



# ADVANCED iOS APPLICATION DEVELOPMENT

---

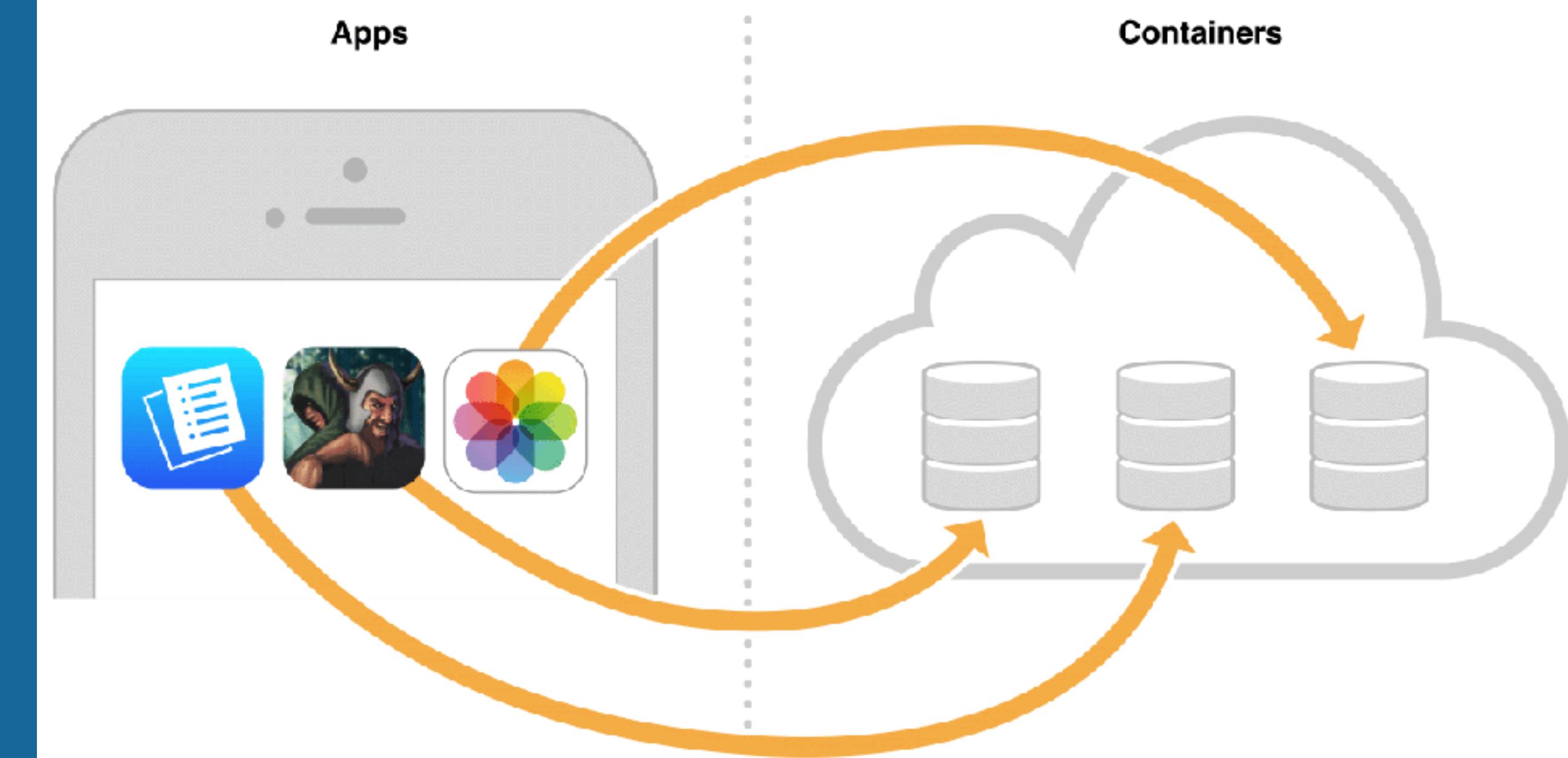
MPCS 51032 • SPRING 2020 • SESSION 3

# CONTAINER

# CLOUDKIT OBJECTS

## CONTAINER

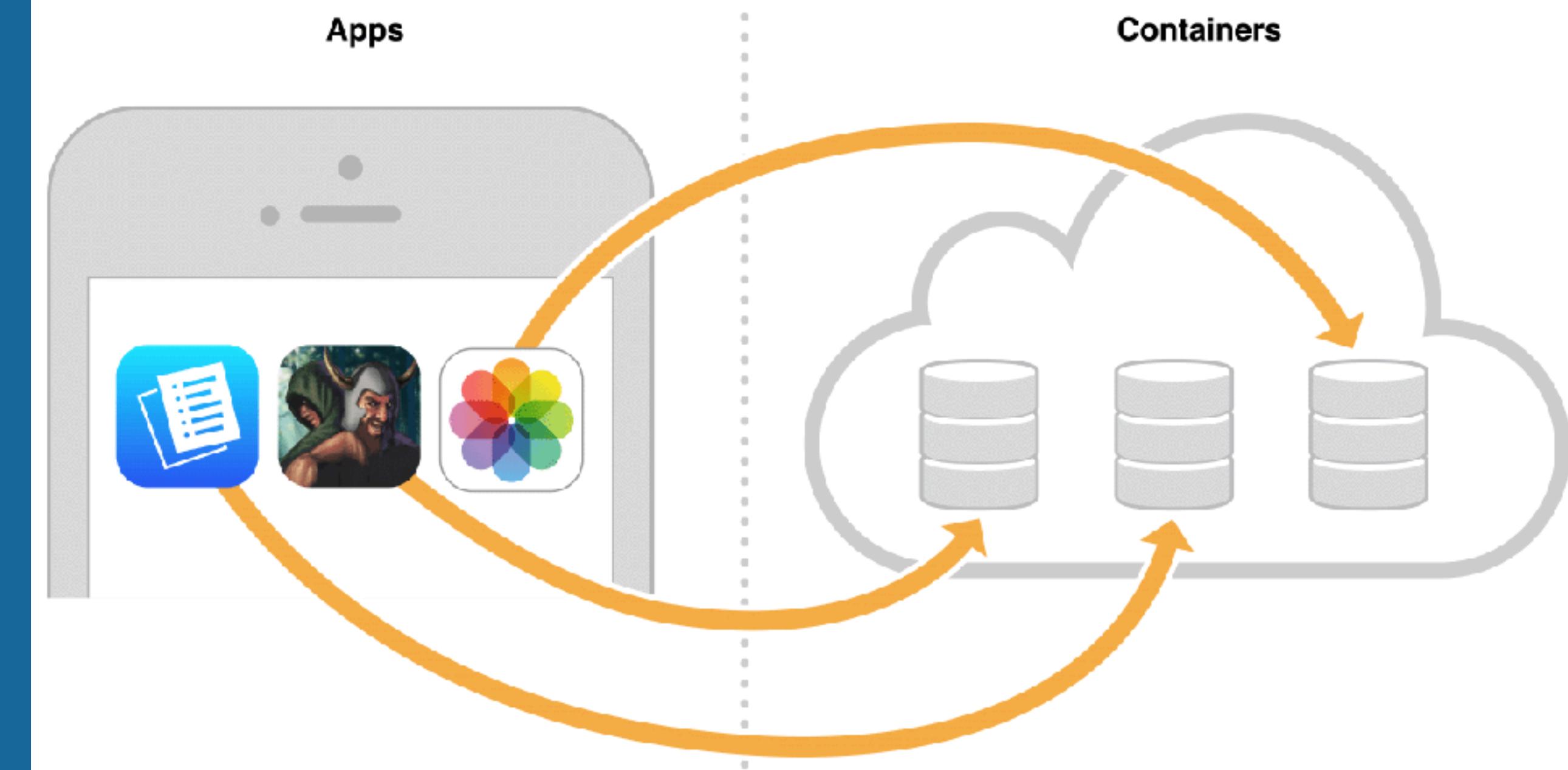
- CKContainer
  - One default container per application
  - Data segregation
  - User encapsulation
  - Managed by the developer via portal



