THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

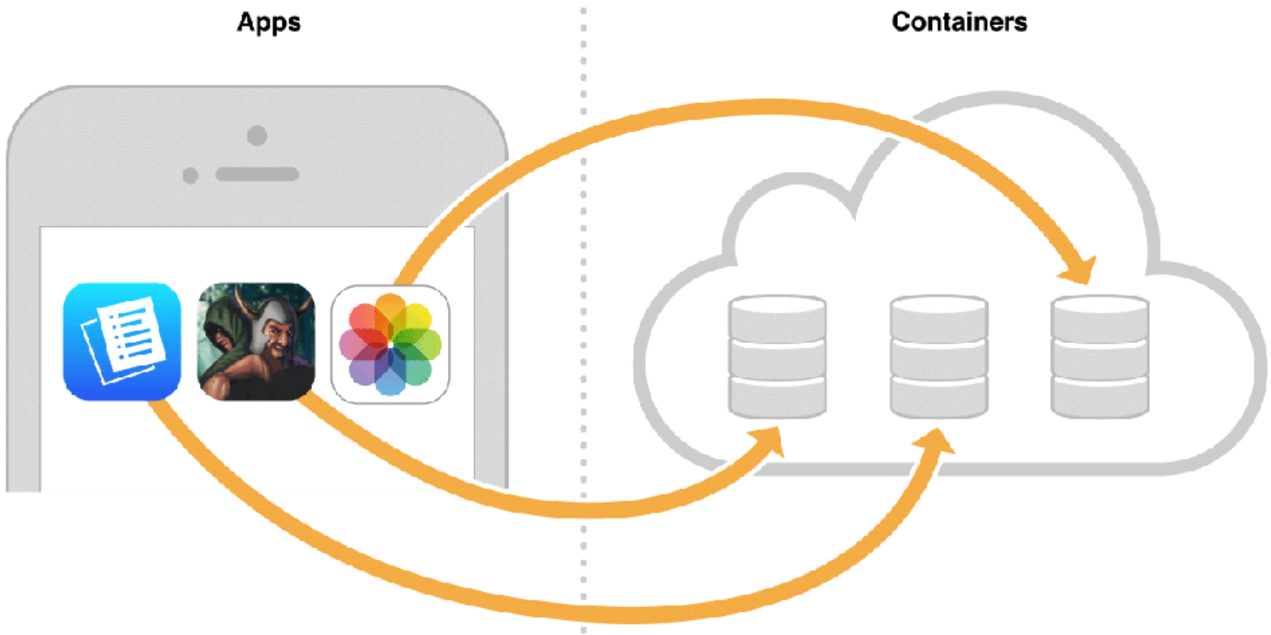
MPCS 51032 • SPRING 2020 • SESSION 3

CONTAINER

CLOUDKIT OBJECTS

CONTAINER

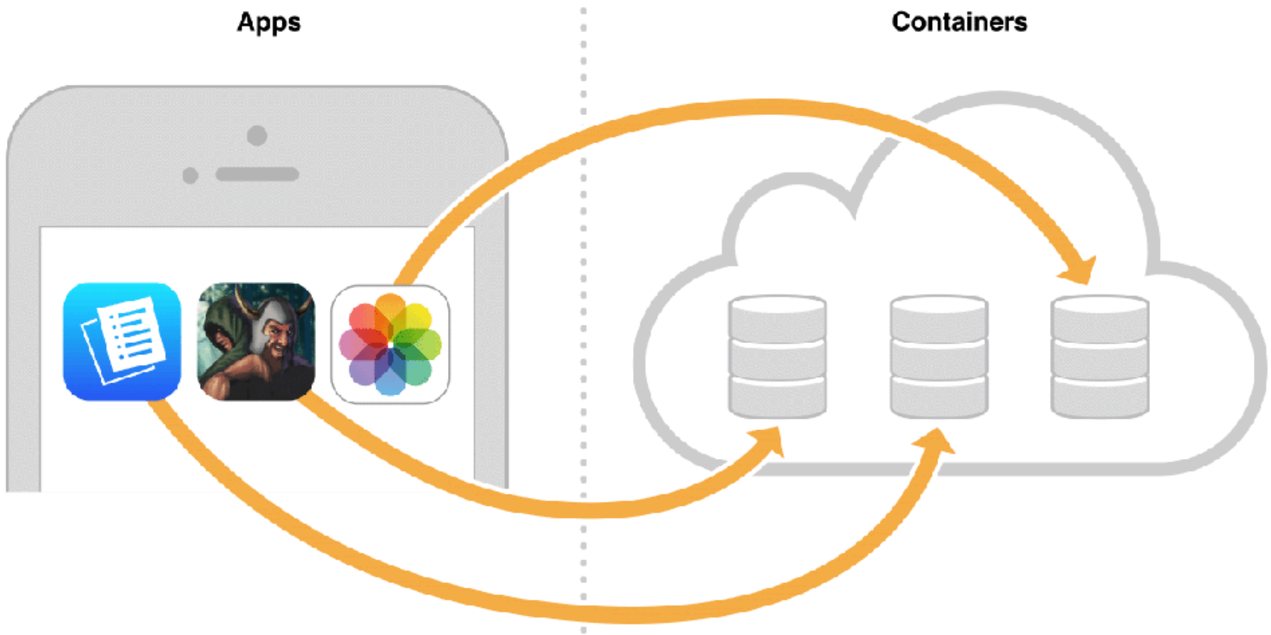
- CKContainer
  - One default container per application
  - Data segregation
  - User encapsulation
  - Managed by the developer via portal



CLOUDKIT OBJECTS

CONTAINER

- CKContainer can be shared between apps from the same developer
- You can create additional custom containers for you app



CLOUDKIT

CONTAINER

```
import CloudKit
```

IT'S JUST THAT EASY

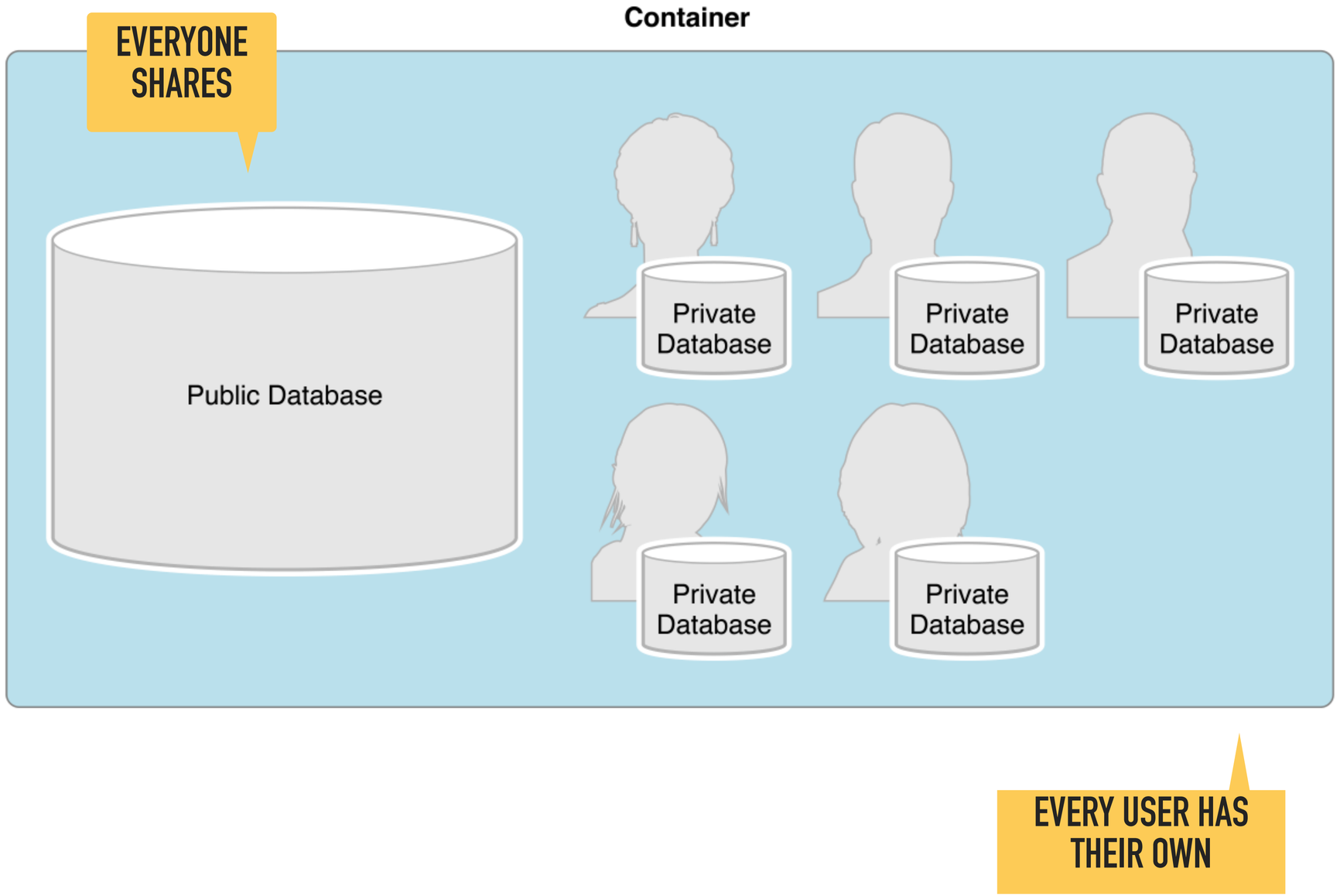
```
let container: CKContainer = CKContainer.default()
```

DATABASE

CLOUDKIT OBJECTS

DATABASE

- CKDatabase
  - Public
  - Private
  - Shared



CLOUDKIT OBJECTS

DATABASE

```
/// Container
let container: CKContainer = CKContainer.default()

/// Databases
let publicDB: CKDatabase = CKContainer.default().publicCloudDatabase
let privateDB: CKDatabase = CKContainer.default().privateCloudDatabase
```

CLOUDKIT OBJECTS

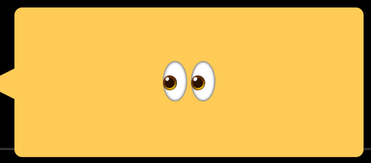
DATABASE

	Public Database	Private Database
Data Type	Shared Data	Current User's Data
Account	Required for Writing	Required
Quota	Developer	User
Default Permissions	World Readable	User Readable
Editing Permissions	iCloud Dashboard Roles	N/A

CLOUDKIT OBJECTS

DATABASE

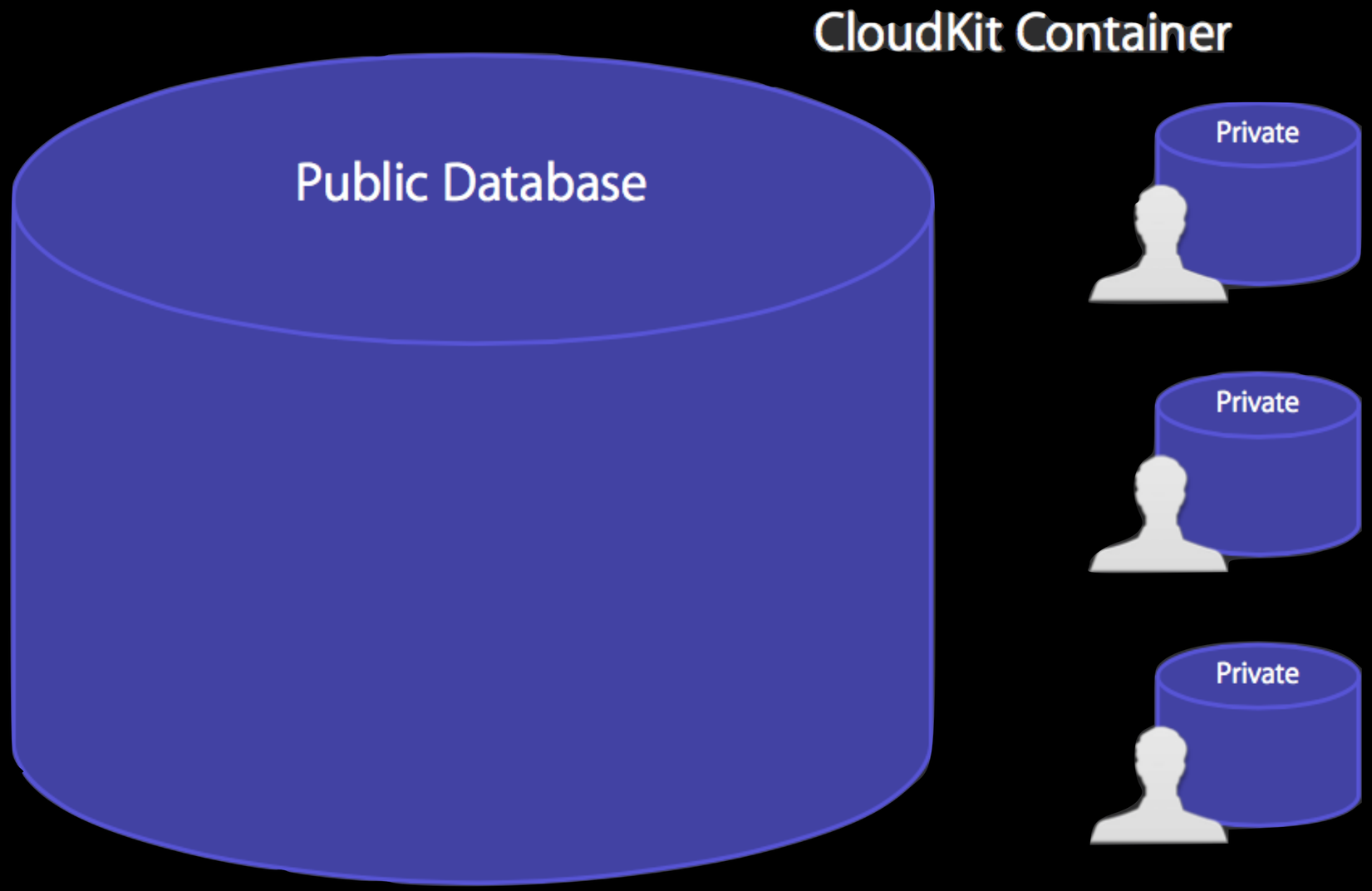
	Public Database	Private Database
Data Type	Shared Data	Current User's Data
Account	Required for Writing	Required
Quota	Developer	User
Default Permissions	World Readable	User Readable
Editing Permissions	iCloud Dashboard Roles	N/A



CLOUDKIT OBJECTS

DATABASE

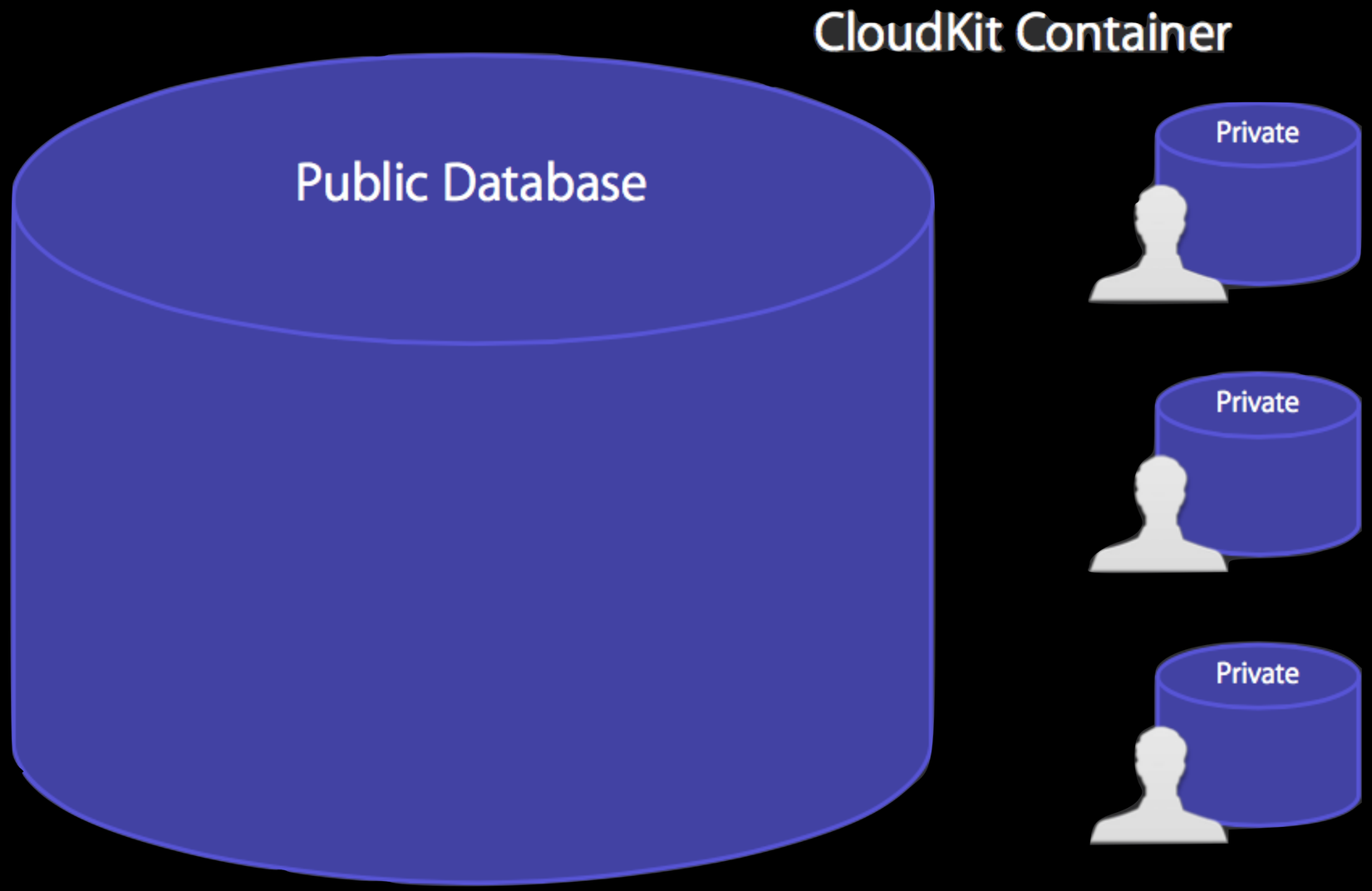
- Public databases
  - No atomic writes
  - No delta downloads
  - Cross zone references



