Overview

Team: Solution Accepted;
Members: Junxian Chen, Wen Chia Yang, Zihua Weng;
Project: realm-core (https://github.com/realm/realm-core)

Changelog

For the resubmission of assignment 2, here are the changes that we made.

1. Switched project from realm-java to realm-core. We found that realm-java is a pure wrapper that doesn't have any actual essential features, thus we chose realm-core instead.
2. Changed feature 1 from "Adding more query methods" in realm-java to "Table" in realm-core. Changed feature 2 from "Adding more encryption methods" in realm-java to "Query" in realm-core.
3. Removed all code blocks because they were not helpful for understanding how different parts were related. We used a higher-level view and analyzed the dependencies and relationships of each essential feature.

Feature 1: Table

Introduction

Realm-core is a library that implements the entire database from scratch. In every database, a table is definitely an important feature because it will be involved in every query. Therefore, we choose “table” as an essential feature and it is located in table.hpp and table.cpp.

In Realm, a database is composed of at least one table. A table is composed of at least one column and also composed of rows greater or equal to 0. For each row, there only contains one value per column. Basically, a table contains a lot of abilities, such as add/remove rows, columns, search index, get/set row/column information, values, and aggregate/searching functions, etc. Figure 1.1 shows a class diagram for the relationship between a table and its subcomponents. We’ll point out some important components inside the Table class.

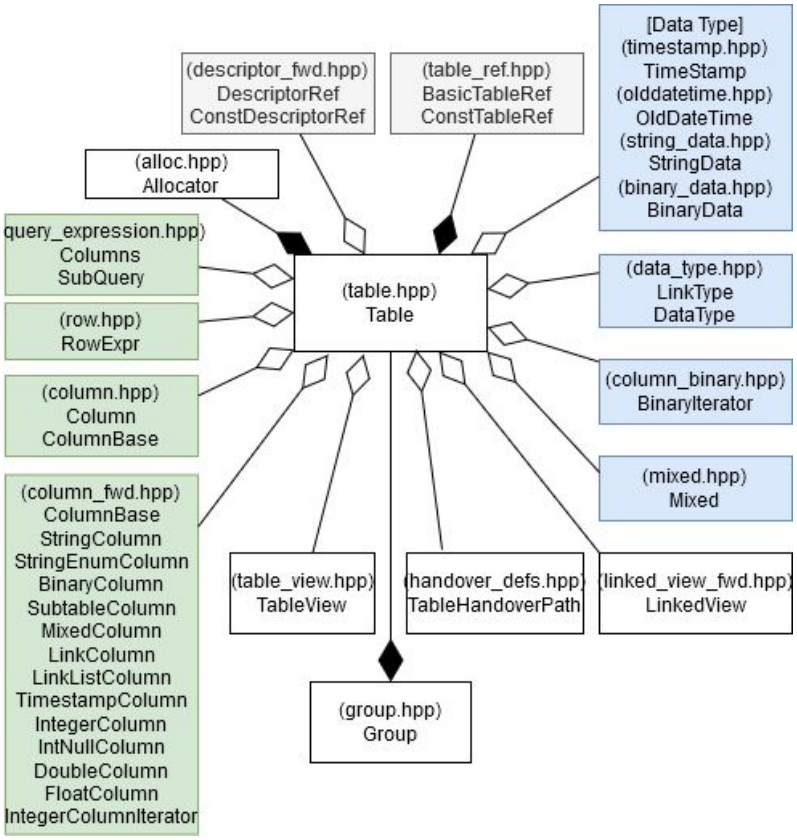

Figure 1.1 Table’s UML class Diagram. Here similar classes or classes from the same header (hpp) files are placed into a box. Names of classes begin with a capitalized letter.

Dependencies

table.hpp

Class Table is a composed component that defines all properties and methods per table. The following files are related to this file to build the concept “table” in Realm.

alloc.hpp

Class Allocator serves as a role that allocates memories of a table in the disk.

descriptor_fwd.hpp

Class DescriptorRef is a dynamic type descriptor for a complex table, and class ConstDescriptor is a dynamic type descriptor for a single table. Here, a table type descriptor is an entity that specifies the dynamic type of a Realm table. Each column descriptor specifies the name and type of the column.

table_ref.hpp

Class TableRef is a reference counting(also known as a pointer) for referencing table accessors. An accessor is a Realm specific term to represent an element that gives access to a specific piece of data. There are different types of accessors, field, row, column accessors, etc. Accessors point to specific data locations, whenever the underlying data is changed, accessors need to be updated. Accessors cache key information in order to improve access efficiency.

timestamp.hpp, olddatetime.hpp, string_data.hpp, binary_data.hpp (Primitive Type Classes)

These classes are basic primitive types that are defined in Realm.

data_type.hpp

Class LinkType is a link type for the column, and class DataType is a data type for the column.

column_binary.hpp

A binary column (BinaryColumn) is a single B+-tree, and the root of the column is the root of the B+-tree. Leaf nodes are either of type ArrayBinary (array of small blobs) or ArrayBigBlobs (array of big blobs).

