Table of Contents

[**1.** **So sánh CoreData và SQLite** 2](#_Toc129644935)

[**1.1** **Các thành phần trong CoreData** 2](#_Toc129644936)

[**1.2** **CoreData Stack** 3](#_Toc129644937)

[**1.2.1** **NSManagedObjectModel** 3](#_Toc129644938)

[**1.2.2** **NSPersistentStoreCoordinator** 4](#_Toc129644939)

[**1.2.3** **Manage Object Context** 4](#_Toc129644940)

[**1.3** **Cách sử dụng CoreData** 6](#_Toc129644941)

[**2.** **NSPrecdicate** 9](#_Toc129644942)

# **So sánh CoreData và SQLite**

SQLite is a relational database. Một cơ sở dữ liệu quan hệ (relational database) là một hệ thống **quản lý cơ sở dữ liệu (database management system)** dựa trên mô hình quan hệ (relational model) của E.F. Codd. Trong mô hình này, dữ liệu được tổ chức thành các bảng (tables) có mối quan hệ với nhau thông qua các khóa ngoại (foreign keys).

**The most important difference between Core Data and SQLite is that SQLite is a database while Core Data is not. Core Data can use SQLite as its persistent store, but the framework itself is not a database**. Core Data is a framework for managing an object graph. An object graph is nothing more than a collection of interconnected(liên kết) objects. The framework excels(vượt trội) at managing complex object graphs.

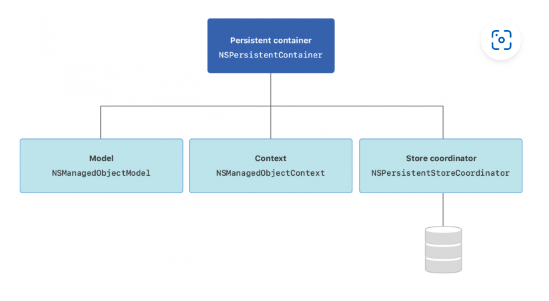
Tóm lại: SQLite là 1 relation database, ta có thể sử dụng Script để trích xuất dữ liệu của Database. Còn CoreData là ORM(Object Relation Model). Nó giúp đơn giản hóa việc truy xuất và lưu trữ dữ liệu trong ứng dụng bằng cách sử dụng đối tượng(class) và quan hệ thay vì sử dụng câu lệnh SQL trực tiếp như trong SQLite. CoreData là một framework của Apple được sử dụng để quản lý dữ liệu trong ứng dụng iOS và macOS.

Vì vậy, SQLite và CoreData đều là những công nghệ được sử dụng để quản lý dữ liệu trong các ứng dụng, tuy nhiên chúng có những đặc điểm và tính năng khác nhau. SQLite là một cơ sở dữ liệu quan hệ truyền thống, trong khi CoreData là một framework quản lý dữ liệu của Apple, có tính tích hợp cao và hỗ trợ nhiều định dạng lưu trữ khác nhau.

## **Các thành phần trong CoreData**

* **Database được CoreData** sử dụng ở đây là **sqlite**.
* **Entity** sẽ tương ứng với các **table** trong Database
* **Attribute** tương ứng với các **field** trong table
* **Relationship** tương ứng với các quan hệ giữa các table trong Database (chỗ này mình chưa chắc)
* **Managed Object** tương ứng với các **record** trong các table
* …

## **CoreData Stack**



Think of the Core Data Stack as a team of main players that run the game. There are 3 main components we must be aware of: **persistent store coordinator, managed** **object model**, **managed** **object context**. Each component has a specific task and coordinates with others to support Core Data.

### **NSManagedObjectModel**

Đây chính là mô hình (model) của Cơ sở dữ liệu của bạn trong project. Nhấn mạnh, nó không phải là cơ sở dữ liệu của bạn. Nhưng ta có thể:

* So sánh mô hình đối tượng được quản lý với giản đồ của cơ sở dữ liệu
* Có chứa thông tin về các mô hình hoặc thực thể của biểu đồ đối tượng
* Thuộc tính của chúng và cách chúng liên quan đến nhau.

Đối tượng NSManagedObjectModel biết về mô hình dữ liệu bằng cách tải một hoặc nhiều tệp mô hình dữ liệu ( \*.xcdatamodel file) trong quá trình khởi tạo của nó**. Ngoài ra, thể hiện trực quan của nó trong project chính là file \*.xcdatamodel. Giúp cho ta:**

* Thiết kế cơ sở dữ liệu
* Tạo các table (Entity)
* Tạo các thuộc tính
* Ràng buộc các quan hệ giữa các table.
* Cấu hình các table
* Mô hình trực quan giữa các table

Ta có thể ánh xạ sang Database như sau:

* Entity ==> Table
* Attribute ==> Field
* Relationship ==> Key/Relationship

Chốt lại: *Để cho****Persistent Store Coordinator****biết được mô hình ở dưới Database như thế nào? Thì****Managed Object Model****đảm đương nhiệm vụ đó. Bên cạnh còn giúp người lập trình có thể****thiết kế****một cách trực quan nhất, mặc dù chã biết tí nào về Database.*

### **NSPersistentStoreCoordinator**

Như tên gọi của nó, đối tượng NSPersistentStoreCoordinator tiếp tục lưu trữ dữ liệu vào đĩa và đảm bảo (các) lưu trữ liên tục và mô hình dữ liệu tương thích. Nó làm cầu nối trung gian tới phần Database lưu trữ (sqlite). Tạo ra một lớp dữ liệu ở giữa để tương tác với phần còn lại của project.

