**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN TỬ VIỄN THÔNG**

**BÁO CÁO**

**LẬP TRÌNH ĐA NỀN TẢNG**

**CHỦ ĐỀ:**

**SQLITE VÀ DRIFT - QUẢN LÝ CƠ SỞ DỮ LIỆU QUAN HỆ**

**Sinh viên thực hiện:**

**01. Bùi Trịnh Thế Viên Lớp: 22KTMT2**  **MSSV:**  106220276

**02. Kim Gia Bảo**  **Lớp: 22KTMT2**  **MSSV:**  106220245

**Người hướng dẫn:**

**TS. Nguyễn Duy Nhật Viễn**

**Đà Nẵng, 2025.**

**BÁO CÁO**

**LẬP TRÌNH ĐA NỀN TẢNG**

**CHỦ ĐỀ:**

**SQLITE VÀ DRIFT - QUẢN LÝ CƠ SỞ DỮ LIỆU QUAN HỆ**

BẢNG PHÂN CÔNG CÔNG VIỆC TRONG NHÓM

|  |  |  |  |
| --- | --- | --- | --- |
| STT | HỌ VÀ TÊN | NHIỆM VỤ | KHỐI LƯỢNG |
| 01 | BÙI TRỊNH THẾ VIÊN | Tìm hiểu SQLite, tạo ứng dụng quản lí sản phẩm bằng SQLite, test Performance | 50% |
| 02 | KIM GIA BẢO | Tìm hiểu Drift, tạo ứng dụng quản lí sản phẩm bằng Drift, test Performance | 50% |

 Link code github: <https://github.com/thevien257/Flutter-SQLite>

**Mục lục**

[I. Giới thiệu 4](#_Toc212377718)

[1.1. Mục tiêu đề tài 4](#_Toc212377719)

[1.2. Phạm vi nghiên cứu 4](#_Toc212377720)

[1.3. Phương pháp thực hiện 4](#_Toc212377721)

[II. CƠ SỞ LÝ THUYẾT 4](#_Toc212377722)

[2.1. Giới thiệu về SQLite 4](#_Toc212377723)

[2.1.1. Đặc điểm và ưu điểm của SQLite 4](#_Toc212377724)

[2.1.2. Các thao tác cơ bản trong SQLite (CRUD, JOIN, Aggregate) 5](#_Toc212377725)

[2.2. Giới thiệu về Drift 5](#_Toc212377726)

[2.2.1. Drift là gì 5](#_Toc212377727)

[2.2.2. Cấu trúc và thành phần chính của Drift 6](#_Toc212377728)

[2.2.3. Ưu điểm và hạn chế so với SQLite 6](#_Toc212377729)

[III. Ứng dụng thử nghiệm 6](#_Toc212377730)

[3.1. Thiết kế ứng dụng (App quản lý sản phẩm) 6](#_Toc212377731)

[3.2. Các hàm benchmark và phương pháp đo hiệu năng 13](#_Toc212377732)

[IV. Kết quả thực nghiệm 13](#_Toc212377733)

[4.1. Kết quả đo thời gian thực thi 13](#_Toc212377734)

[4.2. So sánh chi tiết hiệu năng SQLite vs Drift 14](#_Toc212377735)

[V. Phân tích và đánh giá 14](#_Toc212377736)

[5.1. Đánh giá ưu và nhược điểm của mỗi giải pháp 14](#_Toc212377737)

[5.2. Phân tích hiệu năng trung bình 15](#_Toc212377738)

[5.3. Nhận xét tổng quát 15](#_Toc212377739)

[5.4. Khuyến nghị sử dụng 15](#_Toc212377740)

[VI. Kết luận 16](#_Toc212377741)

[6.1. Tổng kết kết quả đạt được 16](#_Toc212377742)

# **I. Giới thiệu**

## **1.1. Mục tiêu đề tài**

Đề tài hướng tới việc nghiên cứu, thiết kế và đánh giá hiệu năng giữa hai công nghệ cơ sở dữ liệu quan hệ trong Flutter: SQLite và Drift.  
Cụ thể, nhóm tiến hành:

* Tìm hiểu lý thuyết về SQLite và Drift.
* Xây dựng ứng dụng quản lý sản phẩm sử dụng cả hai công nghệ.
* Đo đạc, phân tích và so sánh thời gian thực thi (performance) giữa hai giải pháp qua các phép thử CRUD, JOIN, và Aggregate.
* Đưa ra đánh giá, khuyến nghị sử dụng trong từng trường hợp ứng dụng thực tế.

