OM5

Goals

- ability to have multiple persistent roots in a store, each with pointer-based undo/redo (i.e. undo is just changing a single value, no diff/merge required). The early ObjectMerging prototypes only support pointer-based undo on the whole repository, or selective undo (diff/merge required) on subsets of it, like git.
- *guaranteed* isolation between persistent roots.

Goals

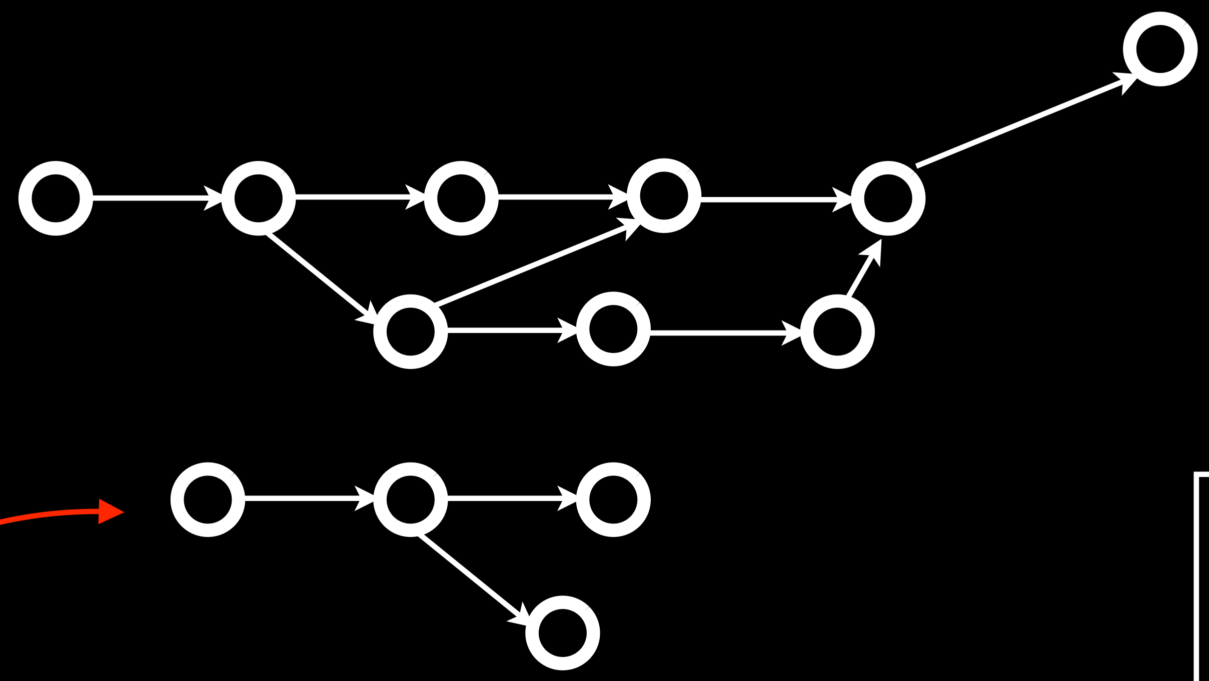
- Unify branch, copy, and persistent root - most users won't use branching. copying documents needs to be the same thing as branching.
- a Branch ≅ a copy of a persistent root
- Persistent root ≅ thin grouping mechanism for branches

Goals

- well-specified data format for objects in the store “property list” is not good enough.

Store Structure

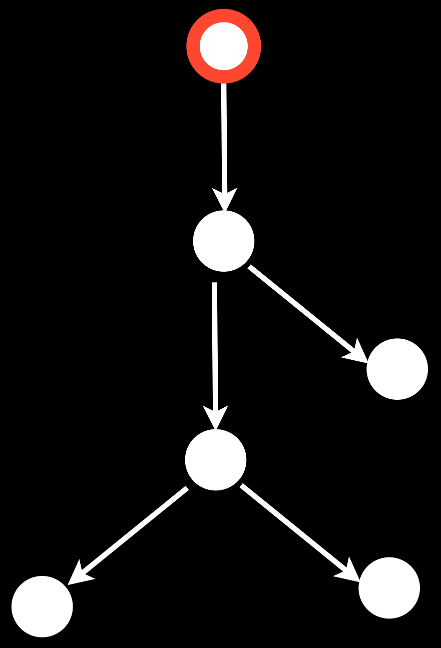
- DAG forest describing the history relationships between commits



The content of the commits in these DAGs is probably unrelated

Commit Structure

- Each commit is identified by a UUID
- A commit contains a tree of Store Items



●	Store item
◉	Root store item

Store Item Structure

- Each store item has a UUID and a set of key/value pairs. Keys are unicode strings.

