Realm

NoSQL mobile database
Replacement for SQLite and Core Data

https://realm.io

What is Realm

- Fast, zero-copy, embedded database
- Object & model-oriented
- Full ACID transactions (Atomic, Consistent, Isolated, Durable)
- Cross-platform file format & C++ core
- Language bindings (Objective C, Swift, Android)
- Launched summer 2014

MVCC

- Multiversion concurrency control
- Each transaction has a snapshot of the database
- Writes are performed as append-only
- It means you can have an immutable view of data
- Reads don’t block reads or writes
- Writes block writes, but not reads

Zero copy

Traditional ORMs must copy

Steps to get data from disk:

- Translates requests into a set of SQL statements
- Perform the query and read all of the data from the rows that matched that query
- Bring all that into memory (memory allocation)
- Deserialize the format that it was on the disk to format that you can represent in memory
- Then convert that into the language level type
- Return the results to the initial requester

Zero copy

Realm skips the copy

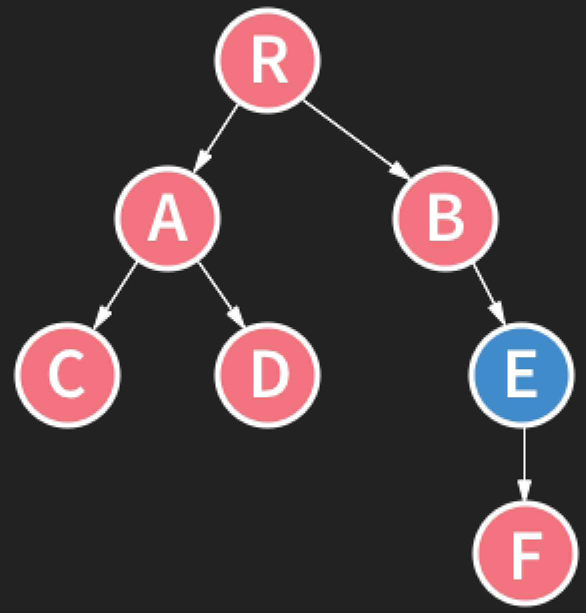
Whole file is memory-mmapped & same format on disk as in-memory (without any deserialization)

- Calculate offset of data to read
- Read from mapped file
- Return raw value from property access

True lazy loading

- In most databases, if you are reading single property, you will load the entire row
- In Realm, you will be load only single property

Relationships in Realm



```
Realm.getR().getB().getE()
```

Auto-Updating Objects & Queries

- Realm objects and Realm query results are live
- Results never have to be re-fetched
- Modifying objects that affect the query will be reflected in the results immediately.

Code example

Swift

```
let puppies = realm.objects(Dog).filter("age < 2")
puppies.count // => 0 because no dogs have been added to the Realm yet
let myDog = Dog()
myDog.name = "Rex“
myDog.age = 1
try! realm.write {
realm.add(myDog)
}
puppies.count // => 1 updated in real-time
// Access the Dog in a separate query
let puppy = realm.objects(Dog).filter("age == 1").first
try! realm.write {
puppy.age = 3
}
// Original Dog object is auto-updated
myDog.age // => 3
puppies.count // => 0
```

Realm features

- Powerful queries (chaining, sorting, aggregation methods, …)
- JSON
- Notifications (listeners)
- Migrations
- Encryption

Current limitations

- RealmObjects can’t be shared between multiple threads
- Realm files can be acceded by a single process at a time
  (Multi-process support is coming soon)
- Sorting and querying on String