CLOUDKIT OBJECTS

DATABASE

- Private databases
  - Atomic writes
  - Delta downloads
  - No cross-zone references

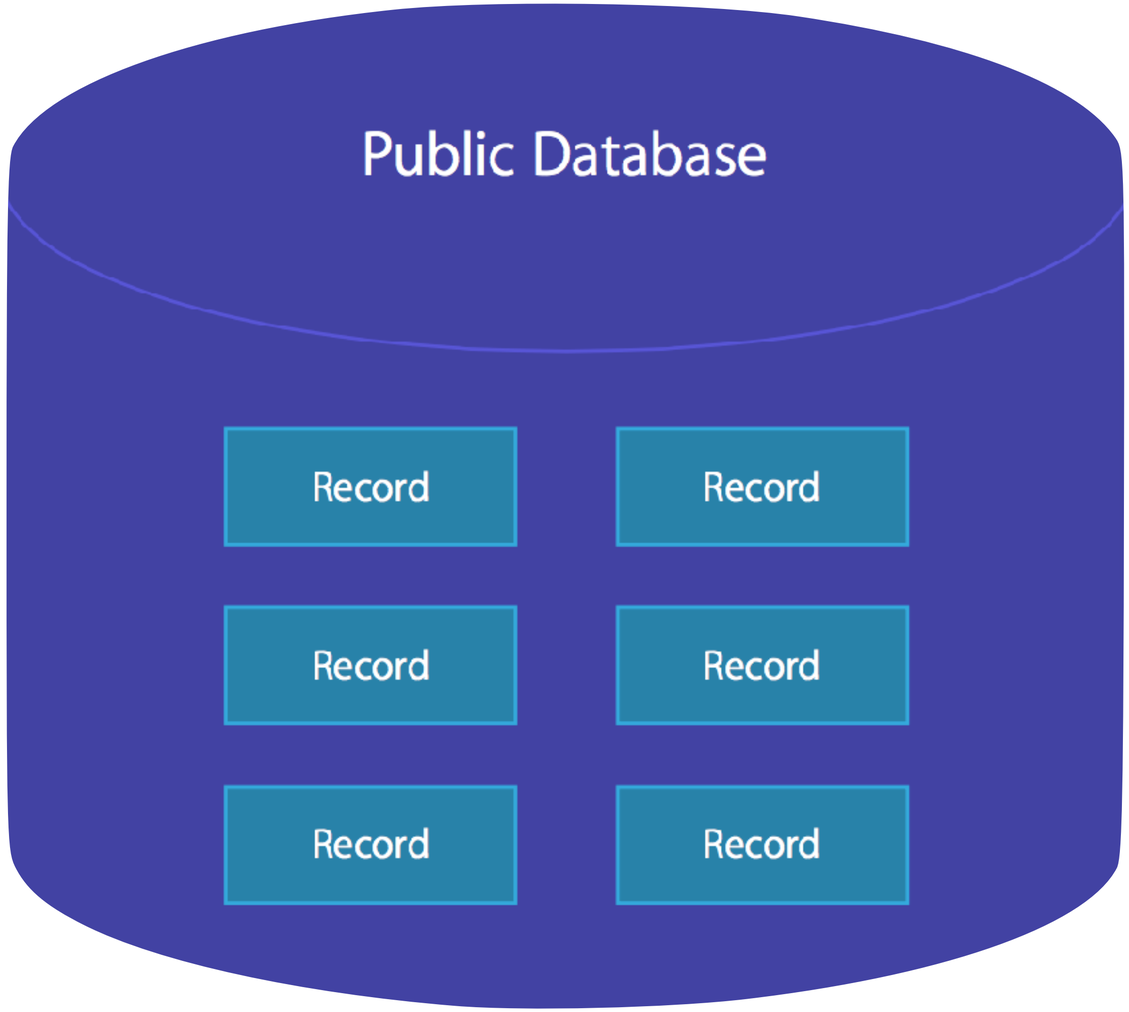


CLOUDKIT OBJECTS

DATABASE

SUBTITLE

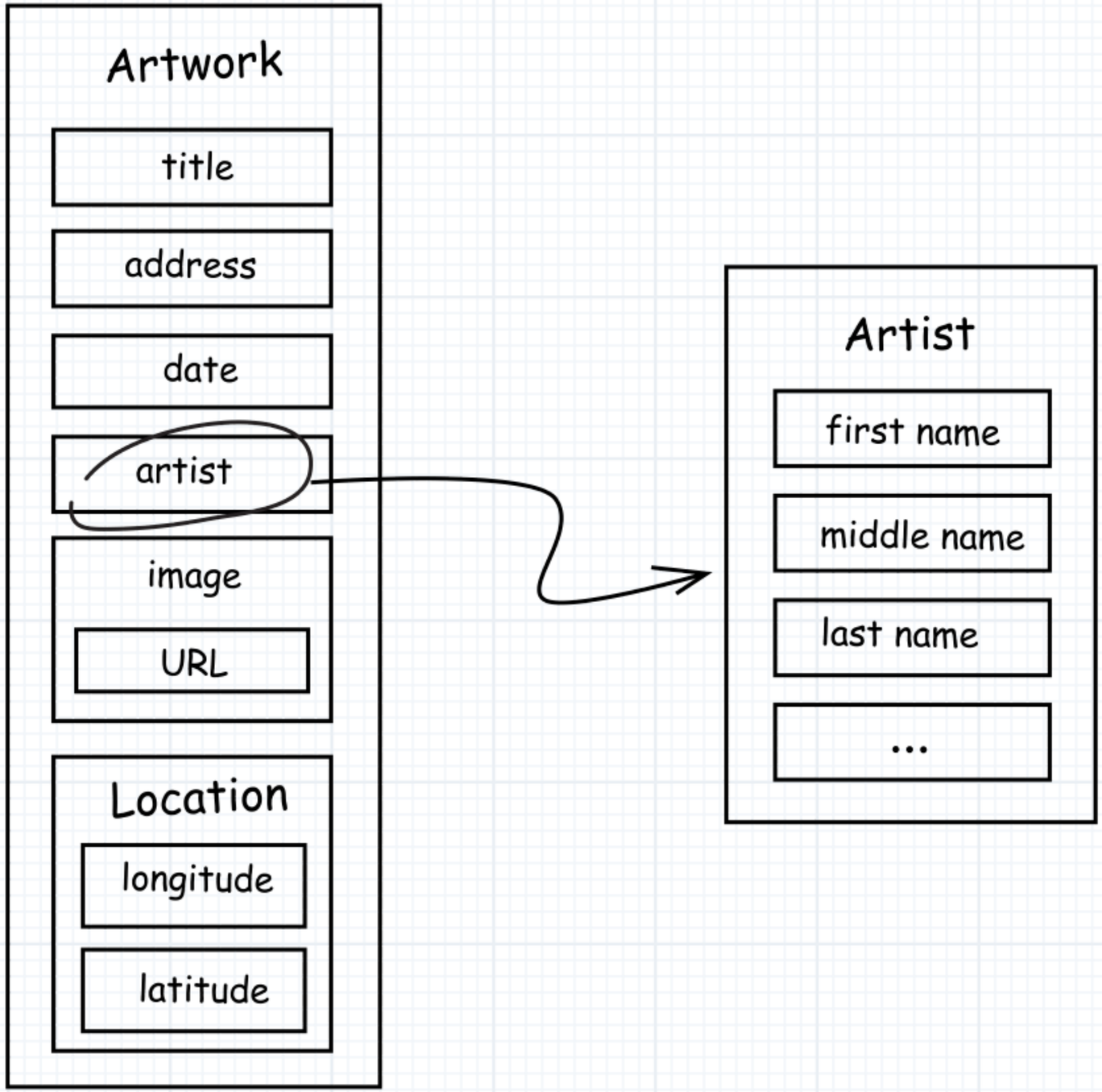
- Structured Data
  - Wraps key/value pairs
  - Types and references
  - Just-in-time schema
  - Metadata
    - Created, edited, etc.



CLOUDKIT OBJECTS

DATABASE

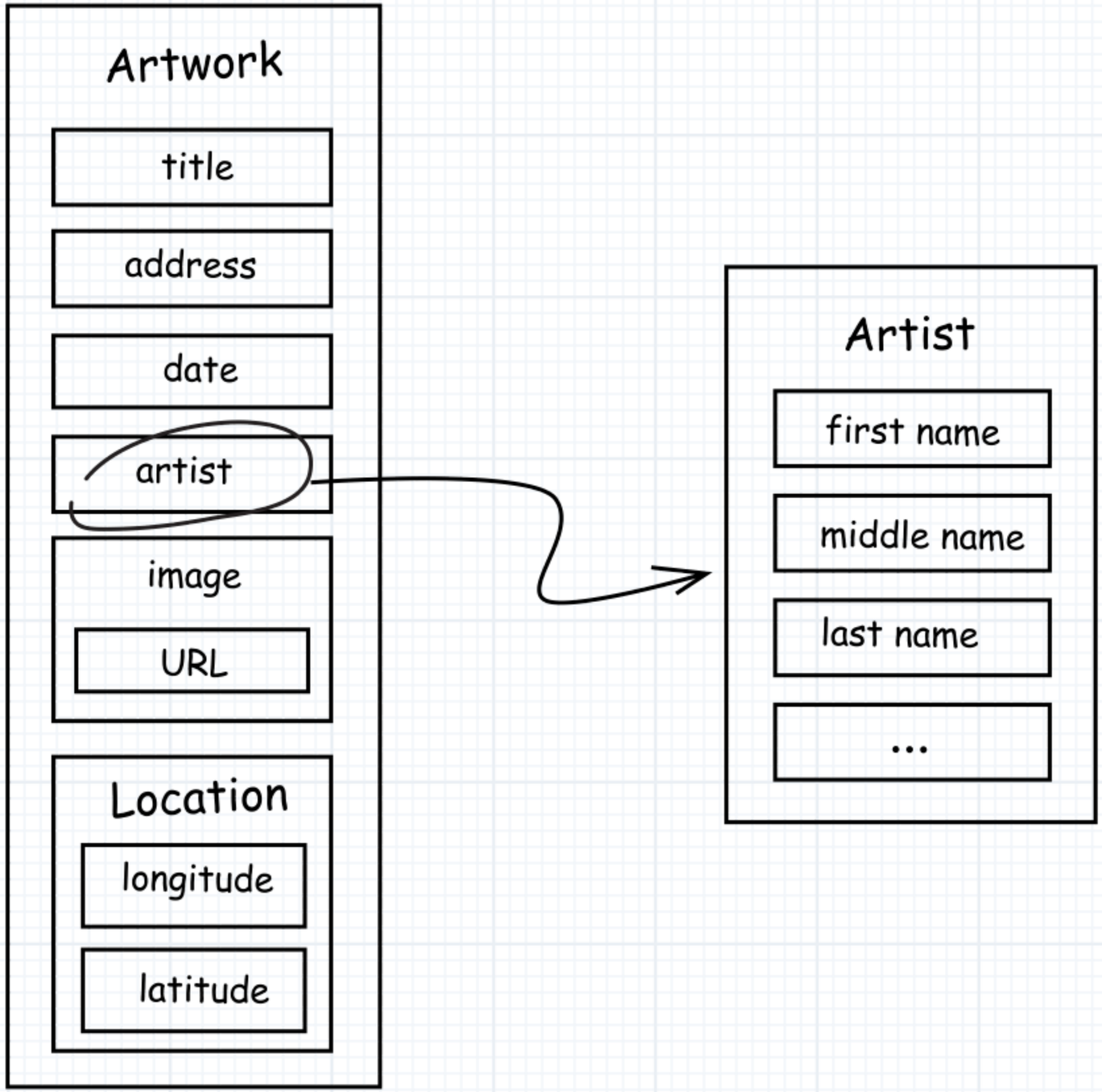
- Schema is built on-demand
- You can edit it in the console



CLOUDKIT OBJECTS

DATABASE

- Changing it may be painful
  - Requires coordination between device and console

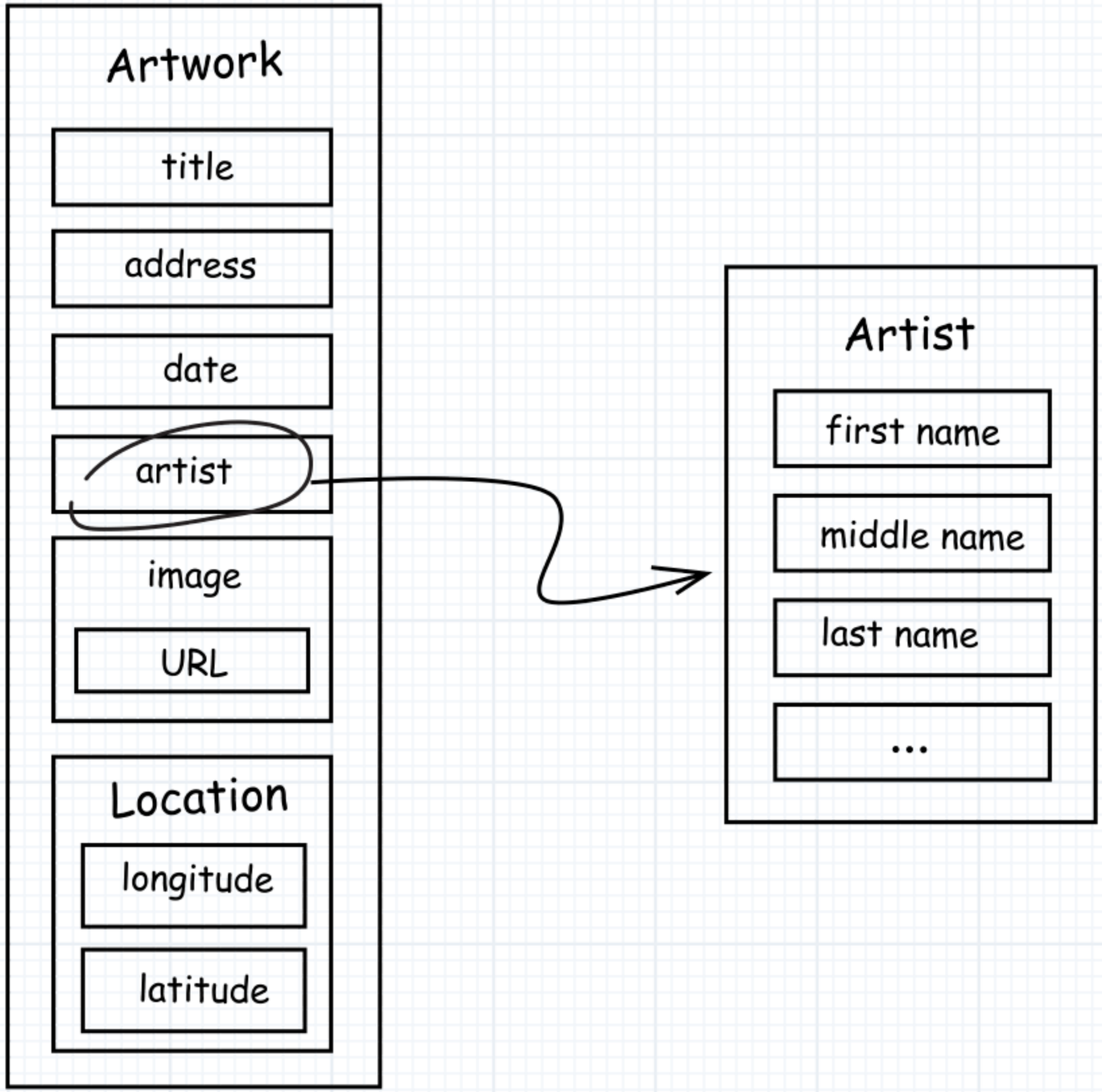


RECORDS

CLOUDKIT OBJECTS

RECORDS

- Record Values
  - String
  - Number
  - Data
  - Date
  - CLLocation
  - CKReference
  - CKAsset
  - Arrays of the above



