
SIGMA
Software


SIGNL4
Software
UNIVERSITY


SIGMA
Software
UNIVERSITY

ЛЕКЦІЯ 19

“Бази даних на прикладі CoreData”

19

SIGMA SOFTWARE UNIVERSITY

Бази даних на прикладі CoreData

CoreData - що це?

Ключовий концепт - Faulting

Ключовий концепт - DB Normal Forms of relationships

Основи

Зв’язки

NSFetchResultController

SIGMA SOFTWARE UNIVERSITY

Бази даних на прикладі CoreData

CoreData - що це?

Ключовий концепт - Faulting

Ключовий концепт - DB Normal Forms of relationships

Основи

Зв’язки

NSFetchResultController

SIGMA SOFTWARE UNIVERSITY

CoreData Що це?

«The Core Data framework provides generalized and automated solutions to common tasks associated with object life-cycle and object graph management, including persistence.»

Основними особливостями є

- управління даними в DB
- управління зв’язками
- автовалідація властивостей
- faulting
- можливість автоматичної інтергації (NSFetchedResultController)
- полегшення копмозиції складних запитів
- merge polices
- легке оновлення та міграція DB

SIGMA SOFTWARE UNIVERSITY

CoreData Чому?

Разом з CoreData:

- приблизно на 50% - 70% менше коду ніж при інших підходах
- знання SQL queries не обов’язкові
- велика оптимізація DB
- швидка інтеграція (графічний інтерфейс)
- прекрасна безпека зберігання, управління пам’яттю та помилками
- безкоштовно
- хороша документація

SIGMA SOFTWARE UNIVERSITY

Бази даних на прикладі CoreData

CoreData - що це?

Ключовий концепт - Faulting

Ключовий концепт - DB Normal Forms of relationships

Основи

Зв’язки

NSFetchResultController

SIGMA SOFTWARE UNIVERSITY

Бази даних на прикладі CoreData

CoreData - що це?

Ключовий концепт - Faulting

Ключовий концепт - DB Normal Forms of relationships

Основи

Зв’язки

NSFetchResultController

SIGMA SOFTWARE UNIVERSITY

Ключовий концепт - Faulting

Faulting is a mechanism Core Data employs to reduce your application's memory usage. A related feature called uniquing ensures that, in a given managed object context, you never have more than one managed object to represent a given record.

Faulting reduces the amount of memory your application consumes. A fault is a placeholder object that represents a managed object that has not yet been fully realized, or a collection object that represents a relationship

Ключовий концепт - Faulting

Faulting дозволяє:

- зменшити к-ть використаної памяті для додатку
- автоматично використовувати на всі об’єкти
- підвищити швидкість доступу в DB

Бази даних на прикладі CoreData

CoreData - що це?

Ключовий концепт - Faulting

Ключовий концепт - DB Normal Forms of relationships

Основи

Зв’язки

NSFetchResultController

SIGMA SOFTWARE UNIVERSITY

Бази даних на прикладі CoreData

CoreData - що це?

