What's New in Core Data on Mac OS X

Session 315

Ben Trumbull
Core Data Engineering Manager

Roadmap

New

- Concurrency
- Auto Save
- Ordered relationships
- iCloud
- Incremental stores
- Developer Tools

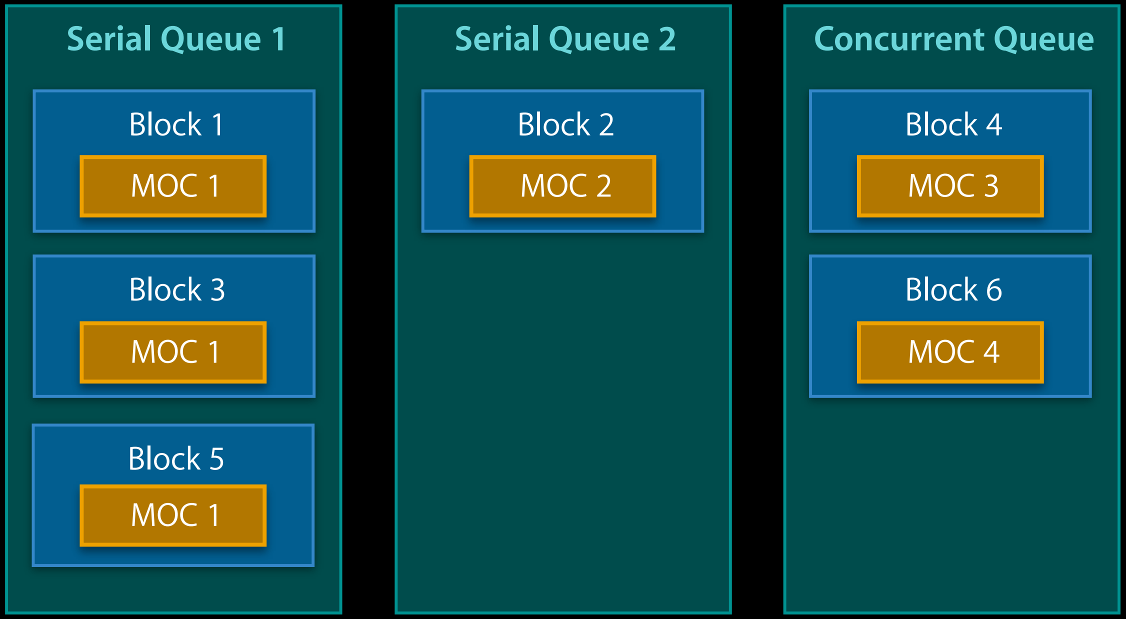
Concurrency

NSManagedObjectContext

- New concurrency types
- Block-based methods
- Nested contexts

Where We Were

Thread confinement



Thread Confinement

- Separate contexts for each thread
- Managed objects owned by their context
- ObjectIDs are safe, immutable value objects

Thread Confinement

- Easy to understand
- Safe and efficient for transactions
- But…
    - Coordination left as exercise to reader
    - Tracking which context goes with which thread
    - Passing changes between threads

What’s a Framework to Do?

Formal Concurrency Policies

- New NSManagedObjectContext initializer
- `-initWithConcurrencyType:`

```
NSConfinementConcurrencyType
NSPrivateQueueConcurrencyType
NSMainQueueConcurrencyType
```

NSConfinementConcurrencyType

- Same behavior and restrictions as 10.4 – 10.6
- Thread confinement
- MOCs only messaged by thread or queue that created them
- Default behavior

NSPrivateQueueConcurrencyType

- New to 10.7 and iOS 5
- Can only be called on its own private queue
- Use `-performBlock:`
- Within block use MOC APIs normally

NSMainQueueConcurrencyType

- Similar to private queue
- Queue is always the main queue
- UI and controllers on main thread can message directly
- Other threads must use `-performBlock:`
- Convenient for receiving results

Queue-based Concurrency

- New Context initializer

```
-initWithConcurrencyType:
NSMainQueueConcurrencyType
NSPrivateQueueConcurrencyType
```

- Block-based API

```
-performBlock:
-performBlockAndWait:
```

-performBlock:

- Asynchronous
- A “user event”
- Convenient autorelease pool
- No support for reentrancy
- Illegal to throw an exception out of your block

-performBlockAndWait:

- Synchronous
- Not an event
- No autorelease pool
- Supports reentrancy
- Illegal to throw an exception out of your block

What’s a User Event?

- Automatic as application main event loop
- Provides
  - Change coalescing
  - Delete propagation
  - Undo
  - NSNotifications
- Time in between calls to `-processPendingChanges`

Queue is Private

- Do not use `dispatch_get_current_queue`
- To use libdispatch or NSOperation APIs
  - Trampoline through your own queue
  - Capture references in your blocks

Interfacing with libdispatch

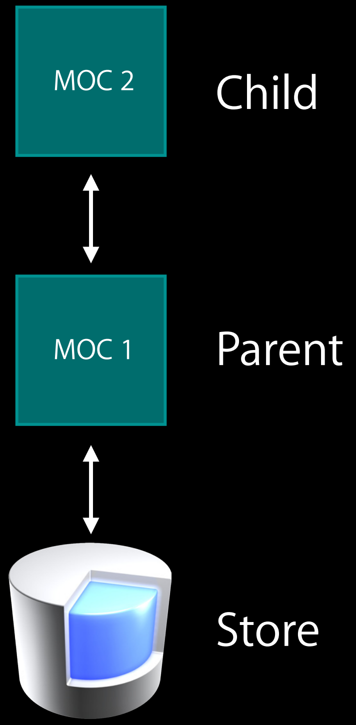
- Create a dispatch group
- Call `dispatch_group_enter`
- Worker block call `dispatch_group_leave`
- Use `dispatch_group_wait` and `dispatch_group_notify` normally

Nested NSManagedObjectContext

Nested Contexts

- Parent Context

  `setParentContext:`



Why Use Nested Contexts?

- Asynchronous saves
- Sharing unsaved changes between MOCs
  - Inheriting changes in a detail inspector
- Background fetching

Asynchronous Save

- Save child
- Asynchronously ask parent to save
- UIManagedDocument

Asynchronous Save

```
NSManagedObjectContext *child, *parent;
parent = [[NSManagedObjectContext alloc]
             initWithConcurrencyType:NSPrivateQueueConcurrencyType];
[child setParentContext:parent];
// ...
[child save:&error];
[parent performBlock:^{
    [parent save:&parentError];
}];
```

Sharing Unsaved Changes

- Shared parent context
- Push to parent
- Pull in peer child

Inheriting Changes in Detail Inspector

- Create a child context
- Save pushes changes into parent
- Fetch incorporates unsaved changes in parent
- Toss child context to cancel detail changes

Things to Remember

- Saving only pushes changes up one level
- Fetching pulls data through all levels
- `-objectWithID:` pulls fewest levels necessary
- Parent contexts must adopt a queue type

Auto Save

Document Saving

- Users had to explicitly save documents
- Needed to save regularly
- Forced to save at inopportune times

Lion Auto Save

New

- Documents automatically save in-place
- UI to avoid unintentional changes
- User initiated saves create a “version”
- Browse through previous versions

NSPersistentDocument

- Fully supports Lion Auto Save
- Browse versions
- Even untitled documents
- Incremental operations on SQLite store

Adopting Lion Auto Save

```
+ (BOOL)autosavesInPlace {
  return YES;
 }
```

New Save Operation Types

- New NSSaveOperationTypes
  - `NSAutosaveInPlaceSaveOperation`
  - `NSAutosaveElsewhereSaveOperation`
- NSPersistentDocument never supported
  - `NSAutosaveOperation`

Overriding NSDocument Write Methods

- Always call super
- Let us do the hard parts
- Handling autosave for databases is tricky

Document File Wrappers

- Recommend document file wrappers
- For new Core Data features
  - iCloud syncing
  - External binary data

Using File Wrappers

- “File Wrappers with Core Data Documents” sample code
- Overrides NSPersistentDocument read/write methods
- URL to store within the document file wrapper
- New Lion Auto Save example coming soon

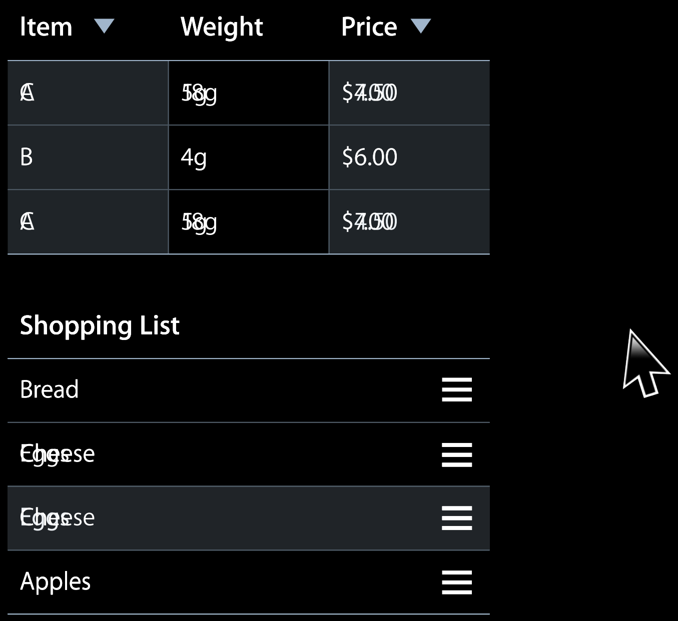
Summary

- Documents automatically save in-place
- Browse through versions
- Simple to enable
- Consider file wrappers

Ordered Relationships

Sorting vs. Ordering

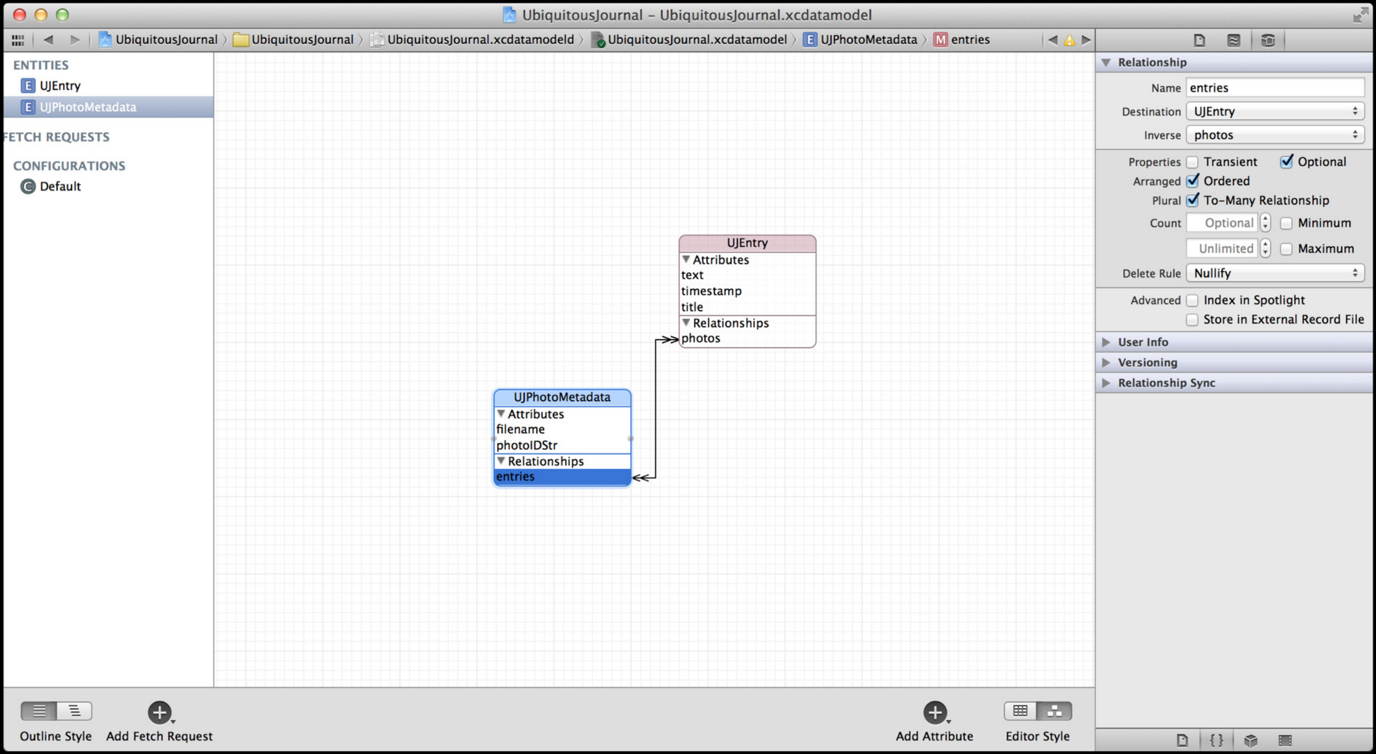
- Sorting by value
  - Derived
  - Change your view
- Arbitrary ordering
  - List
  - Flexible control
  - Not tied to any intrinsic value