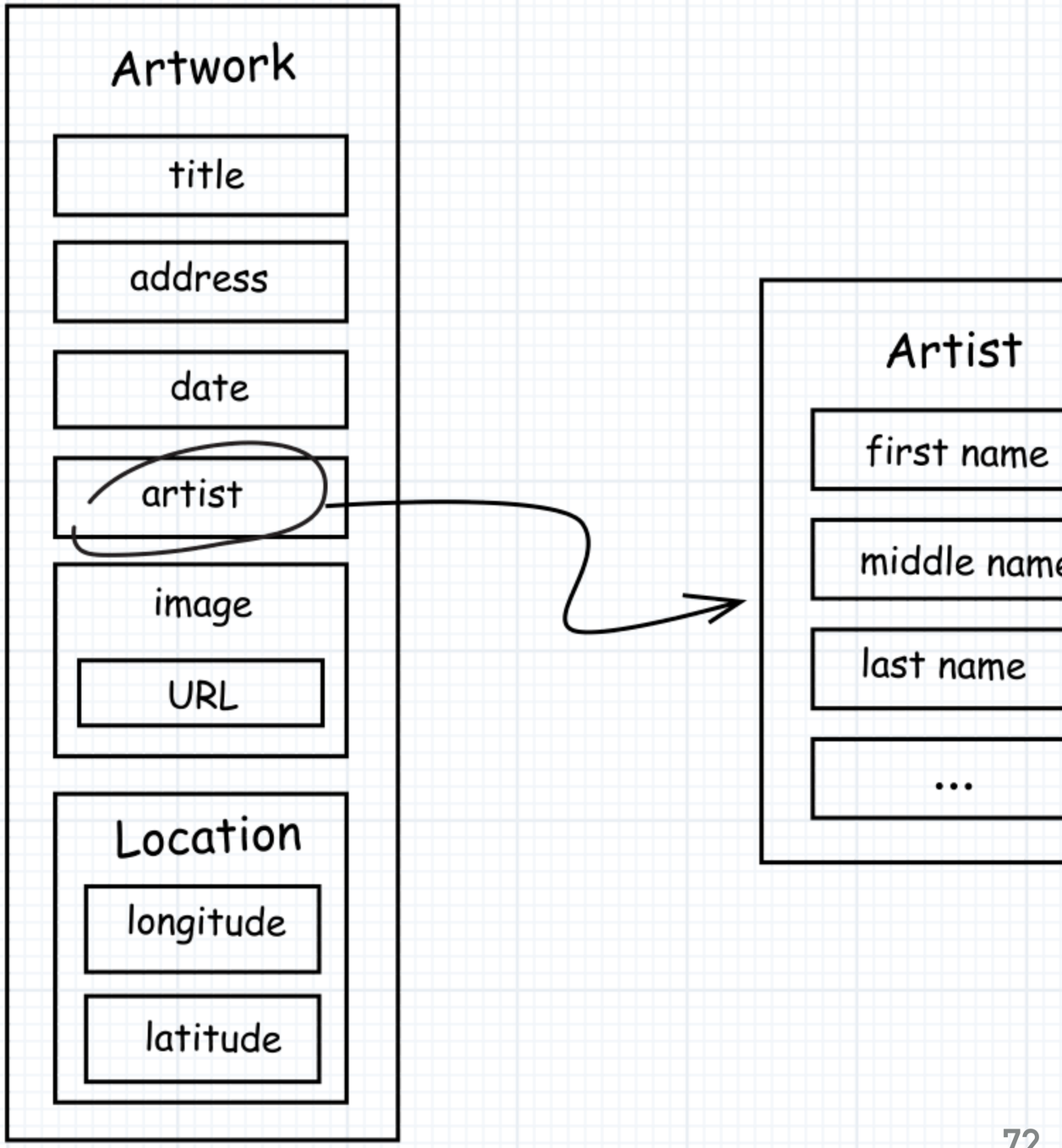
CLOUDKIT OBJECTS

RECORDS

SUBTITLE

- Records are uniquely stored by a record identifier made up of a combination of the "Zone+Name"

SIMILAR TO APP ENGINE PARENT KEY



CLOUDKIT OBJECTS

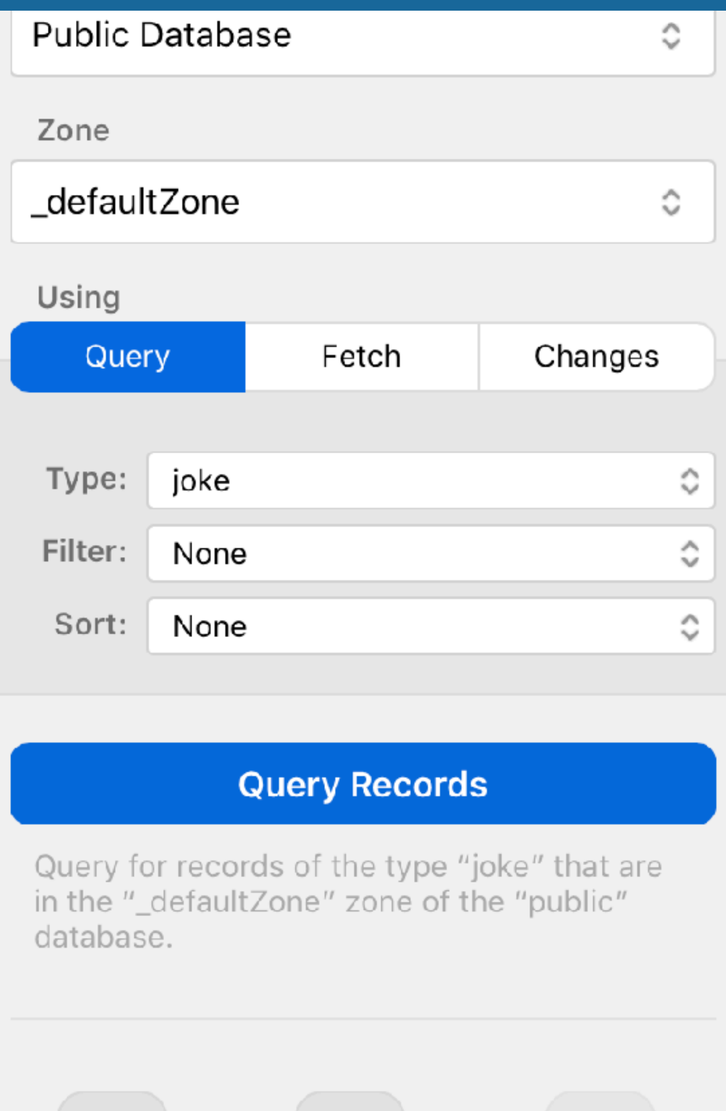
RECORDS

- Meta data is automatically assigned to all records
- Used for many tasks

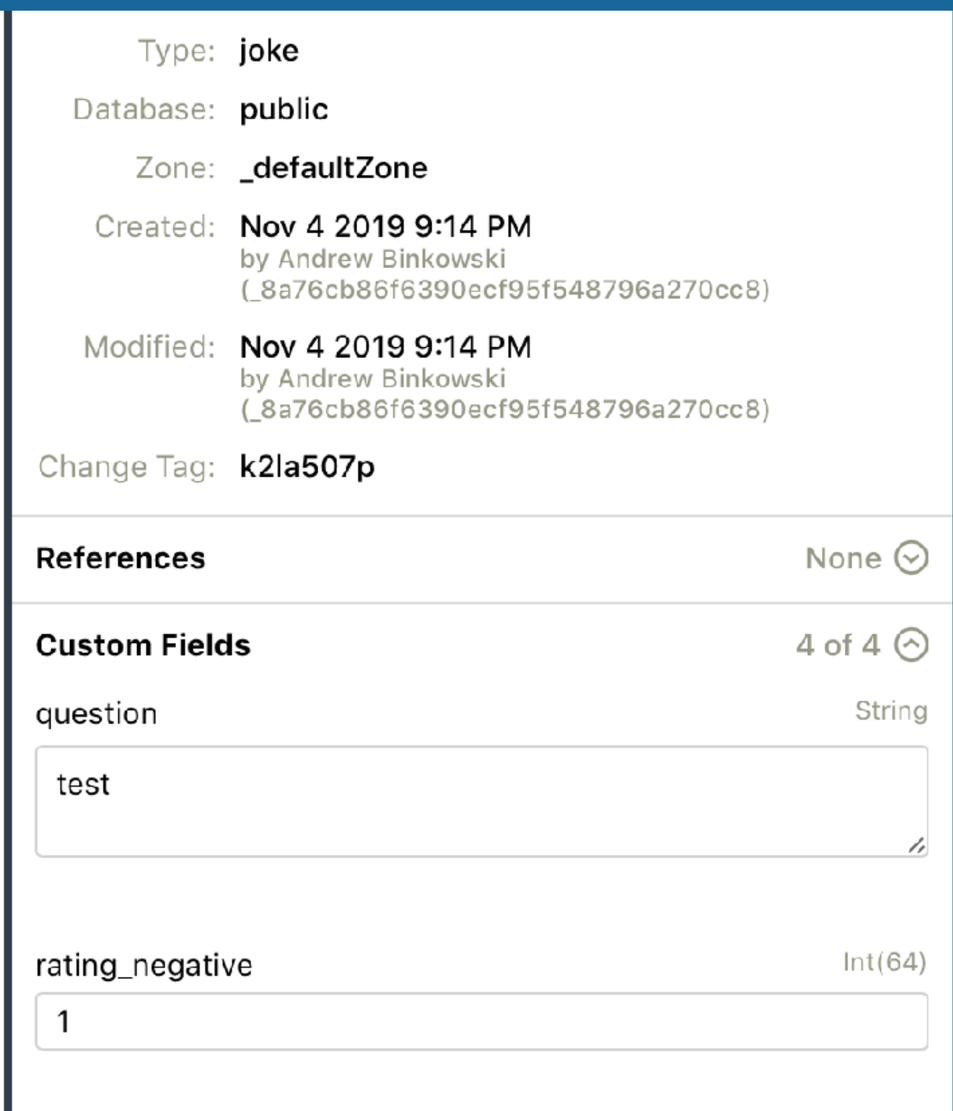
```
/* These create the record in the default zone. */
public init(recordType: String)

public init(recordType: String, recordID: CKRecordID)


public init(recordType: String, zoneID: CKRecordZoneID)


open var recordType: String { get }

@NSCopying open var recordID: CKRecordID { get }


/* Change tags are updated by the server to a unique value every time a record is modified.
   A different change tag necessarily means that the contents of the record are different. */
open var recordChangeTag: String? { get }


/* This is a User Record recordID, identifying the user that created this record. */
@NSCopying open var creatorUserRecordID: CKRecordID? { get }

open var creationDate: Date? { get }


/* This is a User Record recordID, identifying the user that last modified this record. */
@NSCopying open var lastModifiedUserRecordID: CKRecordID? { get }

open var modificationDate: Date? { get }


/*
    In addition to objectForKey: and setObject:forKey:, dictionary-style subscripting (record[key] and record[key] = valu
        used to get and set values.
    Acceptable value object classes are:
        CKReference
        CKAsset
        CLLocation
        NSData
        NSDate
        NSNumber
        NSString
        NSArray containing objects of any of the types above

    Any other classes will result in an exception with name NSInvalidArgumentException.

    Derived field keys are prefixed with '_'. Attempting to set a key prefixed with a '_' will result in an error.

    Key names roughly match C variable name restrictions. They must begin with an ASCII letter and can contain ASCII
     letters and numbers and the underscore character.
    The maximum key length is 255 characters.
*/
open func object(forKey key: String) -> CKRecordValue?
```

CLOUDKIT OBJECTS

RECORDS

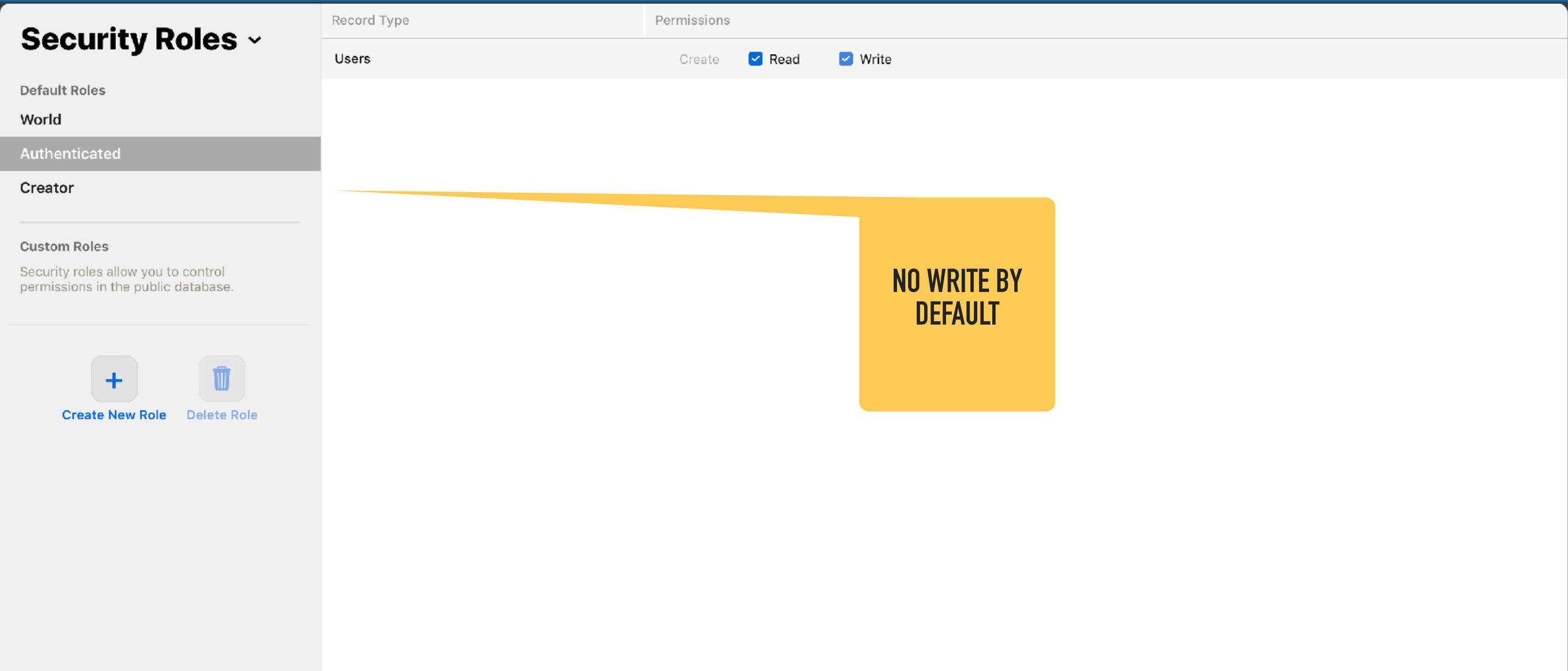


METADATA ON EVERY RECORD



CLOUDKIT OBJECTS

RECORDS



CLOUDKIT OBJECTS

RECORDS

CLOUDKIT OBJECTS

RECORDS

```
/// Create a joke record and save to iCloud using CloudKit
/// - parameter joke: Funny string
/// - remark: Error handling leaves something to be desired
func saveJoke(_ joke: String) {
    let record = CKRecord(recordType: "joke")
    record.setValue(joke, forKey: "question")
    record["response"] = "To get to the other side." as CKRecordValue
    record["rating_positive"] = 0 as CKRecordValue
    record["rating_negative"] = 0 as CKRecordValue

    publicDB.save(record) { (record, error) in
        if let error = error {
            print("Error: \(error.localizedDescription)")
            return
        }
        print("Saved record: \(record.debugDescription)")
    }
}
```

CREATE A RECORD

CLOUDKIT OBJECTS

RECORDS

```
/// Create a joke record and save to iCloud using CloudKit
/// - parameter joke: Funny string
/// - remark: Error handling leaves something to be desired
func saveJoke(_ joke: String) {

  let record = CKRecord(recordType: "joke")
  record.setValue(joke, forKey: "question")
  record["response"] = "To get to the other side." as CKRecordValue
  record["rating_positive"] = 0 as CKRecordValue
  record["rating_negative"] = 0 as CKRecordValue

  publicDB.save(record) { (record, error) in
    if let error = error {
      print("Error: \(error.localizedDescription)")
      return
    }
    print("Saved record: \(record.debugDescription)")
  }
}
```

SAVE THE RECORD

CLOUDKIT OBJECTS

RECORDS

- Create a Joke record in our view controller
- Data is key-value pairs

```
// MARK: - CloudKit
let container: CKContainer = CKContainer.default()
let publicDB: CKDatabase = CKContainer.default().publicCloudDatabase
let privateDB: CKDatabase = CKContainer.default().privateCloudDatabas
```

```
/// Create a joke record and save to iCloud using CloudKit
/// - parameter joke: Funny string
/// - remark: Error handling leaves something to be desired
func saveJoke(_ joke: String) {
  let record = CKRecord(recordType: "joke")
  record.setValue(joke, forKey: "question")
  record["response"] = "To get to the other side." as CKRecordValue

  publicDB.save(record) { (record, error) in
    if let error = error {
      print("🐞: \(error.localizedDescription)")
      return
    }
    print("Saved record: \(record.debugDescription)")
  }
}
```