Ключовий концепт - Faulting

Ключовий концепт - DB Normal Forms of relationships

Основи

Зв’язки

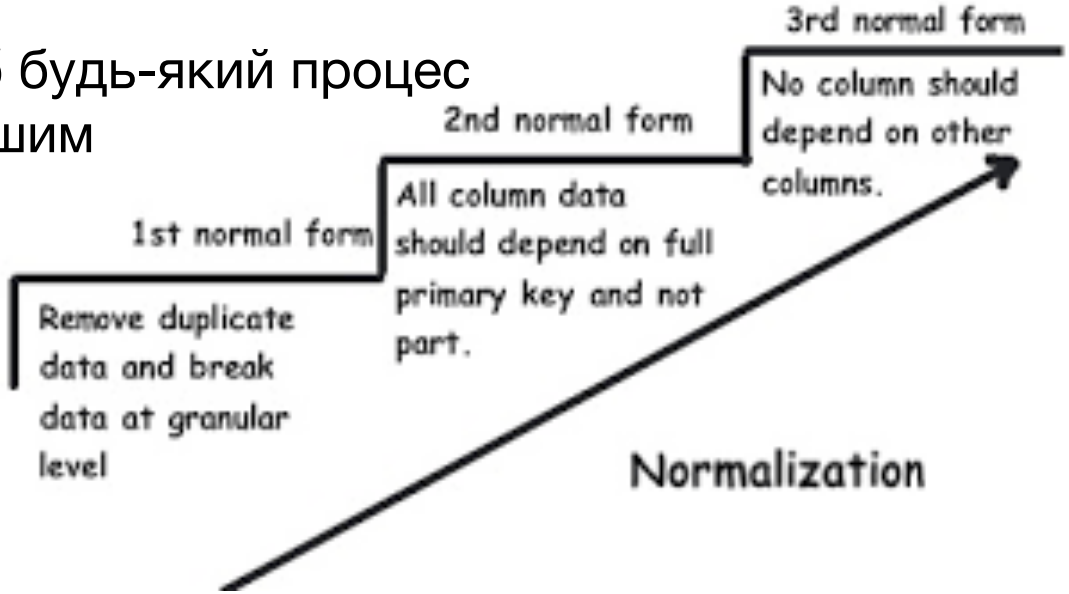
NSFetchResultController

SIGMA SOFTWARE UNIVERSITY

Ключовий концепт - DB Normal Forms of relationship

Нормалізація включає в себе:

- переопис бази даних (таблиць) з меншою к-ті зайвих властивостей
- визначення foreign keys в одних таблицях з посиланням на primary keys в інших
- ізолювання (логічне) даних так, щоб будь-який процес був більш швидким та оптимізованішим
- швидку навігацію і оновлення бази даних через використання вказаних ключів таблиці



Ключовий концепт - DB Normal Forms of relationship

1 Форма

- Кожна таблиця повинна мати основний ключ: мінімальний набір колонок, які ідентифікують запис
- Уникнення повторень груп (категорії даних, що можуть зустрічатись різну кількість разів в різних записах) правильно визначаючи неключові атрибути
- Атомарність: кожен атрибут повинен мати лише одне значення, а не множину значень

UNF	1NF
✓Course Code	✓Course Code
Course Name	Course Name
Tutor Id	Tutor Id
Tutor Name	Tutor Name
✓Student No)	
Student Name)	
Date of Birth)	✓Course Code
Gender)	✓Student No
Last Att Date)	Student Name
	Date of Birth
	Gender
	Last Att Date

wiki

Ключовий концепт - DB Normal Forms of relationship

2 Форма

- Схема бази даних повинна відповідати вимогам першої нормальної форми
- Дані, що повторно з'являються в декількох рядках, виносяться в окремі таблиці

UNF	1NF	2NF
✓Course Code	✓Course Code	✓Course Code
Course Name	Course Name	Course Name
Tutor Id	Tutor Id	Tutor Id
Tutor Name	Tutor Name	Tutor Name
✓Student No)		
Student Name)		
Date of Birth)	✓Course Code	✓Course Code
Gender)	✓Student No	✓Student No
Last Att Date)	Student Name	Last Att Date
	Date of Birth	
	Gender	
	Last Att Date	
		✓Student No
		Student Name
		Date of Birth
		Gender

wiki

Ключовий концепт - DB Normal Forms of relationship

3 Форма

- Схема бази даних повинна відповідати всім вимогам другої нормальної форми

UNF	1NF	2NF	3NF	Entity Name
✓Course Code	✓Course Code	✓Course Code	✓Course Code	
Course Name	Course Name	Course Name	Course Name	Course
Tutor Id	Tutor Id	Tutor Id	*Tutor Id	
Tutor Name	Tutor Name	Tutor Name		
✓Student No)				
Student Name)				
Date of Birth)	✓Course Code	✓Course Code	✓Course Code	
Gender)	✓Student No	✓Student No	✓Student No	Classlist
Last Att Date)	Student Name	Last Att Date	Last Att Date	
	Date of Birth			
	Gender			
	Last Att Date			
		✓Student No	✓Student No	
		Student Name	Student Name	Student
		Date of Birth	Date of Birth	
		Gender	Gender	
			✓Tutor Id	Tutor
			Tutor Name	

wiki

Ключовий концепт - DB Normal Forms of relationship

3 Форма

- Будь-яке поле, що залежить від основного ключа та від будь-якого іншого поля, має виноситись в окрему таблицю

UNF	1NF	2NF	3NF	Entity Name
✓Course Code	✓Course Code	✓Course Code	✓Course Code	
Course Name	Course Name	Course Name	Course Name	Course
Tutor Id	Tutor Id	Tutor Id	*Tutor Id	
Tutor Name	Tutor Name	Tutor Name		
✓Student No)				
Student Name)				
Date of Birth)	✓Course Code	✓Course Code	✓Course Code	
Gender)	✓Student No	✓Student No	✓Student No	Classlist
Last Att Date)	Student Name	Last Att Date	Last Att Date	
	Date of Birth			
	Gender			
	Last Att Date			
		✓Student No	✓Student No	
		Student Name	Student Name	Student
		Date of Birth	Date of Birth	
		Gender	Gender	
			✓Tutor Id	Tutor
			Tutor Name	

wiki

Бази даних на прикладі CoreData

CoreData - що це?