Ordered Relationships

- Assign positions in to-many relationships
- NSOrderedSet
- More like an array than a set
  - Subclass of neither
- Performance impact from ordering and uniquing

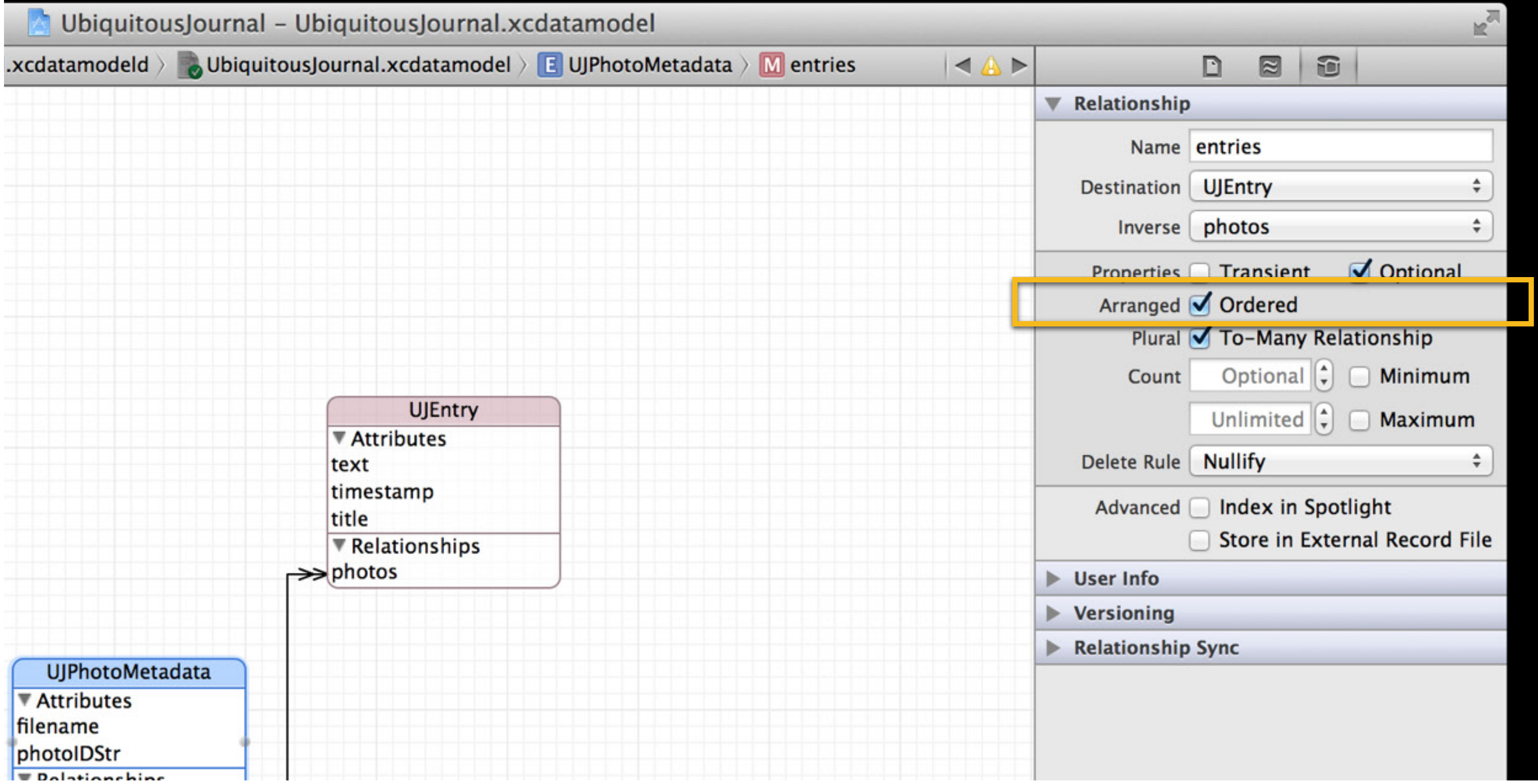
Ordered Relationships



Relationship

Name	entries
Destination	UJEntry
Inverse	photos
Properties	☐ Transient ✓ Optional
Arranged	✓ Ordered
Plural	✓ To-Many Relationship
Count	Optional ☐ Minimum
	Unlimited ☐ Maximum
Delete Rule	Nullify
Advanced	☐ Index in Spotlight
	☐ Store in External Record File

Ordered Relationships



UbiquitousJournal – UbiquitousJournal.xcdatamodel

.xcdatamodeld 〉 UbiquitousJournal.xcdatamodel 〉 UJPhotoMetadata 〉 entries

▼ Relationship

Name: entries

Destination: UJEntry

Inverse: photos

Properties: ☐ Transient ☑ Optional

Arranged: ☑ Ordered

Plural: ☑ To-Many Relationship

Count: Optional ☐ Minimum

Unlimited ☐ Maximum

Delete Rule: Nullify

Advanced: ☐ Index in Spotlight

☐ Store in External Record File

▶ User Info

▶ Versioning

▶ Relationship Sync

Working with Ordered Relationships

- Generate accessors in Xcode 4
- Or use generic mutator

  `mutableOrderedSetValueForKey:`
- Automatic KVC accessors are not available, yet

  `insertEvents:atIndexes:, removeObjectFromEvents:atIndex:`

Observing Changes

- Key Value Observing with ordered collections

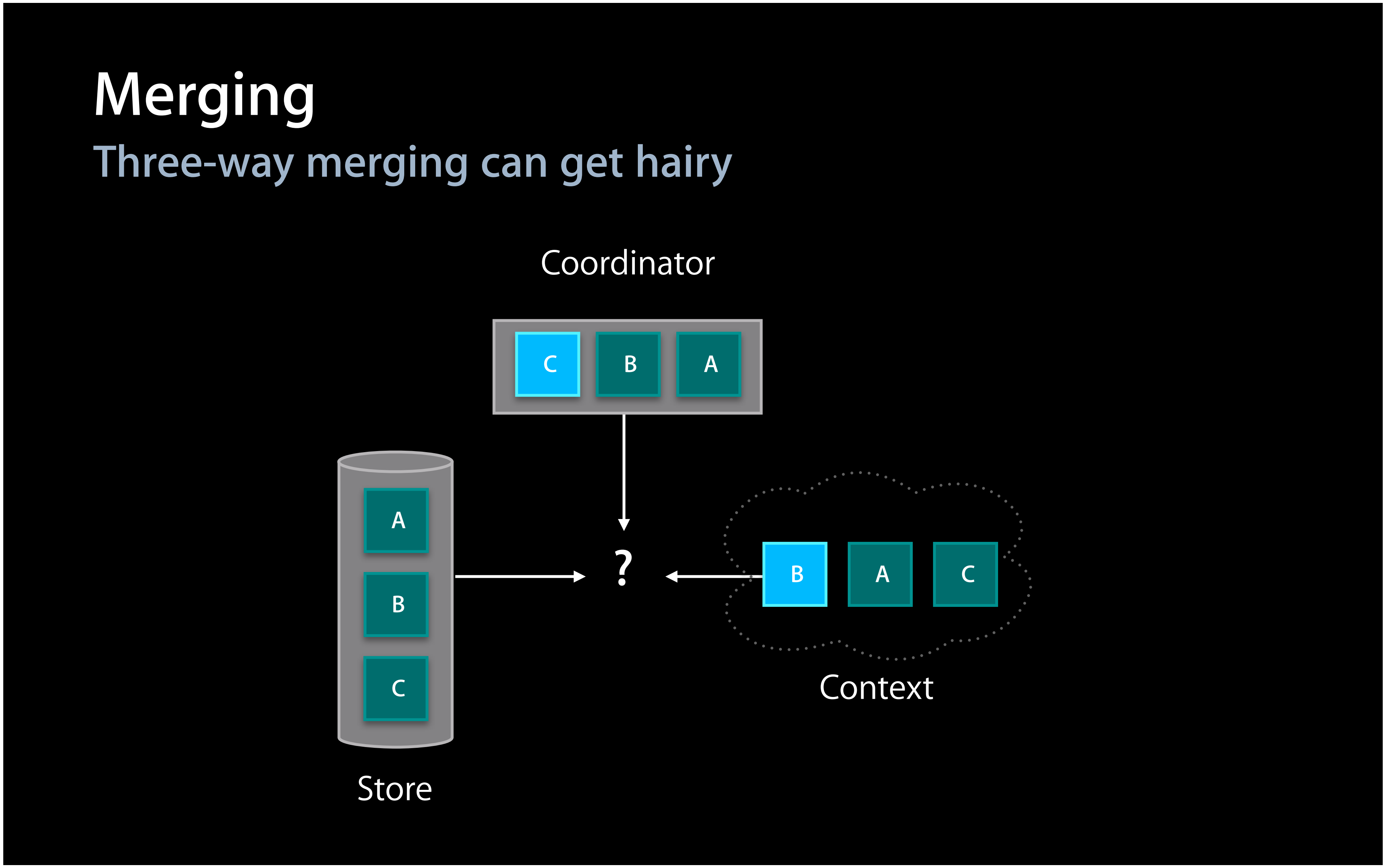
  `observeValueForKeyPath:ofObject:change:context:`