# CLOUDKIT OBJECTS

## CONTAINER

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



# CLOUDKIT

CONTAINER

```
import CloudKit
```

IT'S JUST THAT EASY

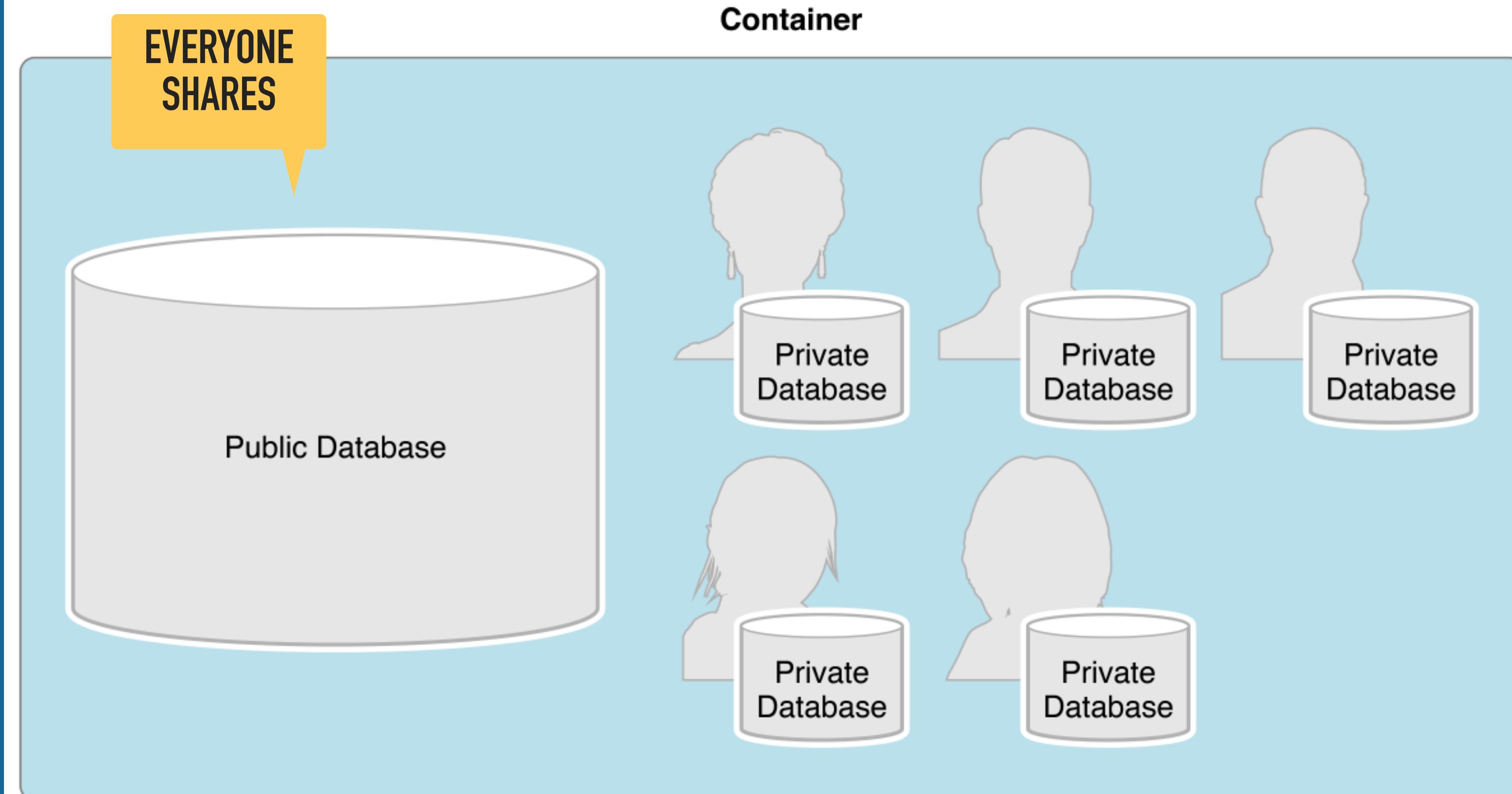
```
let container: CKContainer = CKContainer.default()
```