Junxian Chen, Wen Chia Yang, Zihua Weng
Feb. 22, 2020

mixed.hpp

Class Mixed represents a polymorphic value, which means the value stored in this class might be bool, int64_t, float, double, StringData, BinaryData, OldDateTime or Timestamp.

query_expression.hpp

This file enable to write queries in C++ syntax like: Expression* e = (first + 1 / second >= third + 12.3); Type conversion/promotion semantics is the same as in the C++ expressions, e.g float + int > double == float + (float)int > double.

row.hpp

File row.hpp is a composition of different row properties and operations. Here, table.hpp is associated with Class BasicRowExpr, which is a class that inherited from class RowFuncs. Class RowFuncs defines operations that a row assessor can do, such as getters or setters of int/String/Timestamp/binary/bool/float/double, etc. Class BasicRowExpr, on the other hand, is a special type of row assessor that only existed as temporaries instead of a real row assessor(Class BasicRow).

column.hpp

Class ColumnBase is an abstract that serves as a basic class for all columns. It defines multiple abstract methods such as operations of rows, index, write, etc. According to the documentation, Class Column is a single B+ tree, which is a common data structure that is used in storing data in a database, and the root of the column is the root of the B+ tree. All leaf nodes are arrays.[1.1]

column_fwd.hpp

File column_fwd serves as a mapping file and re-defines several convenient columns.

group.hpp

Group represents a collection of tables (database). A Group is also called a Realm file or "database accessor".

table_view.hpp

According to the documentation, a TableView gives read and write access to the parent table. Class TableView is inherited from TableViewBase, which is a view that is built from the query result.

Junxian Chen, Wen Chia Yang, Zihua Weng

linkedview_fwd.hpp

File linkedview_fwd is a reference file that defines two types of references for LinkView. Class LinkView serves as a role like LinkList.

handover_defs.hpp

Struct TableHandoverPatch is a data structure that contains information about whether the table is sub-table or not, the number of tables, column index, and row index.

Relationship

We also analyze the interactions between a Table and its relevant classes. Here, we draw a sequence diagram that displays the interactions of adding a table as shown in Figure 1.2.