- Change kinds

  `NSKeyValueChangeInsertion`

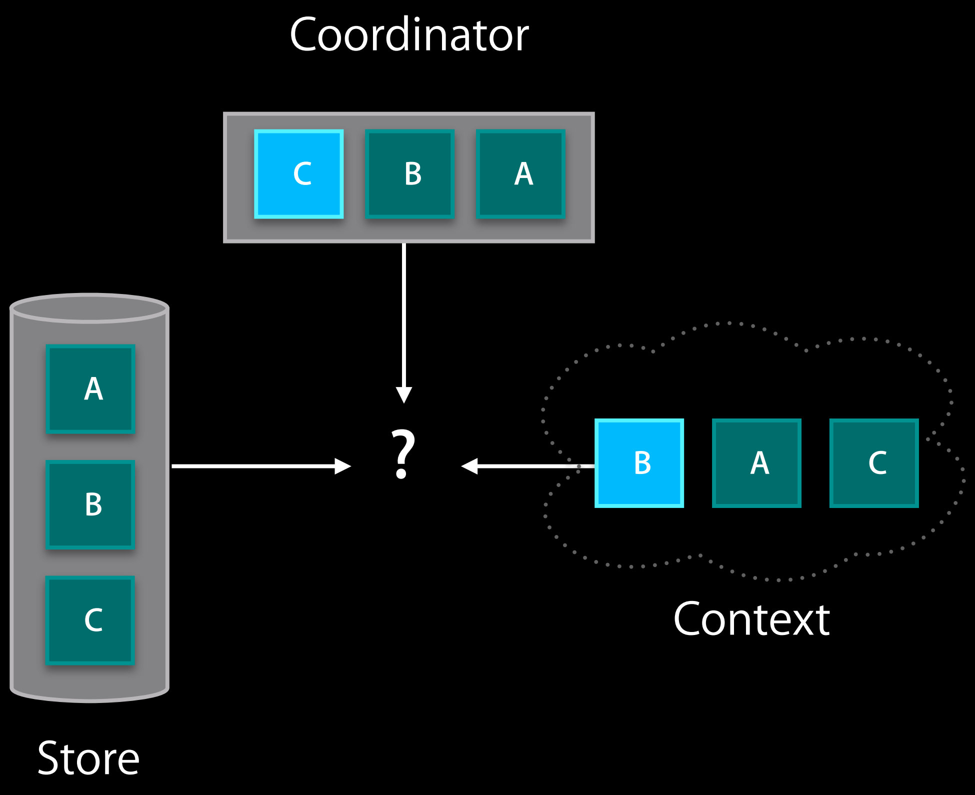
  `NSKeyValueChangeRemoval`

  `NSKeyValueChangeReplacement`



Merging

Three-way merging can get hairy



Merging

- We try to preserve relative ordering
- Performance is much slower than non-ordered
  - Merging existence
  - Merging position

Migration

- Non-ordered to ordered and back
- Lightweight migration gives arbitrary ordering
- Post-process to impose ordering

Summary

- For arbitrary ordering
- Ordered collection KVC/KVO
- Performance
- Use sorted unordered relationships where possible

iCloud

Melissa Turner
Senior Software Engineer

Core Data, iCloud, and You

- Sync data between devices and computers
- Easy integration
- Automatic conflict resolution

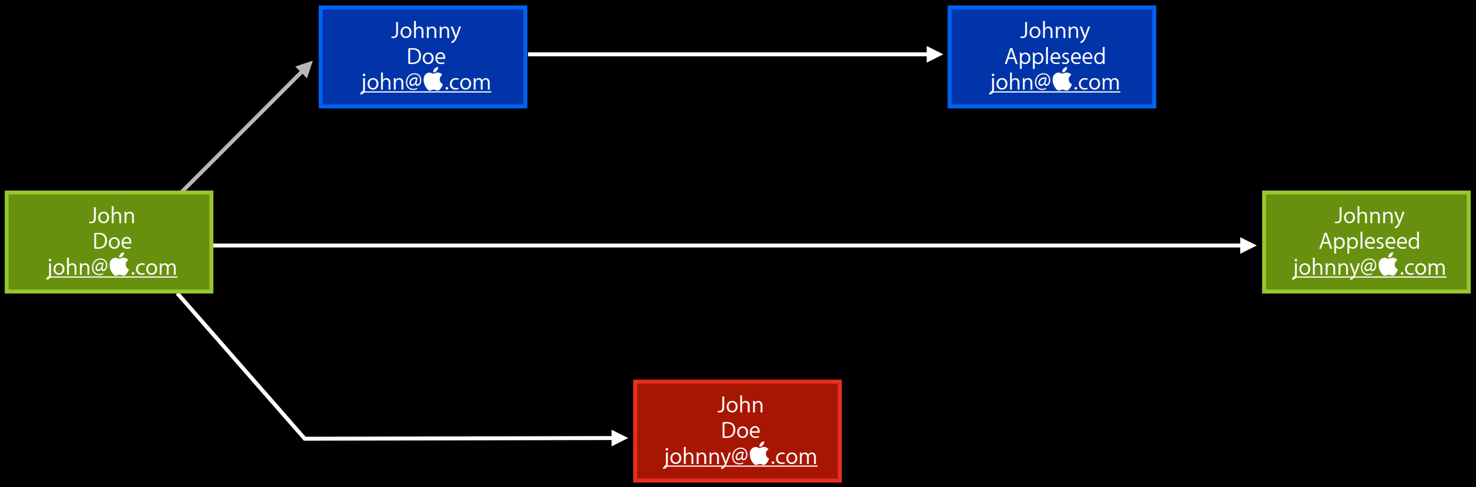
iCloud

What do you get?

- Works with existing stores
- Per record conflict resolution
- Only deltas are sync'd
- Asynchronous import
- Three-way merge

Three-Way Merge

Preserve Changes Between Systems



Less Code

Your part

- Options when adding persistent store
- Respond to import notification

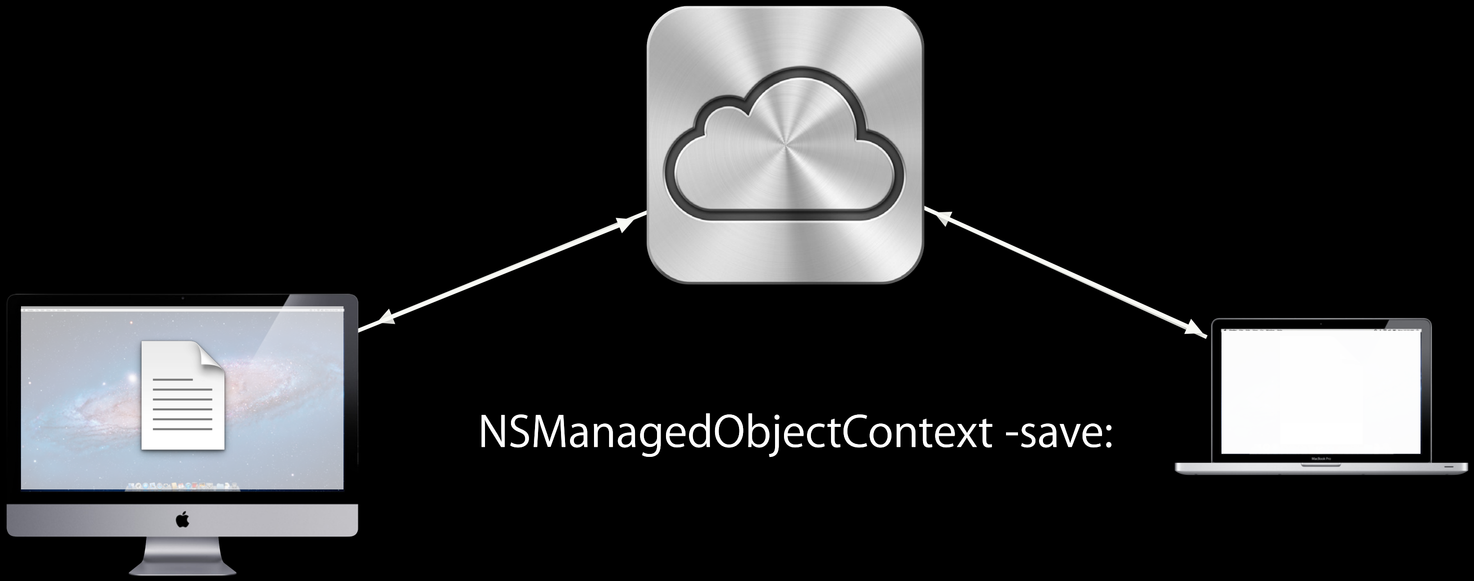
Less Code

Our part

- Handle integration
  - NSFileCoordinator
  - NSFilePresenter
  - NSMetadataQuery
- Export changes
- Import changes

iCloud Demo

What Just Happened?



NSManagedObjectContext -save:

New API

- Persistent Store Options
  - `NSPersistentStoreUbiquitousContentNameKey`
  - `NSPersistentStoreUbiquitousContentURLKey`
- Notification
  - `NSPersistentStoreDidImportUbiquitousContentChangesNotification`

NSPersistentStoreUbiquitousContentNameKey

data.sqlite

/Users/test/data.sqlite

foodtopia inc.

06-30-2011.store

NSPersistentStoreUbiquitousContentURLKey

- Optional
- Provide your own if
  - Ubiquity Container ID != Bundle ID
  - Document syncing
- Opaque Package

NSPersistentStoreUbiquitousContentURLKey

- Defaults to main bundle identifier

```
NSString *bundleID = [[NSBundle mainBundle] bundleIdentifier];
NSURL *contentURL = [[NSFileManager defaultManager]
                       URLForUbiquityContainerID:bundleID];
```

NSPersistentStoreDidImportUbiquitousContentChangesNotification

- Object
  - NSPersistentStoreCoordinator
- User Info
  - `NSInsertedObjects`
  - `NSUpdatedObjects`
  - `NSDeletedObjects`
  - Collections of NSManagedObjectIDs

Responding to an Import

- Similar to `NSManagedObjectContextDidSaveNotification`
- Refresh unchanged objects
- Merge changed objects

Syncing NSPersistentDocument

- Sync document file wrappers
- Use ubiquitous store options
- Don’t sync SQLite files
  - Include “.nosync” in the path
- Set ubiquitous URL path inside document file wrapper

Document Syncing Alternatives

- Atomic stores can sync as whole files
    - SQLite should not be
- Whole store syncing
    - Don’t need ubiquitous store options
    - Last writer wins
    - Use NSFileVersion APIs for conflicts

Tips and Tricks

Good ideas

`NSPersistentStoreDidImportUbiquitousContentChangesNotification`

- Use appropriate merge policy

  `NSMergeByPropertyStoreTrumpMergePolicy`

  `NSMergeByPropertyObjectTrumpMergePolicy`

- Anticipate bandwidth constraints
- Use .nosync

Incremental Stores

Why Do I Care?