## **1.2. Phạm vi nghiên cứu**

Phạm vi của đề tài tập trung vào:

* Ngôn ngữ & nền tảng: Flutter (Dart).
* Cơ sở dữ liệu: SQLite (thư viện sqflite) và Drift (ORM dựa trên SQLite).
* Ứng dụng minh họa: App quản lý sản phẩm gồm bảng Products và Categories.
* Chỉ tiêu đánh giá: Tốc độ xử lý dữ liệu (thời gian thực thi tính bằng mili giây).

Không đi sâu vào các khía cạnh như bảo mật dữ liệu, đồng bộ dữ liệu mạng (Cloud Sync) hoặc thiết kế giao diện UI.

## **1.3. Phương pháp thực hiện**

* Nghiên cứu lý thuyết: Tìm hiểu cơ chế hoạt động của SQLite và Drift thông qua tài liệu chính thức, ví dụ trên pub.dev và API Flutter.
* Thực nghiệm: Xây dựng ứng dụng mẫu trong Flutter, cài đặt hai hệ quản trị (SQLite và Drift), sau đó thực hiện các phép thử hiệu năng trên cùng dữ liệu (10.000 bản ghi).
* Đo lường: Sử dụng lớp Stopwatch của Dart để đo thời gian thực thi các thao tác (insert, query, join, search, aggregate).
* Phân tích – đánh giá: Tổng hợp kết quả, tính hiệu năng trung bình, biểu diễn bằng bảng và biểu đồ so sánh.

# **II. CƠ SỞ LÝ THUYẾT**

## **2.1. Giới thiệu về SQLite**

### **2.1.1. Đặc điểm và ưu điểm của SQLite**

SQLite là hệ quản trị cơ sở dữ liệu quan hệ nhẹ (lightweight RDBMS), được nhúng trực tiếp vào ứng dụng mà không cần máy chủ riêng biệt.  
Các đặc điểm chính:

* Tự chứa (Self-contained): Toàn bộ cơ sở dữ liệu nằm trong một file duy nhất .db.
* Nhúng (Embedded): Không cần cài đặt server, chỉ cần mở file để truy cập.
* Độ ổn định cao: Được sử dụng rộng rãi trong trình duyệt, hệ điều hành, và ứng dụng di động.
* Hiệu năng tốt: Thích hợp cho ứng dụng nhỏ hoặc trung bình.
* Lưu trữ cục bộ (Local storage): Dữ liệu được lưu trực tiếp trên thiết bị người dùng.

### **2.1.2. Các thao tác cơ bản trong SQLite (CRUD, JOIN, Aggregate)**

|  |  |  |
| --- | --- | --- |
| **Thao tác** | **Mục đích** | **Ví dụ / Mô tả** |
| **Create (INSERT)** | Thêm bản ghi mới | db.insert('products', values) |
| **Read (SELECT)** | Truy vấn dữ liệu | db.query('products') |
| **Update (UPDATE)** | Cập nhật bản ghi | db.update('products', values, where: 'id = ?', whereArgs: [id]) |
| **Delete (DELETE)** | Xóa bản ghi | db.delete('products', where: 'id = ?', whereArgs: [id]) |
| **JOIN** | Kết hợp dữ liệu giữa bảng Products và Categories qua categoryId | SELECT p.name, c.name FROM products p JOIN categories c ON p.category\_id = c.id |
| **Aggregate** | Thực hiện phép tính tổng hợp (COUNT, SUM, AVG) | SELECT COUNT(\*) FROM products |

=> SQLite cung cấp độ linh hoạt cao trong viết truy vấn SQL thủ công, tuy nhiên khó bảo trì khi ứng dụng mở rộng.