CLOUDKIT OBJECTS

RECORDS

```swift
import CloudKit

class ViewController: UIViewController {

  // MARK: - CloudKit
  let container: CKContainer = CKContainer.default()
  let publicDB: CKDatabase = CKContainer.default().publicCloudDatabase
  let privateDB: CKDatabase = CKContainer.default().privateCloudDatabase


  override func viewDidAppear(_ animated: Bool) {
    saveJoke("Why did the chicken cross the road?")
  }


  /// Create a joke record and save to iCloud using CloudKit
  /// - parameter joke: Funny string
  /// - remark: Error handling leaves something to be desired
  func saveJoke(_ joke: String) {
    let record = CKRecord(recordType: "joke")
    record.setValue(joke, forKey: "text")
    publicDB.save(record) { (record, error) in
      if let error = error {
        print("🐞: \(error.localizedDescription)")
        return
      }
      print("Saved record: \(record.debugDescription)")
    }

  }

}
```

YOUR FIRST CLOUDKIT ERROR

🎉

```
🐞: This request requires an authenticated account
```

CLOUDKIT OBJECTS

RECORDS

FULLY SUPPORTED BY THE SIMULATOR

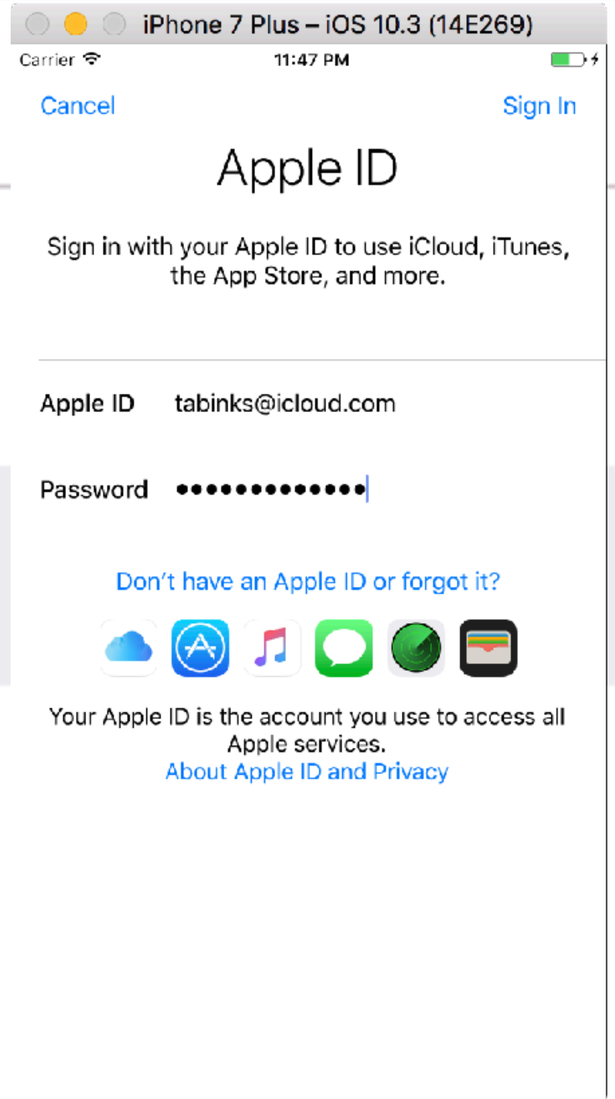
General

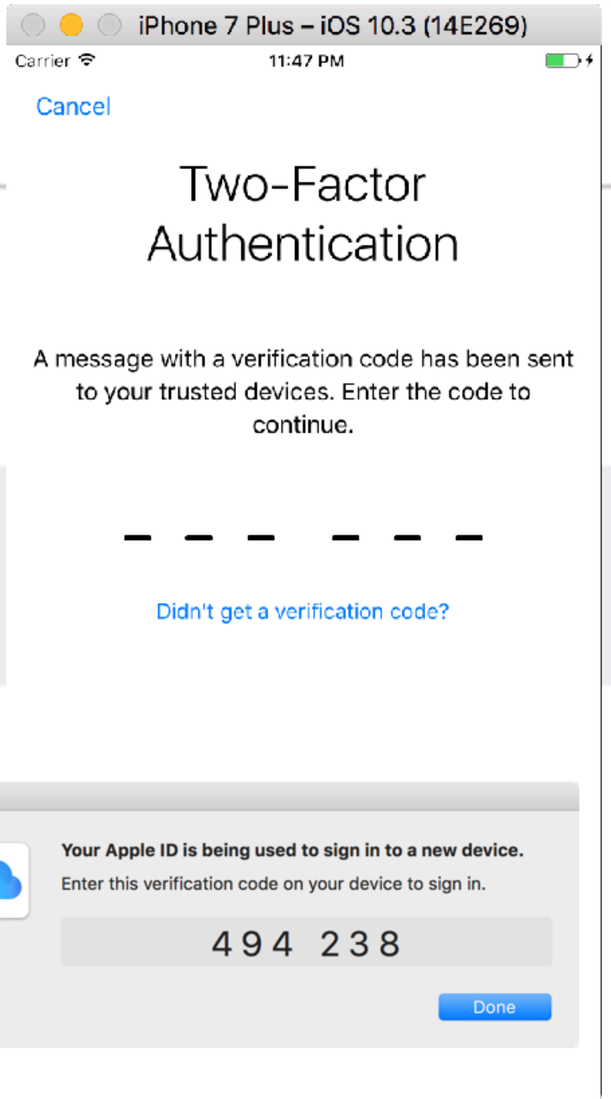
Privacy

NEED TO LOGIN WITH ICLOUD ID

Maps

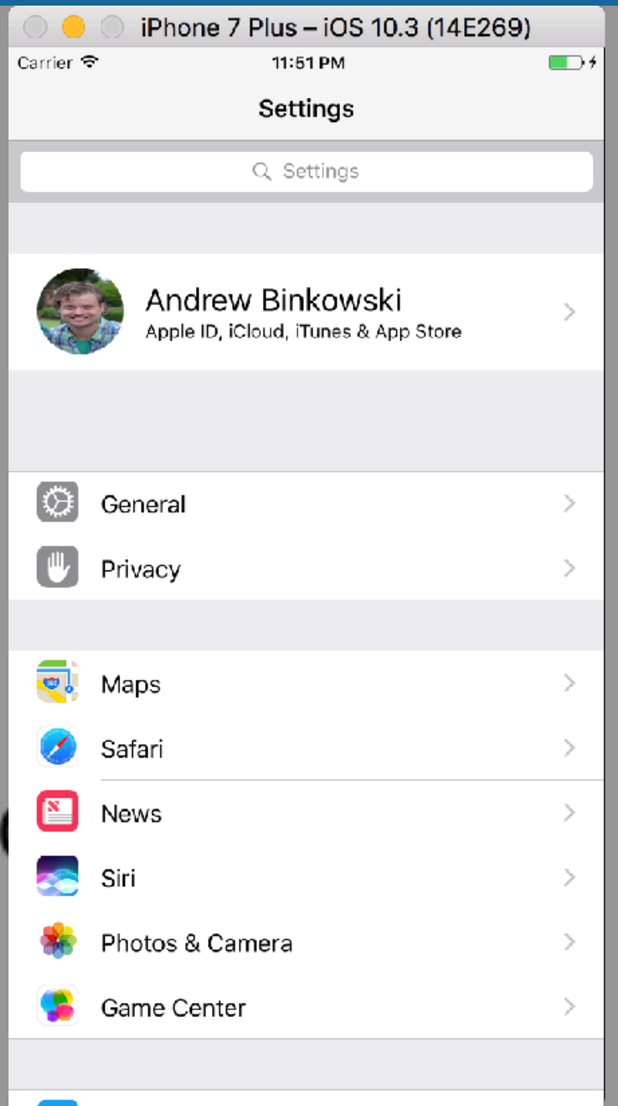
Safari





CLOUDKIT OBJECTS

RECORDS



BE CAREFUL...YOU PROBABLY DON'T WANT TO MERGE

CLOUDKIT OBJECTS

RECORDS

```
  }


  /// Create a joke record and save to iCloud using CloudKit
  /// - parameter joke: Funny string
  /// - remark: Error handling leaves something to be desired
  func saveJoke(_ joke: String) {

    let record = CKRecord(recordType: "joke")
    record.setValue(joke, forKey: "question")
    record["response"] = "To get to the other side." as CKRecordValue

    publicDB.save(record) { (record, error) in
      if let error = error {
        print("Error: \(error.localizedDescription)")
        return
      }
      print("Saved record: \(record.debugDescription)")
    }
  }

}
```

CloudyWithAChanceOfErros

```
Saved record: Optional(<CKRecord: 0x7fad84001550; recordID=025F8A03-2047-4B26-98A0-A97CCA823DDB:
(_defaultZone:__defaultOwner__), recordChangeTag=j28i9sj2, values={
    text = "Why did the chicken cross the road?";
}, recordType=joke>
{
   creatorUserRecordID -> <CKRecordID: 0x610000028d40; recordName=__defaultOwner__,
zoneID=_defaultZone:__defaultOwner__>
   lastModifiedUserRecordID -> <CKRecordID: 0x6100000290c0; recordName=__defaultOwner__,
zoneID=_defaultZone:__defaultOwner__>
```

CLOUDKIT OBJECTS

RECORDS

Records

Database

Public Database

Zone

_defaultZone

Using

Query | Fetch | Changes

Type: joke

Filter: None

Sort: None

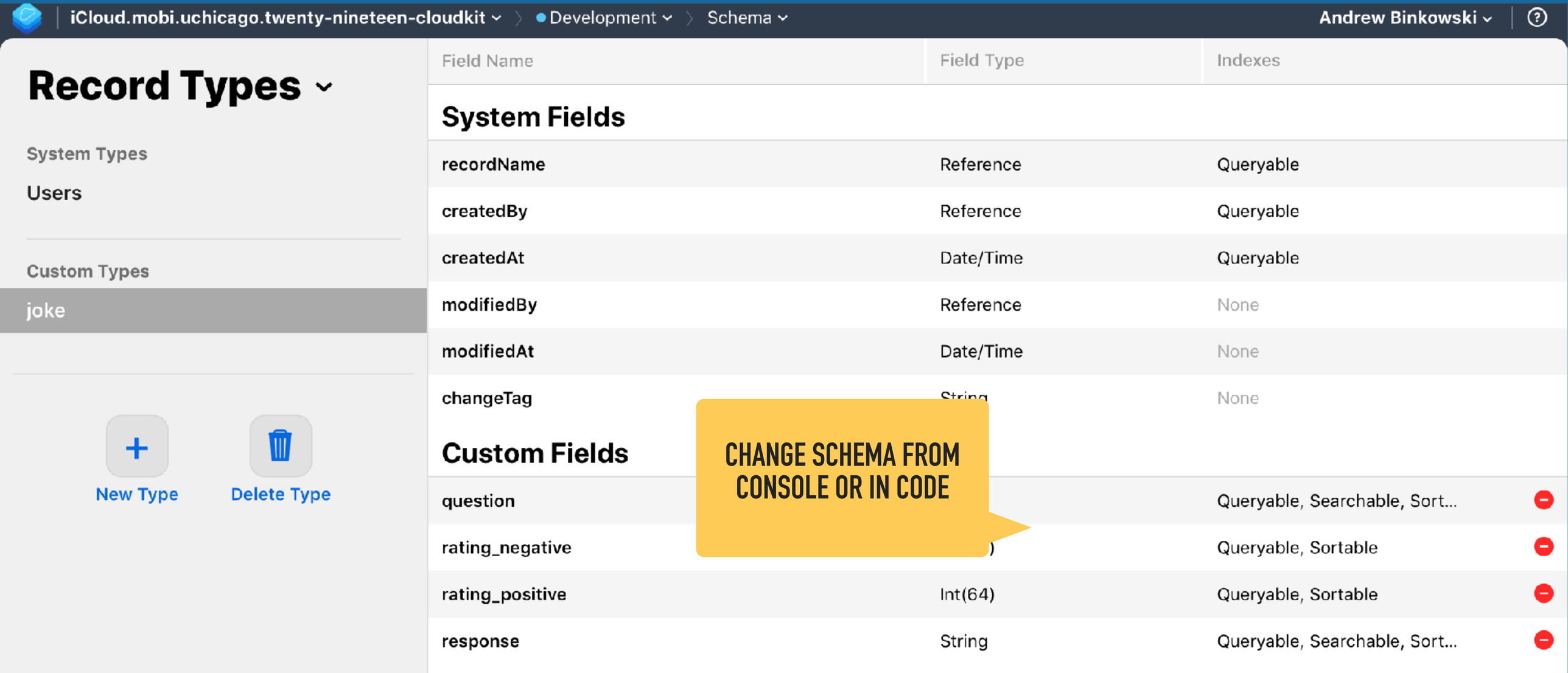
Query Records

Query for records of the type "joke" that are in the "_defaultZone" zone of the "public" database.

Name	Type	Fields	Ch. Tag	Created	Modified
0FEB50A6-282E-4...	joke	**4**: question, response, ...	k2lacelf	2019/11/04, 21:20	2019/11/04, 21:20
AA672AF9-673D-F...	joke	**4**: question, response, ...	k2la5...	2019/11/04, 21:14	2019/11/04, 21:14

CLOUDKIT OBJECTS

RECORDS



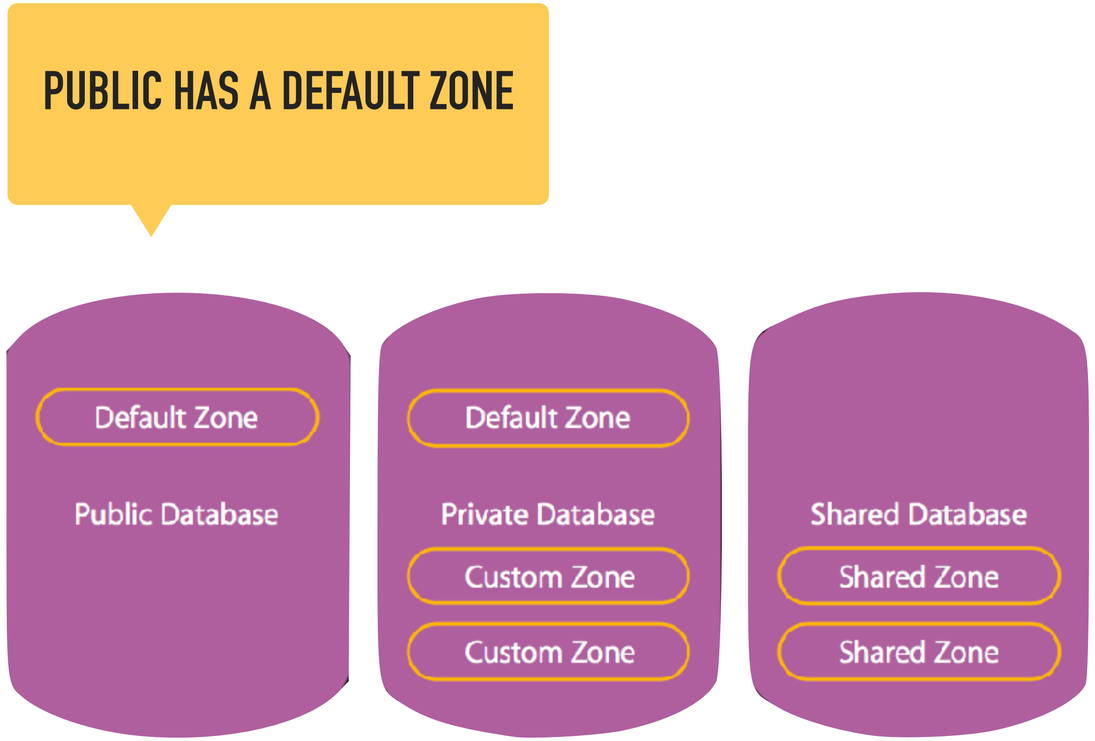
RECORD ZONES

CLOUDKIT OBJECTS

RECORD ZONES

SUBTITLE

- CKRecordZones
  - Zones are a useful way to arrange a discrete group of records
  - Supported only in Private and Shared database
  - Atomic group writes