XML-RPC

Lucene

REST

CouchDB

In Memory

JSON

SQLite

SOAP

XML

PostgreSQL

MongoDB

Binary

ThriftDB

MySQL

LDAP

Incremental Store

- Talk to your data source in its own language

```
{ variety : “Brooks” ,
  reviews : [ { rating : 4 , text :  “Favorite!”} ,
              { rating : 3 , text :  “Best early choice” },
              { rating : 5 , text :  “Season is too short” } ] }
```

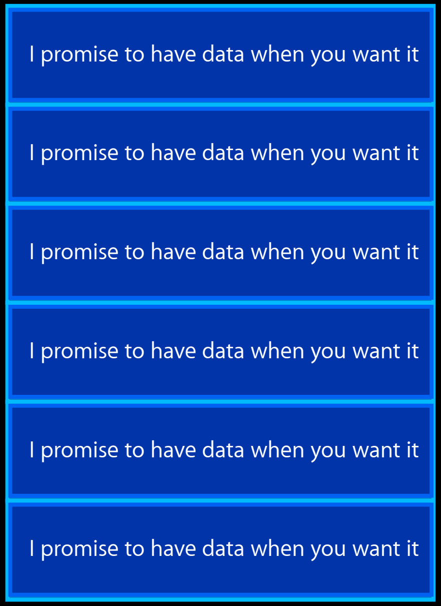
Incremental Store

- Talk to your data source in its own language
- Load only the data you need

Mark Perlson
Tom McNeil
Sumeera Razul
Lea Longo
Trisha Zarin
Greg Apodaka
Elisa Rossi
Jack Simon
Hari Seshaiah
Derrick Thornton

Incremental Store

- Talk to your data source in its own language
- Load only the data you need
- Supports faulting


I promise to have data when you want it
I promise to have data when you want it
I promise to have data when you want it
I promise to have data when you want it
I promise to have data when you want it
I promise to have data when you want it

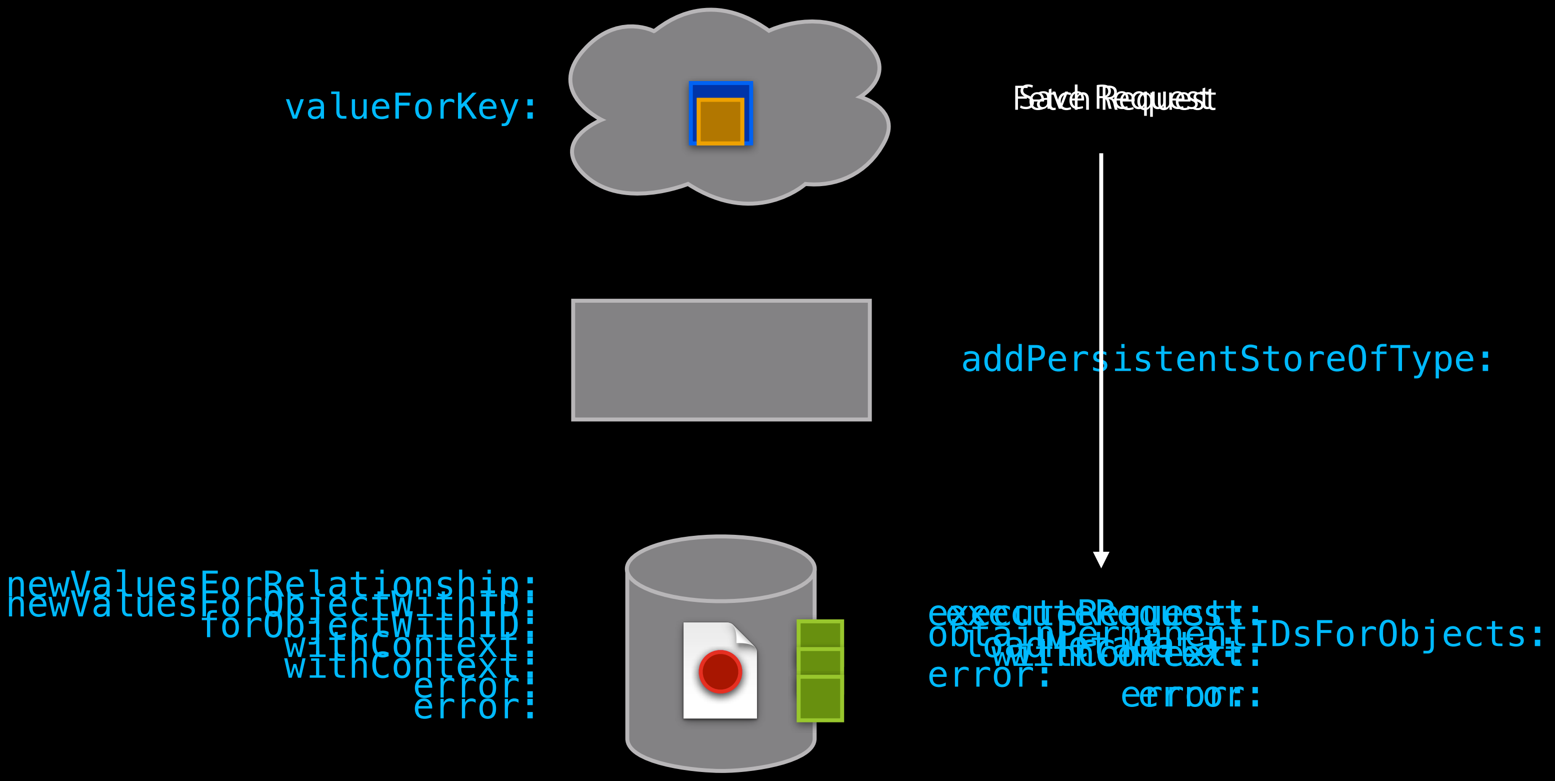
Incremental Store

- Talk to your data source in its own language
- Load only the data you need
- Supports faulting
- Flush unused data

Mark Perlson
Sumeera Razul
Lea Longo
Trisha Zarin
Jack Simon
Derrick Thornton

Control Flow

How does it work?



NSIncrementalStoreNode

Data in a format Core Data can use

```
initWithObjectID:withValues:version:
valueForPropertyDescription:
```

Talking to the Store

NSPersistentStoreRequest and Friends

- New base class
- NSSaveChangesRequest
- Reparented NSFetchRequest

Requesting Data from the Store

NSFetchRequest

- Flags that affect results
- Flags that affect performance
- Graceful degradation

Implementation Details

- Object ID mapping APIs supplied
- Get managed objects from context

`objectWithID:`

Integration Points

- SQL generator not included
  - Canned queries
- JSON provider in Foundation

General Design Tips

- Design to a specific schema
- Balance I/O and memory
  - Cache (API not provided)
- Better to talk to web services

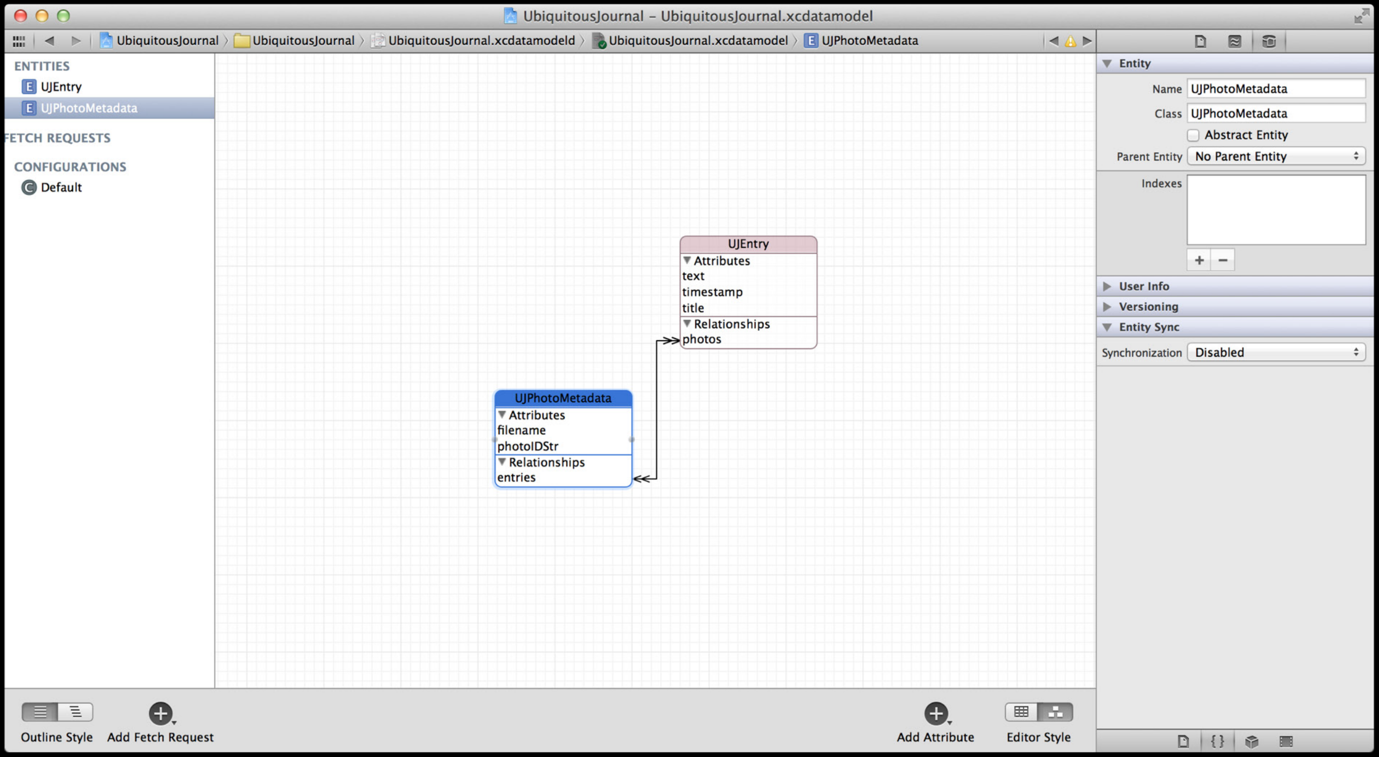
Developer Tools

Xcode 4

- New UI
- Optimized models
- Readable, diffable models
- Scalar accessors

New UI

Diagram View


UbiquitousJournal – UbiquitousJournal.xcdatamodel
UbiquitousJournal › UbiquitousJournal › UbiquitousJournal.xcdatamodeld › UbiquitousJournal.xcdatamodel › UJPhotoMetadata
ENTITIES
UJEntry
UJPhotoMetadata
FETCH REQUESTS
CONFIGURATIONS
Default
Outline Style
Add Fetch Request
Add Attribute
Editor Style
User Info
Versioning

UJEntry
- Attributes
  - text
  - timestamp
  - title
- Relationships
  - photos

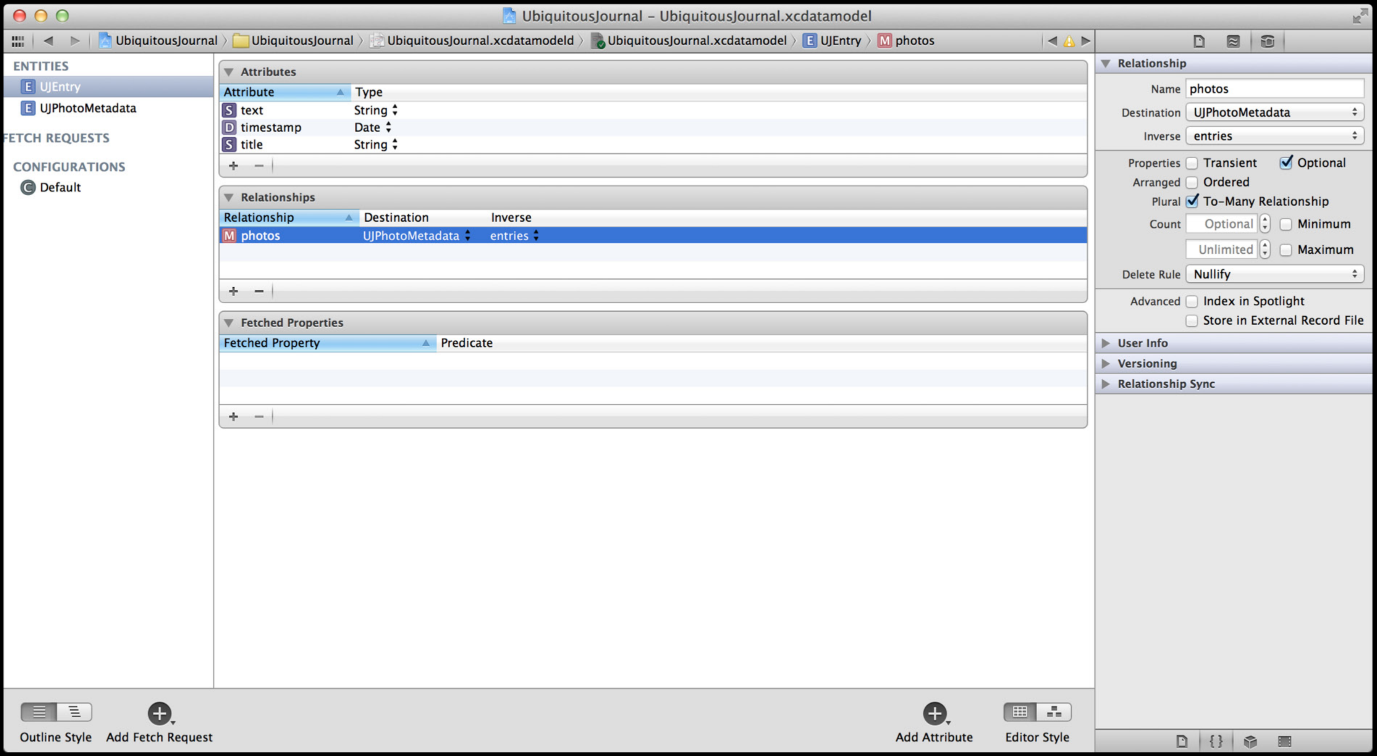
UJPhotoMetadata
- Attributes
  - filename
  - photoIDStr
- Relationships
  - entries