## **2.2. Giới thiệu về Drift**

### **2.2.1. Drift là gì**

Drift (trước đây là Moor) là một thư viện ORM (Object Relational Mapping) được xây dựng trên SQLite, cho phép lập trình viên thao tác với cơ sở dữ liệu bằng cú pháp Dart, không cần viết SQL thủ công.  
Drift hỗ trợ các tính năng:

* Reactive: Giao diện tự động cập nhật khi dữ liệu thay đổi (watch() streams).
* Migration: Hỗ trợ cập nhật cấu trúc cơ sở dữ liệu tự động.
* Type-safe: Kiểm tra kiểu dữ liệu ngay trong quá trình biên dịch.
* Cross-platform: Chạy được trên Android, iOS, Web và Desktop.

Cho phép tương tác cơ sở dữ liệu một cách:

* Type-safe (an toàn kiểu dữ liệu)
* Dễ bảo trì
* Reactive (UI cập nhật tự động khi dữ liệu thay đổi)

### **2.2.2. Cấu trúc và thành phần chính của Drift**

|  |  |
| --- | --- |
| **Thành phần** | **Vai trò** |
| **Table** | Định nghĩa cấu trúc bảng, cột, kiểu dữ liệu, khóa chính/ngoại. |
| **Data Class** | Đại diện cho một bản ghi dữ liệu. |
| **Companion Class** | Dùng để thêm hoặc cập nhật bản ghi. |
| **DAO (Data Access Object)** | Chứa các hàm CRUD, truy vấn tùy chỉnh. |
| **Database Class** | Tập hợp tất cả bảng, DAO, và cấu hình migration. |

* Drift tự động sinh mã trong file \*.g.dart, giúp code rõ ràng, an toàn và dễ bảo trì.

Các CRUID cơ bản của Drift:

|  |  |  |
| --- | --- | --- |
| **Thao tác** | **Hàm chính** | **Mục đích** |
| Create | insert() | Thêm dữ liệu mới |
| Read | select().get() / watch() | Đọc dữ liệu hoặc stream realtime |
| Update | update().write() | Sửa dữ liệu có điều kiện |
| Delete | delete().go() | Xóa dữ liệu |

### **2.2.3. Ưu điểm và hạn chế so với SQLite**

**Ưu điểm:**

* Viết code bằng Dart thuần, không cần SQL thủ công.
* Kiểm tra cú pháp và lỗi biên dịch sớm.
* Hỗ trợ reactive UI, migration, batch và transaction.
* Dễ mở rộng và tái sử dụng trong dự án lớn.

**Hạn chế:**

* Tốc độ thấp hơn trong một số truy vấn phức tạp (JOIN, Aggregate).
* Kích thước ứng dụng tăng do sinh mã và dependency ORM.

# **III. Ứng dụng thử nghiệm**

## **3.1. Thiết kế ứng dụng (App quản lý sản phẩm)**

Ứng dụng thử nghiệm được xây dựng bằng Flutter, mô phỏng hệ thống quản lý sản phẩm với hai bảng:

Categories: Chứa danh mục sản phẩm (id, name, description).

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 1. Danh mục*

Products: Chứa thông tin sản phẩm (id, name, price, categoryId, stock, createdAt).

**A white background with pink and black lines

AI-generated content may be incorrect.**

*Hình 2.Danh sách sản phẩm*

**Quan hệ:** Một danh mục có thể chứa nhiều sản phẩm. Khi xóa danh mục, các sản phẩm liên quan cũng bị xóa theo (KeyAction.cascade).

Các chức năng chính:

* Thêm, sửa, xóa, xem danh sách sản phẩm và danh mục.
* Kết hợp bảng (JOIN) để hiển thị sản phẩm cùng tên danh mục.
* Thực hiện các phép thử hiệu năng trên 10.000 bản ghi dữ liệu.

A screenshot of a phone

AI-generated content may be incorrect.

*Hình 3. Màn hình chính*

A screenshot of a phone

AI-generated content may be incorrect.