### **Manage Object Context**

Đối tượng NSManagedObjectContext quản lý các đối tượng của Entity trên Data Model. Là các thể hiện của lớp NSManagedObject.

*Dễ hiểu thì việc bạn tương tác với Database, nếu phải nhớ các lệnh truy vấn SQL thì khá là đau đầu.****Managed Object Context****giúp bạn biến mọi dữ liệu trong Database thành các class và object.*

Có 2 thành phần quan trọng nhất:

* **NSManagedObjectContext** là class quản lý toàn bộ các đối tượng liên quan tới dữ liệu của Database. Phụ trách thêm các công việc như:
  + Fetch
  + Insert
  + Update
  + Delete
  + Save
  + …
* **NSManagedObject** đây là class để quản lý tới các **record**. Biến dữ liệu trong các record thành các đối tượng của nó. Bạn có thể tạo các class mới kế thừa lại để dễ thao tác trong code.

Hình dung sẽ như thế này:

* **Table** trong Database sẽ được biểu diển là các **Entity** trong Data Model. Và tiếp tục được biển diễn bằng các **NSManagedObject** trong code.
* Các thuộc tính, quan hệ cũng được biểu diễn thành các **property** trong sub-class của NSManagedObject.
* Với sub-class của NSManagedObject thì
  + **Class name** sẽ là tên của Entity hay cũng là tên của table trong Database
* **NSManagedObjectContext** sẽ quản lý tất cả các lớp NSManagedObject.

Tới đây cũng khá nhiều lý thuyết được trình bày. Và tiếp tục sang phần tìm hiểu về cách hoạt động, thì chúng ta sẽ sử dụng code để trình bày và giải thích. Bài này thì sẽ tạo một iOS Project ngay từ đầu.

**Chốt cơ bản lại:**

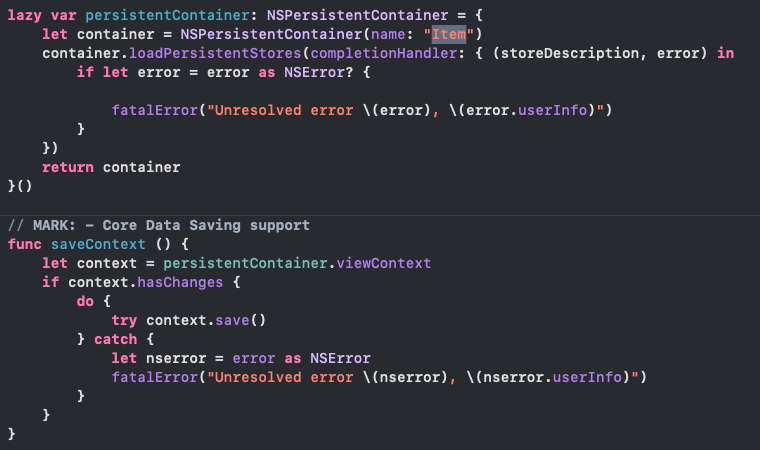
NSManagedObjectModel : Là thằng tạo ra các relation trong database quy định các trường, kiểu dữ liệu, tên biến trong database.

NSPersistentStoreCoordinator: Là thằng có nhiệm vụ ghi dữ liệu vào ổ cứng, là cầu nối trung gian giữa database với app.

NSManagedObjectContext: Là đối tượng mà người lập trình sử dụng để tương tác, lưu giá trị,…

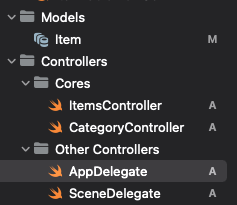
## **Cách sử dụng CoreData**

* Đầu tiên ta khởi tạo CoreData trong AppDelegate



Ta chú ý: let container = NSPersistentContainer(name: “Item”).

Tên ta truyền vào đây phải trùng tên với DataBase:



* Trong Item ta sẽ tạo các Entity và Attributes và Relation liên quan:



* Sau đó ta sẽ tạo biến context trong các Controllers mà ta sử dụng để truy cập dữ liệu

No description available.

