What’s New in Core Data on iOS

Session 303

Adam Swift
Senior Software Engineer

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

Roadmap

- Concurrency
- Data protection
- Ordered relationships
- UIManagedDocument
- iCloud
- Incremental stores
- Developer tools

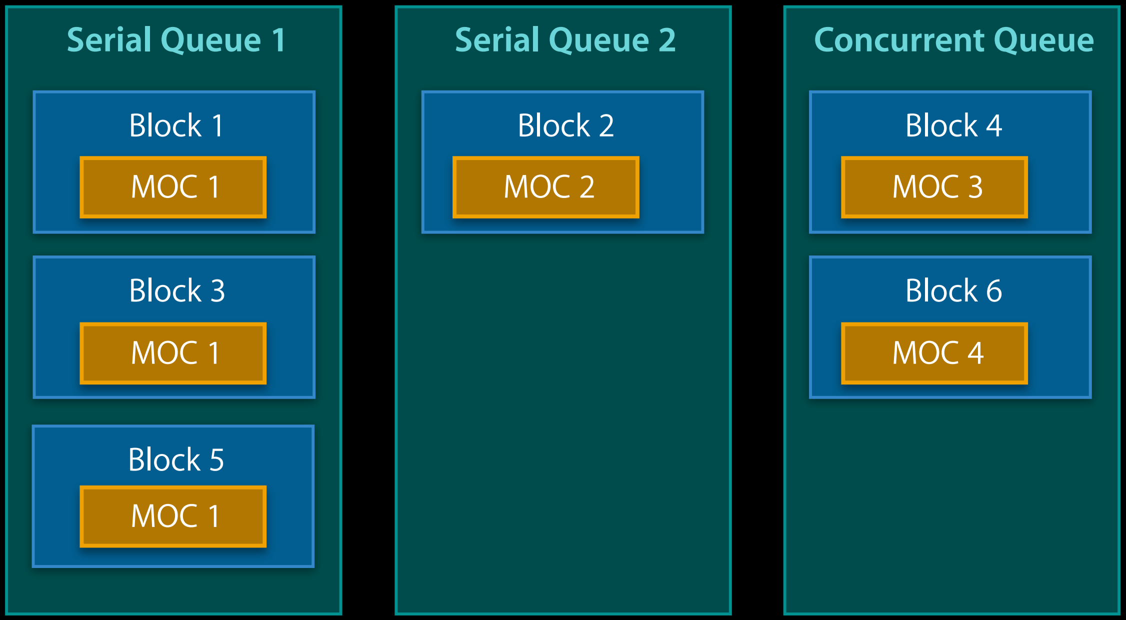
Concurrency

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 iOS 3.0–iOS 4.3
- 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 re-entrancy
- Illegal to throw an exception out of your block

-performBlockAndWait:

- Synchronous
- Not an event
- No autorelease pool
- Supports re-entrancy
- 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

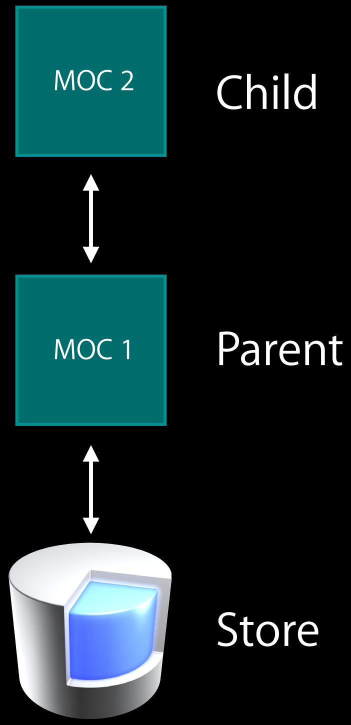
- Create a dispatch group
- Call `dispatch_group_enter`
- Worker block call `dispatch_group_leave`
- Use `dispatch_group_wait` and `dispatch_group_notify` normally

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

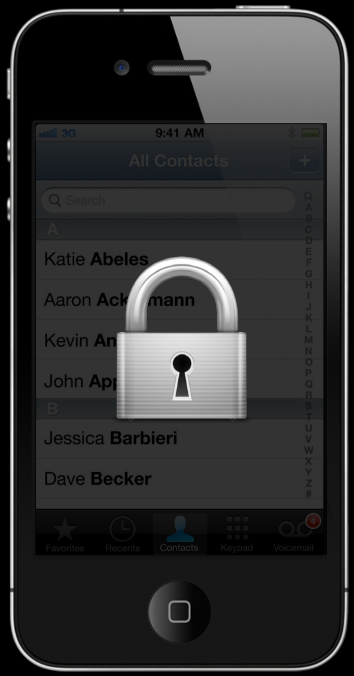
Things to Remember

- 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

Data Protection

Data Protection Intro

- Encrypt user data
- File-level protection
- Tied to user passcode



Data Protection Classes

- Declared in NSFileManager
- First introduced in iOS 4

```
NSFileProtectionNone
NSFileProtectionComplete
```

- New in iOS 5

```
NSFileProtectionCompleteUnlessOpen
NSFileProtectionCompleteUntilFirstUserAuthentication
```

First Boot

NSFileProtection Class	Open	Create	Read/Write
Complete Unless Open	✗	✓	✓
Complete Until First User Authentication	✗	✗	✗

Locked

NSFileProtection Class	Open	Create	Read/Write
Complete Unless Open	✗	✓	✓
Complete Until First User Authentication	✓	✓	✓

Persistent Store Protection

- Persistent store option

  `NSPersistentStoreFileProtectionKey`

- Use NSFileManager values
- Default in iOS 5

  `NSFileProtectionCompleteUntilFirstUserAuthentication`

Usage Patterns

`NSFileProtectionComplete`

- Background tasks can’t access store while locked

`NSFileProtectionCompleteUnlessOpen`

- Background tasks can’t open store while locked

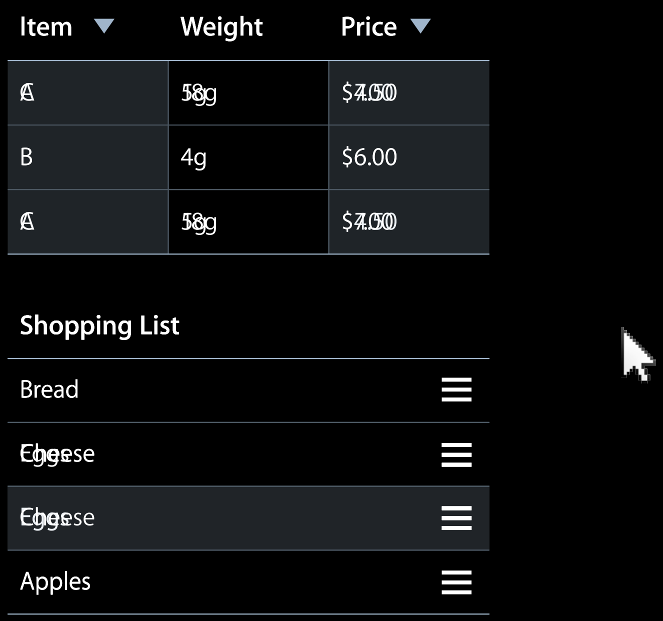
`NSFileProtectionCompleteUntilFirstUserAuthentication`

- Core Location region monitoring may trigger access before first unlock

Ordered Relationships

Sorting vs. Ordering

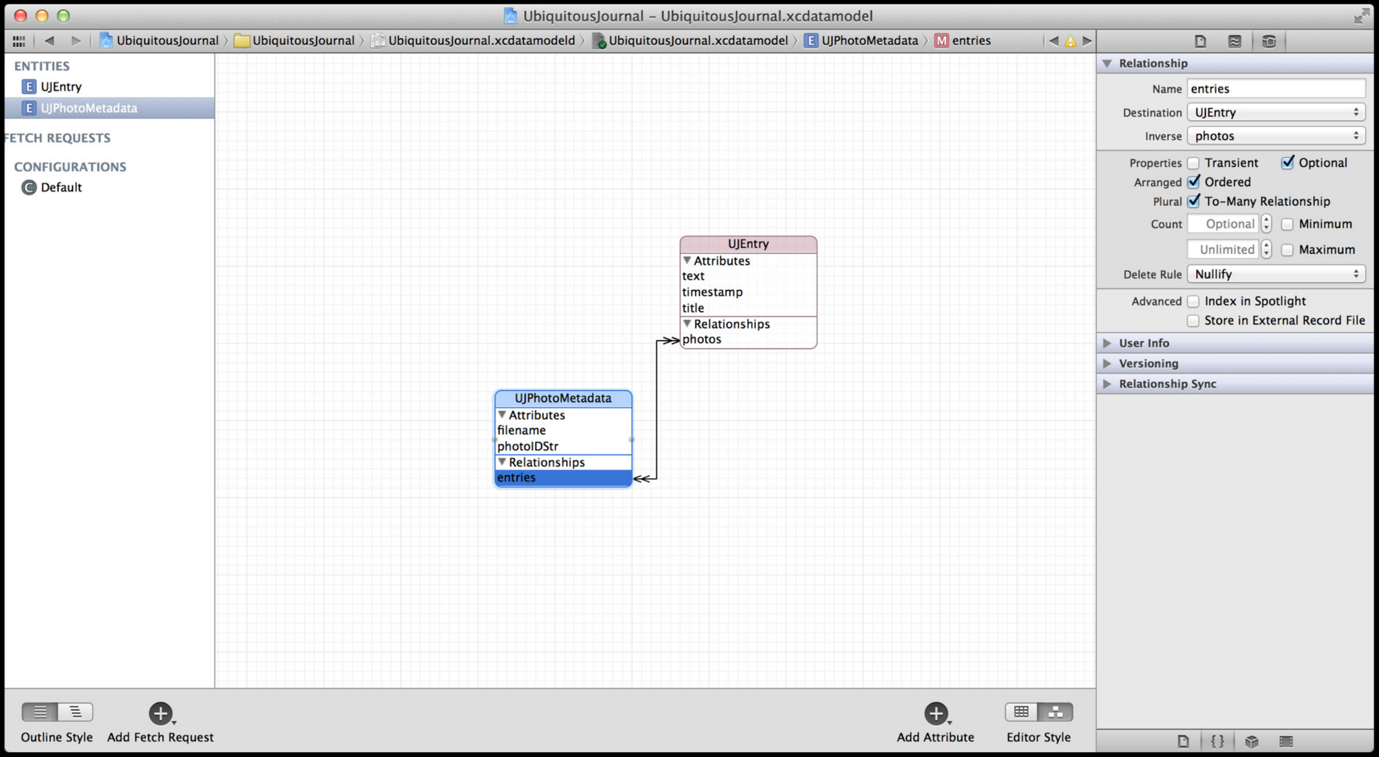
- Sorting by value
  - Derived
  - Change your view
- Arbitrary ordering
  - List
  - Flexible control
  - Not tied to any intrinsic value