# DATABASE

# CLOUDKIT OBJECTS

## DATABASE

- CKDatabase
  - Public
  - Private
  - Shared



EVERY USER HAS  
THEIR OWN

# 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



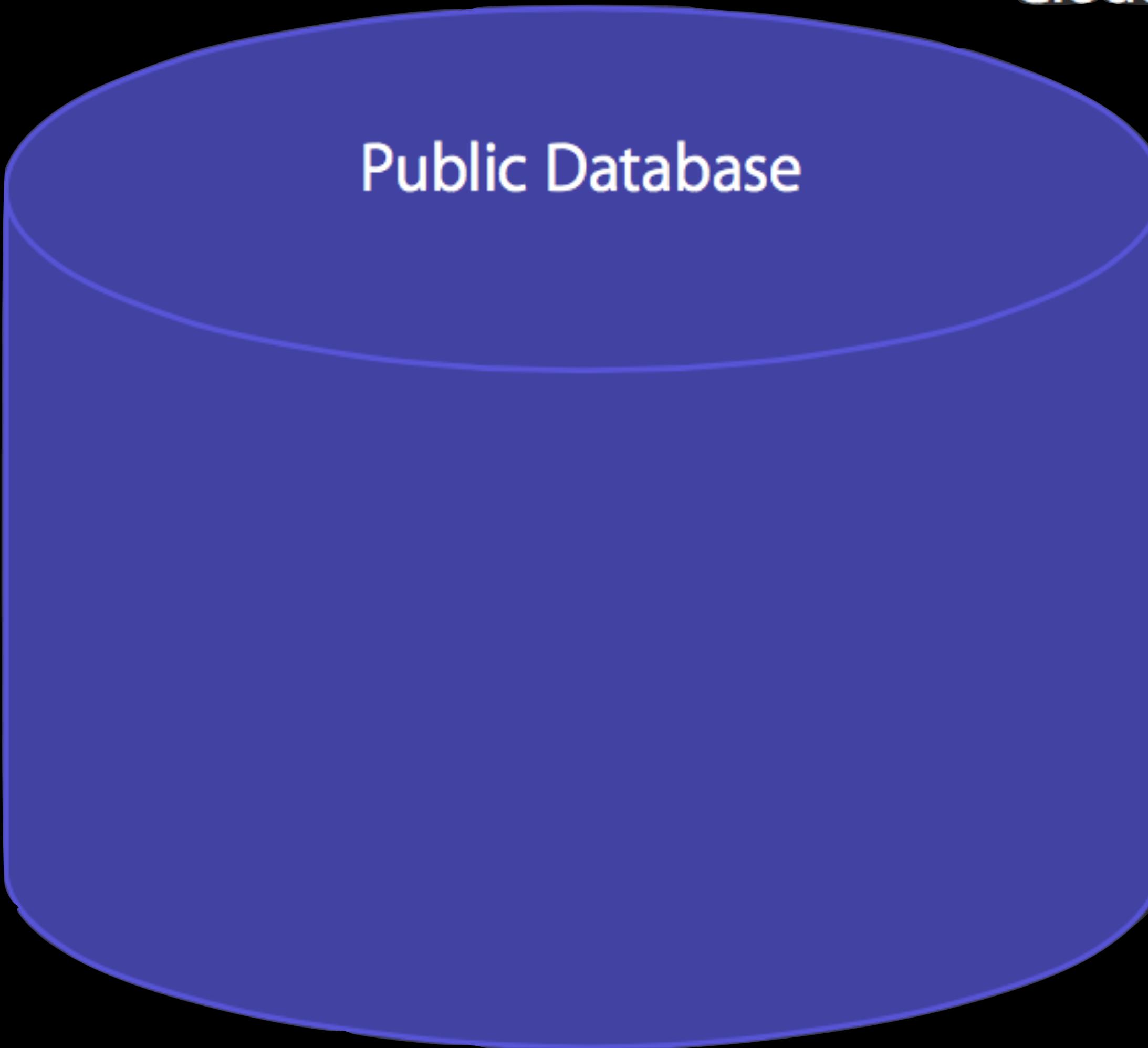
# CLOUDKIT OBJECTS

## DATABASE

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

Public Database

CloudKit Container



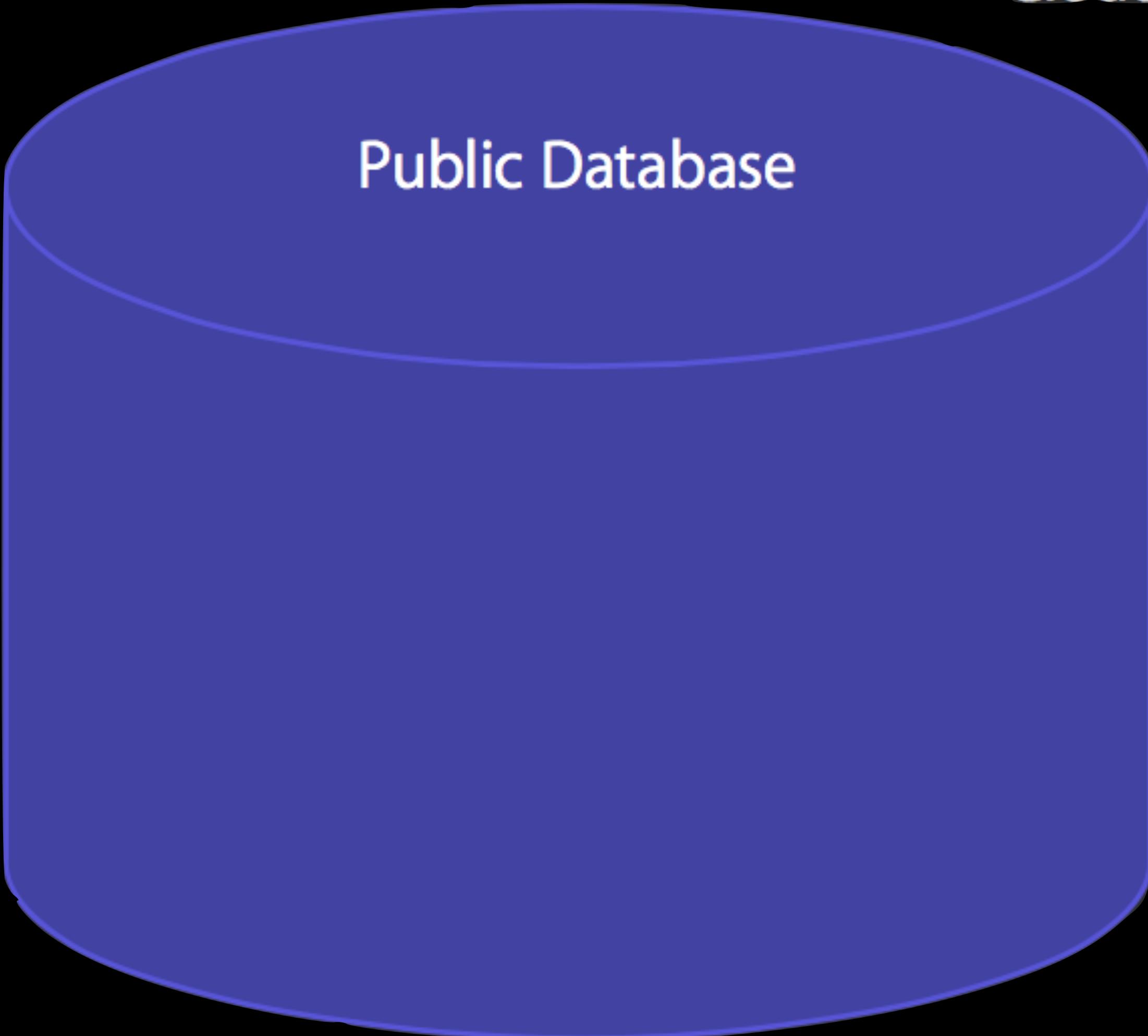