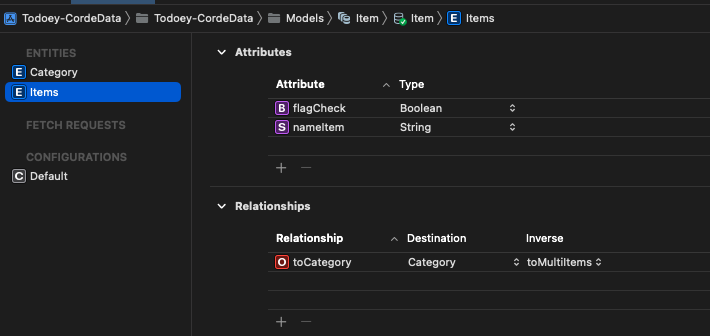
* Tất cả việc ta thao tác với dữ liệu đều qua context, ta sẽ fetch dữ liệu thông qua ví dụ sau. Ở đây ta muốn lấy tất cả dữ liệu Entity có tên **Items** ta sử dụng :



* Lúc này khi **itemArray** sẽ tham chiều tới các dữ liệu mà **context** fetch về. Lúc này khi muốn sửa hoặc xóa dữ liệu ta chỉ cần tsửa xóa dữ liệu thông qua itemArray.
* Khi ta muốn tạo mới dữ liệu mà được context tham chiếu tới ta dùng:



* Ta thấy các thuộc tính **Attributes** ở Entity đều là các thuộc tính bình thường(Int, String, Bool,…) Vậy để trỏ tới 1 Class khác ta sử dụng RelationShips.



Như này biến với các Instance của Items sẽ có 3 thuộc tính: **nameItem, flagCheck và toCategory**



* Với **Category** khi **relationship là tomay**: Khi ta muốn truy cập vào mỗi phần từ Items mà Category Relationship tới.



* Xóa dữ liệu ta cũng xem cách trên.

# **NSPrecdicate**

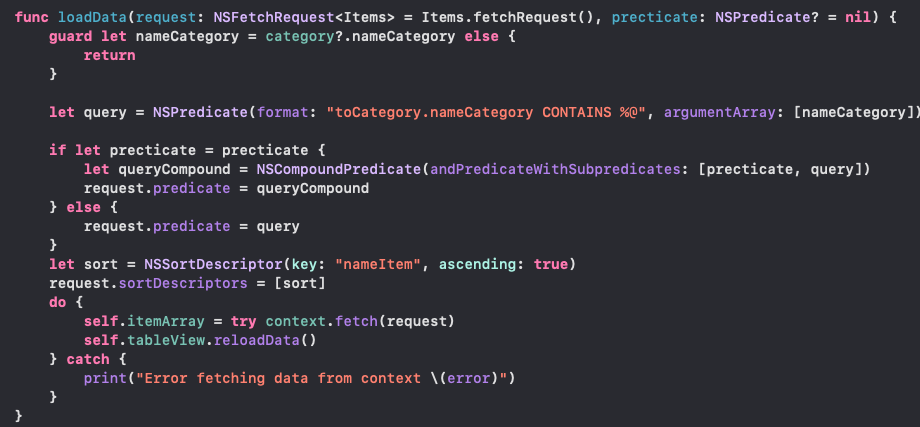
NSPredicate là một đối tượng của Objective-C và Swift trong framework Foundation của Apple, được sử dụng để định nghĩa các biểu thức để lọc và truy vấn dữ liệu.

Format String cho NSPrecticate:

* **%@:** This represents a placeholder for an object**. It can be used for any type of object, including strings, numbers, and dates.**
* **%K:** This represents a placeholder for a key path. Key paths are used to access a property of an object using dot notation. For example, if you have an object with a property called name, you can use %K to refer to that property in a predicate format string: NSPredicate**(format: "%K == %@", "name", "Alice").**
* **%d, %i, %u, %x, %X, %o**: These represent placeholders for integers. %d and %i are interchangeable and represent a signed integer, while %u represents an unsigned integer. %x and %X represent hexadecimal integers (lowercase and uppercase respectively), and %o represents an octal integer.
* **%f, %g, %e:** These represent placeholders for floating-point numbers. %f is used for fixed-point notation, %g is used for scientific notation, and %e is used for exponential notation.
* **%c:** This represents a placeholder for a single character.
* **%s:** This represents a placeholder for a C-style string (a null-terminated array of characters).



Ví dụ:



Ví dụ này sẽ request dữ liệu của tất cả các dữ liệu item(item lúc này có quan hệ relation với Category tên là **toCategory**). Vì **toCategory** có kiểu là **Category** nên nó sẽ có thuộc tính **nameCategory**(thuộc tính này là ta add Attributes trong model).

Tất cả các biến item mà có property thỏa mãn sẽ được lấy.

* **NSCompoundPrecdicate** sử dụng để có thể sử dụng nhiều predicate.
* **NSSortDescriptor:** được sử dụng để sắp xếp dữ liệu lấy về theo key truyền vào.
* Sau khi khởi tạo xong ta sẽ gán các precdicate và compound đó cho request thông qua
* Request.predicate = predicate và request.sortDescriptors = [sort]
* Sau đó ta gọi context.fetch(request) để lấy dữ liệu.