Ключовий концепт - Faulting

Ключовий концепт - DB Normal Forms of relationships

Основи

Зв’язки

NSFetchResultController

SIGMA SOFTWARE UNIVERSITY

Бази даних на прикладі CoreData

CoreData - що це?

Ключовий концепт - Faulting

Ключовий концепт - DB Normal Forms of relationships

Основи

Зв’язки

NSFetchResultController

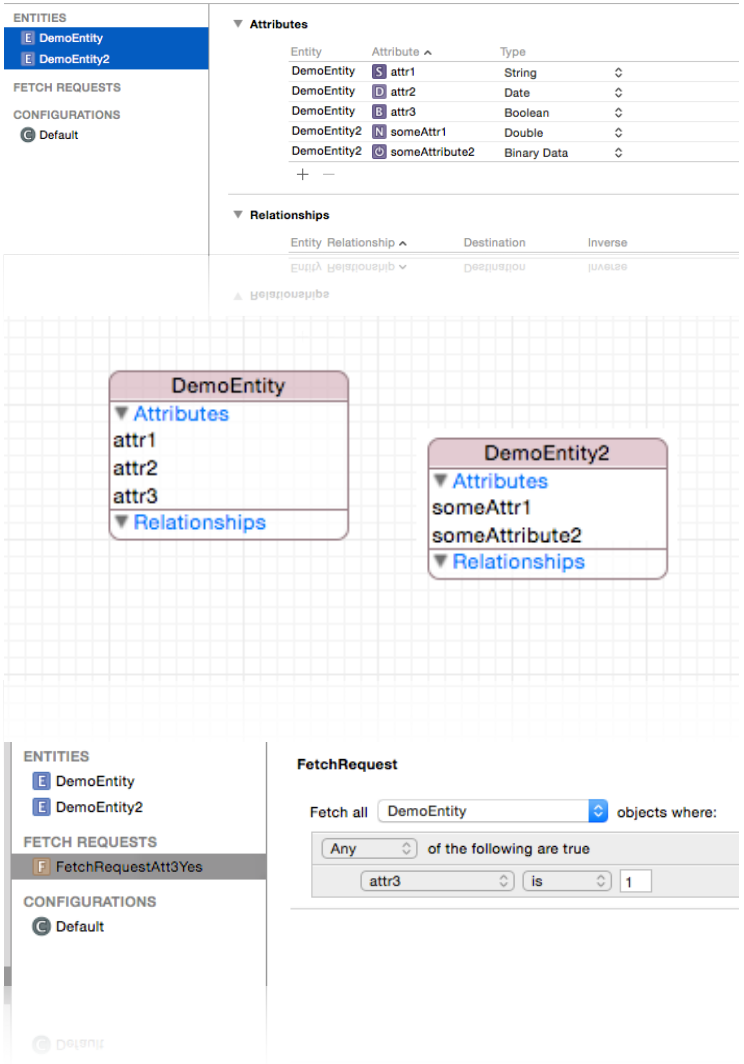
SIGMA SOFTWARE UNIVERSITY

Основи

xCode

xCode - потужний інструмент для створення та управління моделлю для DB.
За допомогою xCode ви можете:

- створити модель у графічному редакторі
- додати необхідні відносини
- визначити fetchRequests
- Намалювати модель для кращого розуміння
- керувати міграцією БД (легким або жорстким)
- та ін

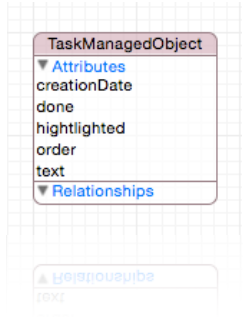


SIGMA SOFTWARE UNIVERSITY

Основи

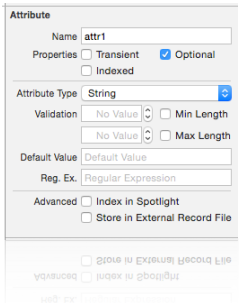
Entities (NSManagedObject)

- Об’єкти які використовуються в CoreData в dataModel.