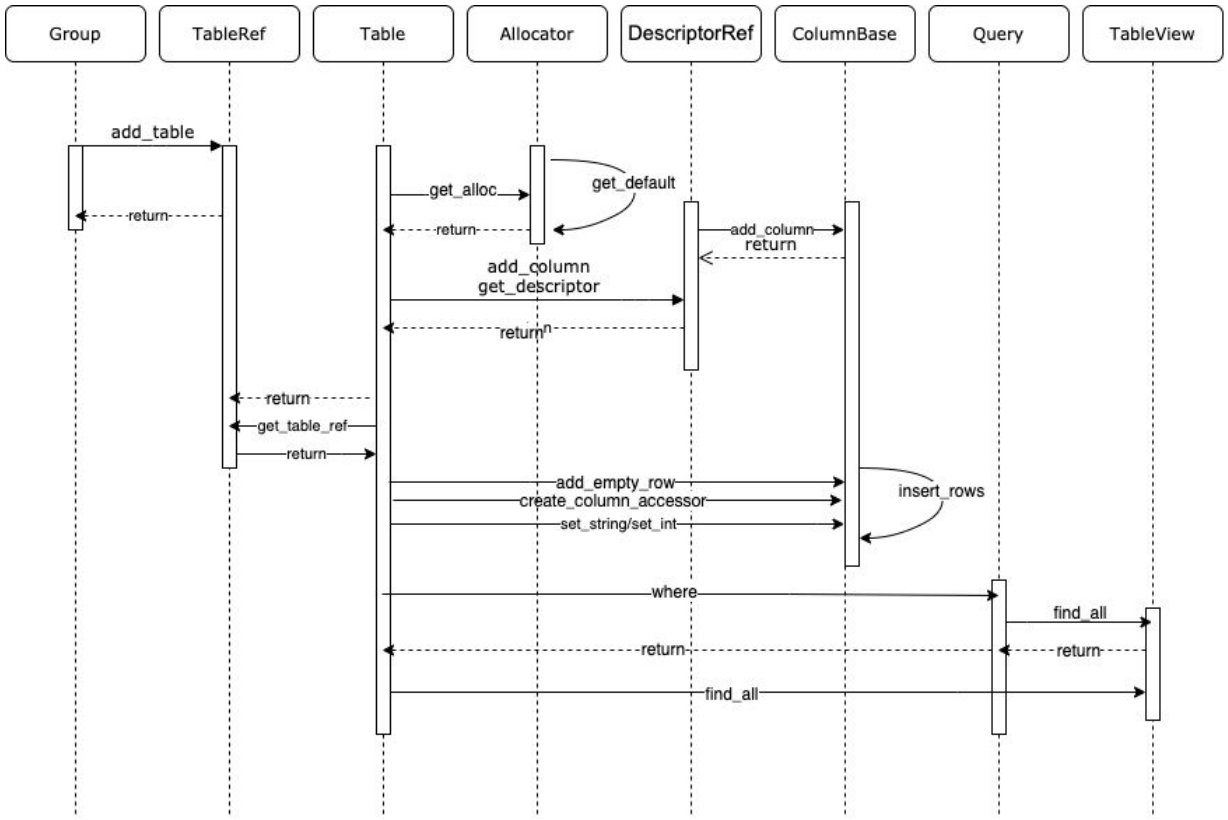

Figure 1.2 Table's Sequence Diagram (add table)

Group could add_table() which returns a TableRef. The TableRef data structure contains the Table. Table is initiated with Allocator return by get_default() method. While adding columns in the table, Table object will first get DescriptorRef and then DescriptorRef add the new

ColumnBase object. While adding rows, Table object calls ColumnBase to insert rows as in Realm, all data in the same column grouped together. By calling where, Table object creates Query object for the following query operations. Find_all method could show the TableView.

Feature 2: Query

Introduction

Query is an essential feature because it is the only way for developers to express their intentions of accessing or modifying data through code.

Under the “/src/realm” folder of realm-core, there are source code files whose names begin with “query”, which is a strong beacon for our research. We decided to start with the Query class declared in query.hpp.

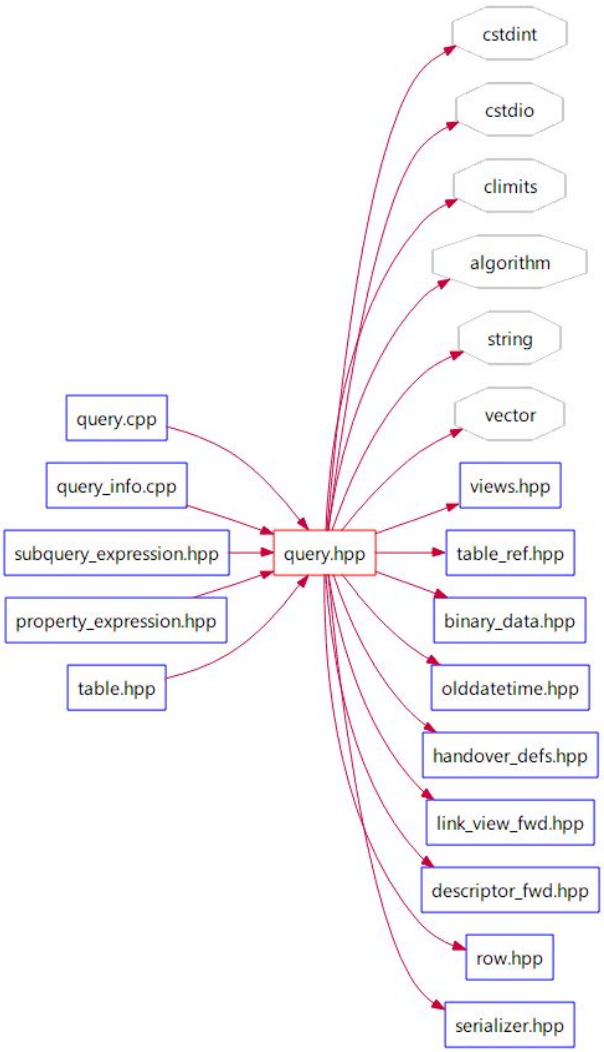

Figure 2.1: The “include” relationships of query.hpp. The directions of arrows indicate how files are included. For example, query.cpp has an arrow that points to query.hpp, which means query.cpp includes query.hpp in its code.