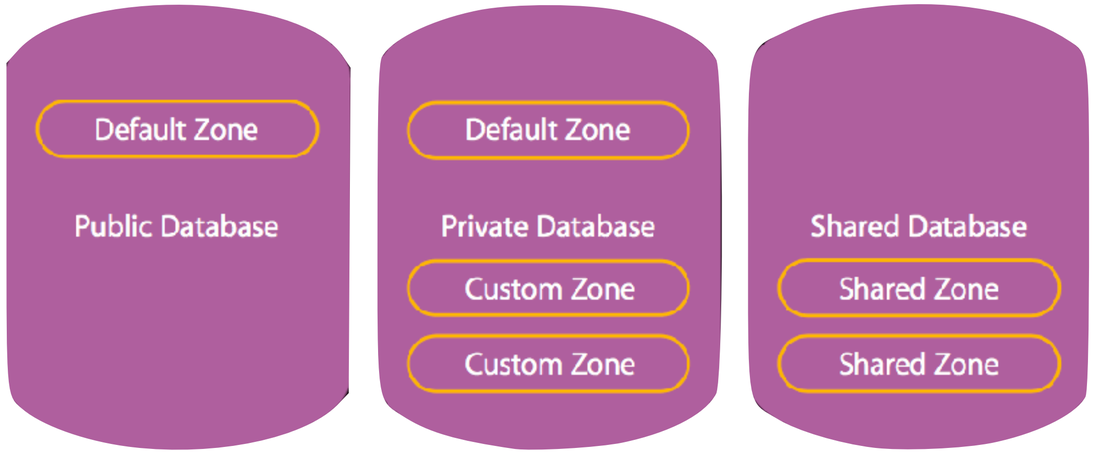


CLOUDKIT OBJECTS

RECORD ZONES

SUBTITLE

- CKRecordZones
  - Zones are a useful way to arrange a discrete group of records
  - Supported only in Private and Shared database
  - Atomic group writes



CLOUDKIT OBJECTS

RECORDS

- Created by the client
- They represent the location of the record
- External data set foreign key

```
//
//  CKRecordID.h
//  CloudKit
//
//  Copyright (c) 2014 Apple
//

#import <Foundation/Foundati

@class CKRecordZoneID;

NS_CLASS_AVAILABLE(10_10, 8_0)
@interface CKRecordID : NSObject <NSSecureCodi

- (instancetype)init NS_UNAVAILABLE;

/* Record names must be 255 characters or less
/* This creates a record ID in the default zon
- (instancetype)initWithRecordName:(NSString *
- (instancetype)initWithRecordName:(NSString *

@property (nonatomic, readonly, strong) NSStri
@property (nonatomic, readonly, strong) CKReco

@end
```

CKRECORDZONEID

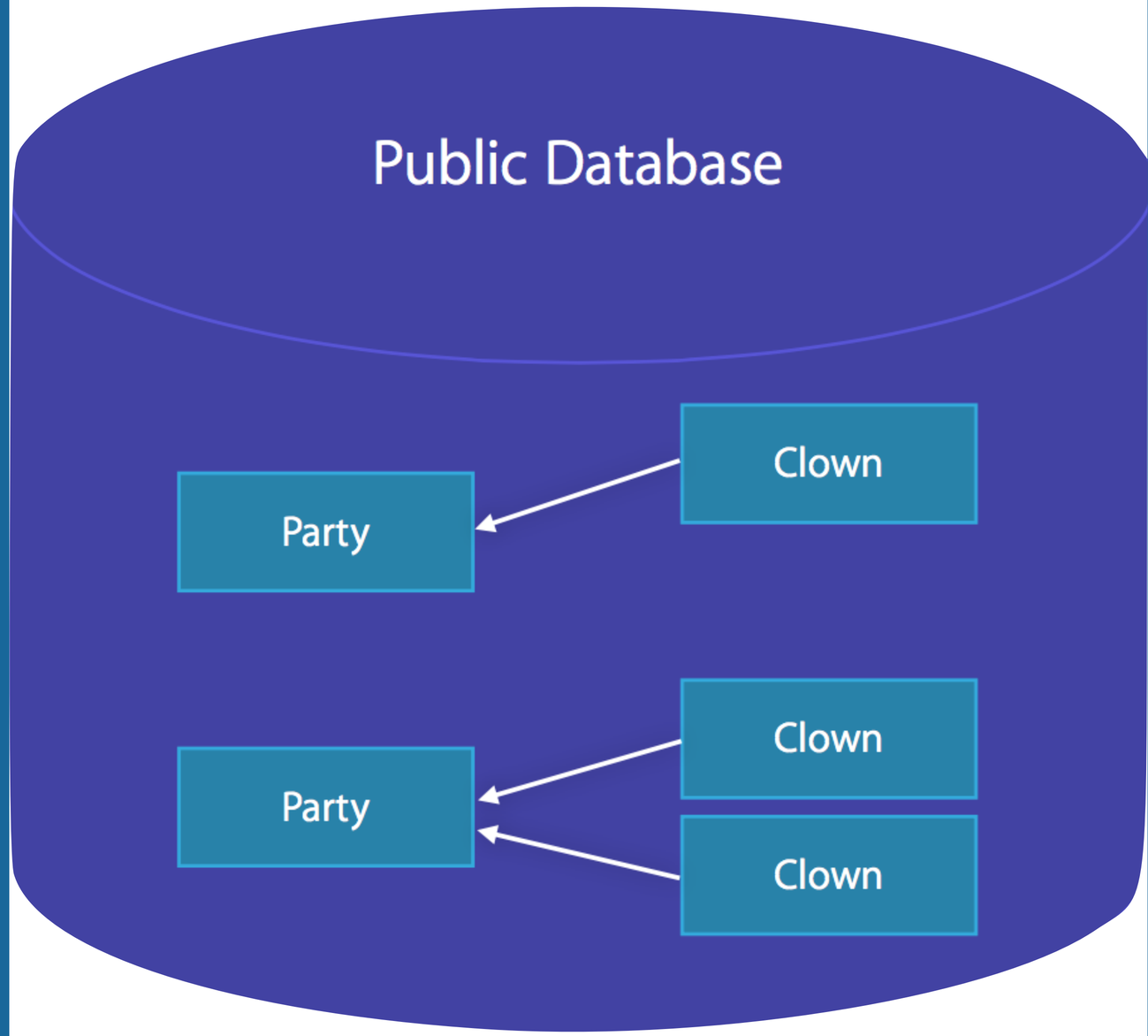
REFERENCES

CLOUDKIT OBJECTS

RECORDS

SUBTITLE

- CKReference
  - Points to a another record
  - Server understands relationship (takes some responsibility for managing; if you want)

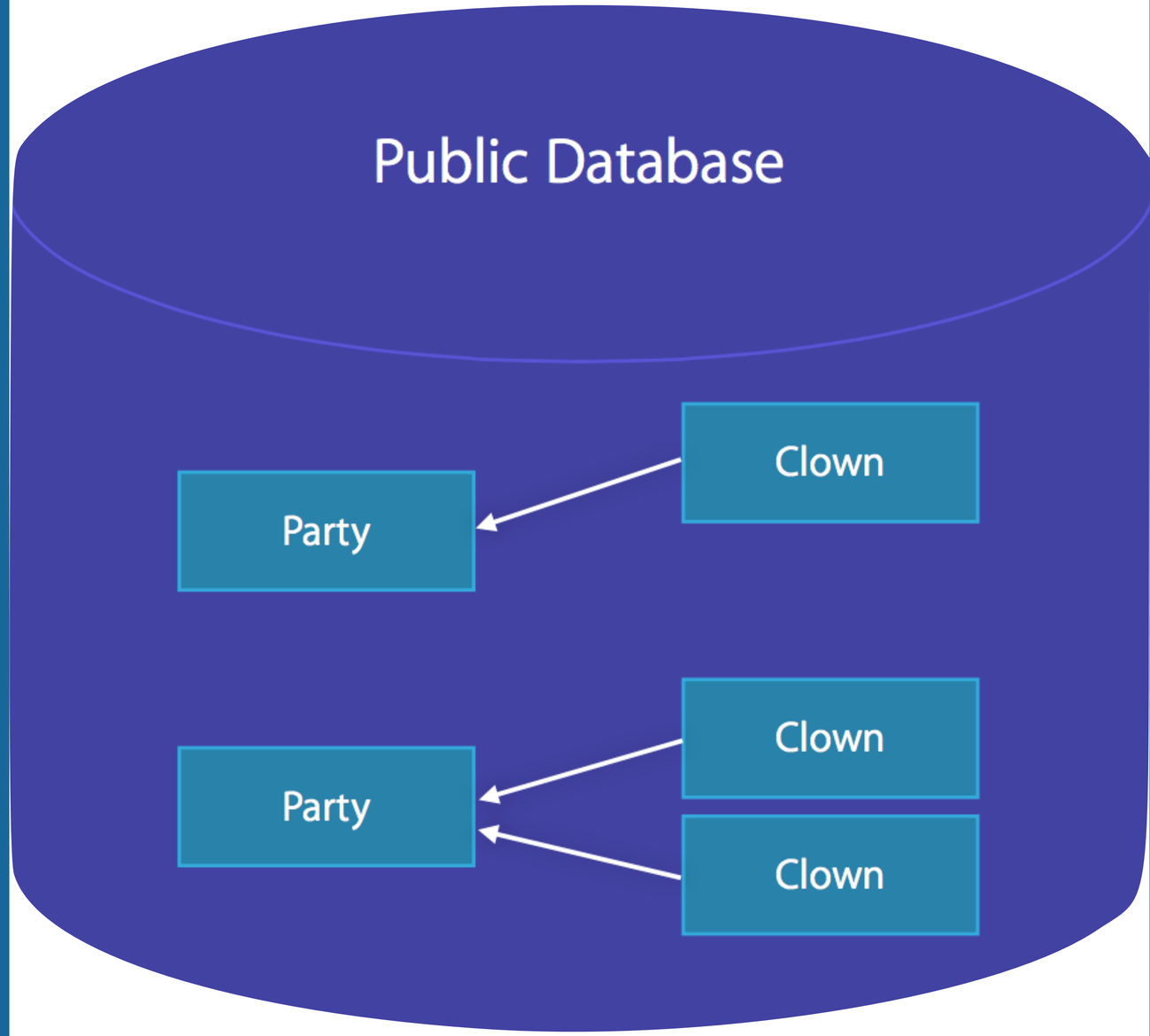


CLOUDKIT OBJECTS

RECORDS

SUBTITLE

- Cascade Deletes
  - What happens when you delete something with a reference?
  - Dangling Pointers
- Use "Back References" strategy

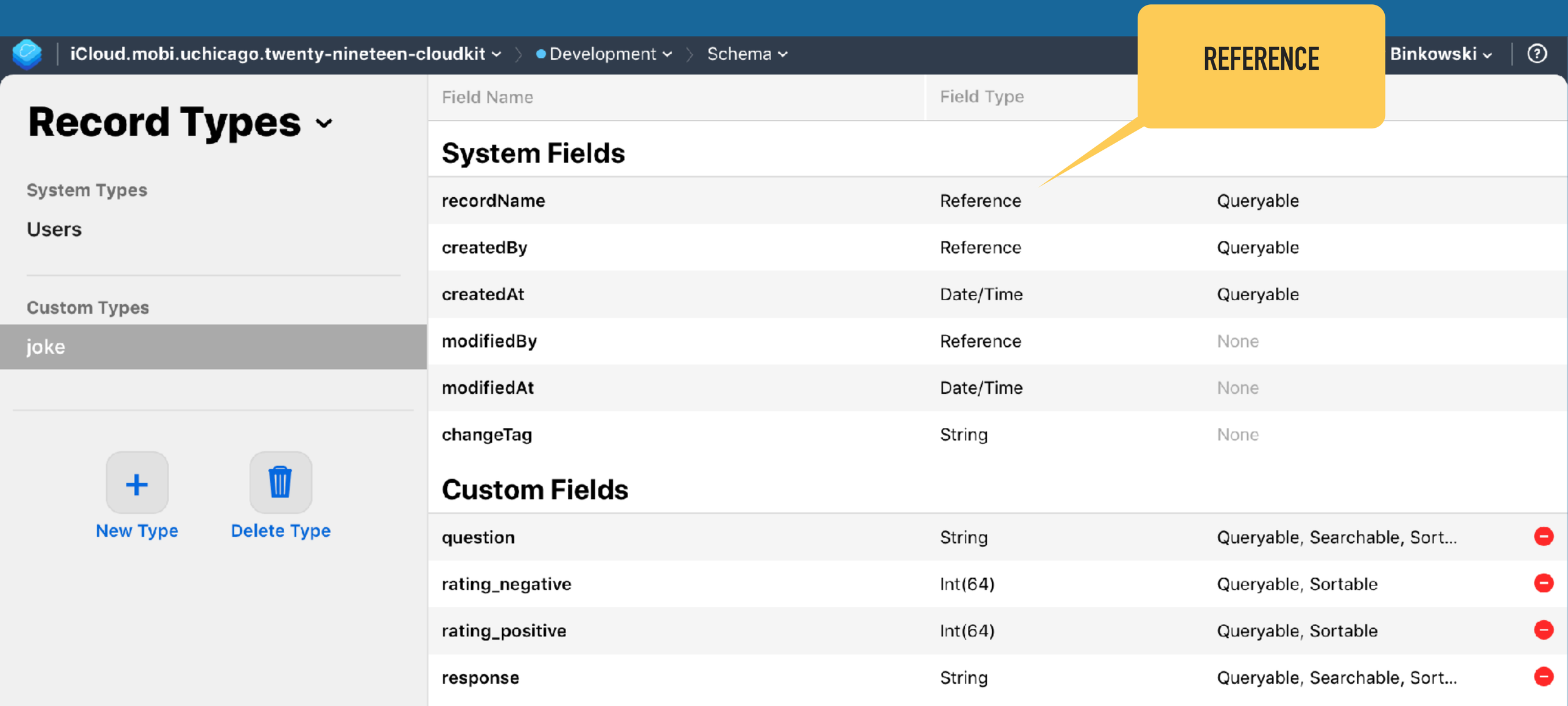


CLOUDKIT API

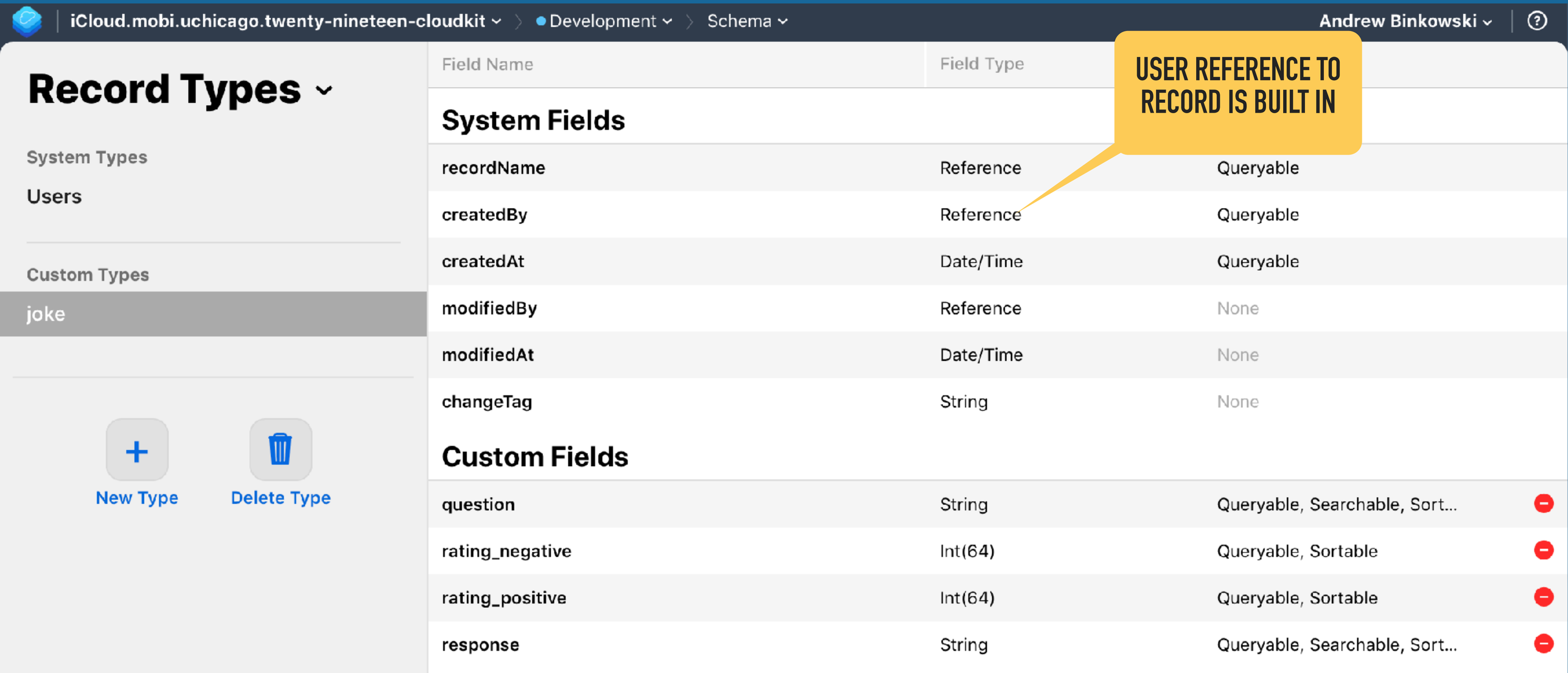
```
// Create a reference with a record
// Set the delete 'action' preference
let reference = CKReference(recordID: recordID,
                            action: .none)
```

