realm is an embedded database** built from the ground up to run directly inside phones, tablets or wearables. It was optimized for the hardware constraints of devices[16].
2. **Realm is an object-oriented database.** Data in Realm is directly exposed as objects and queryable by code, removing the need for Object-relational mapping's riddled with performance & maintenance issues[17].
3. **The core of Realm is written in super-fast C++ and Realm is cross-platform** supporting multiple languages, such as Java, JavaScript, Swift, C#, etc.. Each SDK is designed to use the language and platform features that developers are used to[18].
4. **Realm reads and shows data with zero-copy.** Files are always memory-mapped and the format on disk is readable in memory without having to do any deserialization. When showing data, Realm reads the value from the offset to its length and returns the raw value from the property access[19].
5. **Realm stores properties contiguously linked at the vertical level.** It enables users to load specific properties to memory rather than loading the whole row data into memory as SQLite[20].
6. **Realm uses Multiversion Concurrency Control which provides concurrent access to the database.** Users have full isolation in their transactions without replicating data. Therefore, reading and writing can happen in different spots in the project, on different threads, from different processes[21].

7. **Realm syncs data seamlessly in real-time** with the Realm Object Server without the need to write networking, serialization, or object-relational mapping code. Users can refresh data as fast as needed[22].

On the other hand, the essential non-functional aspects are as follows.

1. **Simple.** It keeps API down to a few classes which enable developers to pick up the database easily[23].
2. **Built-in Encryption**. It keeps users' data safe with AES-256 bit encryption, strong enough for global financial institutions[24].
3. **Fine-Grained Notifications.** Realm will not only send a notification when a change has occurred, but also the precise indexes of the objects that have been inserted, updated, or deleted. Subscribe to notifications to help get updates when data changes, then update UI[25].
4. **Crash Safety.** Realm is similar to gigantic B-tree, and users have the top-level commit. When users are making changes, it forks the B-tree and writes without modifying the existing data. When the transaction confirmed, it moves the pointer to the new branch. When the devices crash, existing data remains the same[26].
5. **Open Source.** The Realm Database is developed in the open. Thousands of developers work with on the GitHub project, plus hundreds of more write add-ons that work with the Realm Database[27].
6. **Extension.** Realm provides Realm Platform extends the mobile database to the server-side. It could sync data in Realm Database among devices and support real-time collaboration[28].

Uniqueness

1. **Brand-new**: Realm is specially designed for mobile platforms. It is built from scratch to support mobile apps. It is NOT related to SQLite or SQL language, NOT a key-value store, NOT an ORM (Object-Relational Mapping), but a new database that persists objects directly on disk to give an extraordinary performance. Its customized core, written in C++, currently has 5 SDKs[29].
2. **Performant**: Its SDKs (or APIs, Wrappers) are optimized to provide fast and low-memory performance. For example, on Apple platforms, there are middle-wares like Results Controller, ORM and SQLite between the actual code and data, while with Realm SDKs you will only need a class called Results to fetch data. With that being said, Realm cuts a lot of middle parts in its design to achieve fast I/O[30].
3. **Elegant**: Realm helps you implement good practices. With the usage of Realm, the architecture of an app can be split into 3 layers: data storage and observation layer, shared application and presentation logic layer, and views and view controlling layer. This organization makes an app message-driven, resilient and responsive[31].
4. **Live**: In the past, data fetched from databases is static, i.e. it is frozen in time at the precise moment when it was retrieved from the disk; as soon as it is fetched, it can be considered outdated. But Realm is a Live Object Database, which means the data you access is always current and fresh[32].
5. **Fine-grained**: Realm SDKs allow developers to provide partial UI updates based on fine-grained notifications. UI components now can know about when and what changes occur to the data. By knowing “when”, UI can be updated promptly to reflect the latest data. By knowing “what”, UI can be controlled in a detailed view. That is to say, for example, only an item in a list needs to be changed, instead of the whole list being refreshed[33].

Issues That We May Be Able To Address

1. **Provide ability to create new RealmConfiguration. Builder from an existing realm configuration** https://github.com/realm/realm-java/issues/3382

Analysis:
This issue is opened by Zak Taccardi because of testing purposes. We think this is practical and doable because if Realm-java has a public API named realm.getConfiguration.newBuilder(), it can reduce the repetitive work for developers.

To address this issue, it might be related to modifying RealmConfiguration.java. Currently, the class only provides a Builder class with one build function. We can create a convenient newBuilder method just copy the existing configurations but with a different file name. It seems not that hard for this issue, but I was wondering if this additional functionality will influence other structures or not.

2. **Inheritance / Polymorphism** https://github.com/realm/realm-java/issues/761

Analysis:
This issue is requested by many developers and Christian Melchior then opened it. We think this feature is useful for Java. For now, many developers use interface inheritance to solve this problem, but they also mentioned that this method won't work for certain compositions.

