GRDB

Erste Erfahrungen

Friedrich Ruynat, 9.1.2025

GRDB = SQLite für Swift

SQLite?

- Lokale SQL-Datenbank
- C-Library – läuft im Prozess
- Am meisten verbreitete Datenbank überhaupt
- Umfassende Regression-Testsuite

GRDB: Grundidee

- Alternative zu Core Data (und Swift Data)
- Value-Type basiert und reines Swift
- Wenig Magic
- Concurrency-safe
- Abstrahiert SQLite sehr umfassend
- Direkter SQLite-Zugriff trotzdem möglich
- Lange dabei (seit 2015) und aktive Entwicklung

Demo

Assoziationen

- Im Datenbank-Schema über *belongs_to* definiert ↩
- Werden nicht automatisch gefetched ↩
- Gebündelte Fetches über Join-Typen ↩

Cached Statements

- Query werden zu SQLite strings übersetzt
- Muss von SQLite wiederum geparst werden, kostet Zeit
- CachedStatements möglich

Value Observation

- Beispiel:
  - Note: Primary Key mit UUID
  - ValueObservation.tracking { Note.fetch(uuid, db) }
- Problem: Observer triggert nun bei **jeder** geänderten Zeile
- Grund:
  - Value Observation trackt Tabelle, Zeile, Spalte
  - Zeilen nur über rowID performant identifiziert
  - Primary Key muss Int64 sein

GRDB – Erste Eindrücke

- Extrem gut dokumentiert
- Entwickler sehr aktiv
- Unterstützt sehr viele SQLite-Features
- Wenig Magic (außer Protocol-Extension-Zoo)
- Value-based: Records Flexibler einsetzbar als Classes
- Wrapping oft sinnvoll: DB-Details oft sichtbar (z.B. Optional-IDs)
- Kein CloudKit-Sync (Swift Data), keine sortieren Assoziationen (Core Data)
- Swift UI: Boilerplate nötig oder GRDBQuery

SQLite noch zeitgemäß?

- SQLite: Super robust, performant, supported
- Aber:
  - Dynamische Queries selten nötig
  - Unnötiges Konvertieren zwischen SQLite und Swift
  - Einige triviale Use-Cases schwer zu lösen (z.B. Ordered Associations)
  - Keine Deduplizierung von Strings
  - Volltextsuche möglich, aber ausbaufähig (Chinesisch…)