WHAT HAPPENS ON DELETE

CLOUDKIT API



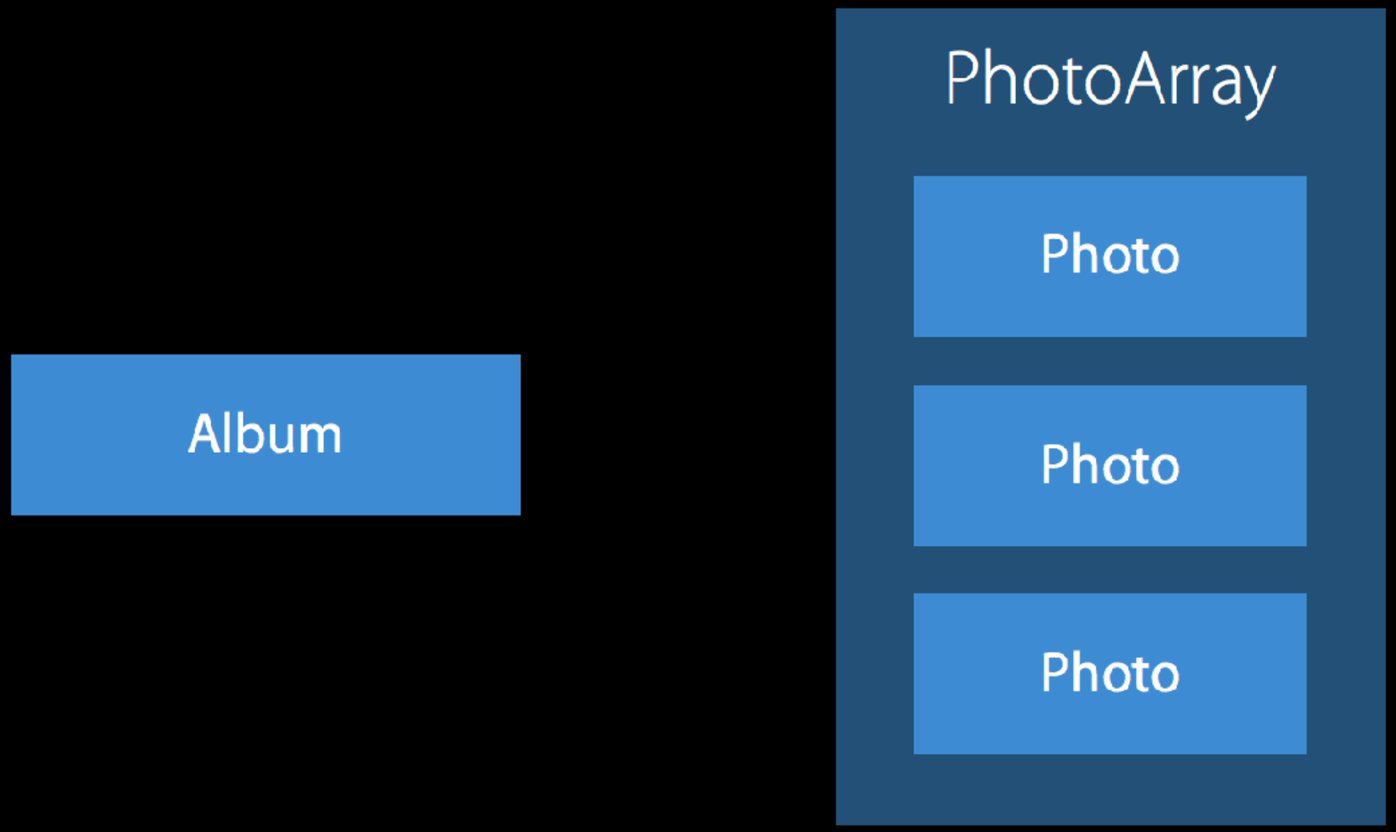
CLOUDKIT API



DATA MODELING

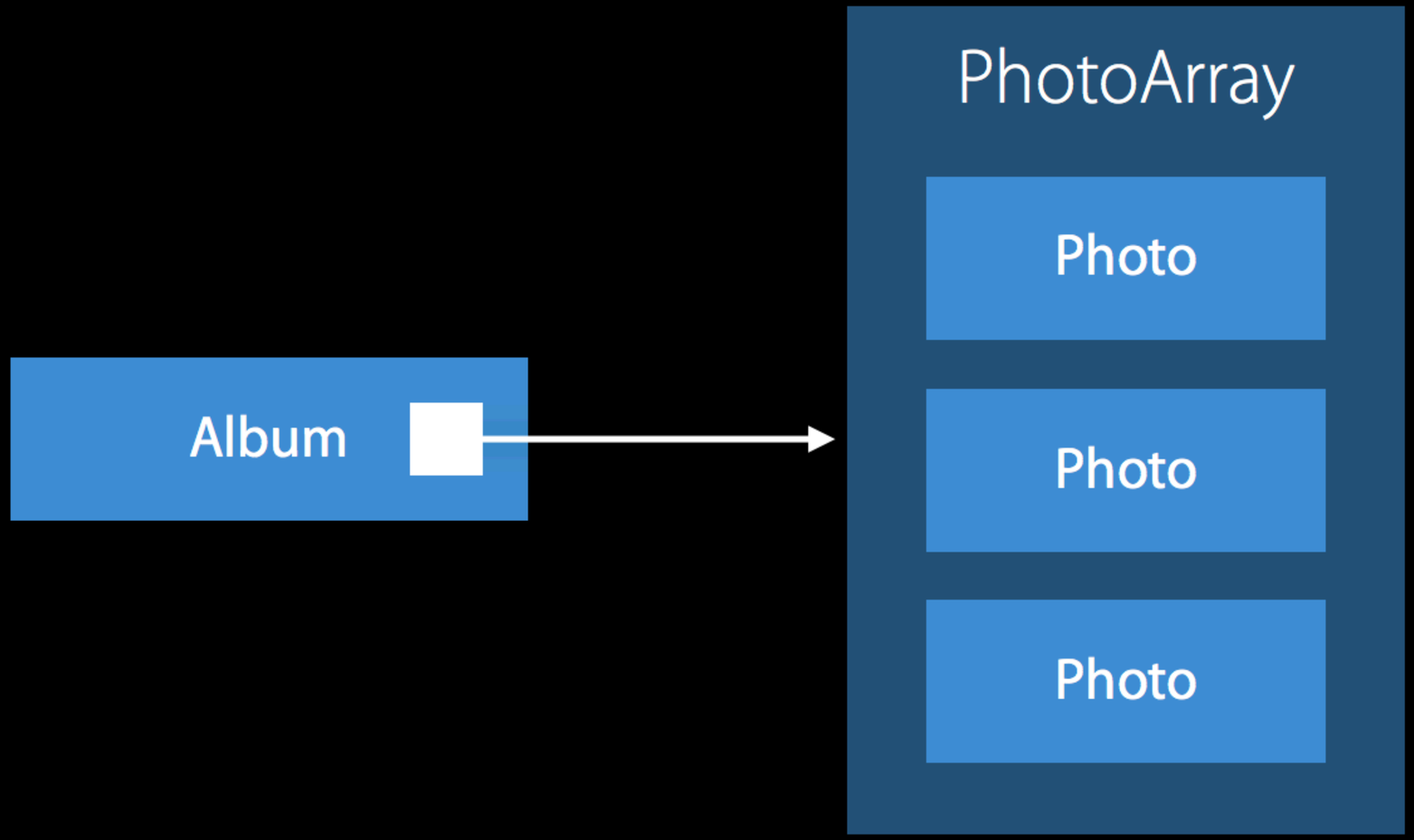
DATA MODELING

- How should 1 to many relationships be handled?



DATA MODELING

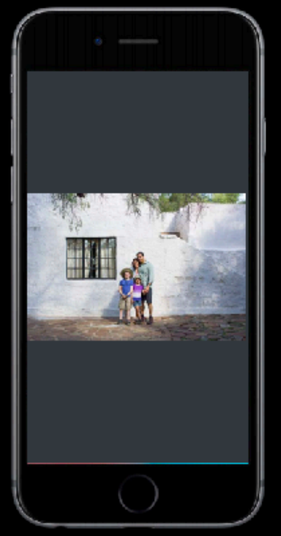
- Typical setup to model



DATA MODELING

PhotoArray = []

Album



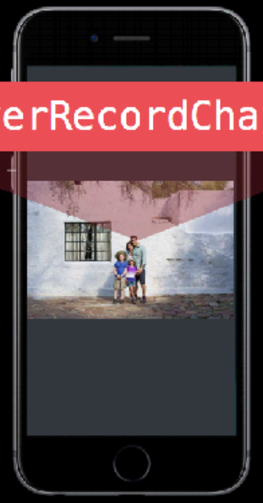
DATA MODELING

PhotoArray = []

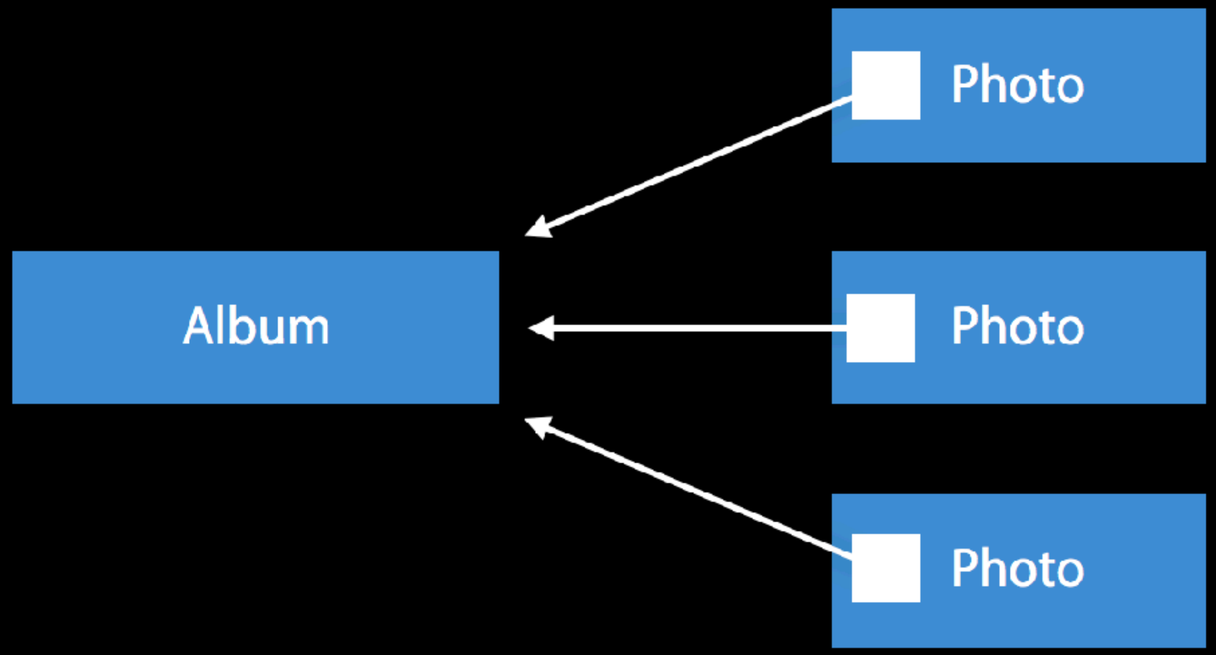
Album

serverRecordChanged

serverRecordChanged



DATA MODELING



USER RECORDS

USER RECORDS

- Identity
- Metadata
- Privacy
- Discovery
  - Users can discover friends who use the application