Attributes

- Properties of each entity - деталі об’єкту

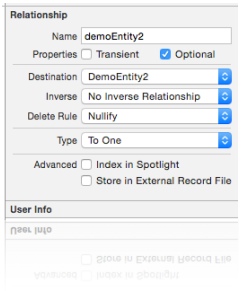


Attribute Options

- Параметри які описують природу поведінки атримубів. Наприклад "Attribute type" або "Default value"

Delete Rules

- Правила які визначають поведінку видалення об’єктів



SIGMA SOFTWARE UNIVERSITY

Основи

Managed object model

Managed object Context

Persistent store coordinator

Основи - Fetch Request та його команда

FetchRequest

OBJ-C

```
NSManagedObjectContext *moc = [self managedObjectContext];
NSFetchRequest *request = [[NSFetchRequest alloc] init];
[request setEntity:[NSEntityDescription entityForName:@"Recipe"
 inManagedObjectContext:moc]];

NSArray *results = [moc executeFetchRequest:request error:&error];
if (error) {
NSLog(@"Error: %@\n%@", [error localizedDescription], [error userInfo]);
return;
}
```

SIGMA SOFTWARE UNIVERSITY

Основи - Fetch Request та його команда

FetchRequest

```
let request = NSFetchRequest<NSDictionary>(entityName: Constants.EntityName.git)
request.resultType = .dictionaryResultType
request.returnsDistinctResults = true
request.propertiesToFetch = [Constants.PropertyName.Git.objectId]

do {
    let object = try persistentContainer.viewContext.fetch(request)
    return object.flatMap({$0.allValues}).compactMap({$0 as? Int})
} catch {
     //помилка
}
```

SIGMA SOFTWARE UNIVERSITY

Основи - Fetch Request та його команда

NSSortDescriptor

```
NSFetchRequest *fetchRequest = [NSFetchRequest fetchRequestWithEntityName:@"Recipe"];
NSSortDescriptor *sort = [[NSSortDescriptor alloc] initWithKey:@"name" ascending:YES];
[fetchRequest setSortDescriptors:[NSArray arrayWithObject:sort]];
```

OBJ-C

Основи - Fetch Request та його команда

NSSortDescriptor

```
let request = NSFetchRequest<NSDictionary>(entityName: Constants.EntityName.git)
var descriptor = NSSortDescriptor(key: "name", ascending: true)
request.sortDescriptors = [descriptor]
```

SIGMA SOFTWARE UNIVERSITY

Основи - Fetch Request та його команда

NSPredicate

```
NSManagedObjectContext *moc = [self managedObjectContext];
NSFetchRequest *request = [[NSFetchRequest alloc] init];
[request setEntity:[NSEntityDescription entityForName:@"Recipe" inManagedObjectContext:moc]];
NSPredicate *predicate = [NSPredicate predicateWithFormat:@"YOUR FILETER"];
[request setPredicate:predicate];
```

OBJ-C

Основи - Fetch Request та його команда

NSPredicate

```
let fetchRequest = NSFetchRequest<QueryEntity>()
fetchRequest.entity = NSEntityDescription.entity(forEntityName: Constants.EntityName.query, in: persistentContainer.viewContext)
let predicate = NSPredicate(format: "\(Constants.PropertyName.Query.qValue) == %@", stringQuery)
fetchRequest.predicate = predicate
```

Бази даних на прикладі CoreData

CoreData - що це?

Ключовий концепт - Faulting

Ключовий концепт - DB Normal Forms of relationships

Основи

Зв’язки

NSFetchResultController

SIGMA SOFTWARE UNIVERSITY

Бази даних на прикладі CoreData

CoreData - що це?

