Thoughts on Object Graph Storage

Louis Gerbarg
GLsoft.mobi

I've never got it when it comes to SQL databases. It's like, why? Just give me a hash table and a shitload of RAM and I'm happy. And then you do something to deal with failures.

James Gosling

What is an Object Graph?

- A series of objects that point to each other
  - Almost everything you deal with in Cocoa is part of a graph
- We usually only really care about graphs that need to be saved (part of the model)

What is an Object Graph?

- In the beginning there was plist
  - It worked okay, but required atomic serialization/deserialization
- Then came CoreData
  - Partial graph writes improved responsiveness
  - Lost some of the "graphiness"

Missing "Graphiness"

- Object graphs should not require queries
- Object graphs should not require users to worry about things like fetch limits
- Object graphs should use native containers and allow idiomatic language constructs for things like iteration

Missing "Graphiness"

- Missing graph features in CoreData are replaced with ORM features by leaking the underlying architecture up through the API
- Encourages people to try to use CoreData like an ORM
- Causes performance issues because CoreData is not an ORM

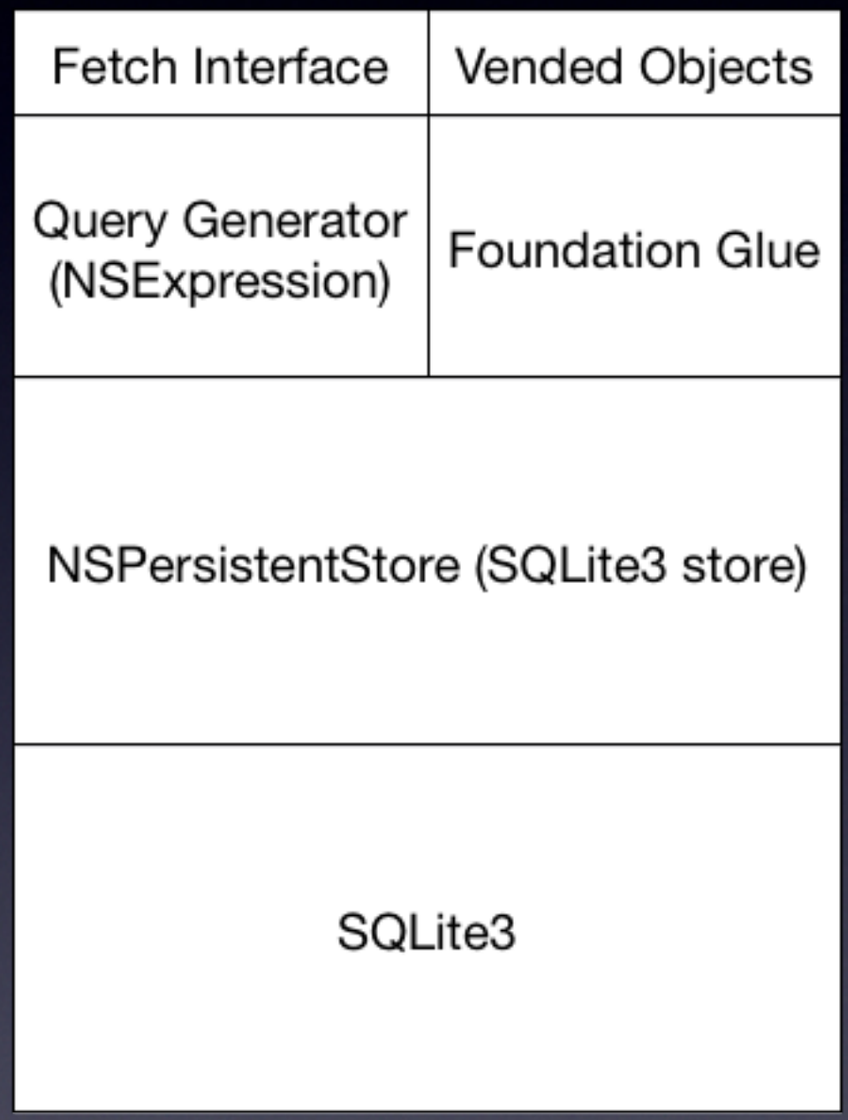
CoreData Architecture

- Applications interface to a store via a context
- Context translates property and relationship accesses into expressions
- Store converts expressions into SQL
- SQLite gets data from the disk