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

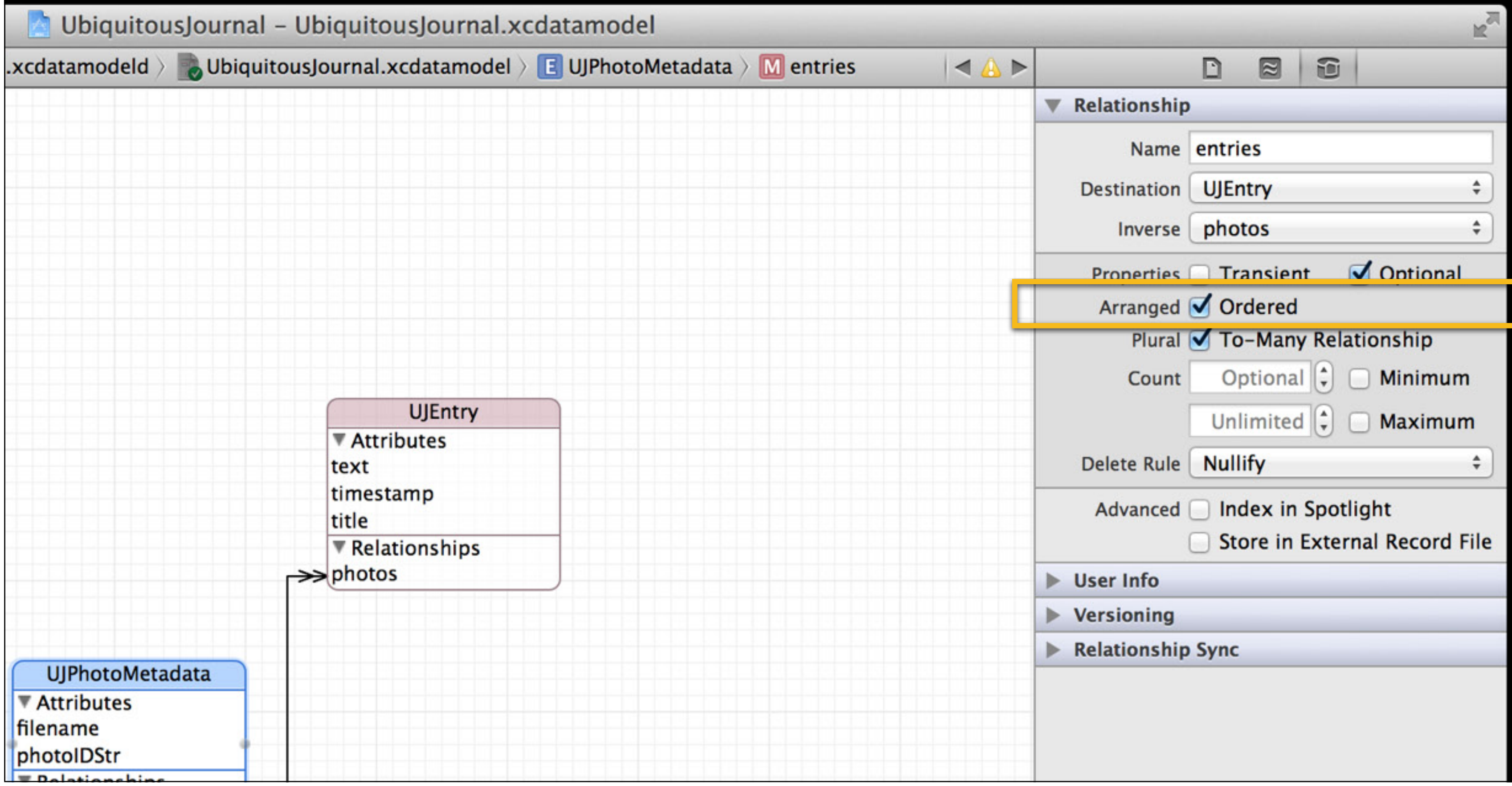
Ordered Relationships



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

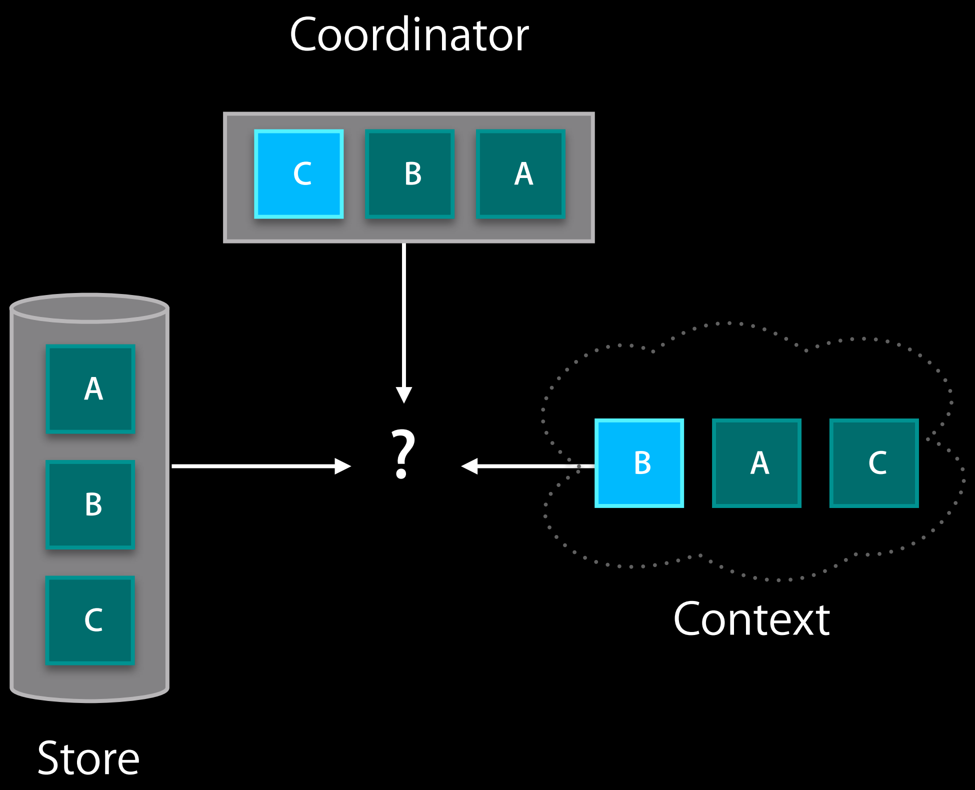
  `NSKeyValueChangeInsertion`

  `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
- Postprocess to impose ordering

Ordered Relationship Recap

- For arbitrary ordering
- Ordered collection KVC/KVO
- Performance

UIManagedDocument

Documents on iOS

- Integrated document architecture
- UIDocument gives you
    - Autosaving
    - iCloud integration
    - Asynchronous I/O

Storing Documents in iCloud Using iOS 5

Presidio
Wednesday 3:15PM

UIManagedDocument

- UIDocument-based
- Adds Core Data features
  - Scale
  - Undo
  - Graph management
  - Searching and sorting
  - Conflict resolution

Using Managed Documents

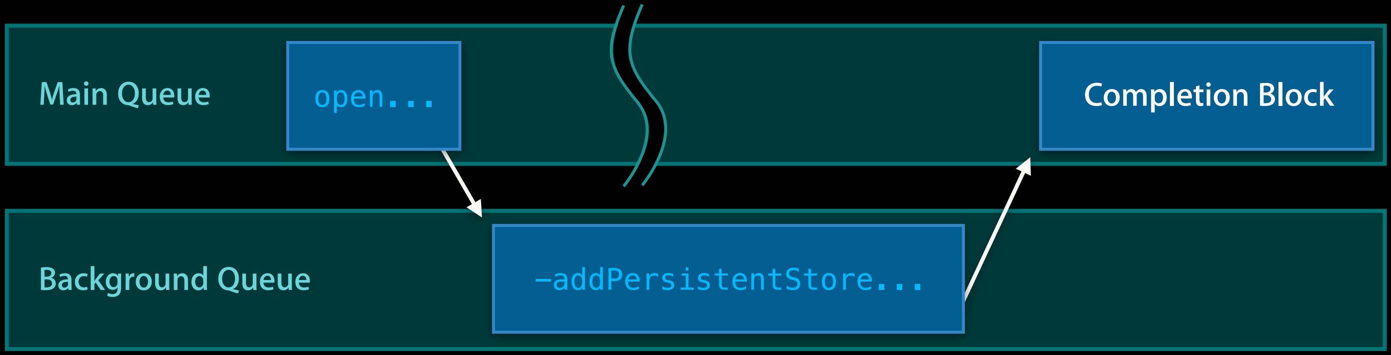
- Concrete UIDocument subclass
- Instantiate and configure
- Read data model from app bundle

Asynchronous I/O

- Don’t freeze UI
- Caller initiates open/read/save
  - Save takes snapshot synchronously
  - Provide completion handler block
- Load/write run on background queue