Ключовий концепт - Faulting

Ключовий концепт - DB Normal Forms of relationships

Основи

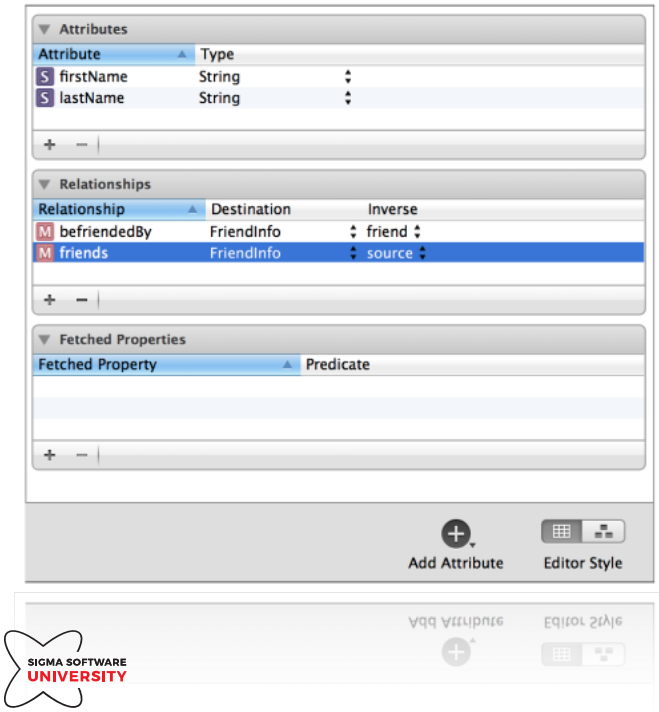
Зв’язки

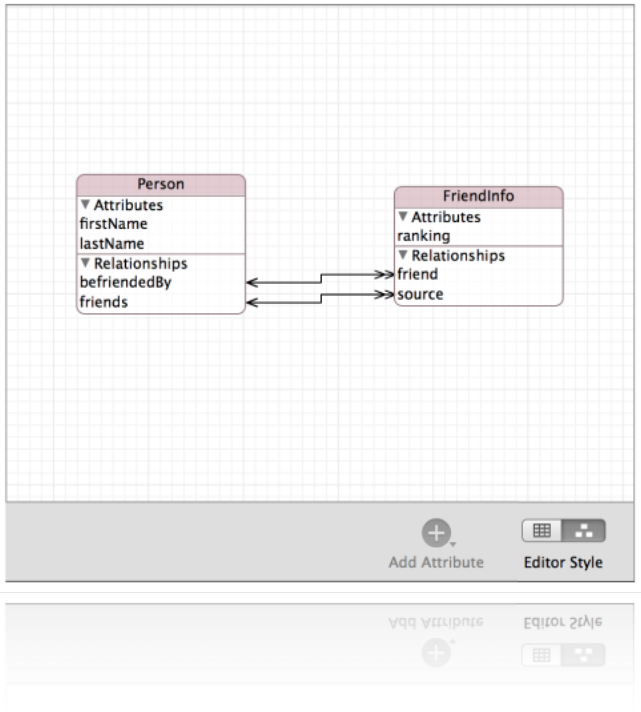
NSFetchResultController

SIGMA SOFTWARE UNIVERSITY

Зв’язки

CoreData підтримує взаємозв'язки типу to-one та to-many, а також fetched properties.





one-to-one

one-to-many

many-to-one

many-to-many

SIGMA SOFTWARE UNIVERSITY

Бази даних на прикладі CoreData

CoreData - що це?

Ключовий концепт - Faulting

Ключовий концепт - DB Normal Forms of relationships

Основи

Зв’язки

NSFetchResultController

SIGMA SOFTWARE UNIVERSITY

Бази даних на прикладі CoreData

CoreData - що це?

Ключовий концепт - Faulting

Ключовий концепт - DB Normal Forms of relationships

Основи

Зв’язки

NSFetchResultController

SIGMA SOFTWARE UNIVERSITY

FetchedResultController

«You use a fetched results controller to efficiently manage the results returned from a Core Data fetch request to provide data for a UITableView object.»

OBJ-C

```
NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] initWithEntityName:@"DemoEntity"];
[fetchRequest setSortDescriptors:@[[NSSortDescriptor sortDescriptorWithKey:@"attr1" ascending:YES]]];

self.fetchedResultsController = [[NSFetchedResultsController alloc] initWithFetchRequest:fetchRequest
managedObjectContext:self.context sectionNameKeyPath:nil cacheName:nil];
[self.fetchedResultsController setDelegate:self];
NSError *fetchError;
[self.fetchedResultsController performFetch:&fetchError];
if (fetchError) {
    NSLog(@"Cant fetch - %@", fetchError.localizedDescription);
} else {
    NSLog(@"Fetched - %i objects", (int)self.fetchedResultsController.fetchedObjects.count);
}
```

SIGMA SOFTWARE UNIVERSITY

FetchedResultController

«You use a fetched results controller to efficiently manage the results returned from a Core Data fetch request to provide data for a UITableView object.»

```
        let request = NSFetchRequest(entityName: "Person")
        let departmentSort = NSSortDescriptor(key: "department.name", ascending: true)
        let lastNameSort = NSSortDescriptor(key: "lastName", ascending: true)
        request.sortDescriptors = [departmentSort, lastNameSort]
        let moc = dataController.managedObjectContext
        fetchedResultsController = NSFetchedResultsController(fetchRequest: request, managedObjectContext: moc,
sectionNameKeyPath: nil, cacheName: nil)
        fetchedResultsController.delegate = self
        do {
            try fetchedResultsController.performFetch()
        } catch {
            fatalError("Failed to initialize FetchedResultsController: \(error)")
        }
```


SIGMA
Software
UNIVERSITY