# CLOUDKIT OBJECTS

## DATABASE

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

Public Database

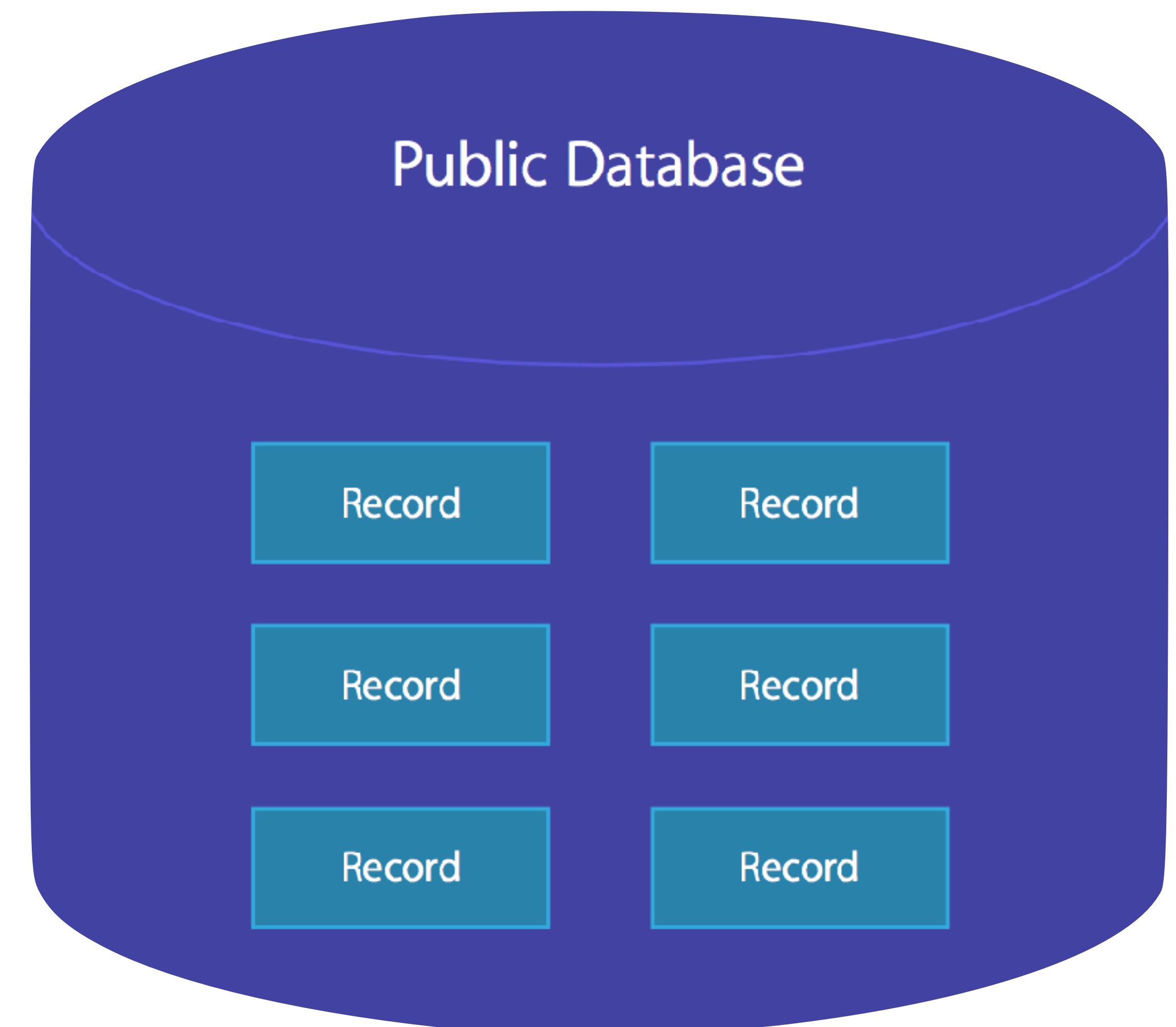
CloudKit Container



# CLOUDKIT OBJECTS

DATABASE  
SUBTITLE

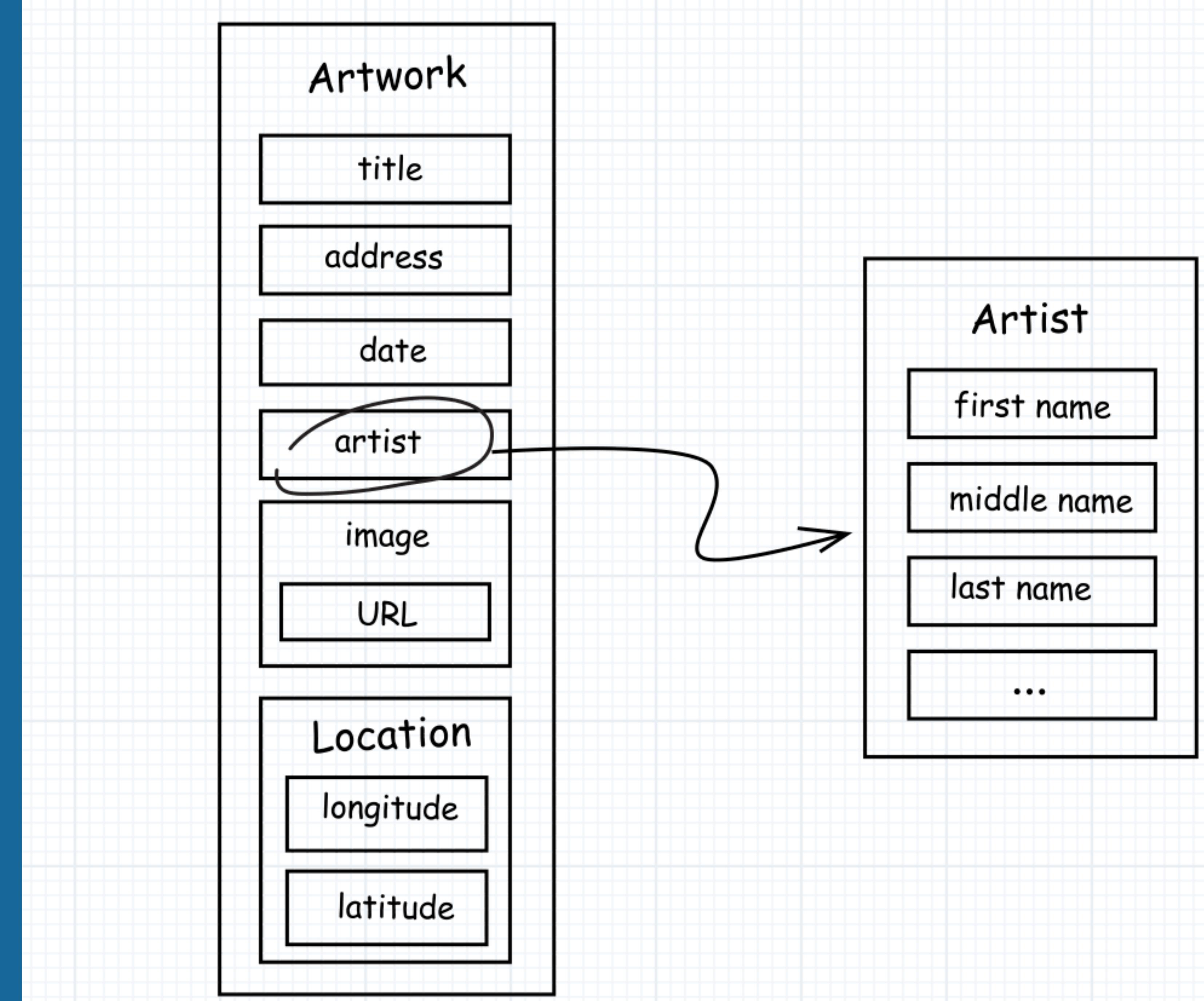
- Structured Data
  - Wraps key/value pairs
  - Types and references
  - Just-in-time schema
  - Metadata
    - Created, edited, etc.