CoreData Architecture

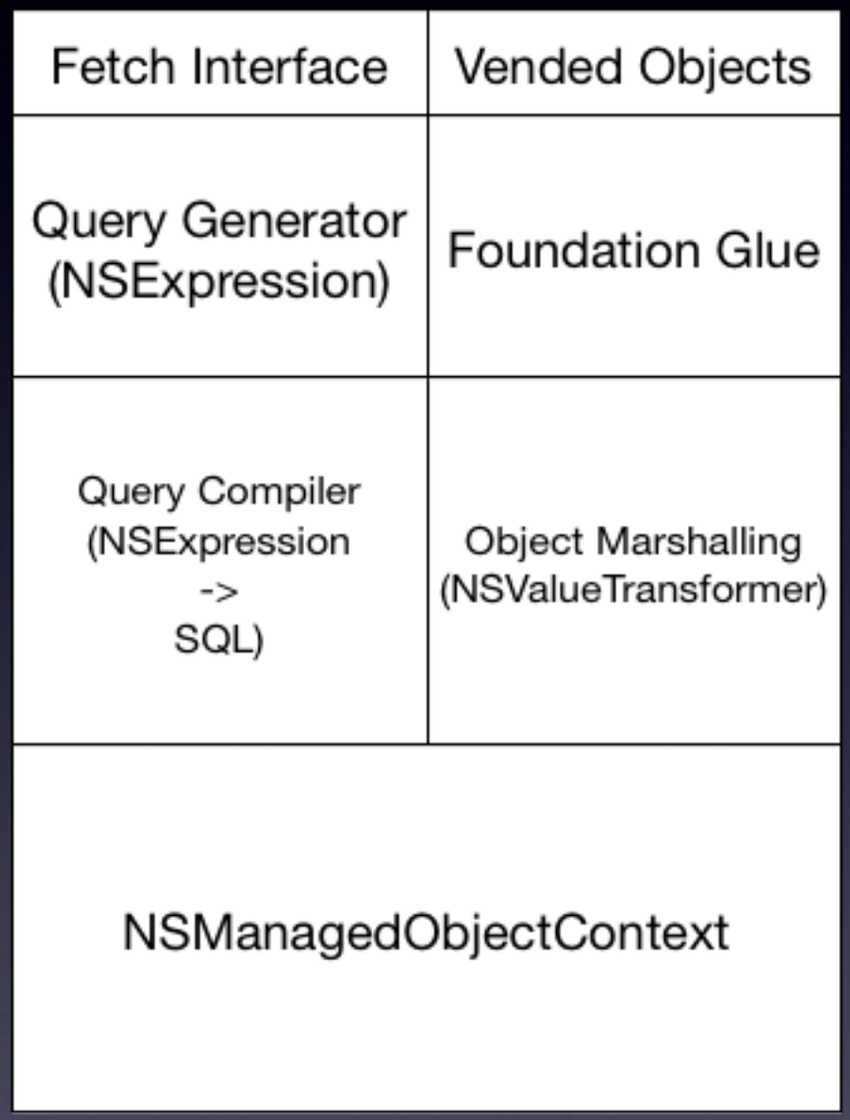
- In order to get vended objects queries need to be generated in order to fault in various bits

- Objects dynamically create accessors that tie those generated queries to standard properties or NSSets