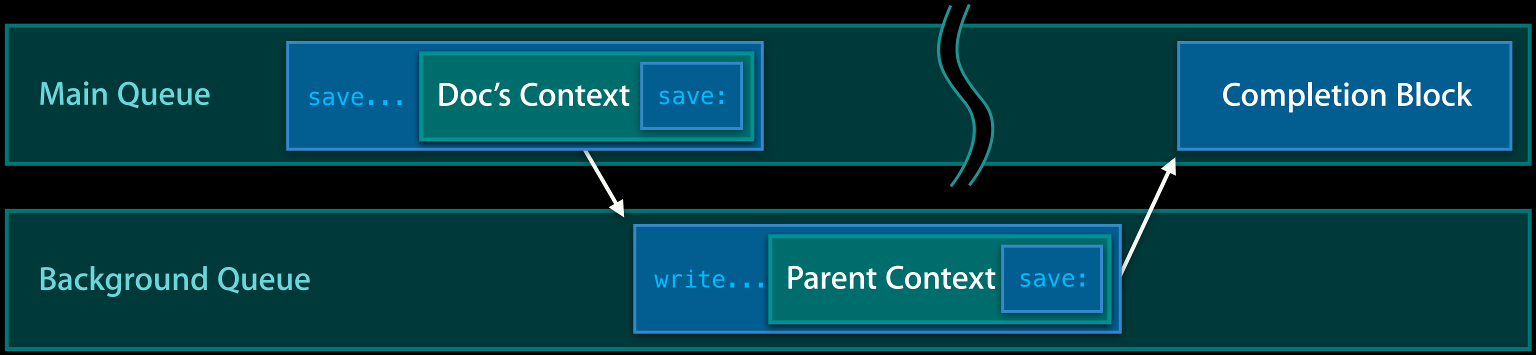
Asynchronous Load

- Simply add store at open
- Add store on background queue
  - Usually no reading
  - Migrate in background



Asynchronous Write

- Nested contexts
- Document's context on main queue
  - Takes snapshot
  - Saves to parent
- Parent context saves to disk on background queue



Unmodeled Data

- Large files should live outside the database
- Handle additional content
- Stored in document file package
- Optional API for subclasses

Additional Content API

- Read additional content

  `readAdditionalContentFromURL:error:`

- Snapshot for save

  `additionalContentForURL:error:`

- Write additional content on background queue

  `writeAdditionalContent:toURL:originalContentURL:error:`

Summary

- UIDocument architecture
- Powerful Core Data features
- Unmodeled content
- iCloud integration

iCloud

Nick Gillett
Software Engineer

Core Data, iCloud, and You

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

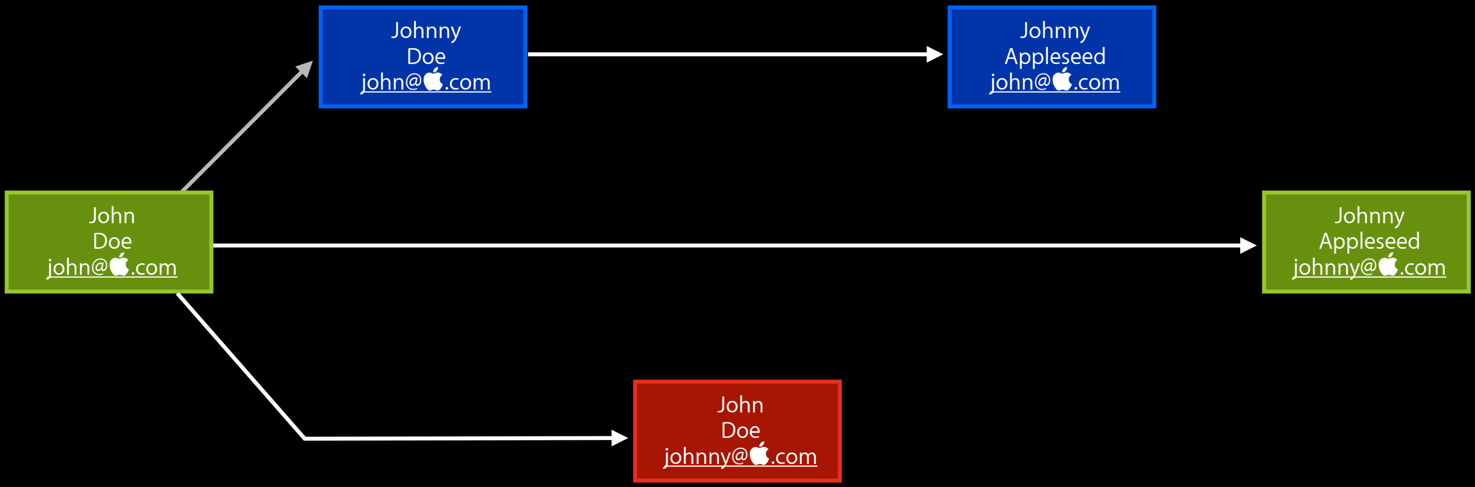
iCloud

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

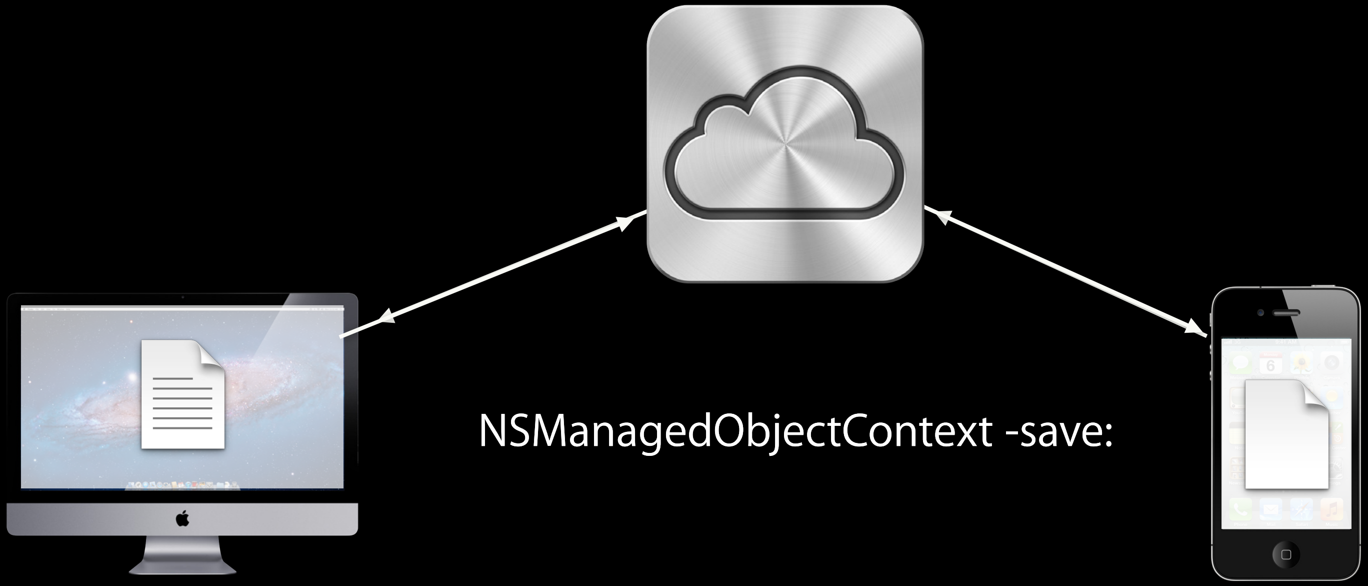
- Options when adding persistent store
- Respond to import notification

Less Code

Our part

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

How does this work?


NSManagedObjectContext -save:

New API

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

NSPersistentStoreUbiquitousContentNameKey

datanqlite

/Users/test/data.sqlite

foodtspanic.

06302011.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

NSPersistentStoreDidImportUbiquitousContentChangesNotification

Responding to an import

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

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 UIDocument conflict resolution APIs