USER RECORDS

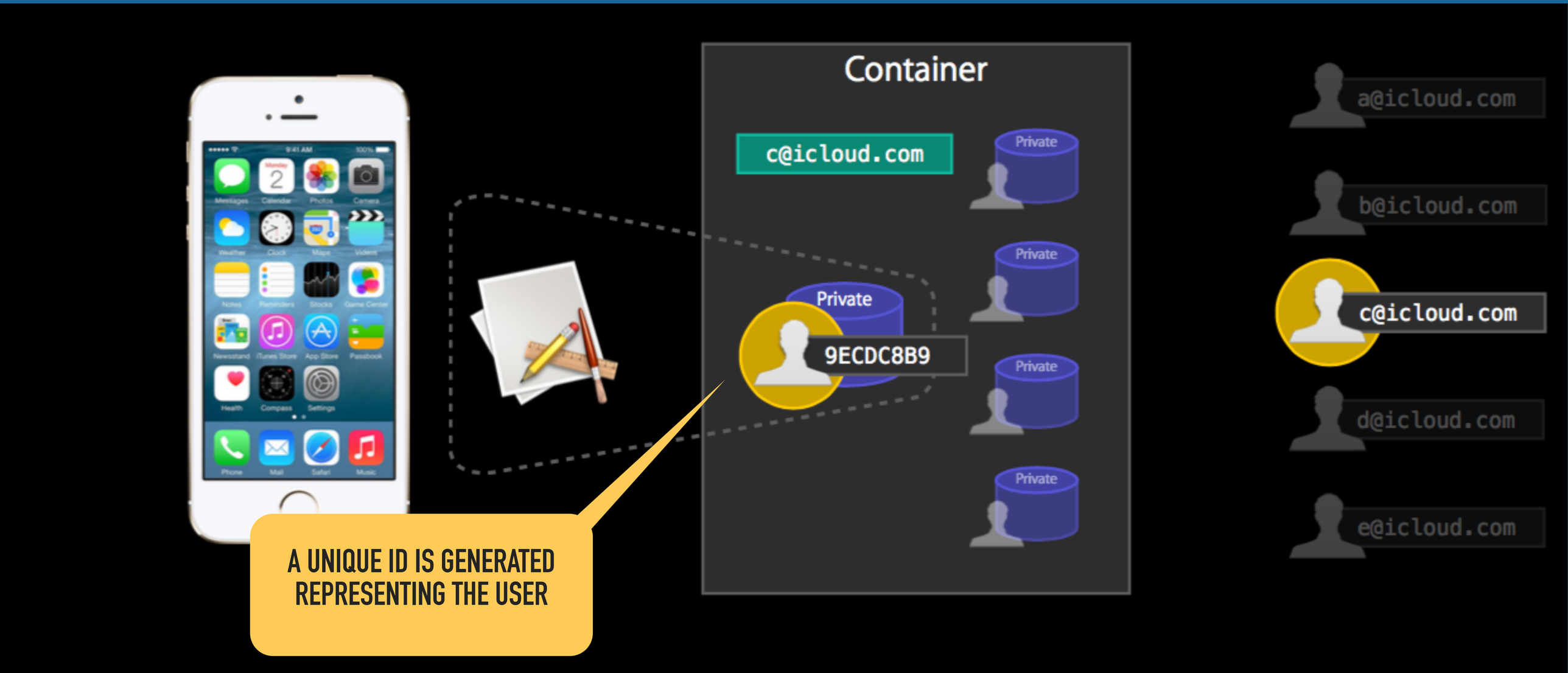
Private Private Private

Private Private Private

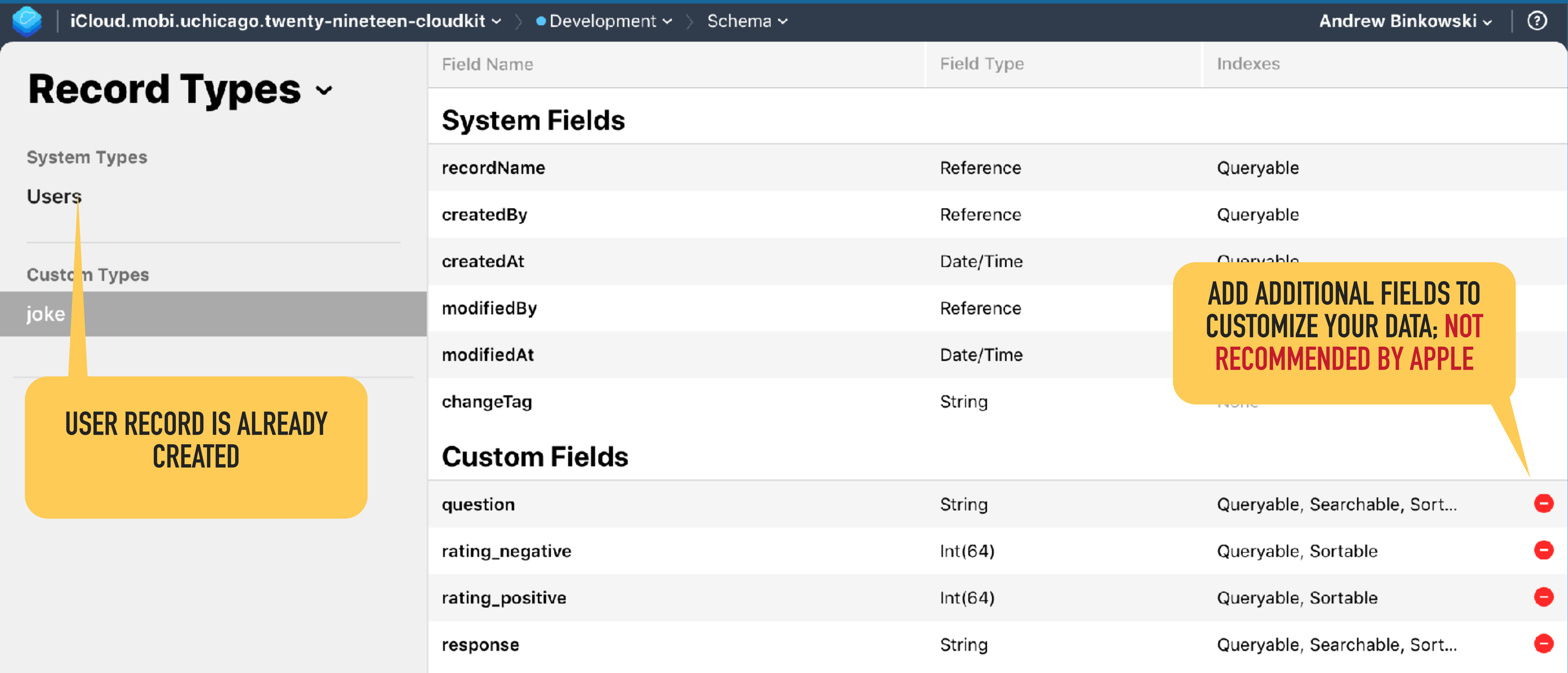
Private Private Private

IDENTITY INFORMATION ABOUT USERS

USER RECORDS

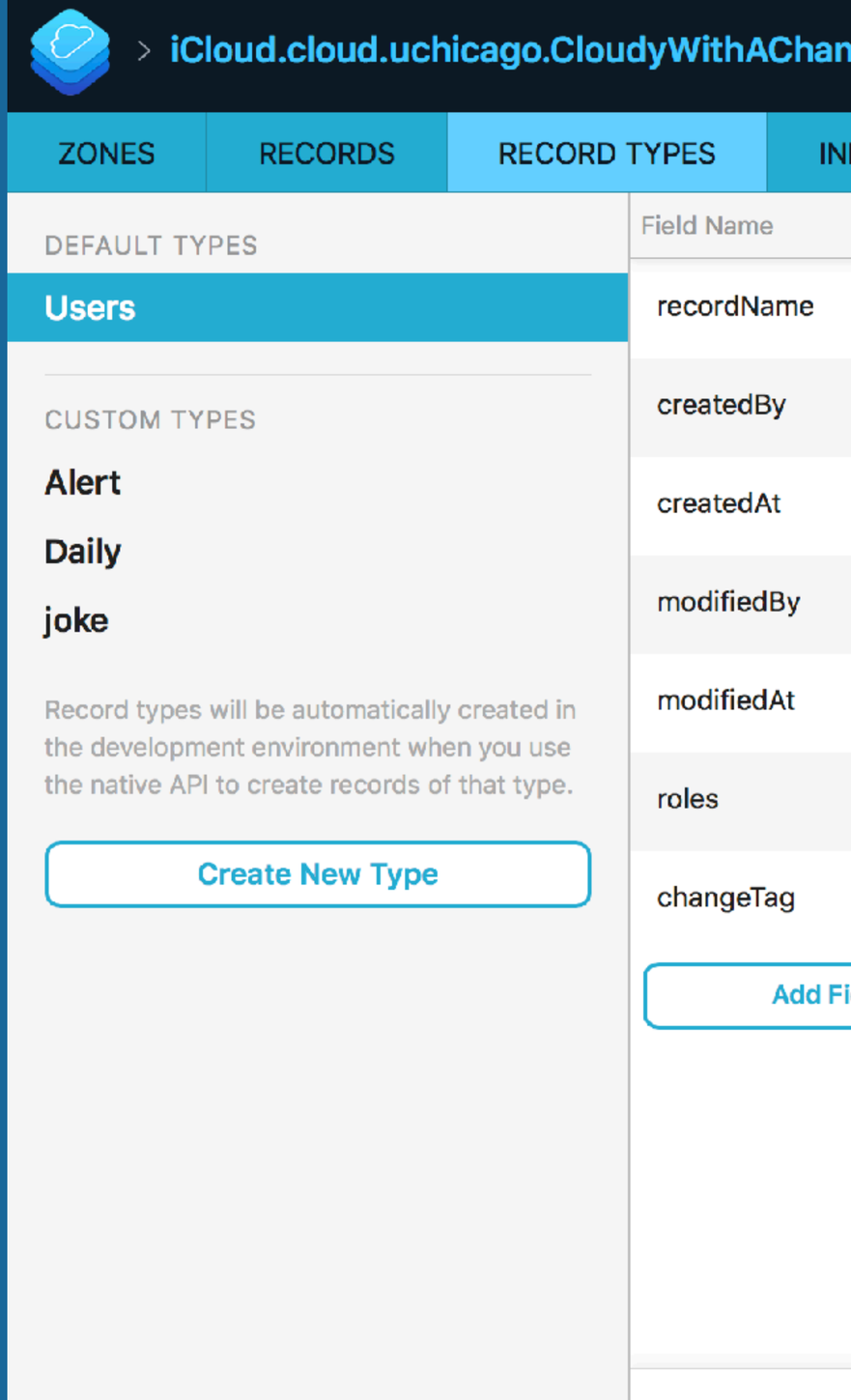


USER RECORDS



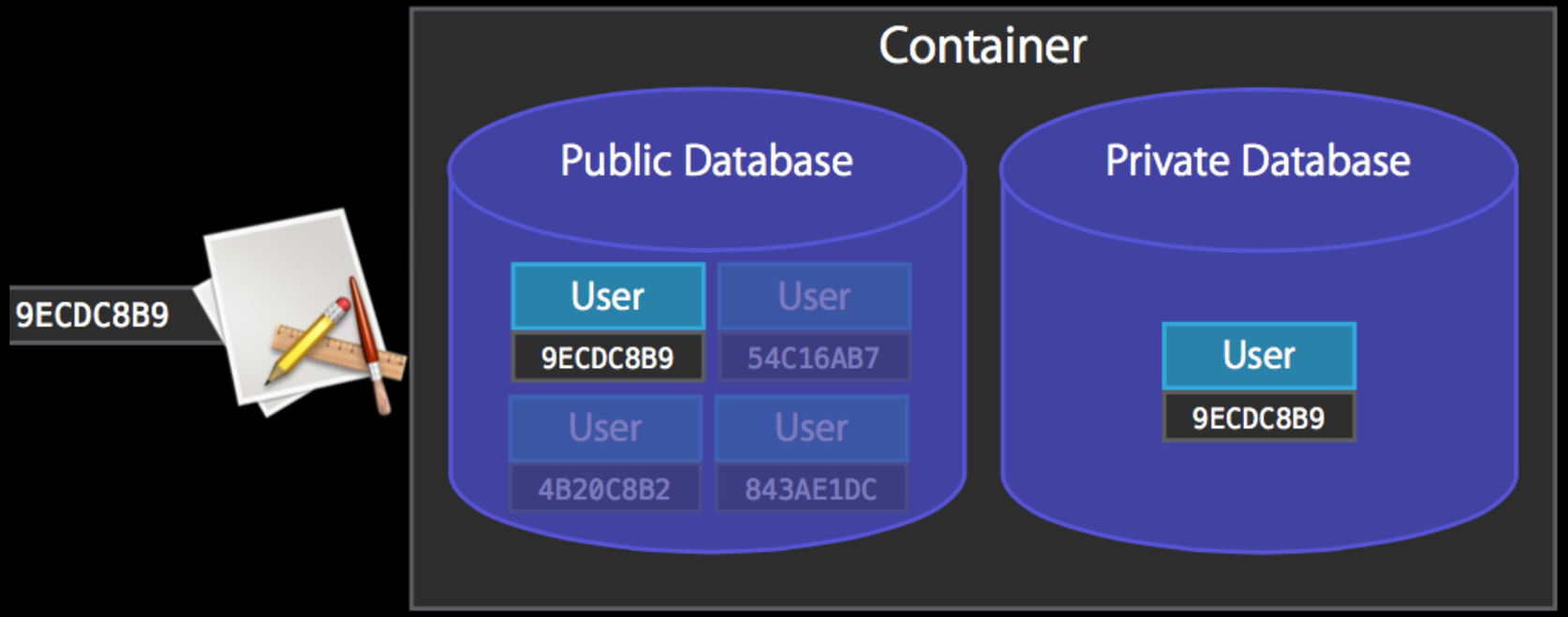
USER RECORDS

- Instead, to associate a User with other data
  - Create a `User` record type on other record that has a reference to the `UserId`
  - Store any other data
  - 



CloudKit User Accounts

- User metadata is a CKRecordTypeUser record
  - One for each database



UNIQUE TO EACH APPLICATION; WILL NOT BE THE SAME FOR SAME USER IN DIFFERENT APPLICATIONS FROM THE SAME DEVELOPER

USER RECORDS

GET A USER RECORD

```
let container = CKContainer.default()

container.fetchUserRecordID() {
      recordID, error in

      if error != nil {
        print(error!.localizedDescription)
        complete(nil, error as NSError?)
      } else {
        // We have access to the user's record
        print("fetched ID \(recordID?.recordName ?? "")")
        complete(recordID, nil)
      }
}
```

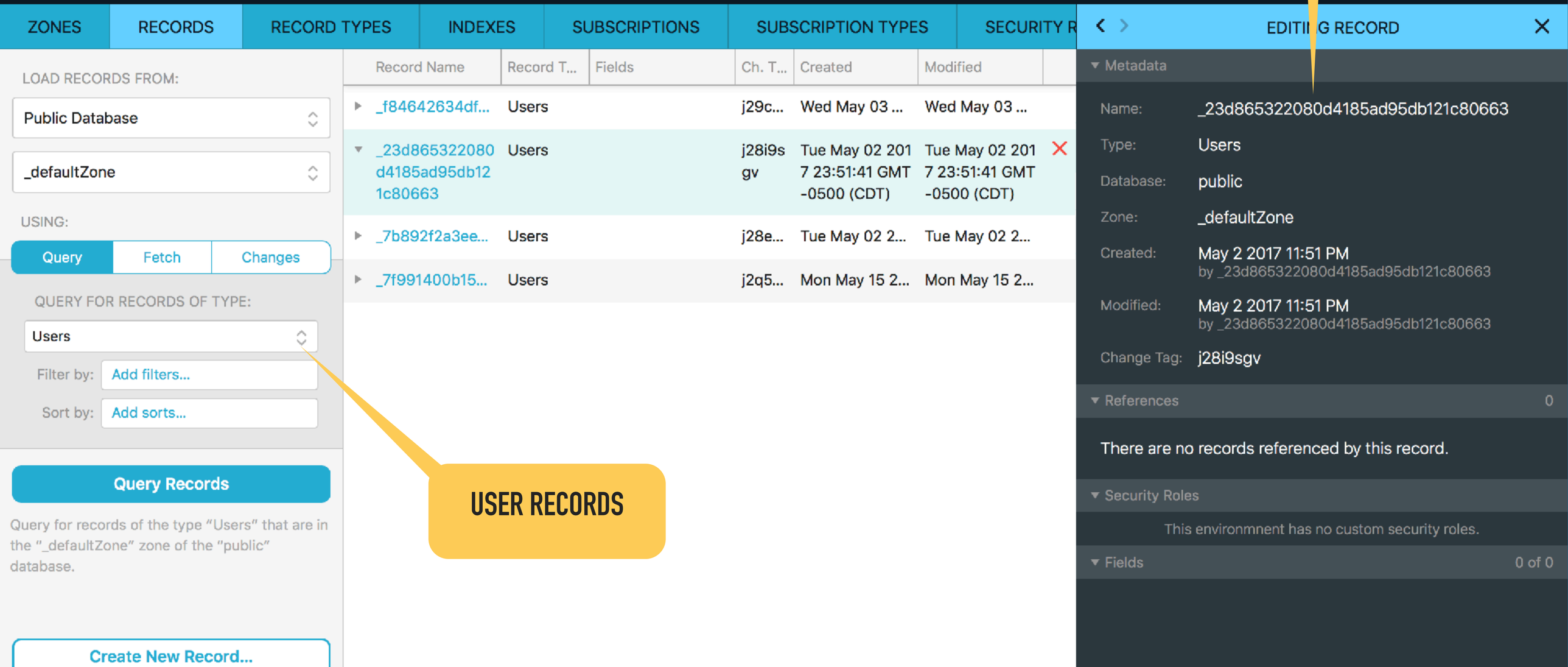
USER RECORDS

```
let container = CKContainer.default()

container.fetc
        recordID

        if error
          print(
          complete(nil, error as NSError?)
        } else {
          // We have access to the user's record
          print("fetched ID \(recordID?.recordName ?? "")")
          complete(recordID, nil)
        }
}
```

"_23D865322080D4185AD95DB121C80663"

USER RECORDS

"_23D865322080D4185AD95DB121C80663"



USER RECORDS

```
/// Get the users `RecordId`
/// - parameters complete: A completion block passing two parameters
func getUserRecordId(complete: @escaping (CKRecordID?, NSError?) -> ()) {

  let container = CKContainer.default()
  container.fetchUserRecordID() {
    recordID, error in

    if error != nil {
      print(error!.localizedDescription)
      complete(nil, error as NSError?)
    } else {
      // We have access to the user's record
      print("fetched ID \(recordID?.recordName ?? "")")
      complete(recordID, nil)
    }
  }
}
```

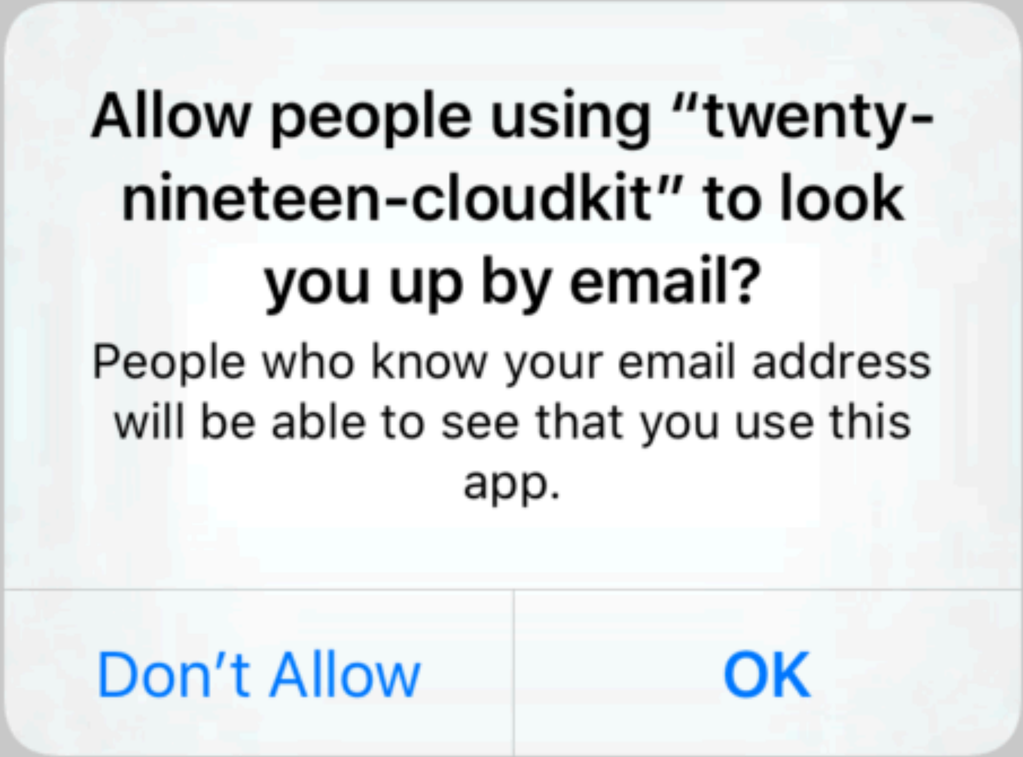
```
getUserRecordId { (recordID, error) in
  if let userID = recordID?.recordName {
    print("iCloudID: \(userID)")
    self.getJokesByCurrentUser(recordID!)
  } else {
    print("iCloudID: nil")
  }
}
```

USER RECORDS

SUBTITLE

- User privacy
  - No disclosure by default
  - Disclosure requested by application