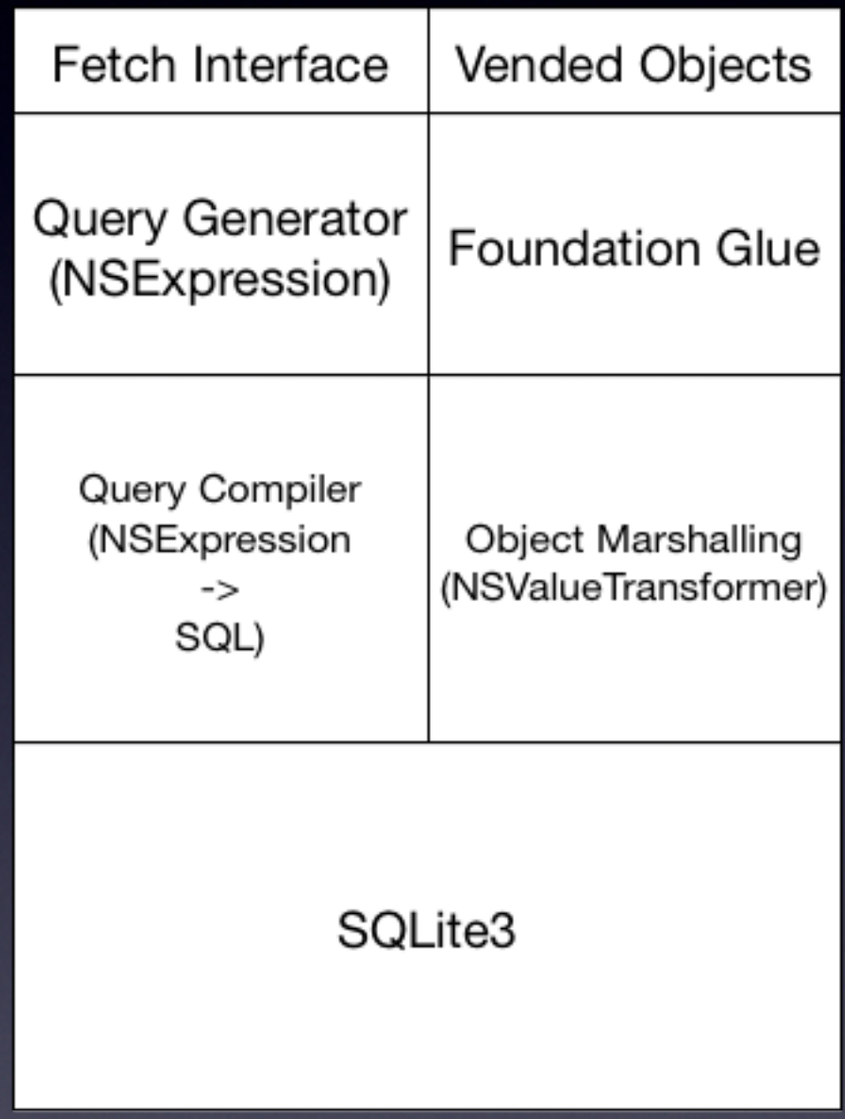
CoreData Architecture

- SQL store takes those built expressions and recompiles them into SQL commands.
- SQLite does not support all of Cocoa's types, so there are built in converters as well as loadable ones to marshall data