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” } ] }
```

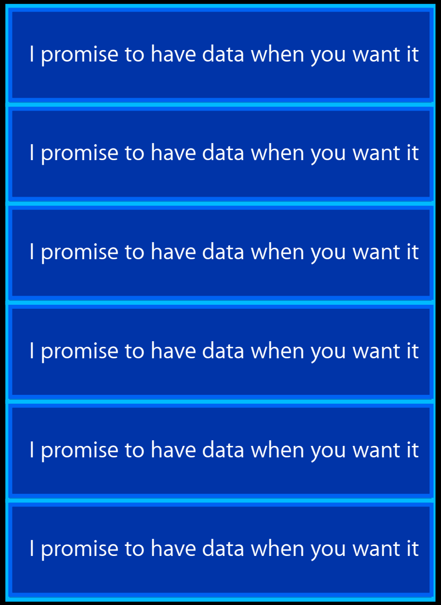
Incremental Store

- 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

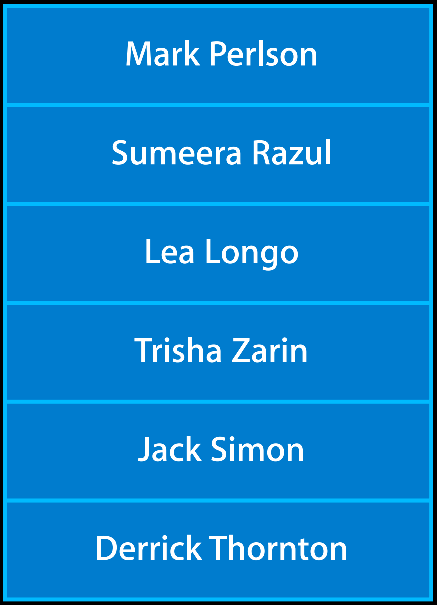
Incremental Store

- 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

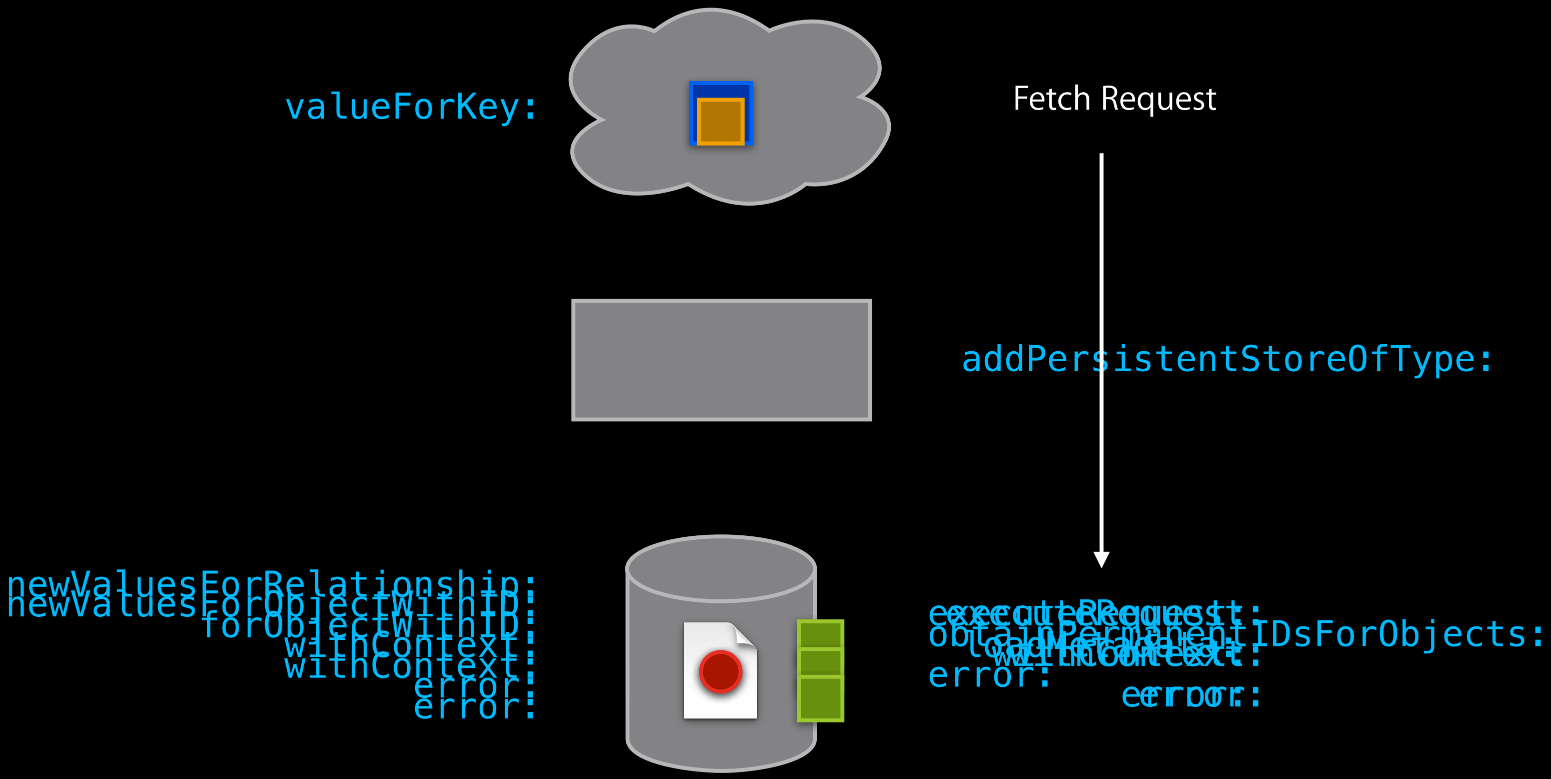
Incremental Store

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



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

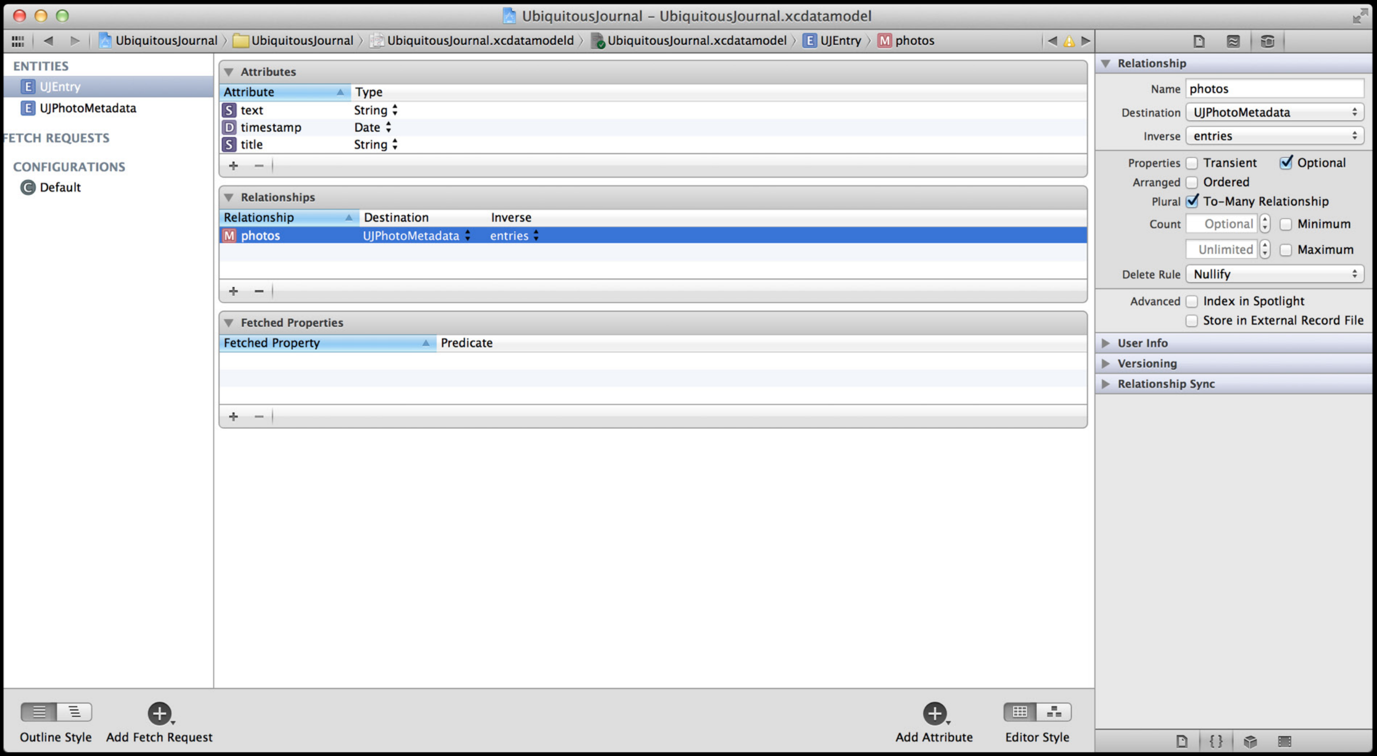
Developer Tools

Xcode 4

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

New UI

Table View



UbiquitousJournal – UbiquitousJournal.xcdatamodel

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

ENTITIES
- E UJEntry
- E UJPhotoMetadata

FETCH REQUESTS

CONFIGURATIONS
- C Default

Attributes

Attribute	Type
S text	String
D timestamp	Date
S title	String