Dependencies

query.hpp

This header file declares all methods of the Query class and implements two special inline Query methods “equal” and “not_equal”. An official document [2.1] indicates that a Realm query consists of two parts, conditions and actions. By reading through the declaration of the Query class, we can verify this because we have found both conditions and actions there, such as conditions like equal, not_equal, greater, less, between, and actions like find, find_all, count, sum, average, maximum, minimum.

assert.hpp

A macro named REALM_ASSERT defined in assert.hpp is used by query.hpp at line 438.
This file serves as a utility for writing assertion statements.

binary_data.hpp

The header query.hpp includes binary_data.hpp at line 39 without further usages.
This file declares a class BinaryData which is a reference to a chunk of binary data.

descriptor_fwd.hpp

The class Query has a member named “m_current_descriptor”, whose type is ConstDescriptorRef declared in descriptor_fwd.hpp.
This file is the forward declaration of class Descriptor, which is the accessor for table type descriptors. A table type descriptor is an entity that specifies the dynamic type of a Realm table.

group.hpp

Two methods of class Query named “apply_and_consume_patch” and “apply_patch” need a Group reference as the parameter “dest_group” (destination group).
This file declares the class Group, which is a collection of named tables.

handover_defs.hpp

The class Query gives QueryHandoverPatch an alias as HandoverPatch, and references HandoverPatch in multiple methods.
QueryHandoverPatch is a structure defined in handover_defs.hpp. It contains three pointers to different types of patches and a vector of QueryNodeHandoverPatch pointers.

link_view_fwd.hpp

The header query.hpp includes link_view_fwd.hpp at line 42, but it does not explicitly use link_view_fwd.hpp in its declaration of the class Query.
link_view_fwd.hpp is a forward declaration of the class LinkView. There it declares the class LinkView, a standard shared pointer type of LinkView as LinkViewRef, and finally a constant LinkViewRef as ConstLinkViewRef. As written on line 41 of link_view.hpp[2.2], the class LinkView is a special Linked List structure.

olddatetime.hpp

The class Query has many methods for writing conditional logic, some of which are for comparing date. query.hpp needs olddatetime.hpp to acquire details of the OldDateTime class. The class OldDateTime is a data type for columns for storing date and time information.

query_engine.hpp

query.hpp does not include query_engine.hpp, but it does have a method named “root_node” that returns a ParentNode pointer. ParentNode is a class declared in query_engine.hpp, and it is the superclass of all the different Nodes declared in query_engine.hpp.

row.hpp

query.hpp includes row.hpp at line 44 without further usages.
Row is an important concept in a database because it represents a record on a table. Inside row.hpp we can see the declarations of RowBase, RowFuncs and BasicRow.

serializer.hpp

query.hpp includes serializer.hpp at line 45 without further usages.
serializer.hpp contains a collection of “print_value” functions designed for different data types. Its purpose is to convert all data types into one unified format (std::string).

string_data.hpp

There are two inline functions “equal” and ”not_equal” that cast “const char*” to StringData in their function bodies. These two functions are shortcuts for equal(StringData(c_str)) and not_equal(StringData(c_str)), and are needed to avoid unwanted implicit conversion of char* to bool. Also, StringData is the data type for string comparisons in a Realm Query.
StringData is a reference to a chunk of character data.

table.hpp

Tables are used as a parameter for Query constructors.
table.hpp declares the class Table, one of the most important classes in Realm.

table_ref.hpp

The class Query has a member variable named “m_table”, which is a TableRef type. This variable has a setter and a getter.
TableRef is different from Table because TableRef is a typedef of BasicTableRef<Table>. A BasicTableRef is a reference-counting "smart pointer" for referring to table accessors.

table_view.hpp

The class Query has a member “m_source_table_view”, which is a TableViewBase pointer. This member can only be set by passing parameters through constructors.
Traditionally speaking, views are virtual tables built on top of tables or other views. Similarly, Realm views are built through queries against either tables or another view.

views.hpp

The class Query has a member “m_view”, which is a RowIndexes pointer. It points to the base class of the restricting view.
The class RowIndexes is for common functionality of ListView and LinkView which inherit from it. Besides RowIndexes, views.hpp also declares several Descriptors.

Relationship

Searching is one of the most important queries. After reading the function body of the “find” method, we may produce a sequence diagram that matches a certain situation.