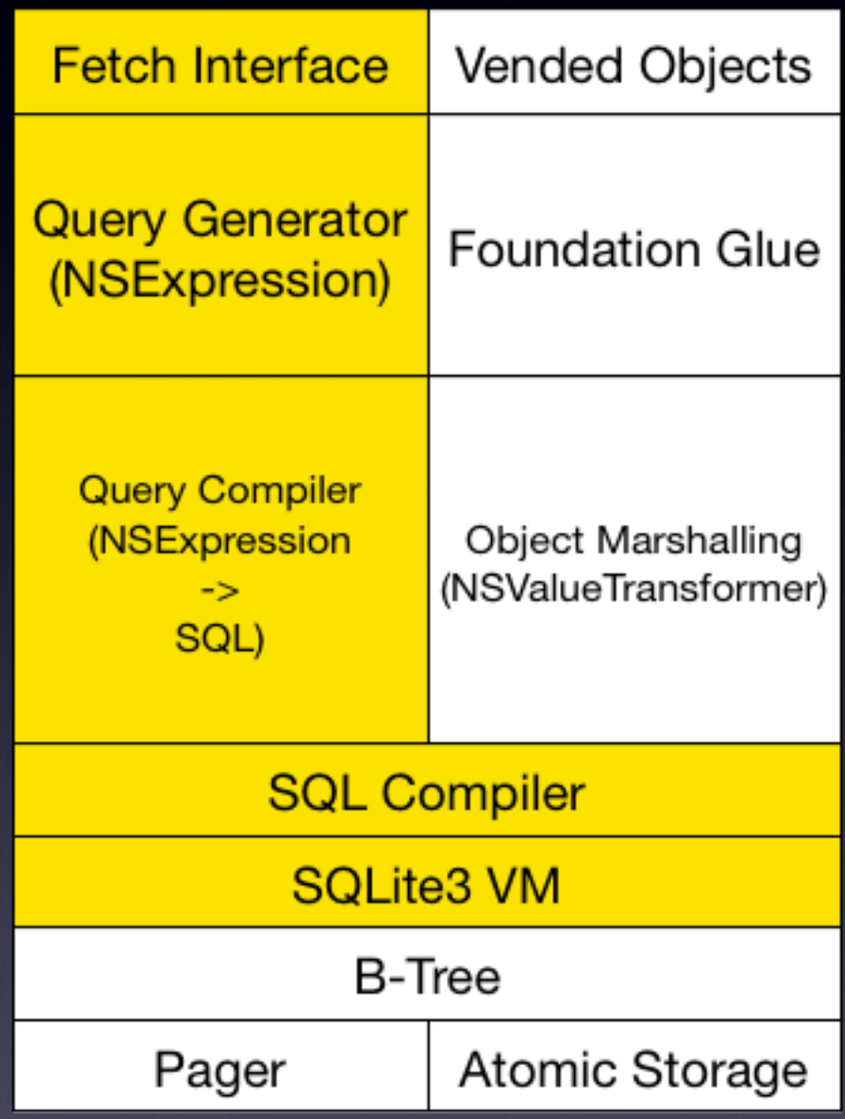
CoreData Architecture

- SQLite 3 compiles SQL into internal byte codes
- Has an internal VM to run those byte codes
- Implements B-Trees atop a replaceable atomic storage layer



Telescope it!

- Let's rearrange the internal pieces without worrying about the layer boundaries
- First off look how much effort is spent handling SQL
  - Only exposed to compensate for missing Graphiness
  - Let's get rid of it

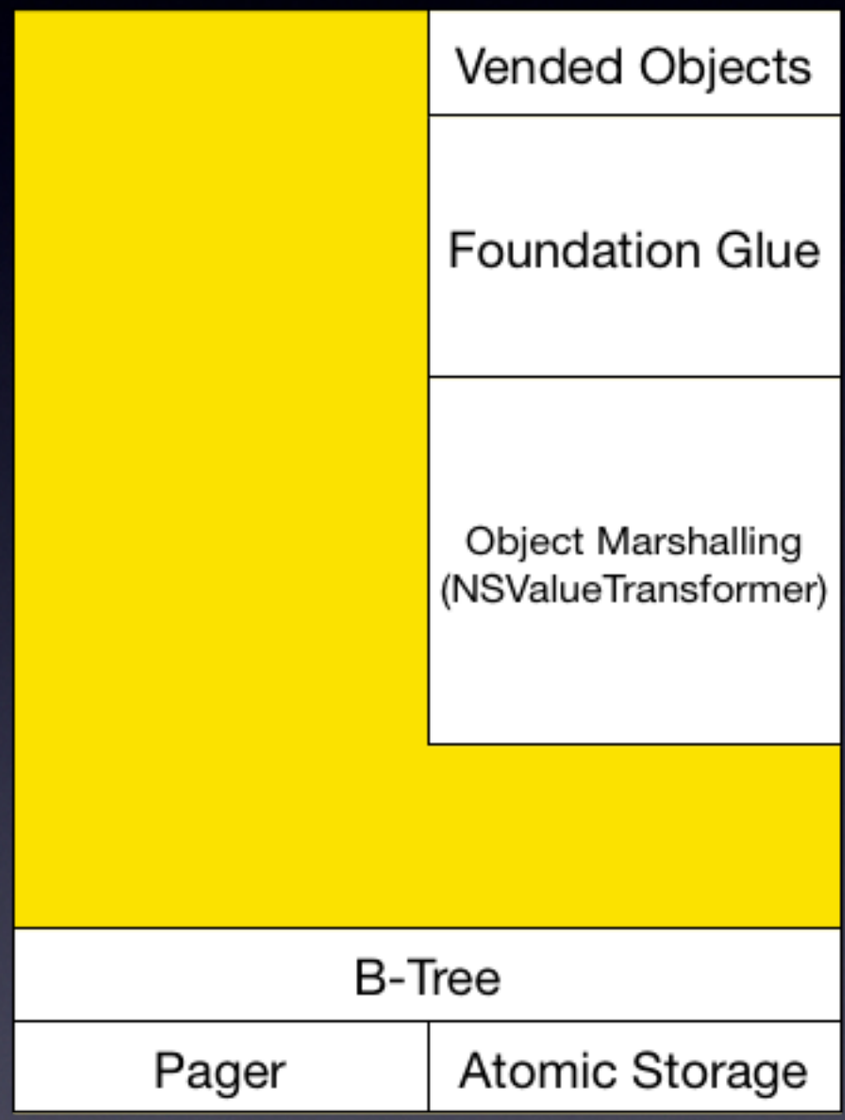


Telescope it!