Relationships

Relationship	Destination	Inverse
M photos	UJPhotoMetadata	entries

Fetched Properties

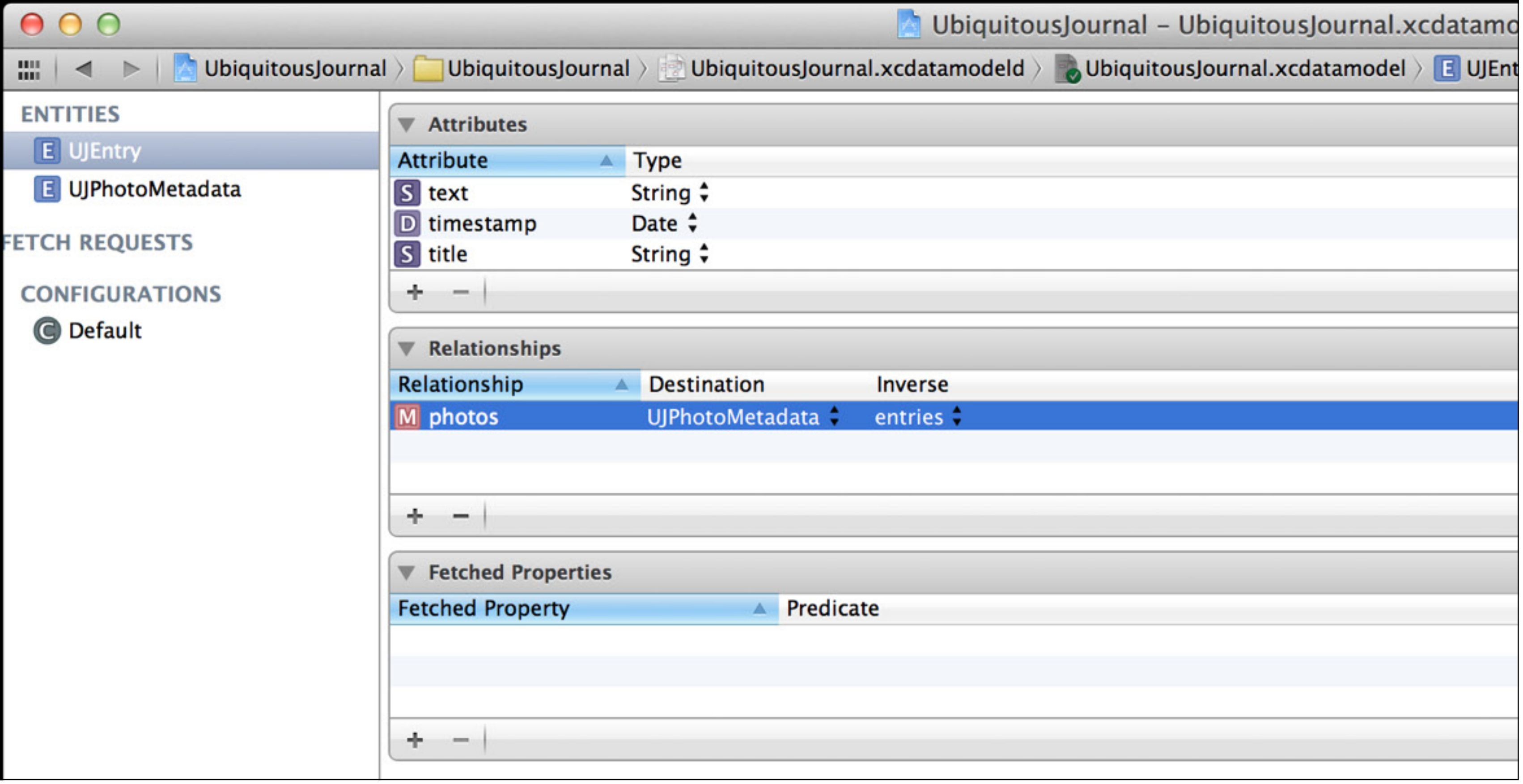
Fetched Property	Predicate

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
Relationships
Fetched Properties

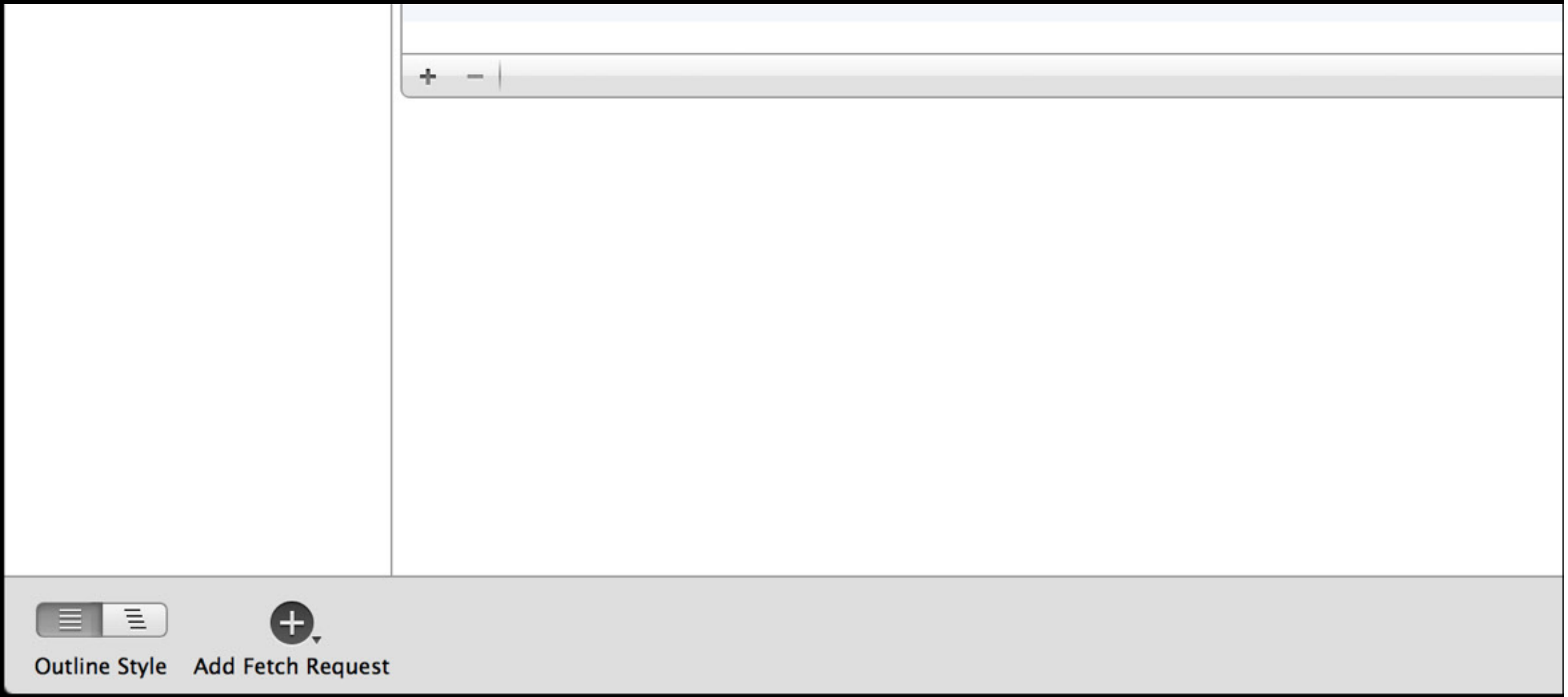
Attribute	Type
text	String
timestamp	Date
title	String

Relationship	Destination	Inverse
photos	UJPhotoMetadata	entries

Fetched Property	Predicate

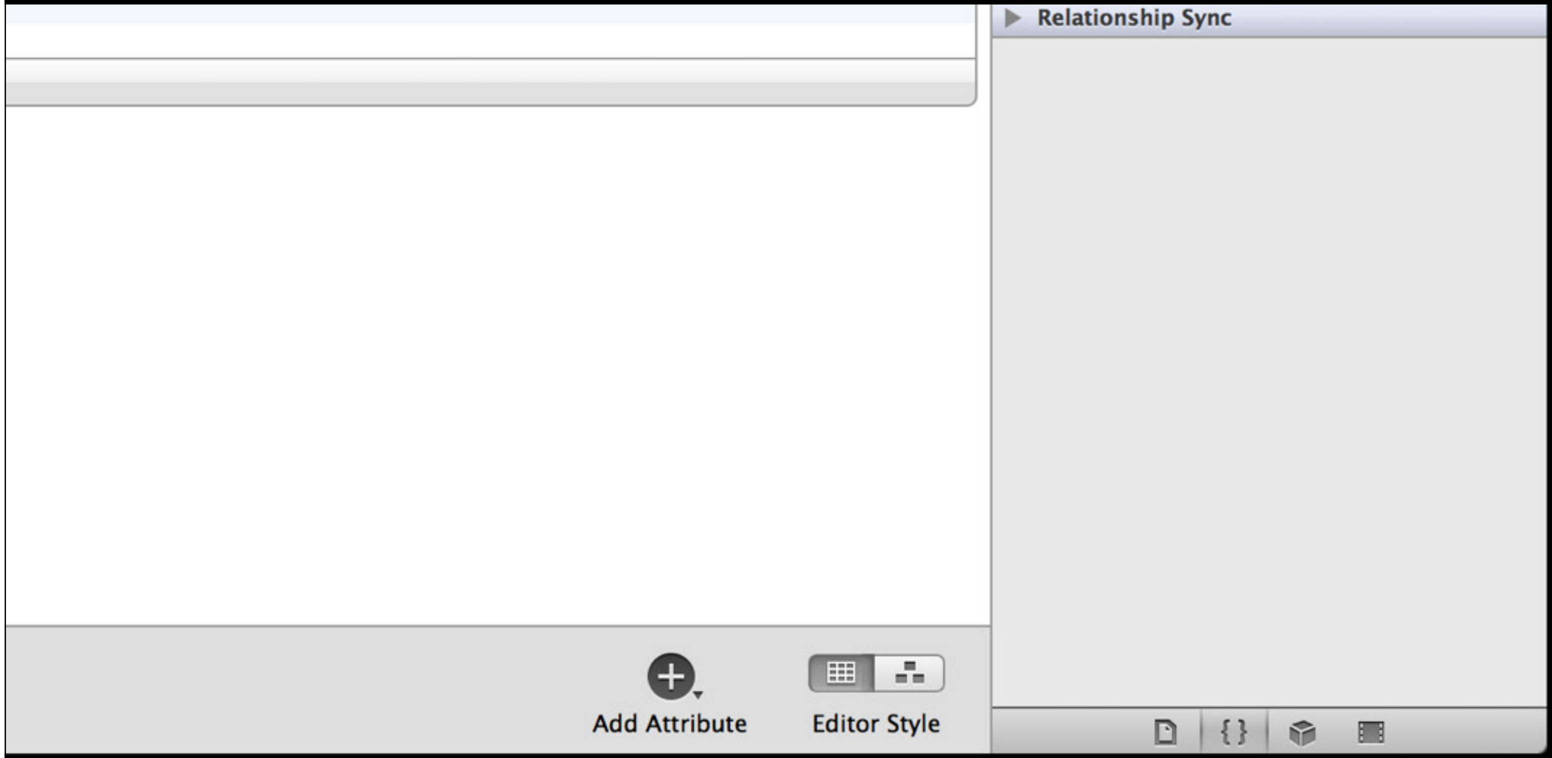
New UI

Table View



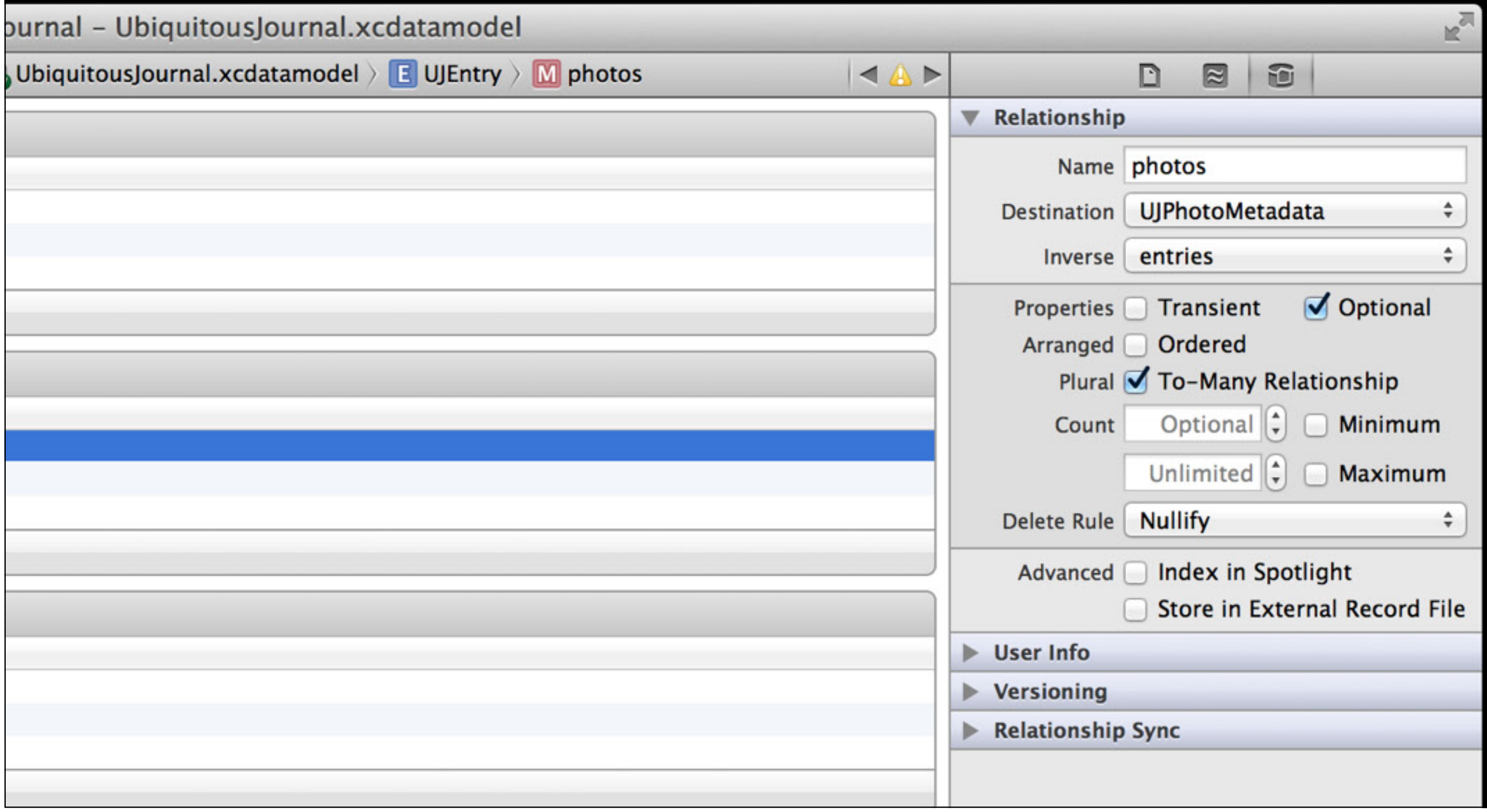
New UI

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

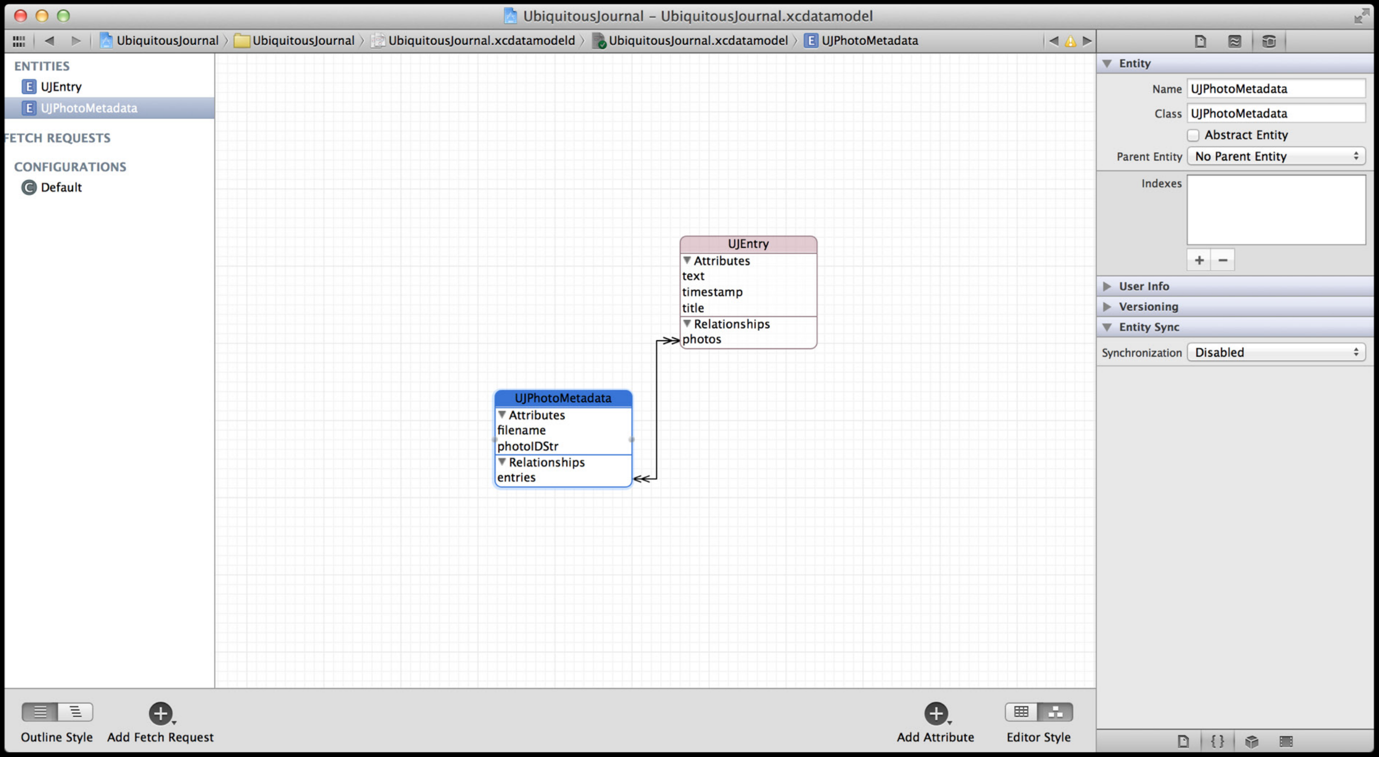
▶ User Info

▶ Versioning

▶ Relationship Sync

New UI

Diagram View



Name: UJPhotoMetadata
Class: UJPhotoMetadata
Abstract Entity
Parent Entity: No Parent Entity
Indexes
Synchronization: Disabled

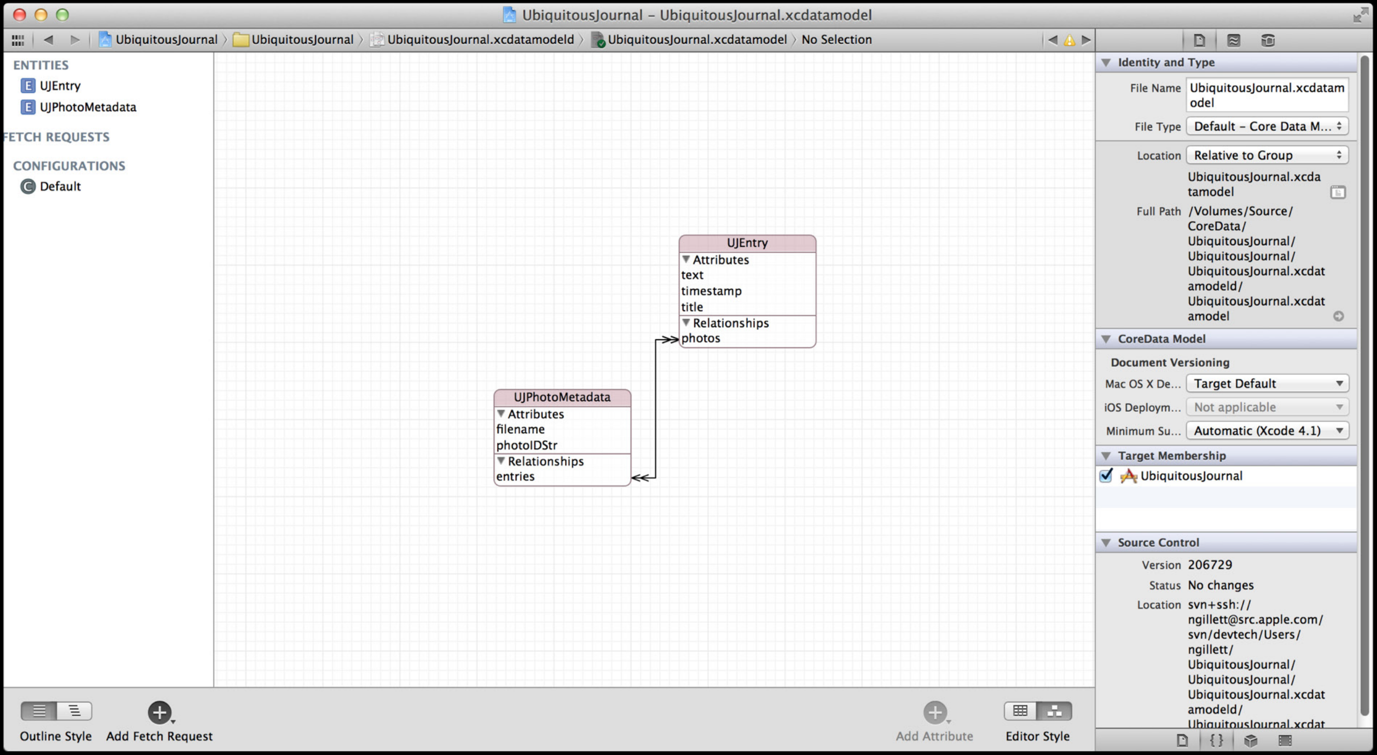