# CLOUDKIT OBJECTS

## DATABASE

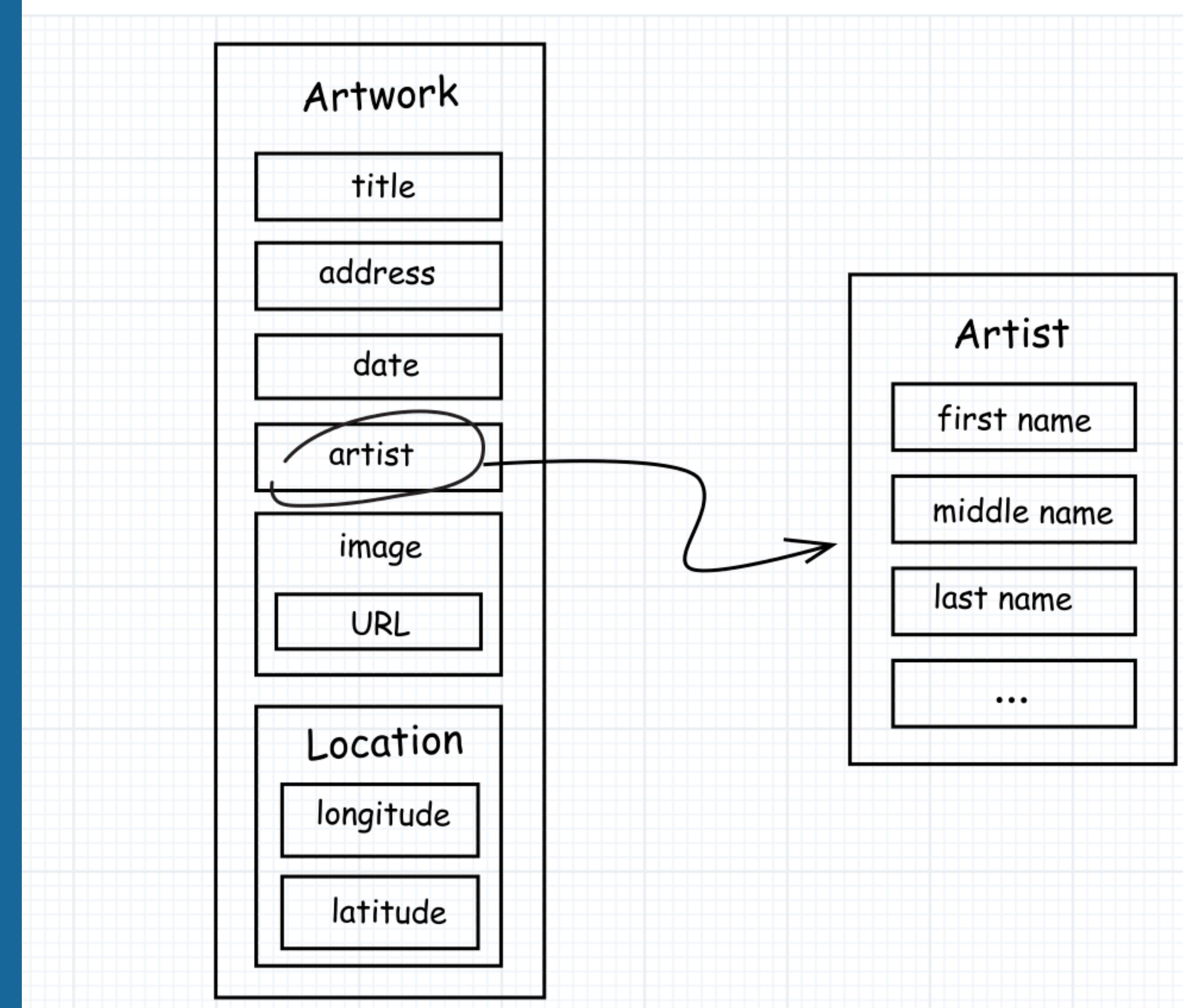
- Schema is built on-demand
- You can edit it in the console