Entity
Name: UJPhotoMetadata
Class: UJPhotoMetadata
Abstract Entity: ☐
Parent Entity: No Parent Entity
Indexes

Entity Sync
Synchronization: Disabled

New UI

Table View



UbiquitousJournal – UbiquitousJournal.xcdatamodel

UbiquitousJournal › UbiquitousJournal › UbiquitousJournal.xcdatamodeld › UbiquitousJournal.xcdatamodel › UJEntry › photos

ENTITIES
- UJEntry
- UJPhotoMetadata

FETCH REQUESTS

CONFIGURATIONS
- Default

Attributes

Attribute	Type
text	String
timestamp	Date
title	String

Relationships

Relationship	Destination	Inverse
photos	UJPhotoMetadata	entries

Fetched Properties

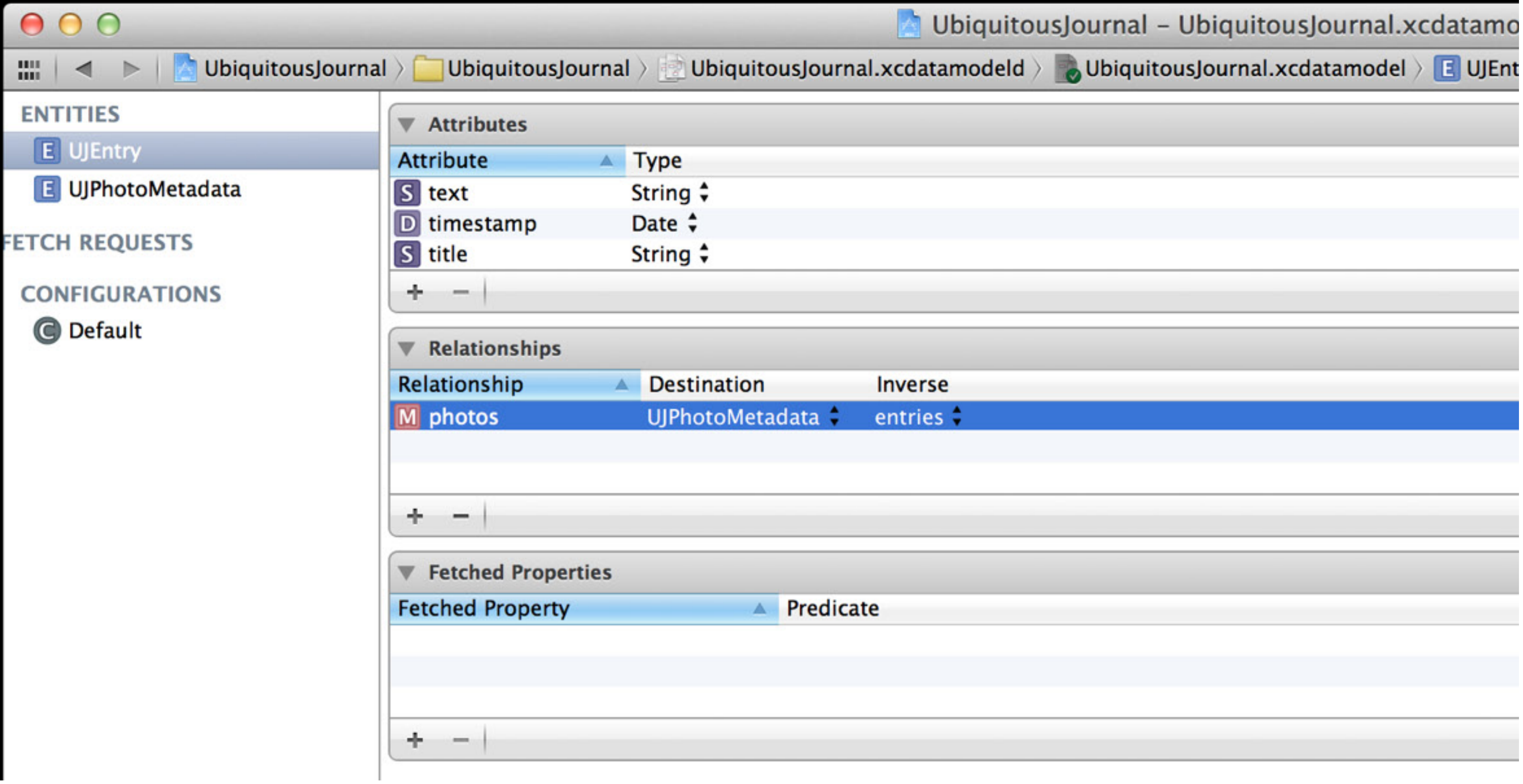
Fetched Property	Predicate

Relationship

Name	photos
Destination	UJPhotoMetadata
Inverse	entries
Properties	☐ Transient ✓ Optional
Arranged	☐ Ordered
Plural	✓ To-Many Relationship
Count	Optional ☐ Minimum
	Unlimited ☐ Maximum
Delete Rule	Nullify
Advanced	☐ Index in Spotlight
	☐ Store in External Record File

New UI

Table View


UbiquitousJournal – UbiquitousJournal.xcdatamo

UbiquitousJournal 〉 UbiquitousJournal 〉 UbiquitousJournal.xcdatamodeld 〉 UbiquitousJournal.xcdatamodel 〉 UJEnt

ENTITIES
- UJEntry
- UJPhotoMetadata

FETCH REQUESTS

CONFIGURATIONS
- Default

Attributes

Attribute	Type
text	String
timestamp	Date
title	String

Relationships

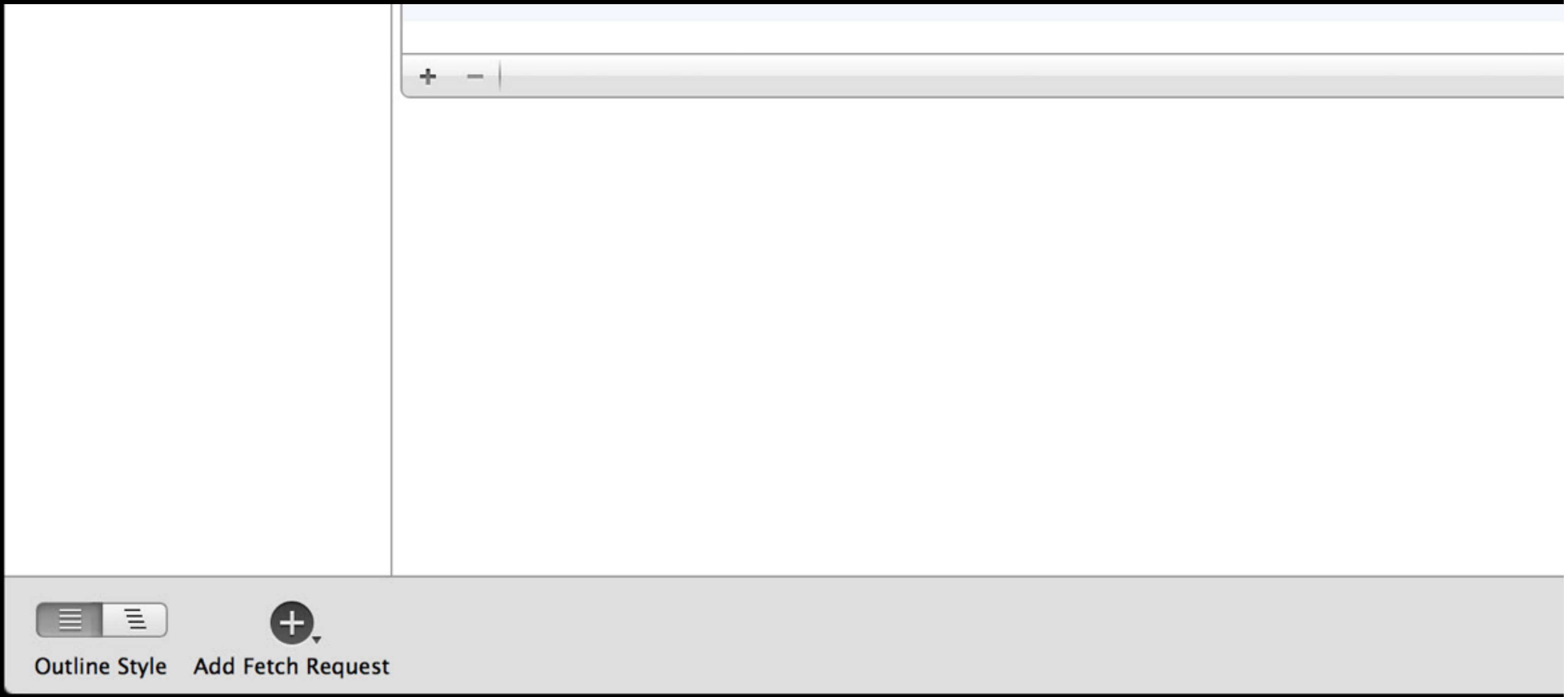
Relationship	Destination	Inverse
photos	UJPhotoMetadata	entries