Optimized Model Format

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

User Readable Xcode 4 Models

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

Readable Models


UbiquitousJournal – UbiquitousJournal.xcdatamodel

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

ENTITIES
- UJEntry
- UJPhotoMetadata

FETCH REQUESTS

CONFIGURATIONS
- Default

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

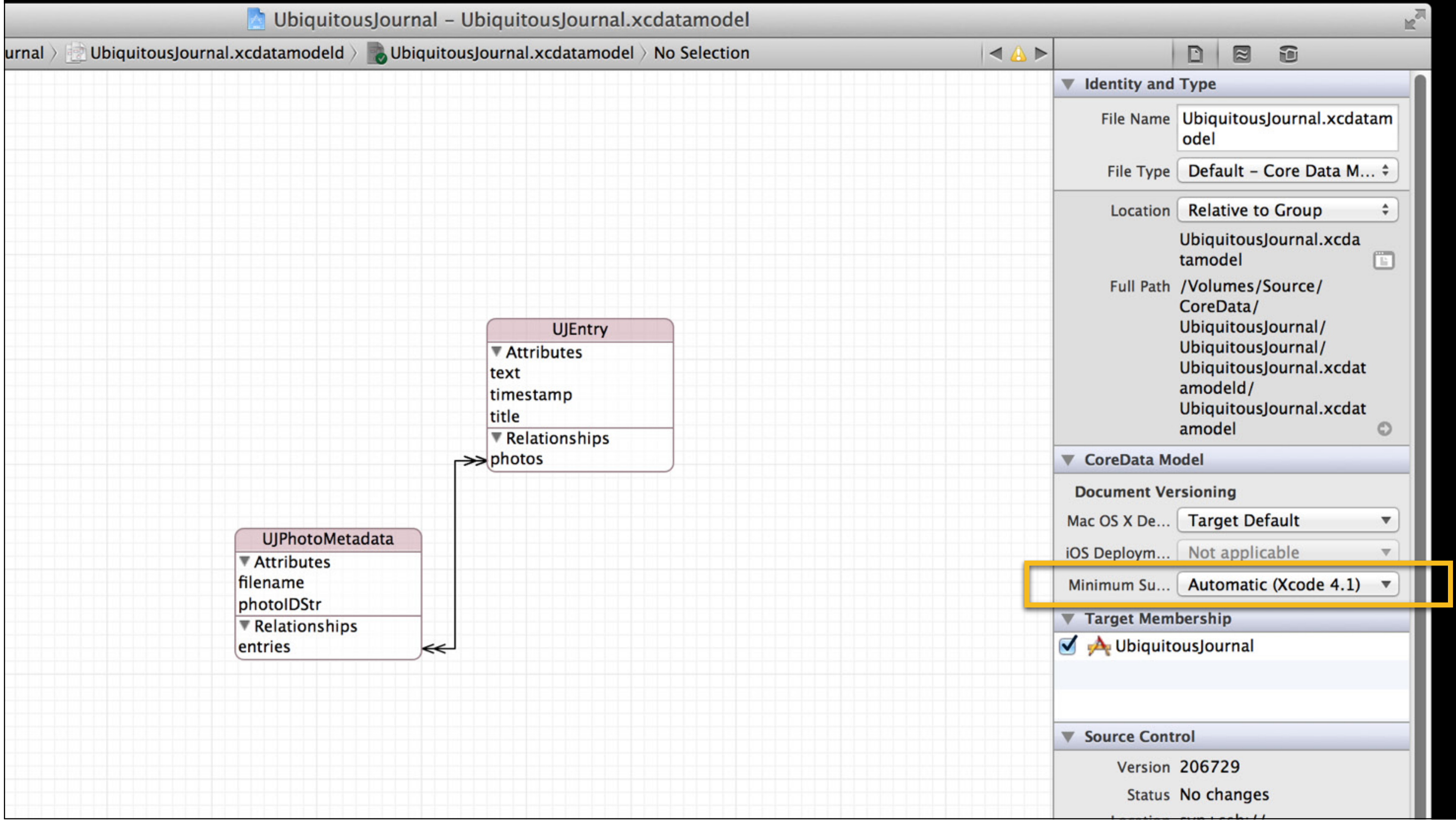
Mac OS X De...	Target Default
iOS Deploym...	Not applicable
Minimum Su...	Automatic (Xcode 4.1)

Target Membership
- [✓] UbiquitousJournal

Source Control

Version	206729
Status	No changes
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

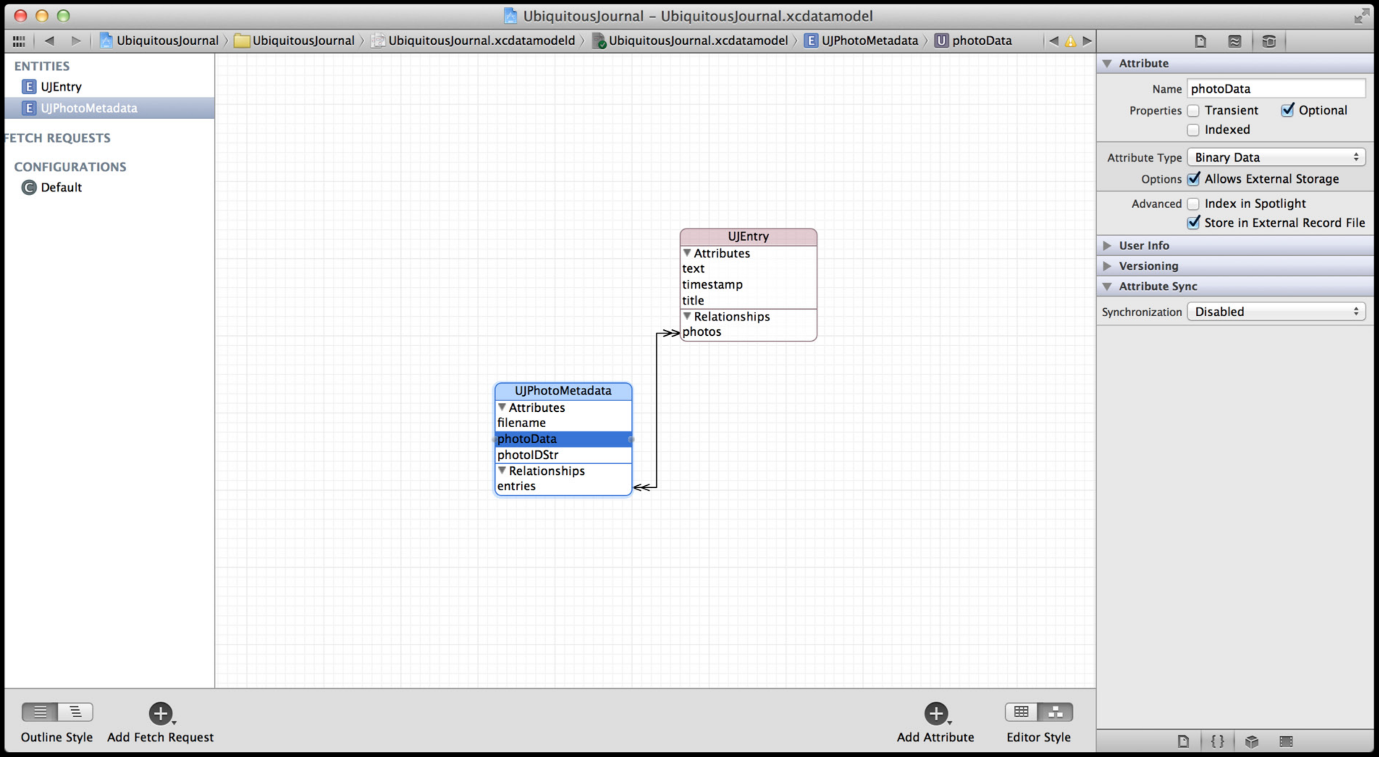
Scalar Accessors

- 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

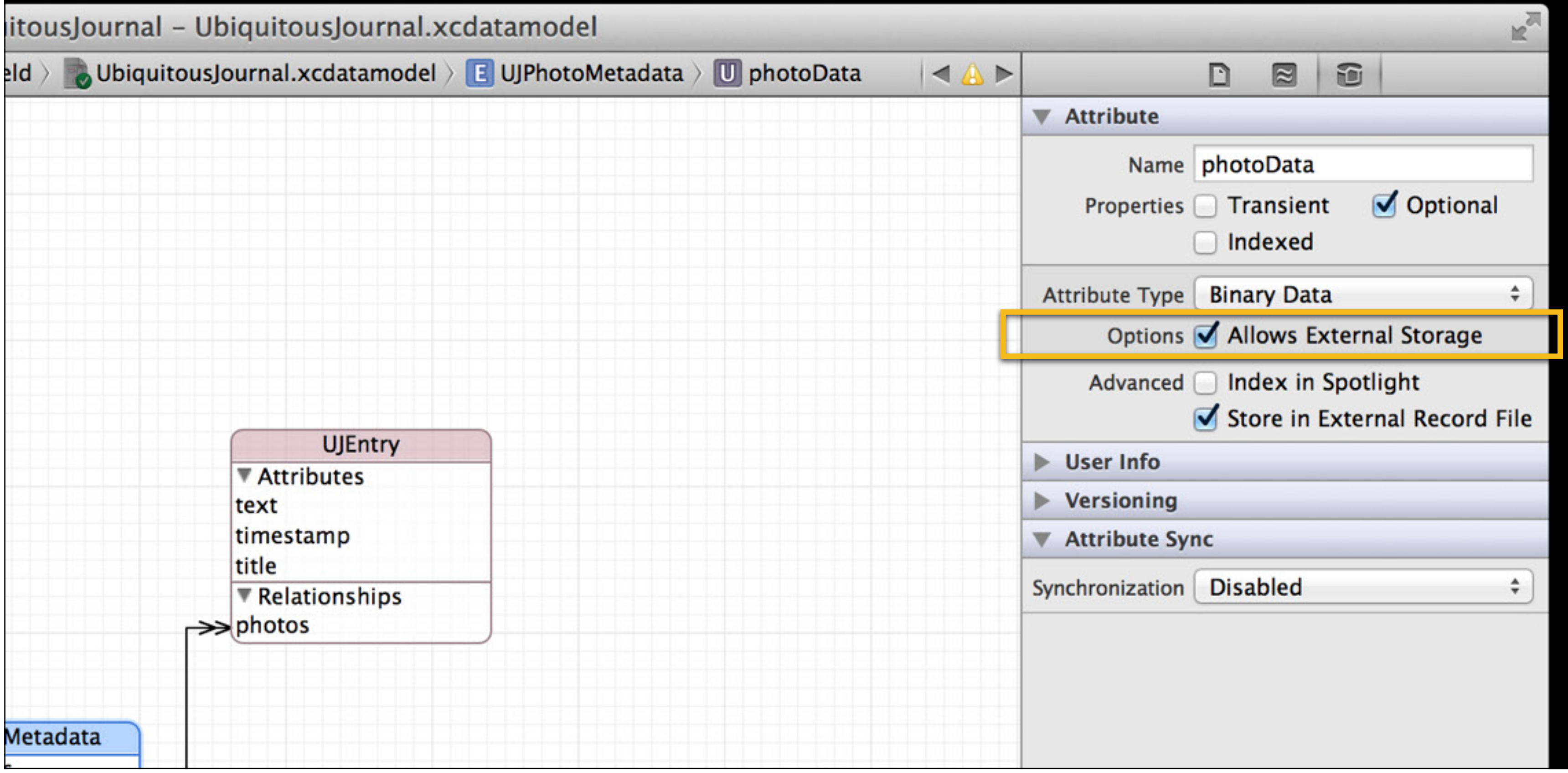