Store Item Structure

Value type

Primitive

kCOPrimitiveTypeInt64
kCOPrimitiveTypeDouble
kCOPrimitiveTypeString
kCOPrimitiveTypeFullTextIndexableString
kCOPrimitiveTypeBlob
kCOPrimitiveTypeCommitUUID
kCOPrimitiveTypePath
kCOPrimitiveTypeEmbeddedItem

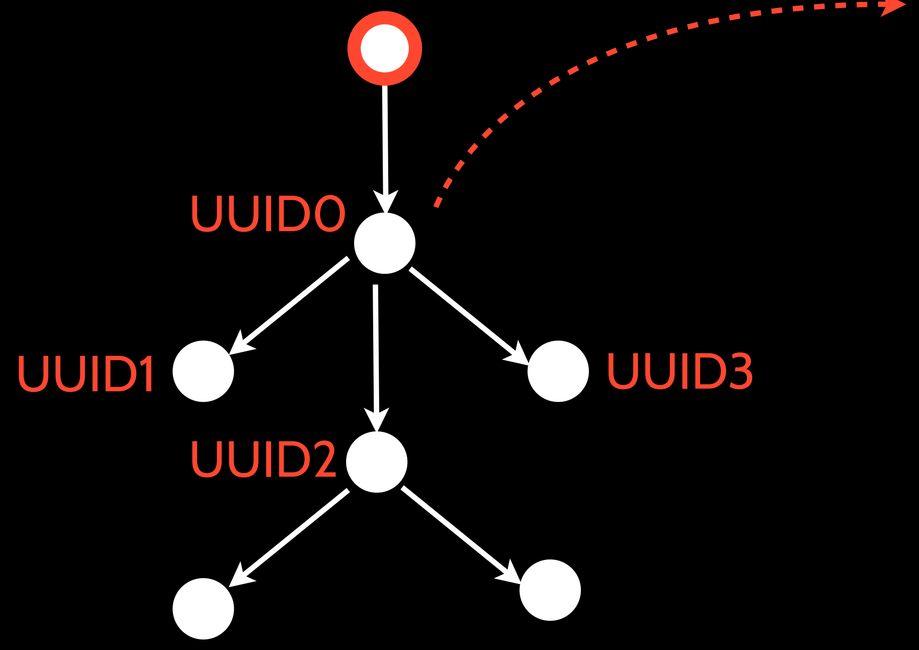
Multivalued

a primitive type plus the following flags:

kCOContainerOrdered	YES/NO
kCOContainerAllowsDuplicates	YES/NO

Commit Structure

- The tree structure of store items is defined by values of type kCOPrimitiveTypeEmbeddedItem

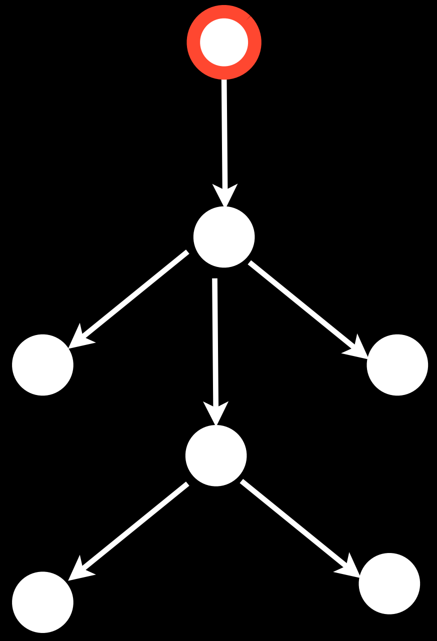


UUID0

attribute	value	type
name	"My Documents"	String
icon	UUID1	Embedded Item
contents	(UUID2, UUID3)	(Unordered Unique Container of Embedded Item)

Commit Structure

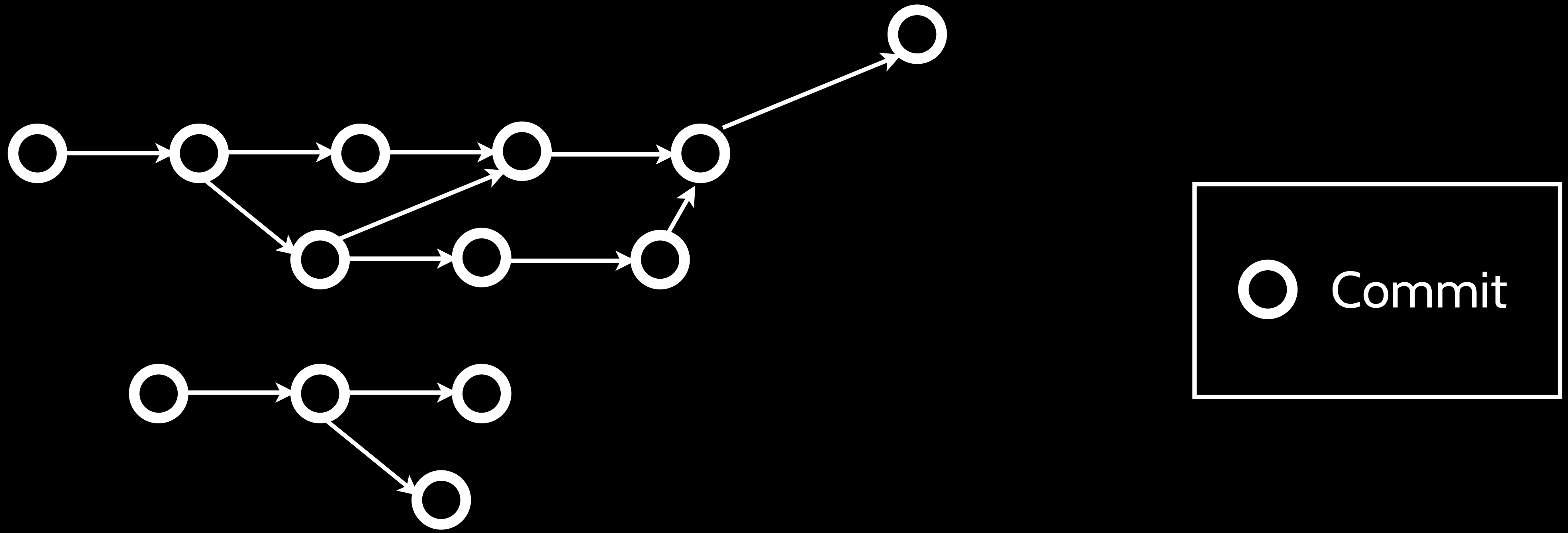
- The set of items in a commit is defined by looking at the root item and including all of its Embedded Items, and those items' embedded items, etc.