Fetched Properties

Fetched Property	Predicate

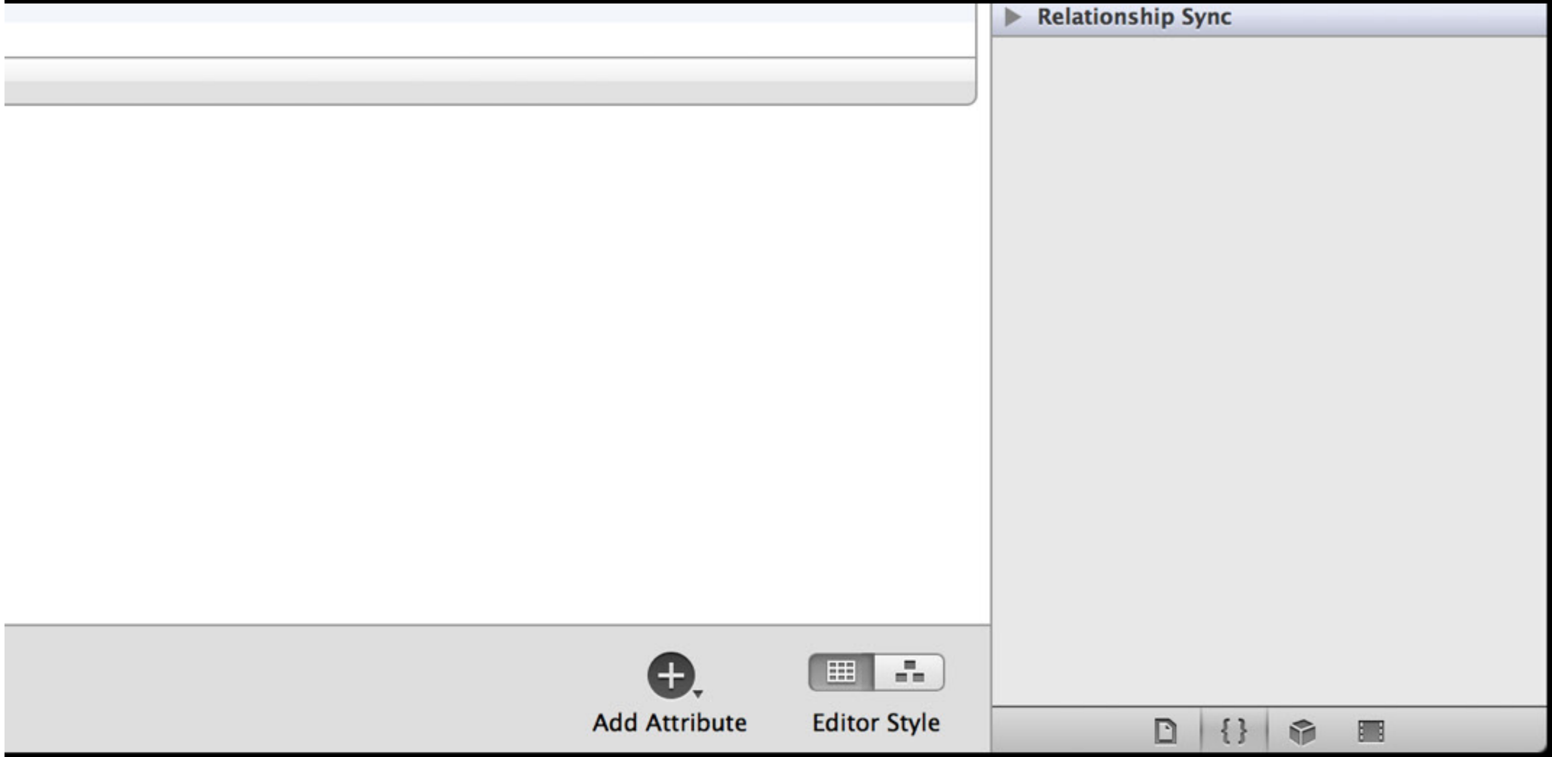
New UI

Table View



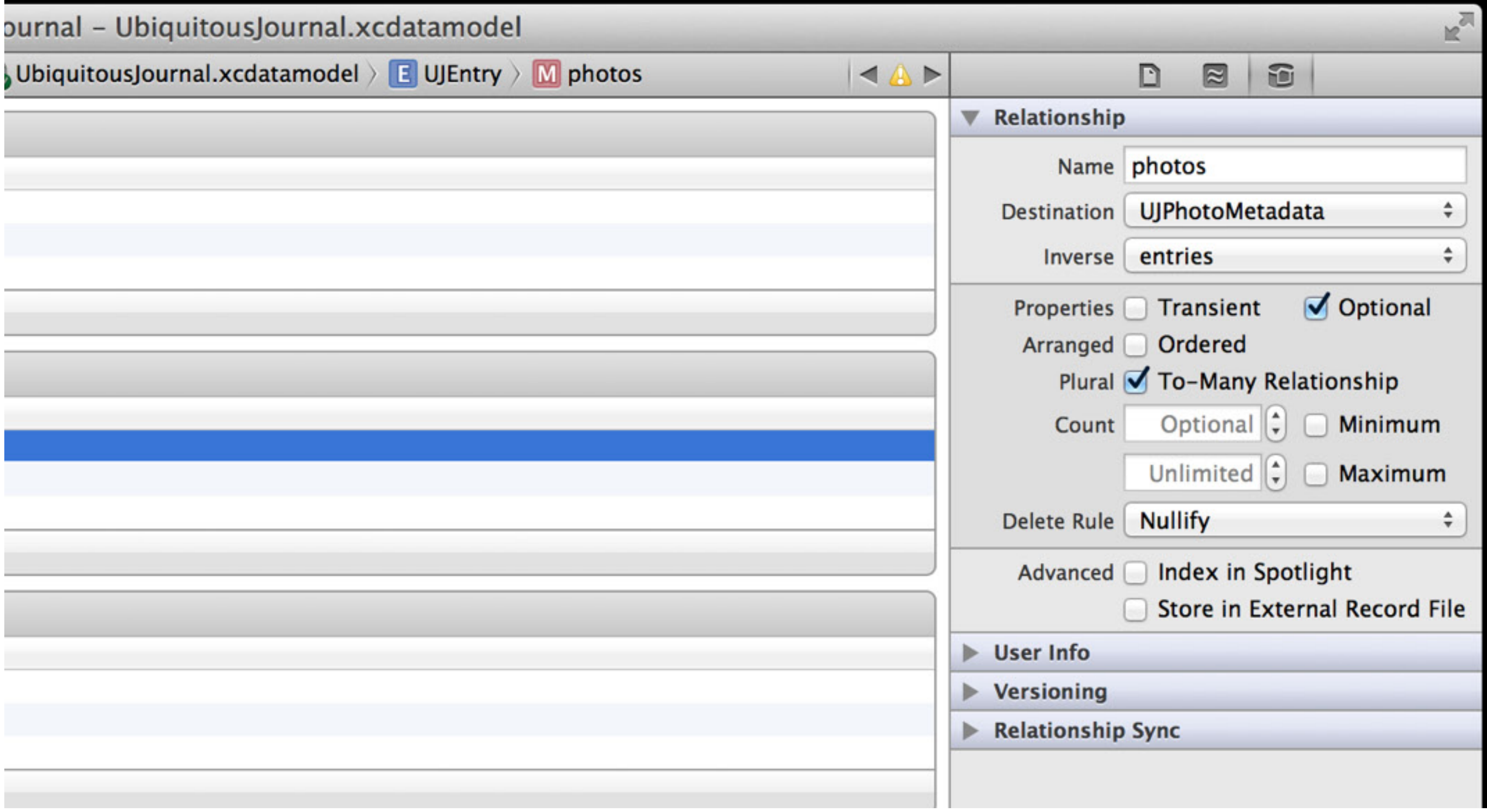
New UI

Table View



New UI

Table View



ournal – UbiquitousJournal.xcdatamodel

UbiquitousJournal.xcdatamodel 〉 E UJEntry 〉 M photos

Relationship

Name	photos
Destination	UJPhotoMetadata
Inverse	entries
Properties	☐ Transient ✓ Optional
Arranged	☐ Ordered
Plural	✓ To-Many Relationship
Count	Optional ☐ Minimum
	Unlimited ☐ Maximum
Delete Rule	Nullify
Advanced	☐ Index in Spotlight
	☐ Store in External Record File

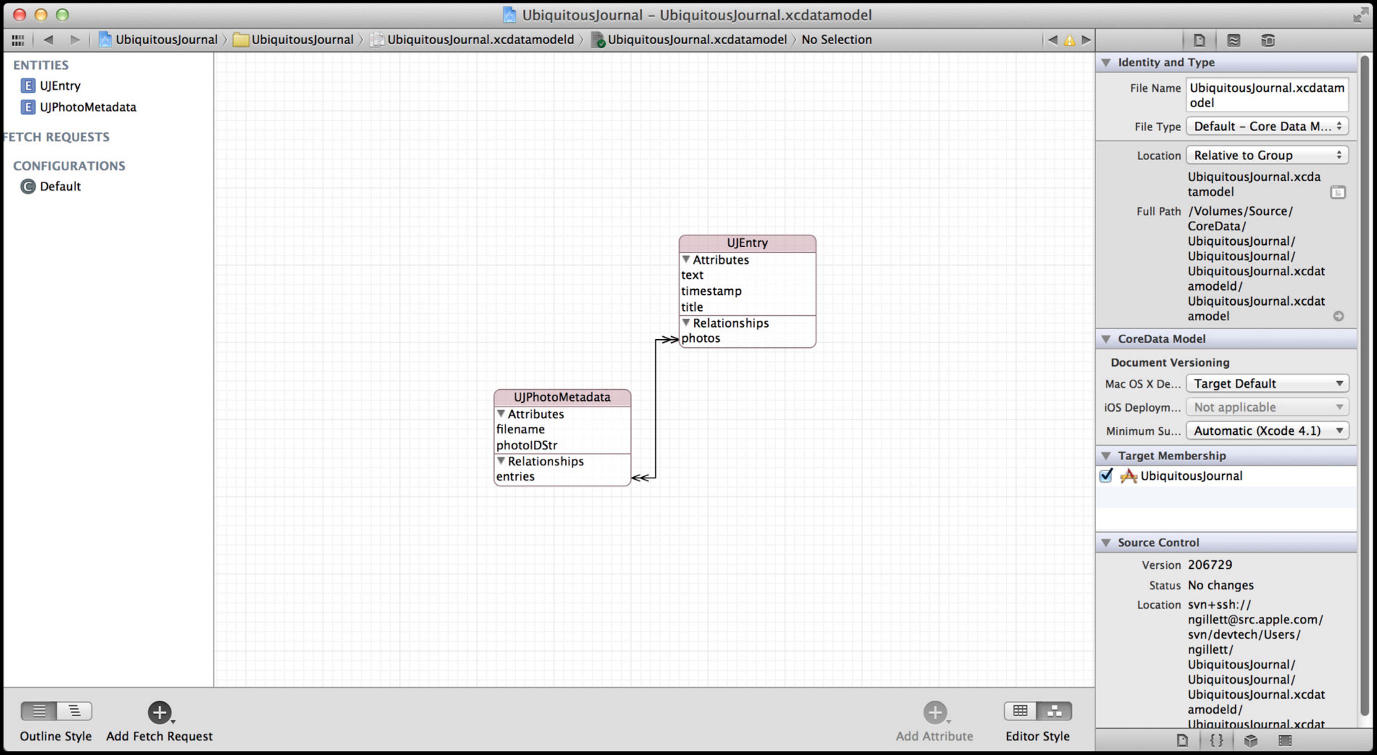
Optimized Model Format

- Speed up model loading
- Automatic with Xcode 4
- Lives in parallel with versioned models

Human Readable Xcode 4 Models

- Automatic in Xcode 4
  - Transparent upgrade from old format
- XML based
- Works with your favorite diff tools

Readable Models



Identity and Type

File Name: UbiquitousJournal.xcdatamodel
File Type: Default – Core Data M...
Location: Relative to Group
UbiquitousJournal.xcdatamodel
Full Path: /Volumes/Source/CoreData/UbiquitousJournal/UbiquitousJournal/UbiquitousJournal.xcdatamodeld/UbiquitousJournal.xcdatamodel