Attribute
- Name: photoData
- Properties: [ ] Transient [✓ Optional] [ ] Indexed
- Attribute Type: Binary Data
- Options: [✓ Allows External Storage]
- Advanced: [ ] Index in Spotlight [✓ Store in External Record File]

User Info

Versioning

Attribute Sync
- Synchronization: Disabled

External Binary Data



itousJournal – UbiquitousJournal.xcdatamodel

eld 〉 UbiquitousJournal.xcdatamodel 〉 E UJPhotoMetadata 〉 U photoData

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

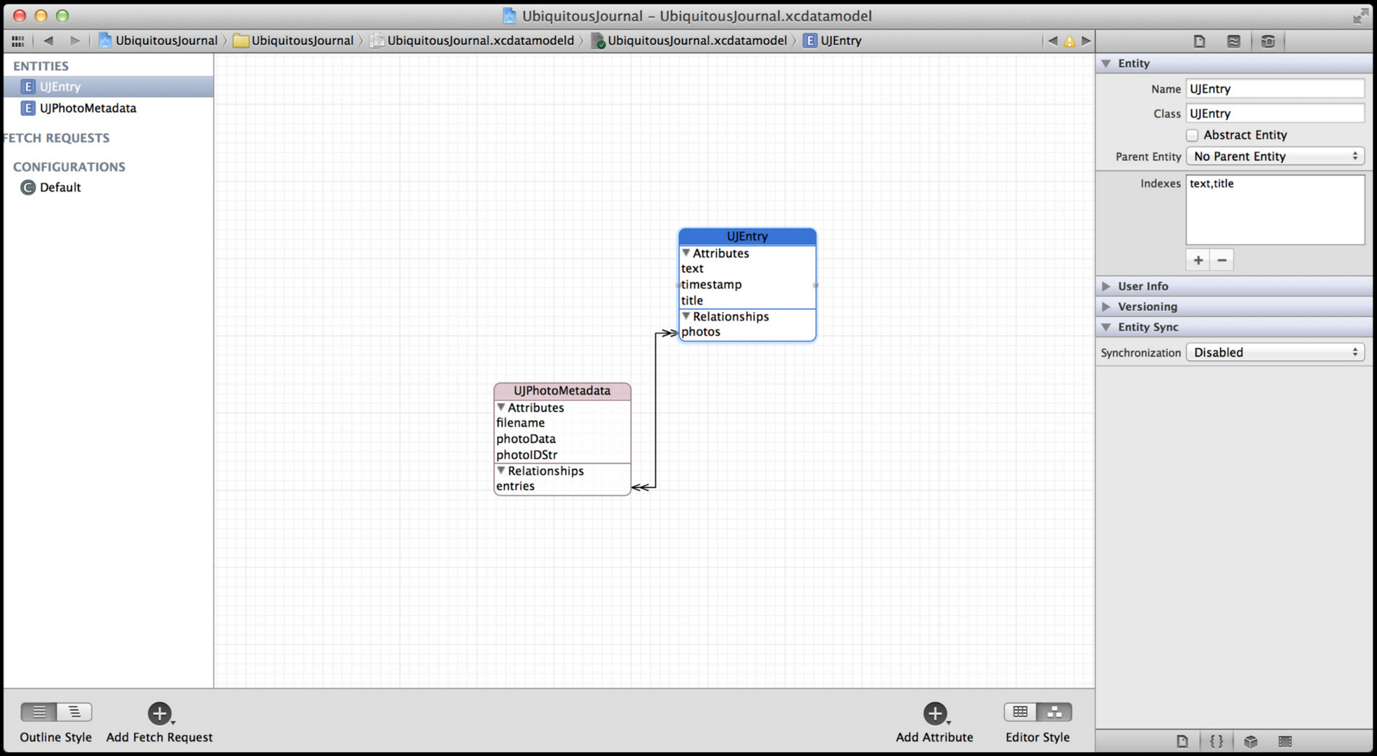
Attribute Sync

Synchronization: Disabled

Compound Indexes

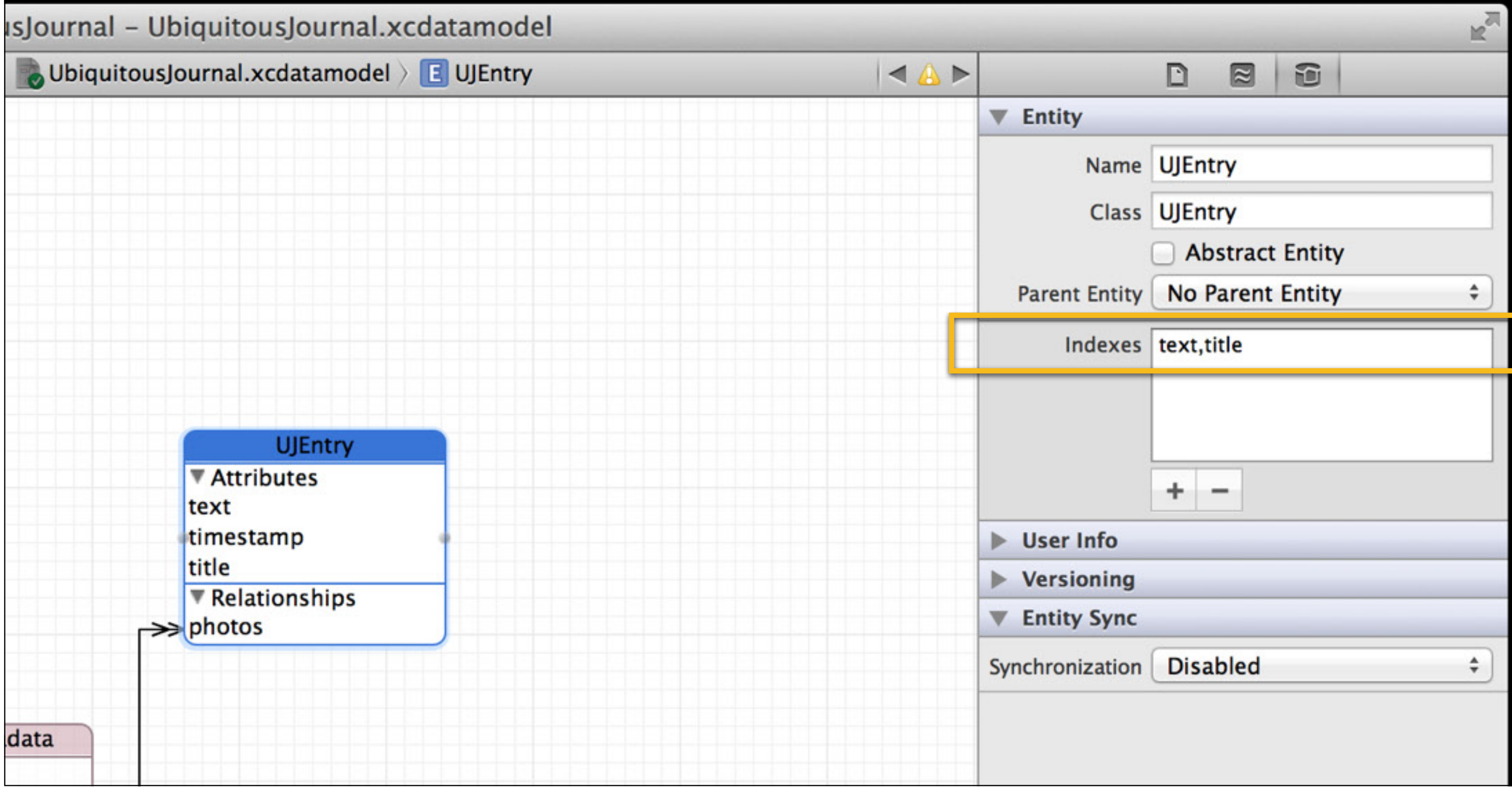
- Index across multiple properties
- Supported by SQLite store

Compound Indexes



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

Compound Indexes


sJournal – UbiquitousJournal.xcdatamodel

UbiquitousJournal.xcdatamodel 〉 E UJEntry

UJEntry
▼ Attributes
text
timestamp
title
▼ Relationships
photos

data

▼ Entity

Name	UJEntry
Class	UJEntry
	☐ Abstract Entity
Parent Entity	No Parent Entity
Indexes	text,title

▶ User Info
▶ Versioning
▼ Entity Sync

Synchronization: Disabled

Summary

- Amazing new iOS 5 features
  - iCloud
  - Documents
  - Data Protection
  - Incremental Stores
  - Ordered Relationships
- Feedback: forums, bug reports, enhancement requests

http://bugreport.apple.com

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

What's New in Core Data on Mac OS X	Nob Hill Thursday 11:30AM
Storing Documents in iCloud Using iOS 5	Presidio Wednesday 3:15PM
Taking Advantage of File Coordination	Pacific Heights Tuesday 4:30PM
Introducing Automatic Reference Counting	Presidio Tuesday 4:30PM

Earlier Sessions

iCloud Storage Overview	Presidio Tuesday 11:30AM

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