# CLOUDKIT OBJECTS

## DATABASE

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

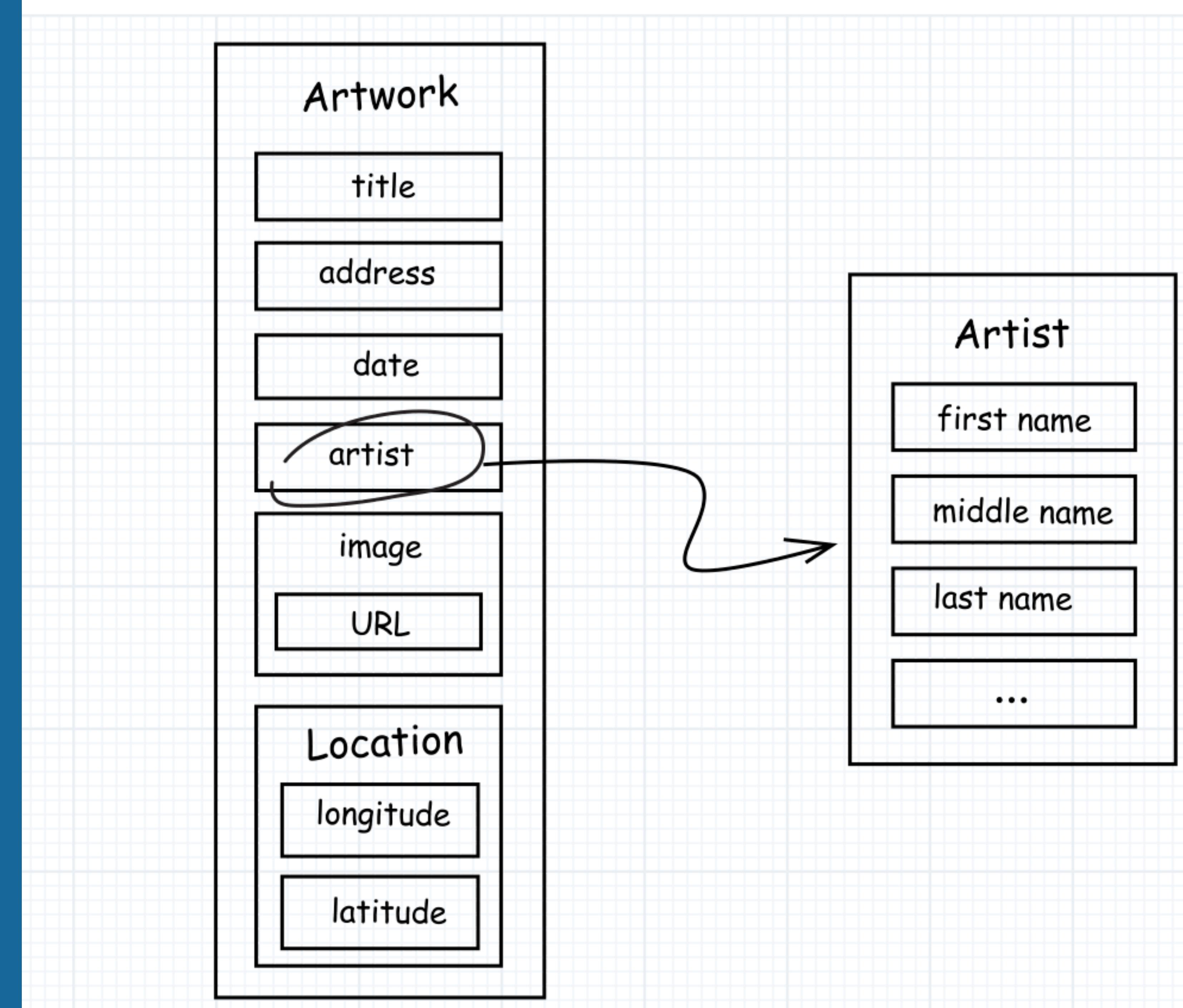


# RECORDS

# CLOUDKIT OBJECTS

## RECORDS

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

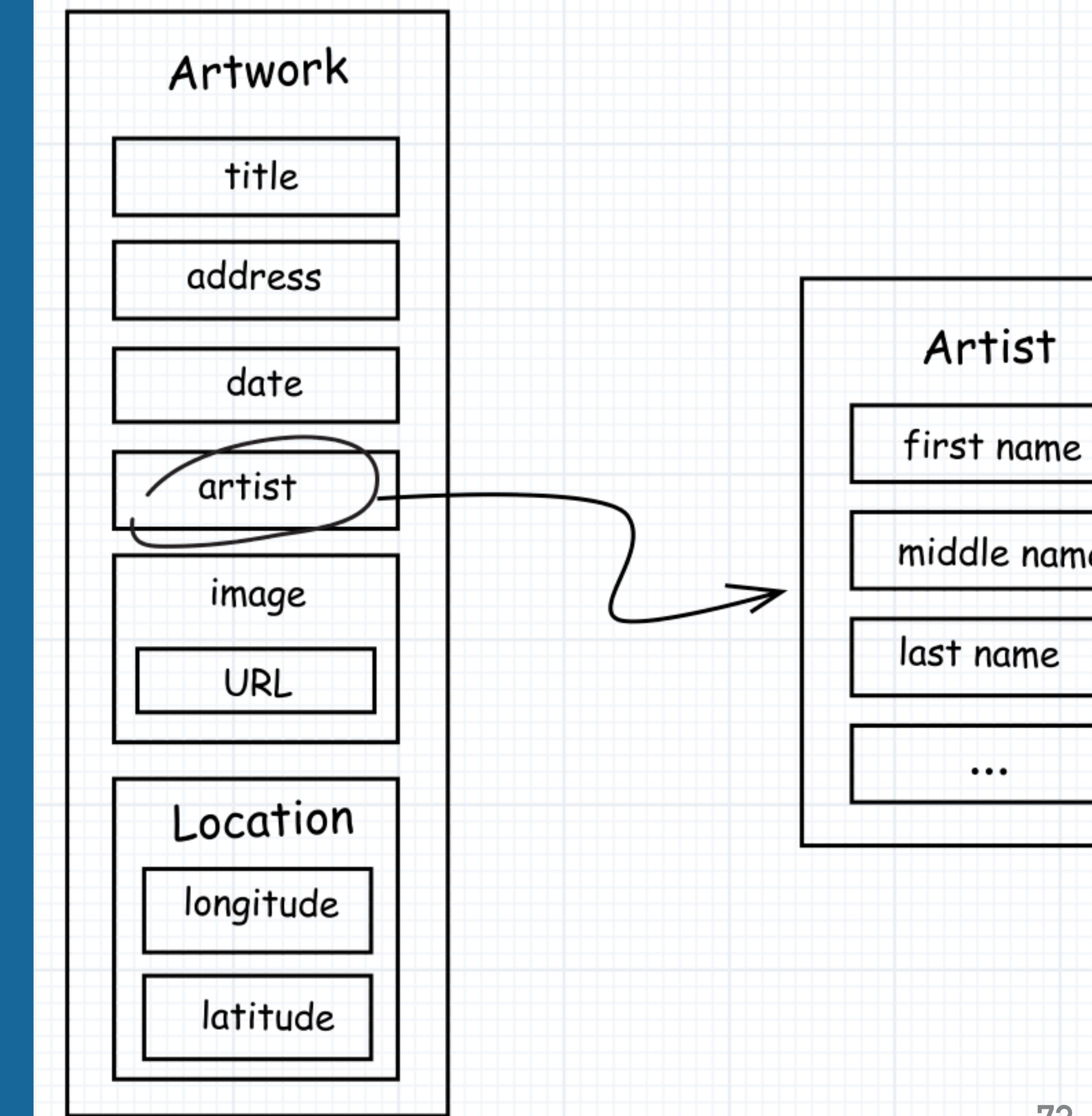


# CLOUDKIT OBJECTS

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] = value)
   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

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.

METADATA ON  
EVERY RECORD

Type: joke  
Database: public  
Zone: \_defaultZone  
Created: Nov 4 2019 9:14 PM  
by Andrew Binkowski  
(\_8a76cb86f6390ecf95f548796a270cc8)

Modified: Nov 4 2019 9:14 PM  
by Andrew Binkowski  
(\_8a76cb86f6390ecf95f548796a270cc8)

Change Tag: k2la507p

**References**

None ⓘ

**Custom Fields**

4 of 4 ⓘ

question

String

test

rating\_negative

Int(64)

1

# CLOUDKIT OBJECTS

## RECORDS

**Security Roles** ▾

Default Roles

World

Authenticated

Creator

---

Custom Roles

Security roles allow you to control permissions in the public database.