*Hình 4. Thêm sản phẩm*

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 5. Thêm danh mục*

A screenshot of a phone

AI-generated content may be incorrect.

*Hình 6. Chỉnh sửa sản phẩm*

A screenshot of a computer screen

AI-generated content may be incorrect.

*Hình 7. Xoá sản phẩm*

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 8. Kết quả category hiển thị trên file database*

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 9. Kết quả product hiển thị trên file database*

A screenshot of a phone

AI-generated content may be incorrect.

*Hình 10. Xoá toàn bộ dữ liệu*

A screenshot of a phone

AI-generated content may be incorrect.

*Hình 11. Hiển thị toàn bộ dữ liệu trên tab SQLite*

A screenshot of a phone

AI-generated content may be incorrect.

*Hình 12. Hiển thị toàn bộ dữ liệu trên tab Drift*

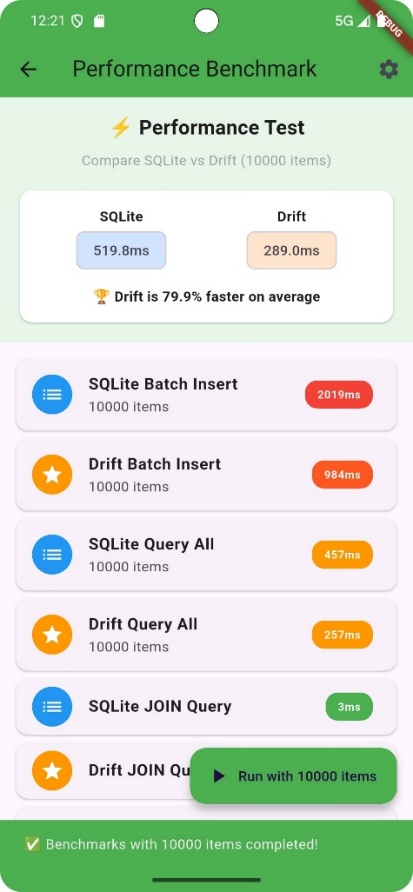
## **3.2. Các hàm benchmark và phương pháp đo hiệu năng**

Các hàm kiểm thử hiệu năng bao gồm:

* benchmarkSQLiteInsert() / benchmarkDriftInsert(): đo tốc độ thêm dữ liệu hàng loạt.
* benchmarkSQLiteQuery() / benchmarkDriftQuery(): đo tốc độ đọc toàn bộ dữ liệu.
* benchmarkSQLiteJoin() / benchmarkDriftJoin(): đo tốc độ kết hợp bảng.
* benchmarkSQLiteSearch() / benchmarkDriftSearch(): tìm kiếm theo tên sản phẩm.
* benchmarkSQLiteAggregate() / benchmarkDriftAggregate(): tính toán tổng hợp.

**Công cụ đo:**  
Sử dụng Stopwatch của Dart để ghi lại thời gian thực thi từng phép thử (tính bằng mili giây).

# **IV. Kết quả thực nghiệm**



*Hình 13. Kết quả chạy benchmark*

## **4.1. Kết quả đo thời gian thực thi**

|  |  |  |  |
| --- | --- | --- | --- |
| **Phép đo** | **SQLite** | **Drift** | **Nhận xét** |
| Batch Insert | 2019 ms | 984 ms | Drift nhanh hơn ~2 lần |
| Query All | 457 ms | 257 ms | Drift tối ưu nhờ caching |
| JOIN Query | 3 ms | 64 ms | SQLite tối ưu hơn ở JOIN |
| Search (LIKE) | 116 ms | 84 ms | Drift xử lý tốt nhờ async |
| Aggregate | 4 ms | 56 ms | SQLite nhanh hơn trong phép tính SQL |

## **4.2. So sánh chi tiết hiệu năng SQLite vs Drift**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **SQLite** | **Drift** |
| CRUD cơ bản | Nhanh | Nhanh hơn nhờ ORM tối ưu |
| JOIN | Tốt hơn | Chậm hơn do lớp trừu tượng |
| Aggregate | Rất nhanh | Chậm hơn đáng kể |
| Reactive UI | Không hỗ trợ | Hỗ trợ sẵn (stream) |
| Dễ bảo trì | Khó (viết SQL) | Dễ (Dart code) |