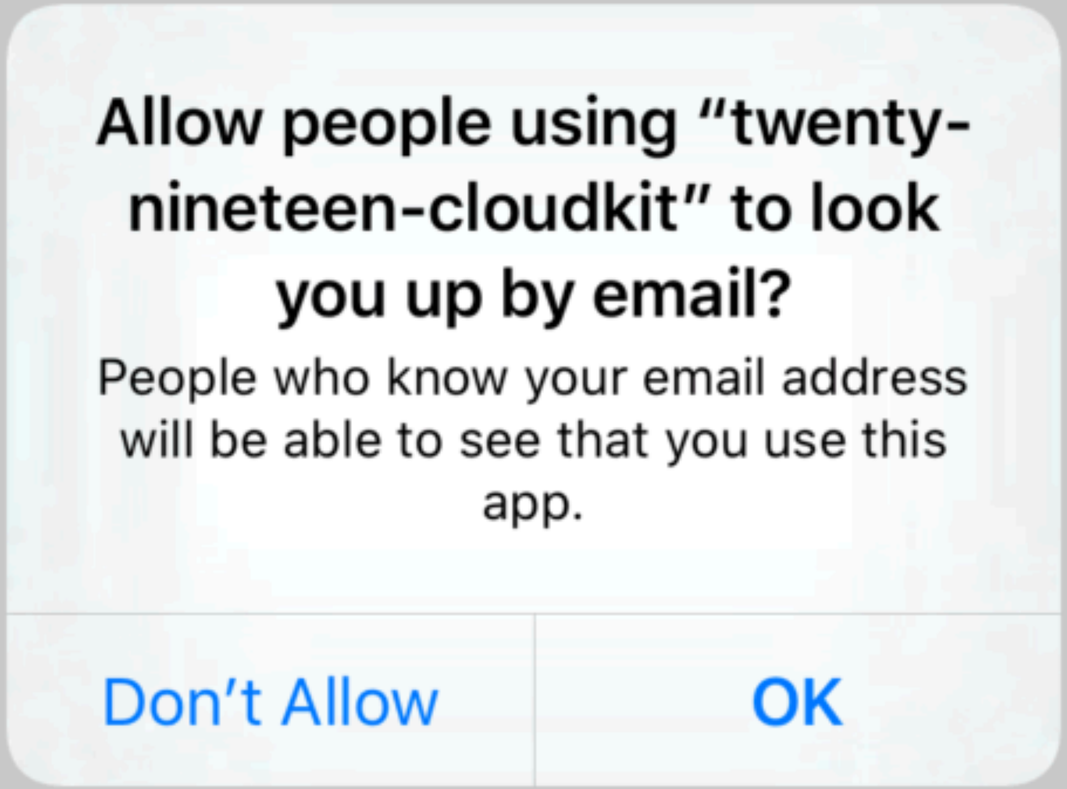
- Opting in allows you to be discoverable by your friends



USER RECORDS

SUBTITLE

- Input
  - User RecordID
  - Email address
  - Entire address book
- Output
  - User RecordID
  - First and last names
- Personally identifying information
  - Requires opt-in



USER RECORDS

GET USER NAME ASSOCIATED WITH ICLOUD ACCOUNT

```
/// Get the users `RecordId`
/// - parameters complete: A completion block passing two parameters
open func getUserIdentity(complete: @escaping (String?, NSError?) -> ()) {

  container.requestApplicationPermission(.userDiscoverability) { (status, error) in
    self.container.fetchUserRecordID { (record, error) in

      if error != nil {
        print(error!.localizedDescription)
        complete(nil, error as NSError?)

      } else {
        //print("fetched ID \(record?.recordName ?? "")")
        self.container.discoverUserIdentity(withUserRecordID: record!, completionHandler: { (userID, error) in
          let userName = (userID?.nameComponents?.givenName)! + " " + (userID?.nameComponents?.familyName)!
          print("CK User Name: " + userName)
          complete(userName,nil)
        })
      }
    }
  }
}
```

USER RECORDS

```
            print("CK User Name: " + userName)
        })
      }
    }
```

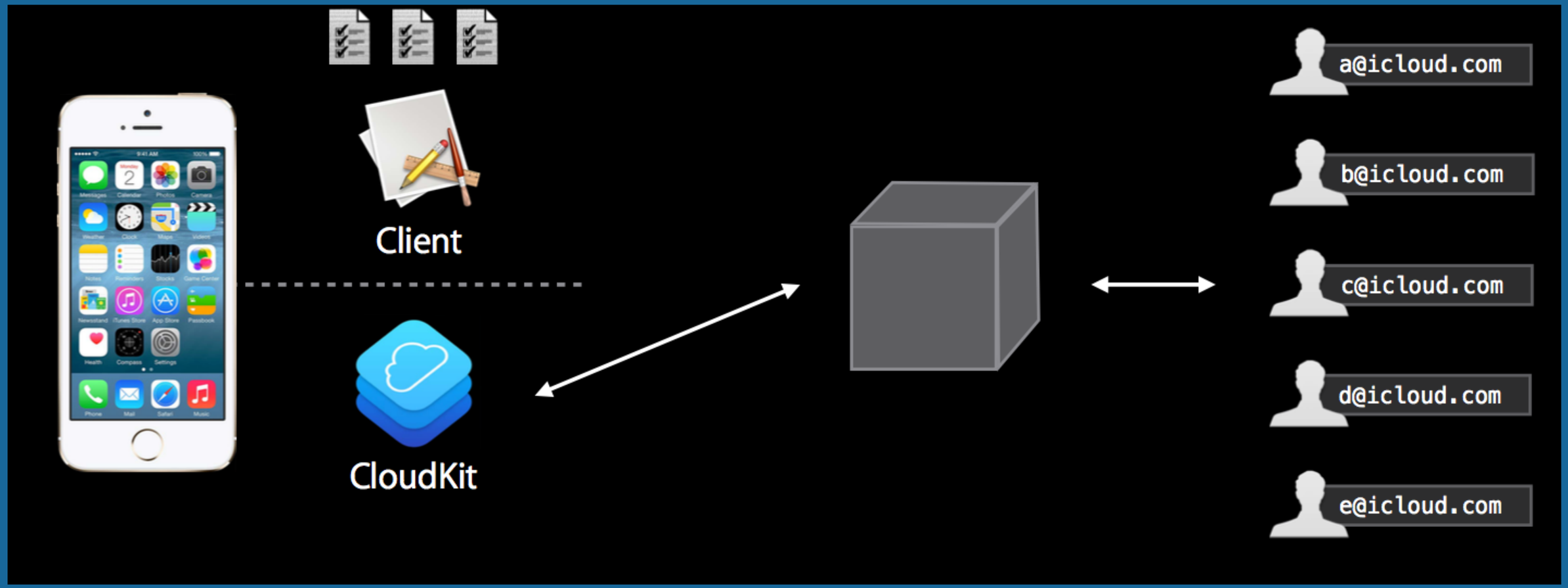
```
CKContainer.default().requestApplicationPermission(.userDiscoverability) { (status, error) in
  CKContainer.default().fetchUserRecordID { (record, error) in
    CKContainer.default().discoverUserIdentity(withUserRecordID: record!, completionHandler: { (userID, error) in
      userName = (userID?.nameComponents?.givenName)! + " " + (userID?.nameComponents?.familyName)!
      print("CK User Name: " + userName)
    })
  }
}
```

```
    }

    func getJokesByCurrentUser(_ recordID: CKRecordID) {
      // The user is a reference, so we need to query against a reference
      let reference = CKReference(recordID: recordID, action: .none)
      let predicate = NSPredicate(format: "creatorUserRecordID == %@", reference)
```

CloudyWithAChanceOfErros

```
fetched ID _23d865322080d4185ad95db121c80663
iCloudID: _23d865322080d4185ad95db121c80663
CK User Name: Andrew Binkowski
```

USER RECORDS

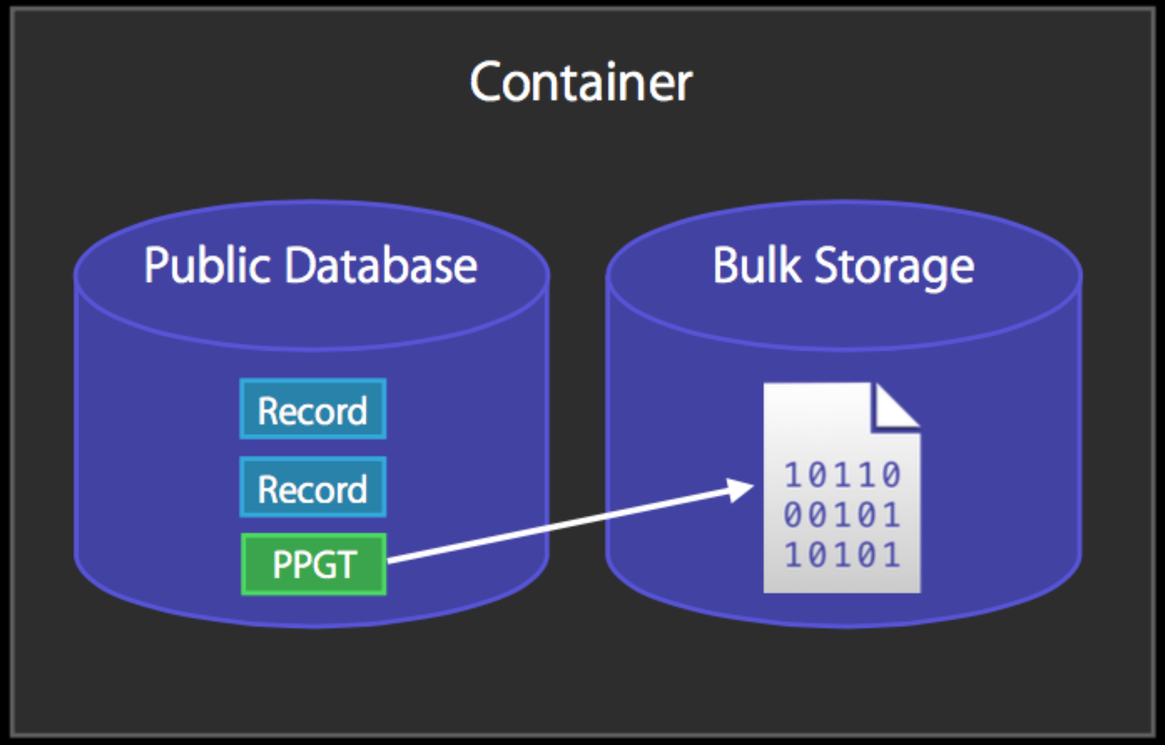


- You can get all friends who use the same app (discover users api)

ASSETS

ASSETS

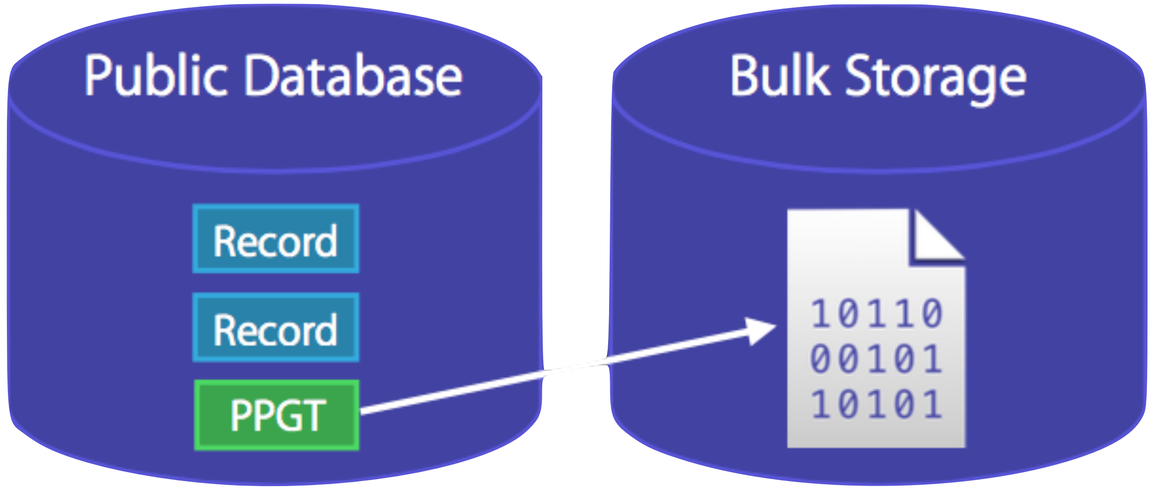
- CKAssets is bulk storage for CloudKit



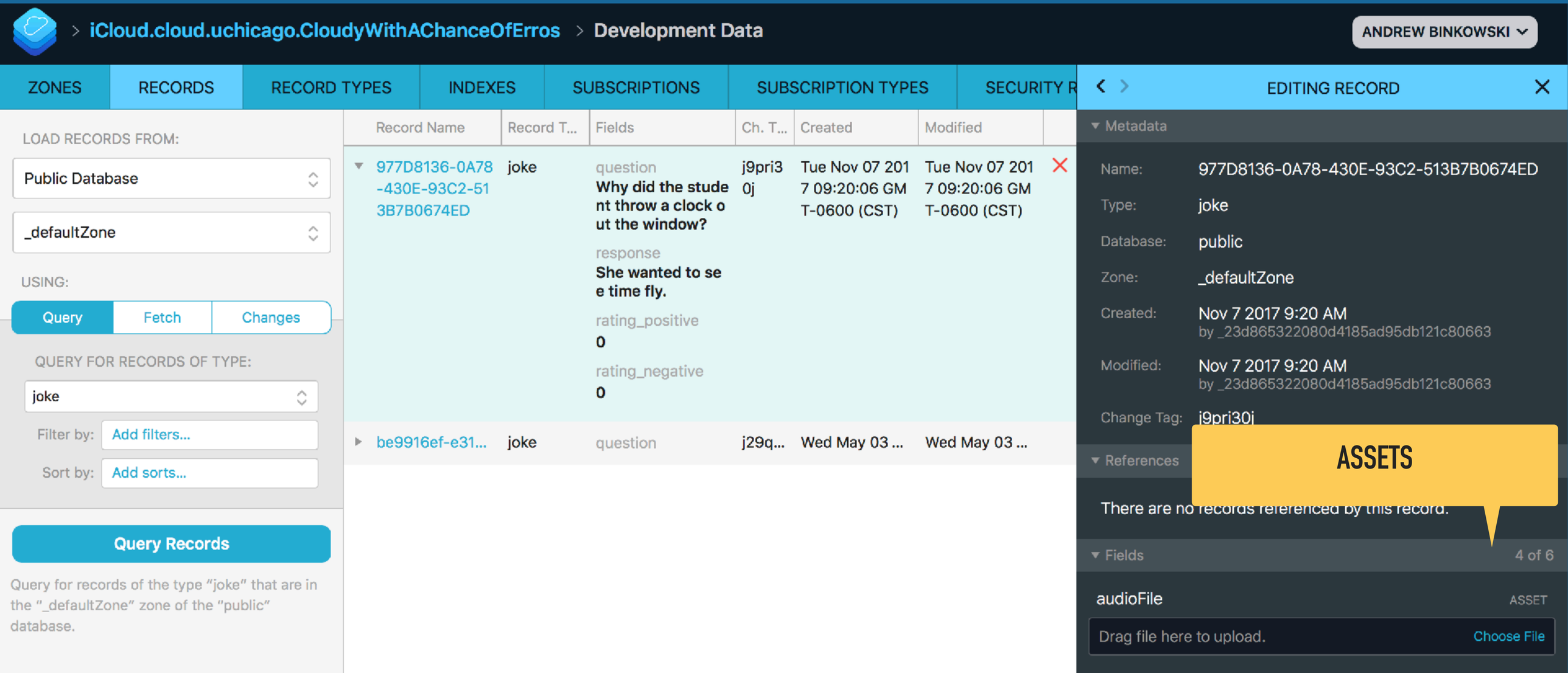
ASSETS

SUBTITLE

- CKAsset
  - Large, unstructured data
  - Files on disk
  - Owned by a CKRecord
  - Garbage collected
    - Deletes with a record
  - Efficient uploads and downloads



ASSETS


iCloud.cloud.uchicago.CloudyWithAChanceOfErros
Development Data
ANDREW BINKOWSKI
ZONES
RECORDS
RECORD TYPES
INDEXES
SUBSCRIPTIONS
SUBSCRIPTION TYPES
SECURITY R
LOAD RECORDS FROM:
Public Database
_defaultZone
USING:
Query
Fetch
Changes
QUERY FOR RECORDS OF TYPE:
joke
Filter by:
Add filters...
Sort by:
Add sorts...
Query Records
Query for records of the type "joke" that are in the "_defaultZone" zone of the "public" database.
Record Name
Record T...
Fields
Ch. T...
Created
Modified
977D8136-0A78-430E-93C2-513B7B0674ED
joke
question
Why did the student throw a clock out the window?
response
She wanted to see time fly.
rating_positive
0
rating_negative
0
j9pri30j
Tue Nov 07 2017 09:20:06 GMT-0600 (CST)
Tue Nov 07 2017 09:20:06 GMT-0600 (CST)
be9916ef-e31...
joke
question
j29q...
Wed May 03 ...
Wed May 03 ...
EDITING RECORD
Metadata
Name: 977D8136-0A78-430E-93C2-513B7B0674ED
Type: joke
Database: public
Zone: _defaultZone
Created: Nov 7 2017 9:20 AM by _23d865322080d4185ad95db121c80663
Modified: Nov 7 2017 9:20 AM by _23d865322080d4185ad95db121c80663
Change Tag: j9pri30j
References
There are no records referenced by this record.
Fields
4 of 6
audioFile
ASSET
Drag file here to upload.
Choose File
ASSETS

ASSETS

```
do {
    let data = UIImagePNGRepresentation(myImage)!
    try data.writeToURL(tempURL, options: NSDataWritingOptions.AtomicWrite)
    let asset = CKAsset(fileURL: tempURL)
    record["myImageKey"] = asset
}
catch {
    print("Error writing data", error)
}
```

NEED TO CONVERT FILE TO A CKASSET AND THEN ASSOCIATE WITH A RECORD

ASSETS

```
if let asset = record["myImageKey"] as? CKAsset,
       data = NSData(contentsOfURL: asset.fileURL),
       image = UIImage(data: data)
{
    // Do something with the image
}
```

RETRIEVE AN IMAGE ASSOCIATED WITH A RECORD;
RETURNED A FILE THAT NEEDS TO BE ACTED ON IMMEDIATELY

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 3