To address this issue, we thought the code in RealmObject.java, RealmObjectProxy.java, and ProxyState.java are most related to this issue. We assumed that we might need to add new abilities in ProxyState because this class provides multiple methods relating to Row or Column information mapping to the RealmObject.

3. **Add RealmGeoPoint as first-class citizen** https://github.com/realm/realm-java/issues/1772

Analysis:
This issue is opened by dadino. As he mentioned, some libraries, such as MongoDB or Parse, already supported geographical query, but Realm didn't.

To address this issue, we think there are two ways. For one, we can add a new class called GeogQuery to implement all geographical queries, such as a query for latitude, longitude or a point. But if we do this way, for other languages that might need to add its own GeogQuery class. Therefore, the better way is to do this in the realm-core library.

4. **@SerializedName annotation for defining Class/Field names** https://github.com/realm/realm-java/issues/1470

Analysis:
This issue is opened by Chen Mulong. The other one named TR4Android followed up this issue and further requested to add @SerializedObjectId(s)Name annotation for supporting fields in an object in JSON. Christian Melchior, the core maintainer, said that this is a cross-platform issue between GSON and Realm, and it is actually a big issue.

To address this issue, we thought its related to the annotations. Hence, we trace code in realm-annotations-processor/RealmProcessor.kt. In this class, we found @SupportedAnnotationTypes to define every annotation. Further, we also needed to add annotation process logic in this class. That's how we think we can address this issue. However, after diving deep into the issue, we found that Christian Melchior replied (https://github.com/realm/realm-java/issues/1470#issuecomment-253271489) and pointed out every class that would be influenced if adding the new annotation. It's so complicated that we need to further consider the RealmObject and its relationship with the schema.

5. **New feature: findLast(int), findFirst(int), findRandom(int)** https://github.com/realm/realm-java/issues/1050

Analysis:

This issue is opened by Sani crysan. He suggested to add findLast(int), findFirst(int) and findRandom(int).

To address this issue, we traced the code into TableQuery.java. In this class, it defines every query methods here. We can implement new functions here by checking if it's a valid query and then call native c++ methods. Or we can just combine the existing query result and implement them as Sani's request.

References

[1] Realm about https://realm.io/about

[2][23][24] realm-java README https://github.com/solution-accepted/realm-java

[3] definition of stakeholder http://www.businessdictionary.com/definition/stakeholder.html

[4] Realm customers https://realm.io/customers

[5] Realm blog https://realm.io/blog/mongodb-to-acquire-realm-the-future-is-bright/

[6] realm-java contributors https://github.com/realm/realm-java/graphs/contributors

[7] LinkedIn: Christian Melchior https://www.linkedin.com/in/christianmelchior/

[8] LinkedIn: Mulong Chen https://www.linkedin.com/in/mulong-chen-632a5281/?originalSubdomain=cn

[9] LinkedIn: Emanuele Zattin https://www.linkedin.com/in/emanuelezattin/

[10] LinkedIn: Kenneth Geisshirt https://www.linkedin.com/in/kennethgeisshirt/?originalSubdomain=dk

[11] LinkedIn: Makoto Yamazaki https://www.linkedin.com/in/zaki50/

[12] LinkedIn: Nikolche Mihajlovski https://www.linkedin.com/in/nikolche-mihajlovski-a5862824/

[13] LinkedIn: Nabil Hachicha https://www.linkedin.com/in/nabil-hachicha-829a8052/?originalSubdomain=uk

[14] LinkedIn: Kristian Spangsege https://www.linkedin.com/in/kristianspangsege/?originalSubdomain=dk

[15] LinkedIn: BRIAN MUNKHOLM https://www.linkedin.com/in/brianmunkholm/?originalSubdomain=dk

[16][17] Realm-java https://github.com/realm/realm-java

[18] Realm Academy https://academy.realm.io/posts/realm-object-centric-present-day-database-mobile-applications/

[19][20][21][26] Realm's Core DB Engine https://academy.realm.io/posts/jp-simard-realm-core-database-engine/

[22] What is Realm Platform? https://docs.realm.io/sync/what-is-realm-platform

[25] Realm Updates https://academy.realm.io/posts/live-objects-fine-grained-notifications-realm-update/

[27] Realm Database https://realm.io/products/realm-database

[28] Crafting Collaborative Apps with Realm https://academy.realm.io/posts/craft-collaborative-apps-with-realm/

[29] Realm is an Object-Centric Modern Database for Mobile Apps
https://academy.realm.io/posts/realm-object-centric-present-day-database-mobile-applications

[30] The Realm API is Optimized for Performance & Low Memory Use
https://academy.realm.io/posts/realm-api-optimized-for-performance-and-low-memory-use/

[31] The Realm SDK Enables Clean and Easy Separation of Concerns https://academy.realm.io/posts/realm-sdk-clean-easy-separation-of-concerns/

[32][33] Live Objects and Fine-Grained Notifications: Realm Updates https://academy.realm.io/posts/live-objects-fine-grained-notifications-realm-update/