---

 **Create New Role**

 **Delete Role**

Record Type	Permissions
Users	Create <input checked="" type="checkbox"/> Read <input checked="" type="checkbox"/> Write

**NO WRITE BY  
DEFAULT**

# CLOUDKIT OBJECTS

## RECORDS

### Record Types ▾

System Types

Users

Custom Types

joke



New Type



Delete Type

#### System Fields

recordName	Reference	Queryable	
createdBy	Reference	Queryable	
createdAt	Date/Time	Queryable	
modifiedBy	Reference	None	
modifiedAt	Date/Time	None	
changeTag		None	
Custom Fields	RECORDS ARE DEFINED PROGRAMMATICALLY OR IN THE CONSOLE		
question		Queryable, Searchable, Sort...	-
rating_negative		Queryable, Sortable	-
rating_positive	Int(64)	Queryable, Sortable	-
response	String	Queryable, Searchable, Sort...	-

⊕ Add Field

↳ Edit Indexes

# 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
}
```

CREATE A RECORD

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

# 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().privateCloudDatabase

/// 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

```
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

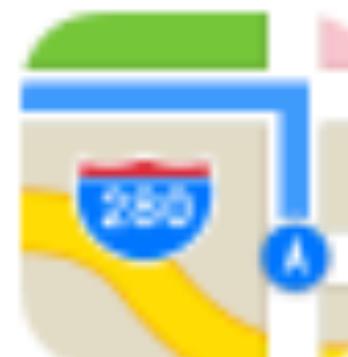


## General



## Privacy

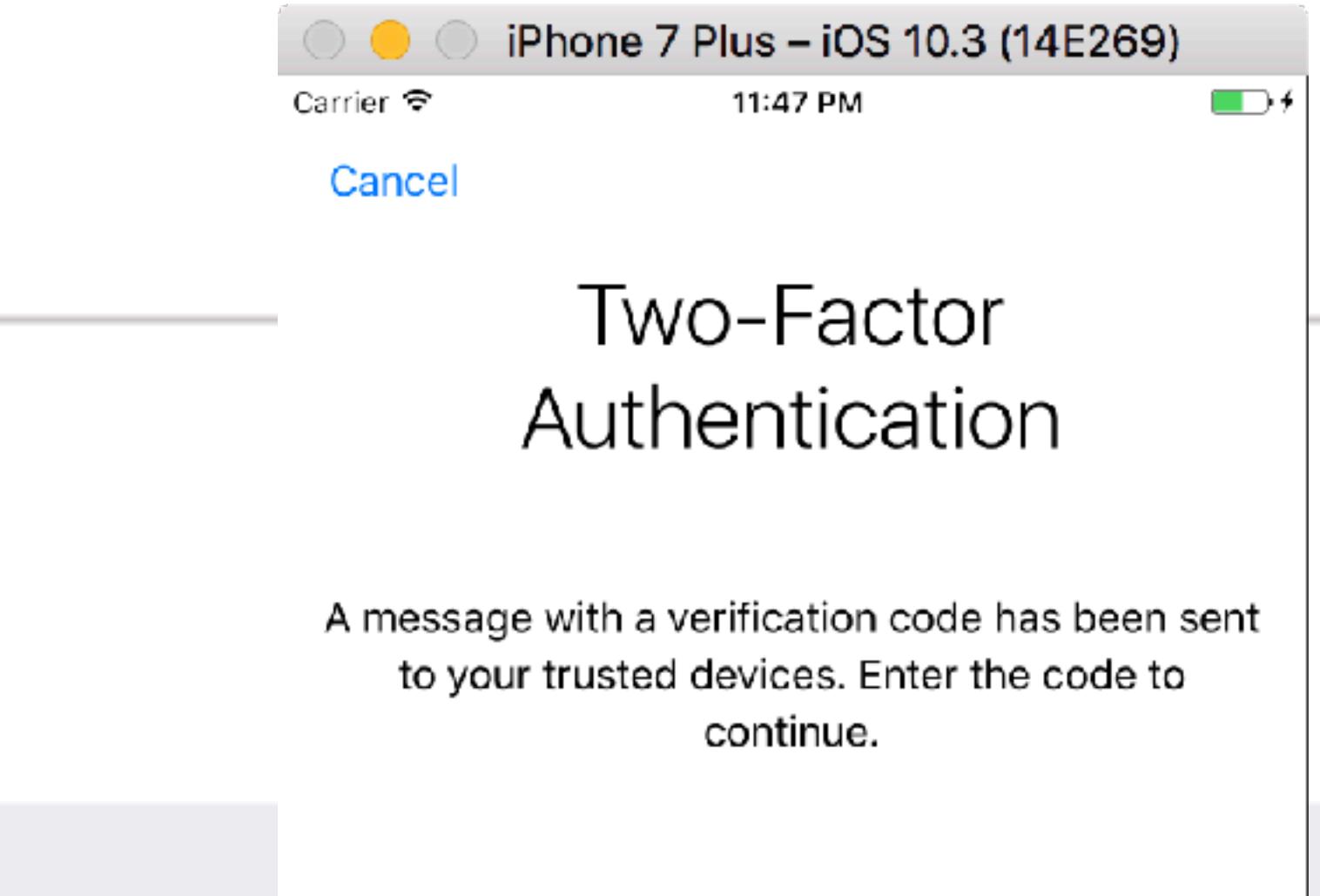
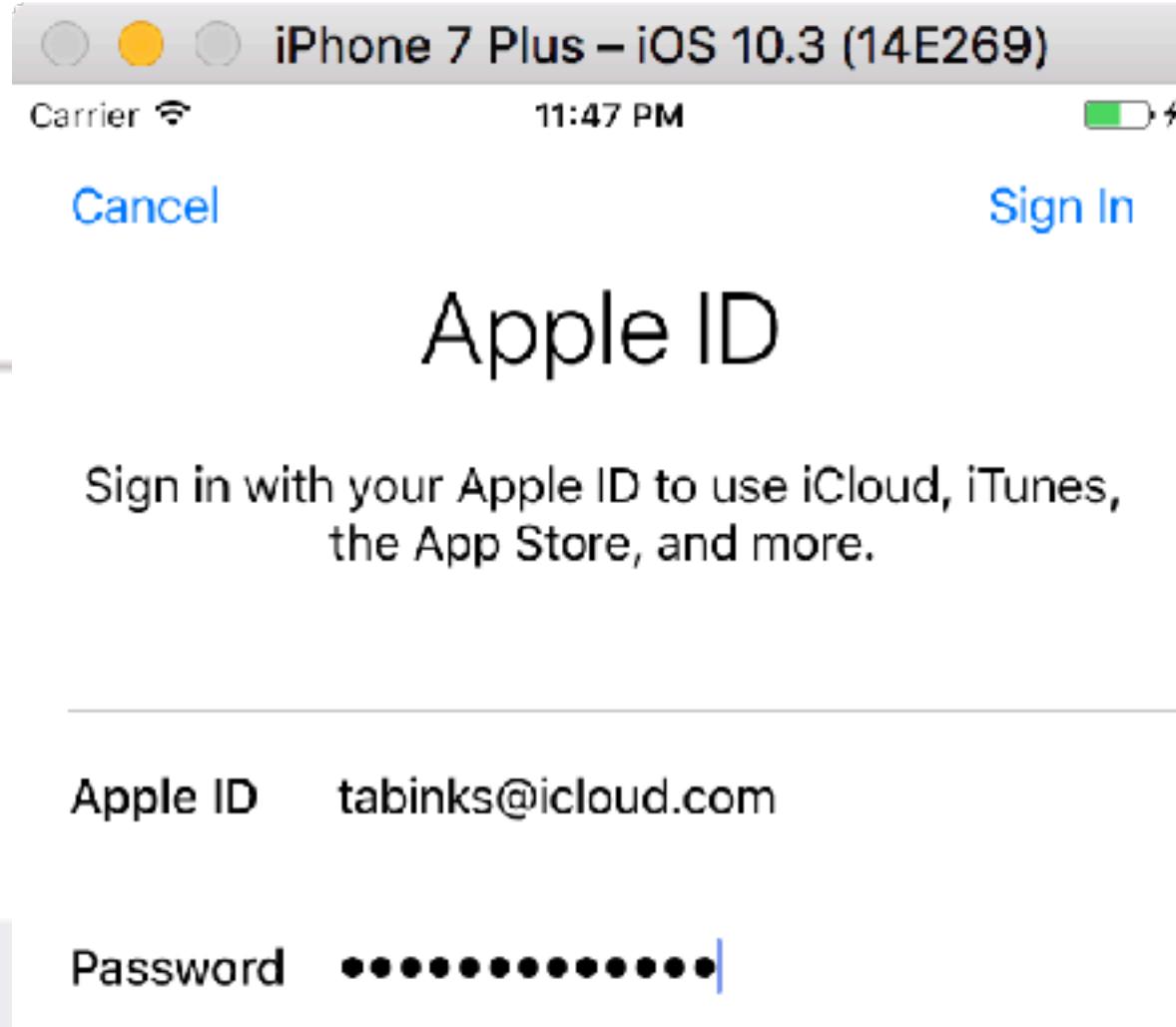
NEED TO LOGIN WITH  
ICLOUD ID