CoreData Model

Document Versioning
Mac OS X De...: Target Default
iOS Deploym...: Not applicable
Minimum Su...: Automatic (Xcode 4.1)

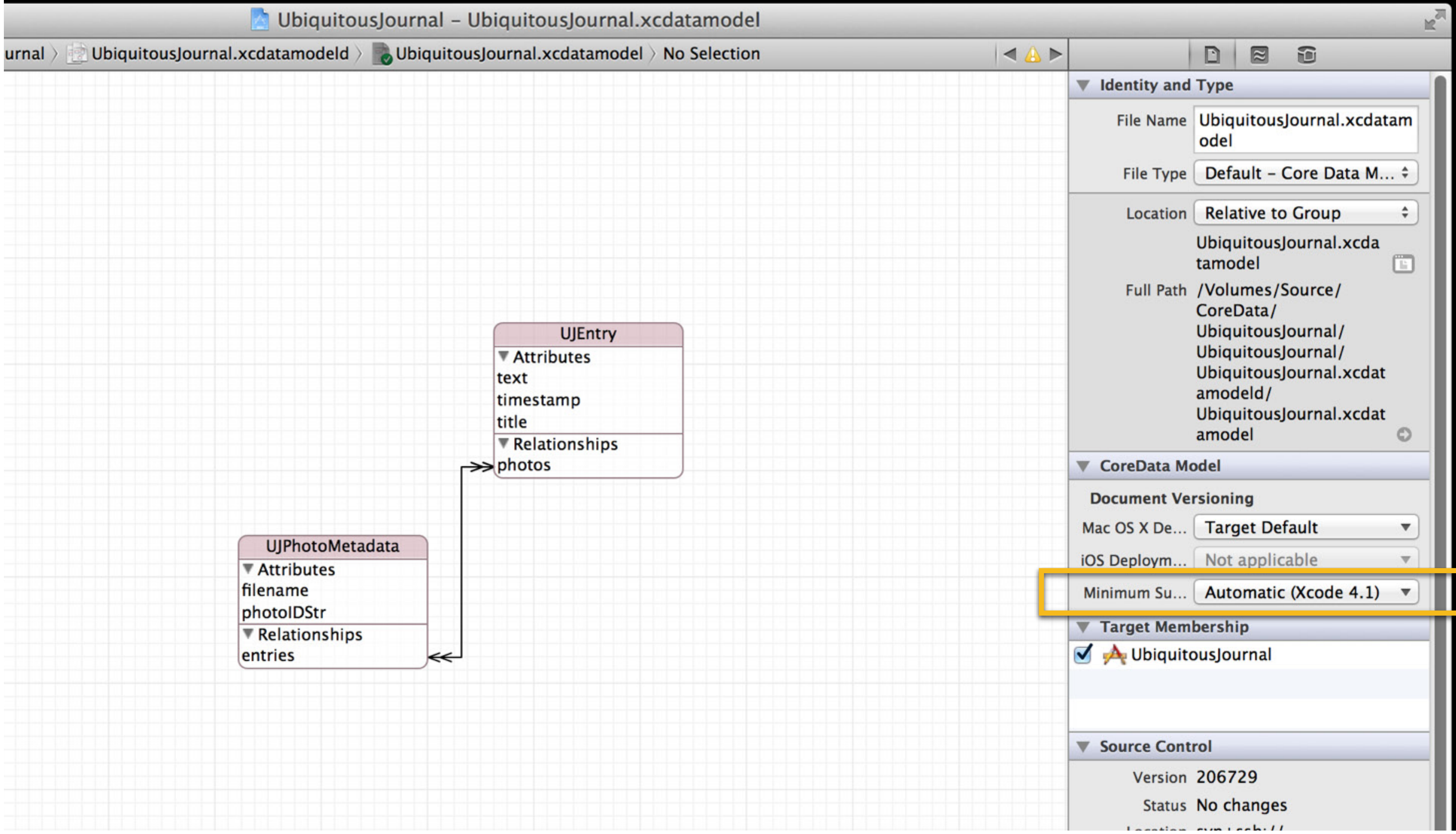
Target Membership

☑ UbiquitousJournal

Source Control

Version: 206729
Status: No changes
Location: svn+ssh://ngillett@src.apple.com/svn/devtech/Users/ngillett/UbiquitousJournal/UbiquitousJournal/UbiquitousJournal.xcdatamodeld/UbiquitousJournal.xcdat

Readable Models



UbiquitousJournal – UbiquitousJournal.xcdatamodel

urnal 〉 UbiquitousJournal.xcdatamodeld 〉 UbiquitousJournal.xcdatamodel 〉 No Selection

UJEntry
▼ Attributes
text
timestamp
title
▼ Relationships
photos

UJPhotoMetadata
▼ Attributes
filename
photoIDStr
▼ Relationships
entries

▼ Identity and Type

File Name	UbiquitousJournal.xcdatamodel
File Type	Default – Core Data M…
Location	Relative to Group
	UbiquitousJournal.xcdatamodel
Full Path	/Volumes/Source/CoreData/UbiquitousJournal/UbiquitousJournal/UbiquitousJournal.xcdatamodeld/UbiquitousJournal.xcdatamodel

▼ CoreData Model

Document Versioning

Mac OS X De…	Target Default
iOS Deploym…	Not applicable
Minimum Su…	Automatic (Xcode 4.1)

▼ Target Membership

✓ UbiquitousJournal

▼ Source Control

Version	206729
Status	No changes

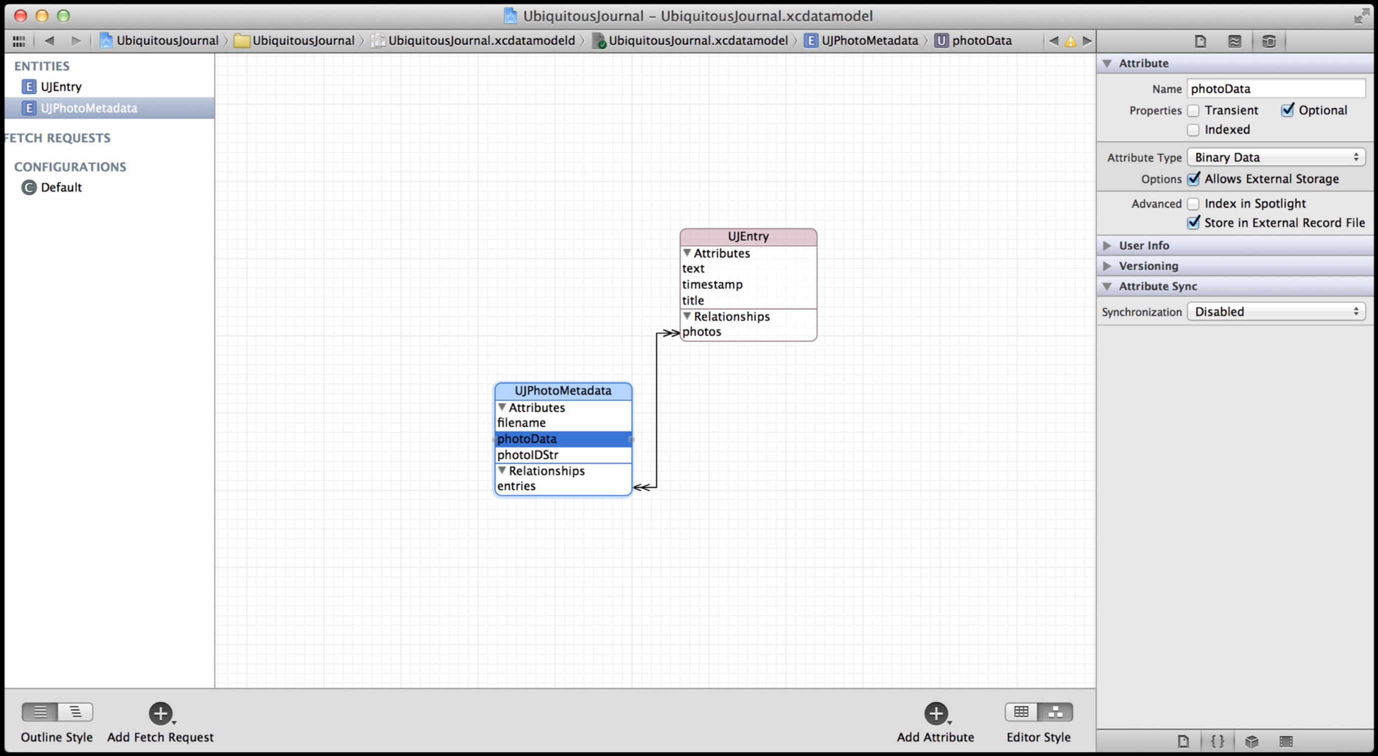
Scalar Accessors

- Avoid overhead of value object construction
- Checkbox during method creation

Automatic Reference Counting

- Makes memory management easier
- No need to implement or call retain and release
- Opt-in per project
  - New project templates enable by default
- Opt-out per file
- Go see the session or watch it on iTunes

External Binary Data


UbiquitousJournal – UbiquitousJournal.xcdatamodel

UbiquitousJournal › UbiquitousJournal › UbiquitousJournal.xcdatamodeld › UbiquitousJournal.xcdatamodel › UJPhotoMetadata › photoData

ENTITIES
- UJEntry
- UJPhotoMetadata

FETCH REQUESTS

CONFIGURATIONS
- Default

UJEntry
- Attributes: text, timestamp, title
- Relationships: photos

UJPhotoMetadata
- Attributes: filename, photoData, photoIDStr
- Relationships: entries