Junxian Chen, Wen Chia Yang, Zihua Weng
Feb. 22, 2020

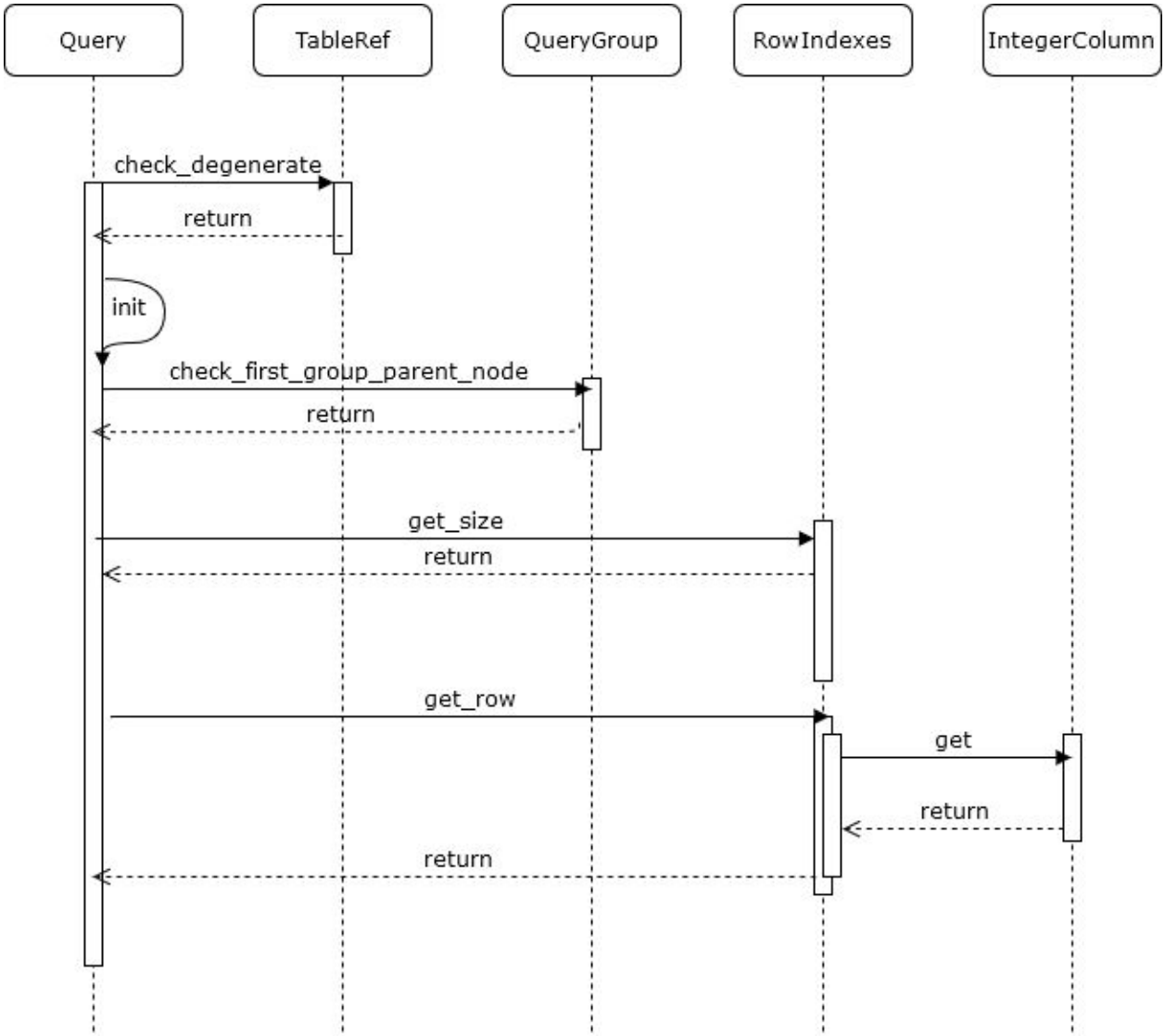

Figure 2.2: A sequence diagram for searching query

From this diagram we can tell a Query can have interactions with classes like TableRef, QueryGroup, RowIndexes, and IntegerColumn. First, the Query wants to know if its TableRef is degenerate. If so, then this process will end by returning “not_found”. Secondly, the query wants to know if it is created with conditions (because sometimes a Query is created without conditions by calling table.where()) by checking its QueryGroup array. Thirdly, if the check returns false, then it will try to fetch rows from RowIndexes, and returns the first row whose index is greater than the given parameter.

Reference

[1.1] Definition of Column https://github.com/realm/realm-core/blob/master/src/realm/column.hpp

[1.2] Definition of TableView
https://github.com/realm/realm-core/blob/master/src/realm/table_view.hpp
[2.1] query_engine
https://github.com/realm/realm-core/blob/master/doc/development/query_engine.pdf
[2.2] link_view.hpp:41
https://github.com/realm/realm-core/blob/master/src/realm/link_view.hpp#L41

Junxian Chen, Wen Chia Yang, Zihua Weng