- ⇒ no "floating" items allowed
- It is illegal for the same item to be Embedded in multiple places

Store Structure

- Now we can store item trees in commits, organized by their history relationship

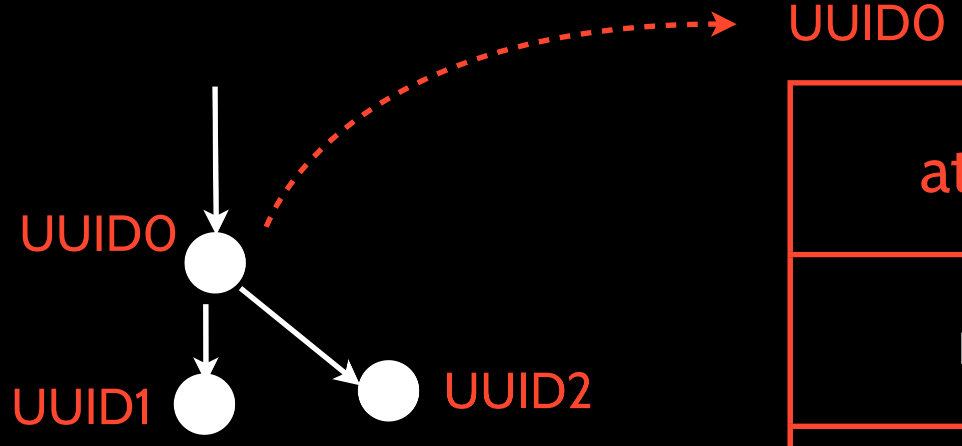


- How do we represent the "current state" of the store? Persistent roots? Pointer-based undo?

Persistent Root

- Just a tree of store items with a known structure/ interpretation.
- Chosen to give us all of the properties we want...
  - thin grouping mechanism for branches, which can be copied in/out trivially.
  - Copying a branch/persistent root has the desired semantics (copy can be subsequently modified without affecting the source) "for free"

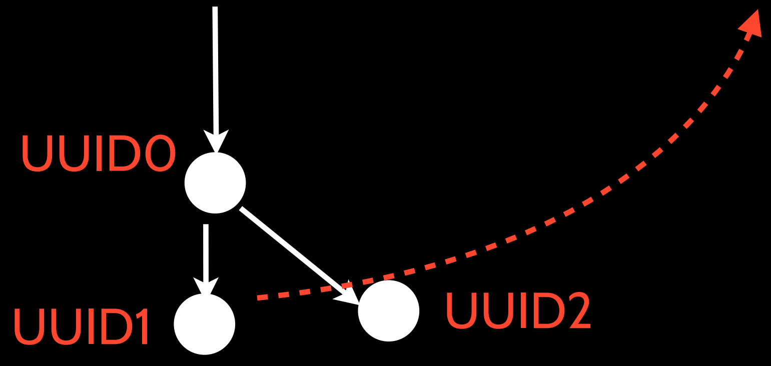
Persistent Root Example



attribute	value	type
name	"My Documents"	String
contents	(UUID1, UUID2)	(Unordered Unique Container of Embedded Item)
type	"persistentRoot"	String
currentBranch	UUID1	Path (a weak reference)

This identifies a persistent root called "My Documents" with two branches. The current branch is UUID1.

Persistent Root Example



UUID1

attribute	value	type
name	“Default Branch”	String
type	“branch”	String
currentVersion	UUID X	Commit UUID
head	UUID Y	Commit UUID
tail	UUID Z	Commit UUID

These are implementation details of undo/redo {

This is the important part... it says that the contents of the persistent are stored in the commit with UUID X