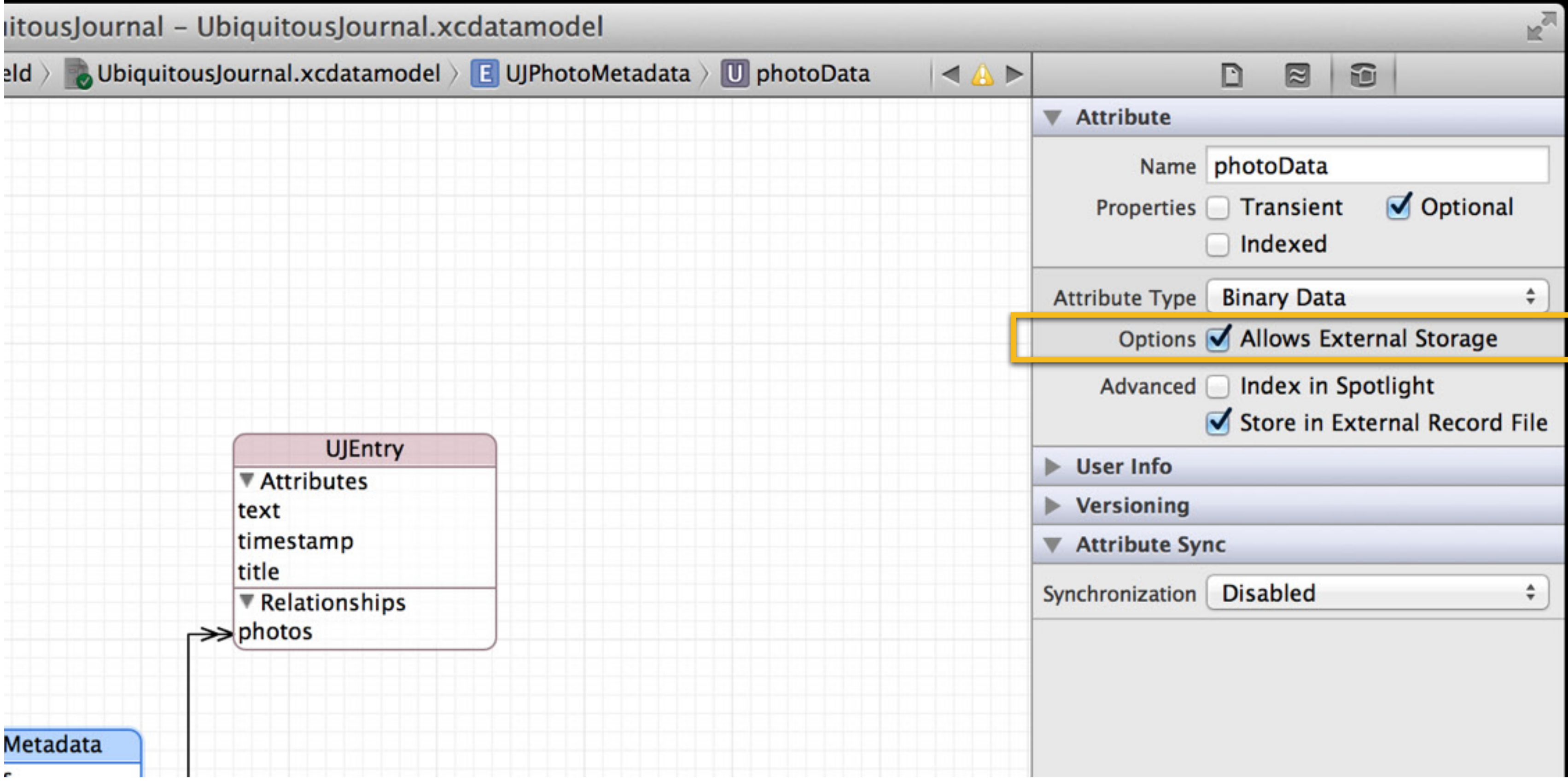
Attribute
- Name: photoData
- Properties: Transient [ ], Optional [✓], Indexed [ ]
- Attribute Type: Binary Data
- Options: Allows External Storage [✓]
- Advanced: Index in Spotlight [ ], Store in External Record File [✓]

User Info

Versioning

Attribute Sync
- Synchronization: Disabled

External Binary Data



Attribute

Name: photoData

Properties: [ ] Transient [✓ Optional] [ ] Indexed

Attribute Type: Binary Data

Options: [✓ Allows External Storage]

Advanced: [ ] Index in Spotlight [✓ Store in External Record File]

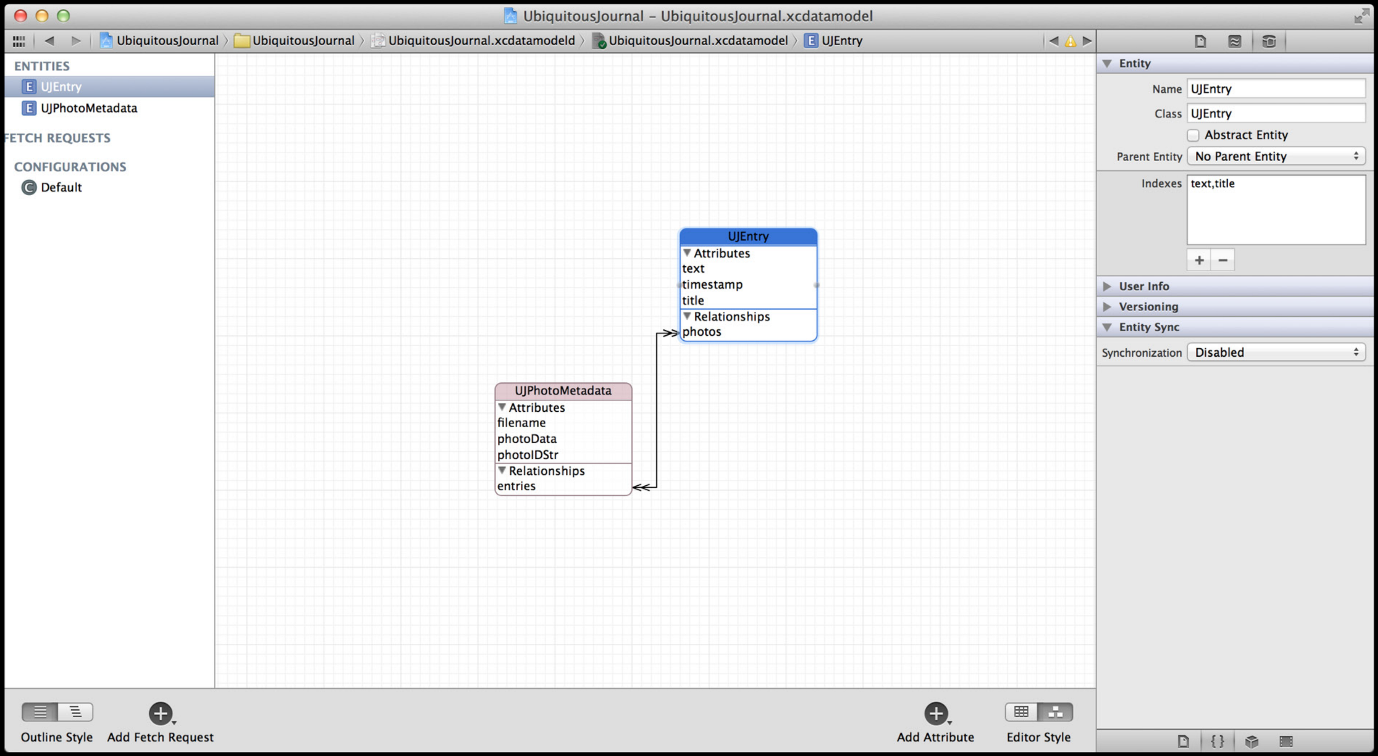
Attribute Sync

Synchronization: Disabled

Compound Indexes

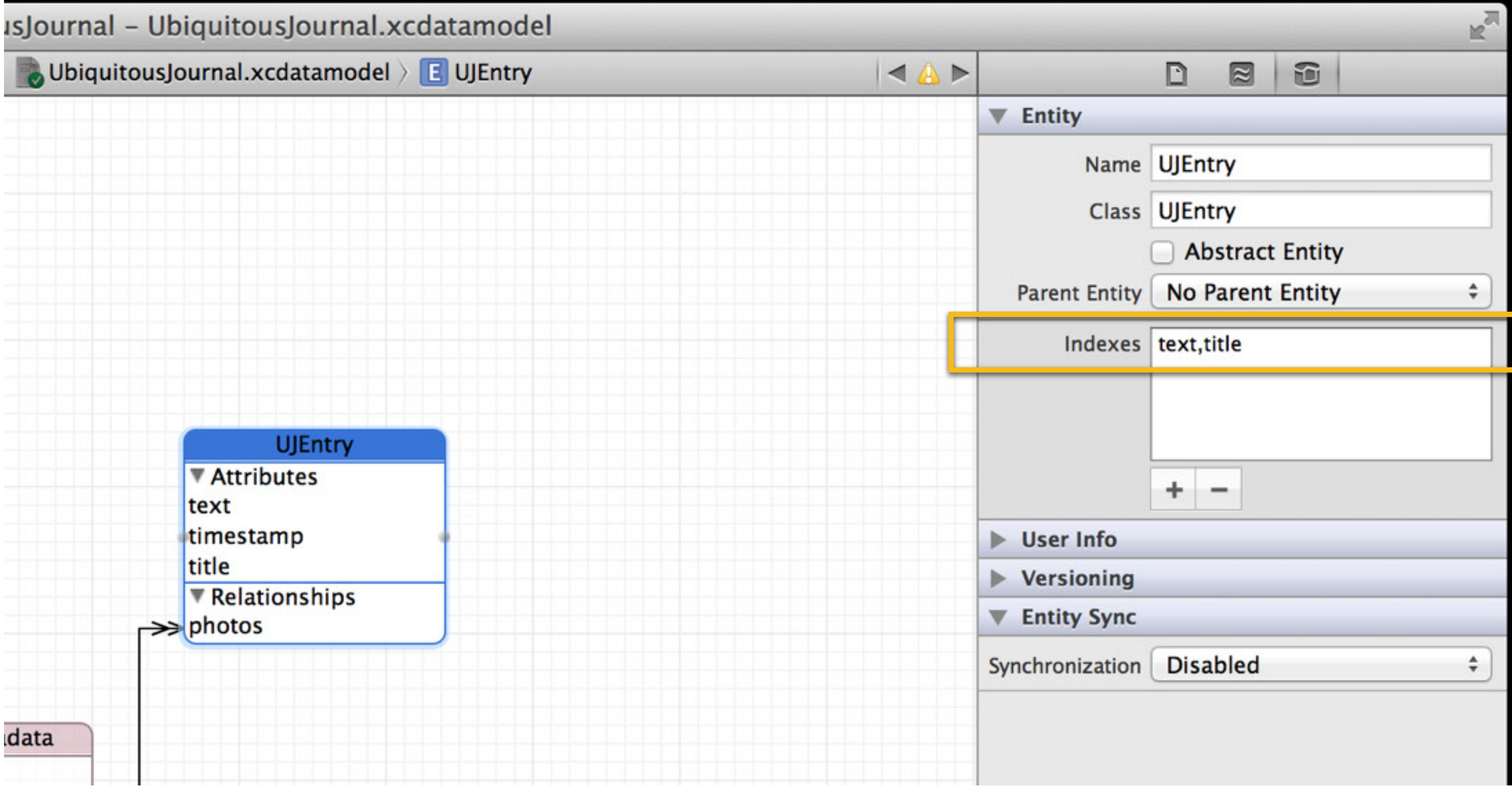
- Index across multiple properties
- Supported by SQLite store

Compound Indexes



Name: UJEntry
Class: UJEntry
Abstract Entity
Parent Entity: No Parent Entity
Indexes: text,title
Synchronization: Disabled

Compound Indexes



usJournal – UbiquitousJournal.xcdatamodel

UbiquitousJournal.xcdatamodel 〉 UJEntry

Entity

Name: UJEntry

Class: UJEntry

Abstract Entity

Parent Entity: No Parent Entity

Indexes: text,title

User Info

Versioning

Entity Sync

Synchronization: Disabled

Summary

- Concurrency
- Auto Save
- Ordered relationships
- iCloud
- Incremental stores
- Developer Tools

http://bugreport.apple.com

- We don’t know unless you tell us
- Bugs fixed faster with
  - Steps to reproduce
  - Sample project
- Also use for
  - Feature requests
  - Enhancement requests
  - Performance issues
  - Documentation requests

More Information

Michael Jurewitz
Developer Tools Evangelist
jurewitz@apple.com

Core Data Documentation
Programming Guides, Examples, and Tutorials
http://developer.apple.com

Apple Developer Forums
http://devforums.apple.com

Related Sessions

iCloud Storage Overview	Presidio Tuesday 11:30AM
Auto Save and Versions in Mac OS X 10.7 Lion	Pacific Heights Tuesday 3:15PM
Taking Advantage of File Coordination	Pacific Heights Tuesday 4:30PM
Introducing Automatic Reference Counting	Presidio Tuesday 4:30PM

Labs

Core Data Lab	Developer Tools Lab B Tuesday 4:30PM
Core Data Lab	Developer Tools Lab B Wednesday 4:30PM
Core Data Lab	Developer Tools Lab A Thursday 2:00PM

WWDC2011