Hiệu năng trung bình:

|  |  |  |
| --- | --- | --- |
|  | **Trung bình** | **So sánh** |
| **SQLite** | 519.8 ms | — |
| **Drift** | 289.0 ms | Nhanh hơn 79.9% |

# **V. Phân tích và đánh giá**

## **5.1. Đánh giá ưu và nhược điểm của mỗi giải pháp**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **SQLite** | **Drift** |
| **Mức trừu tượng** | Cấp thấp – viết SQL thủ công | Cấp cao – dùng cú pháp Dart |
| **Định nghĩa bảng** | Lệnh SQL | Class Dart kế thừa Table |
| **Truy vấn dữ liệu** | SELECT \* FROM ... | select(table).get() |
| **Kiểm tra cú pháp** | Không có | Kiểm tra khi biên dịch |
| **Reactive (watch)** | Không hỗ trợ | Có – UI tự cập nhật |
| **Migration** | Viết SQL thủ công | Có MigrationStrategy |
| **Transaction / Batch** | Thủ công | Có sẵn API hỗ trợ |
| **Độ linh hoạt** | Cao | Trung bình |
| **Tốc độ JOIN/Aggregate** | Tốt hơn | Chậm hơn |
| **Quan hệ bảng** | Tự quản lý | Hỗ trợ references() |
| **Sinh mã tự động** | Không | Có file \*.g.dart |
| **Đa nền tảng** | Android/iOS | Android, iOS, Web, Desktop |
| **Phù hợp cho** | Dự án nhỏ, SQL thuần | Dự án vừa/lớn |

## **5.2. Phân tích hiệu năng trung bình**

Tổng hợp kết quả cho thấy:

* Drift nhanh hơn trung bình ~79.9% trên hầu hết các tác vụ CRUD thông thường.
* Drift tận dụng ORM + Stream API, giúp tăng tốc độ và hỗ trợ reactive UI trong Flutter.
* SQLite thuần vẫn vượt trội ở các truy vấn phức tạp như JOIN và Aggregate, nhờ tối ưu hóa SQL engine trực tiếp.
* Khi ứng dụng cần tính năng reactive, an toàn kiểu dữ liệu, và tích hợp tốt với Flutter, Drift là lựa chọn tốt hơn.
* Nếu yêu cầu hiệu năng thuần túy, hoặc các phép xử lý dữ liệu phức tạp, SQLite gốc có lợi thế.

## **5.3. Nhận xét tổng quát**

Drift phù hợp cho các dự án có giao diện động (reactive UI) và cần quản lý dữ liệu phức tạp nhưng dễ bảo trì.

SQLite thích hợp với ứng dụng nhỏ, đơn giản, nơi hiệu năng thuần túy là ưu tiên hàng đầu.

Cả hai đều ổn định, đáng tin cậy và dễ tích hợp trong Flutter.

## **5.4. Khuyến nghị sử dụng**

|  |  |
| --- | --- |
| **Mục đích ứng dụng** | **Nên chọn** |
| Ứng dụng nhỏ, học tập, SQL thuần | **SQLite** |
| Ứng dụng trung bình – lớn, cần reactive UI | **Drift** |
| Ứng dụng đa nền tảng (Web, Desktop) | **Drift** |
| Cần hiệu năng cực cao với truy vấn phức tạp | **SQLite** |

# **VI. Kết luận**

## **6.1. Tổng kết kết quả đạt được**

Hoàn thành việc tìm hiểu và triển khai hai công nghệ SQLite và Drift trong Flutter.

Xây dựng thành công ứng dụng quản lý sản phẩm thử nghiệm.

Thực hiện benchmark với 10.000 bản ghi và so sánh chi tiết hiệu năng.

Kết quả: Drift nhanh hơn trung bình 79.9% trong CRUD, trong khi SQLite mạnh hơn ở JOIN và Aggregate.