- Foundation glue and object marshaling will need to expand

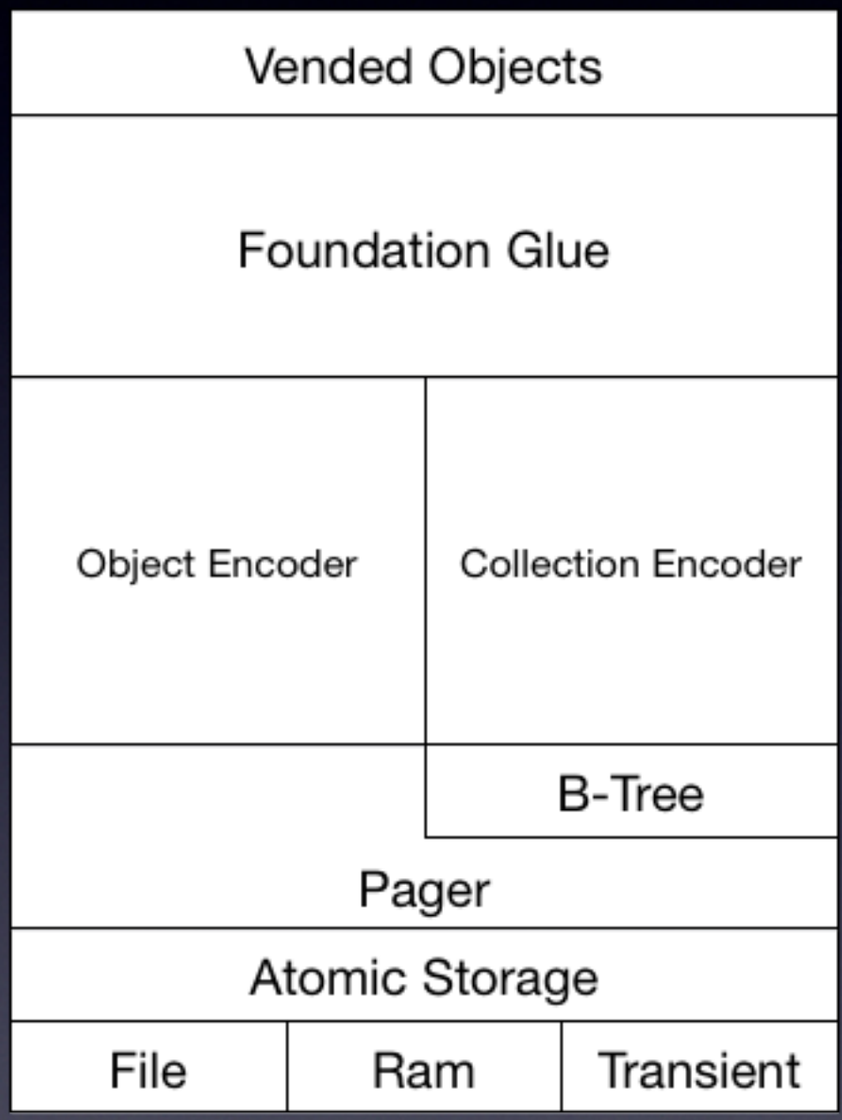
- Requires new object marshaling code that talks to B-trees and implements collections

- Direct pager access for non-collection objects



Telescope it

- New atomic storage layer
- Based on SHA224 hashes
  - Similar to git, hg, bzr
- Supports centralized, p2p synch with minimal client code
- Supports in ram commits
- Supports transient commits



GraphObjects

- Features
  - Self describing types
  - Immutable storage
  - Graph synchronization
  - Fast saves
  - Transient saves when low on RAM

- Types
  - NSString
  - NSNumber
  - NSDate
  - NSURL
  - NSData
  - Scalar integers
- Collections
  - NSMutableSet
  - NSMutableArray
  - NSMutableDictionary

GraphObjects

- Tech preview available today
  - Very raw
  - Expect a final version early next year
- Opensource
  - MIT license
  - Does not include sync code

GraphObjects

```
//Basic usage

LGGOGraph *graph = [[LGGOGraph alloc] initWithURL:URL];

LGGOGraphContext *context = [graph newContext];

graph.rootObject = someObject; // plist types and LGGOObjects

NSError *error = nil;

[graph save:&error];
```

GraphObjects

- Available today
  - NSString
  - NSNumber (59 bit integers)
  - Memory store
  - "Hack" arrays
- Next week
  - Custom Objects
    - Embedded scalars
- December
  - B+Trees
    - NSMutableArray
    - NSMutableSet
    - NSMutableDictionary
  - Disk saves (mostly complete in preview)
  - NSDate
  - NSURL
  - NSData

This presentation was made with 100% natural iPad products

May contain materials processed with Keynote and OmniGraffle.