## Maps



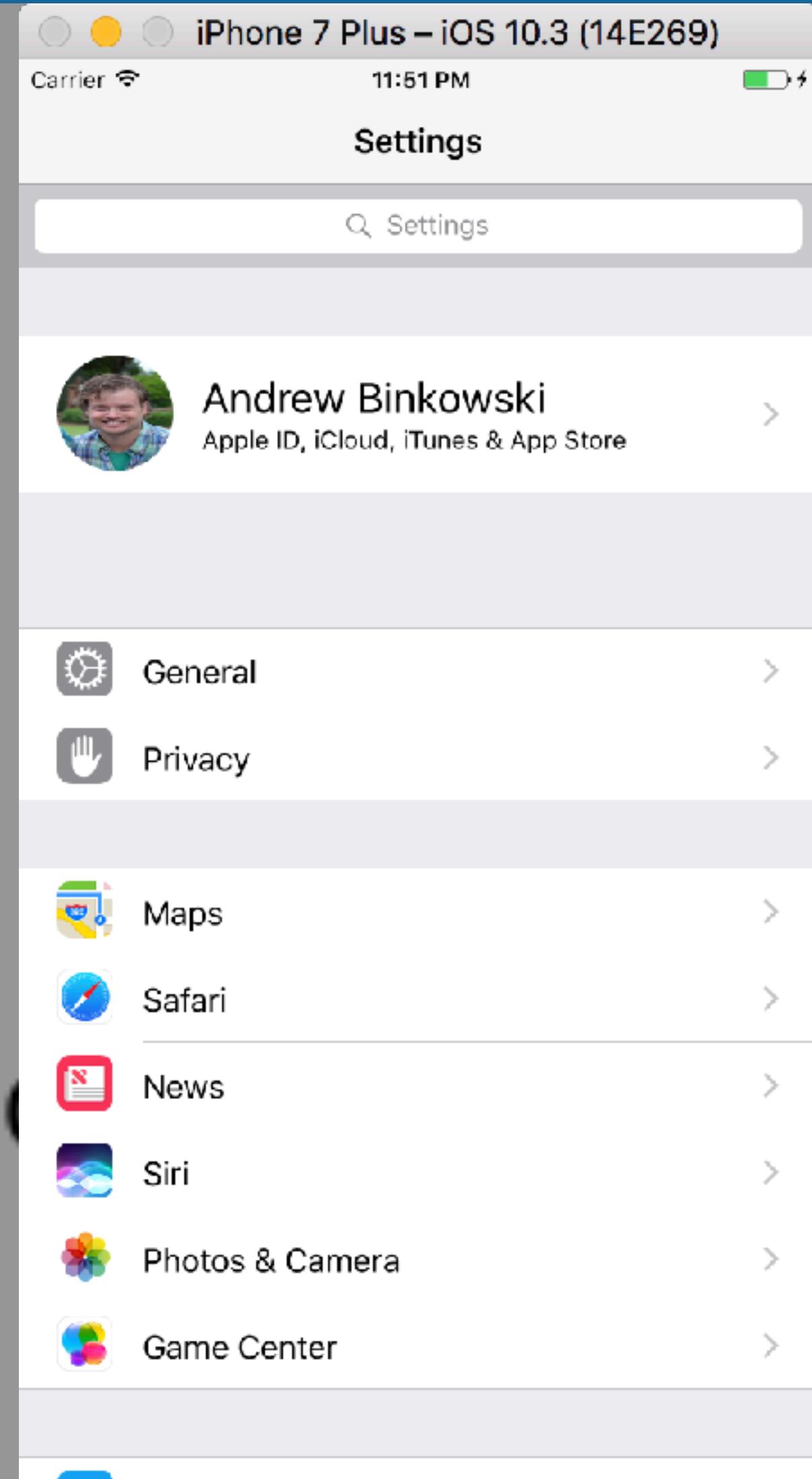
## Safari



# CLOUDKIT OBJECTS

## RECORDS

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



# CLOUDKIT OBJECTS

## RECORDS

```
    // saveCloudKit("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: "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)")  
        }  
    }  
}
```



The screenshot shows the Xcode interface with the CloudyWithAChanceOfErrors project selected. In the bottom left, the CloudKit objects list displays a single record: "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>". The record ID is 025F8A03-2047-4B26-98A0-A97CCA823DDB, and the record type is joke.

```
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
▶ OFEB50A6-282E-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

iCloud.mobi.uchicago.twenty-nineteen-cloudkit › ● Development › Schema › Andrew Binkowski | ?

### Record Types ▾

- System Types
- Users
- Custom Types
- joke

[New Type](#) [Delete Type](#)

Field Name	Field Type	Indexes
<b>System Fields</b>		
recordName	Reference	Queryable
createdBy	Reference	Queryable
createdAt	Date/Time	Queryable
modifiedBy	Reference	None
modifiedAt	Date/Time	None
changeTag	String	None
<b>Custom Fields</b>		
question		Queryable, Searchable, Sort...
rating_negative		Queryable, Sortable
rating_positive	Int(64)	Queryable, Sortable
response	String	Queryable, Searchable, Sort...

CHANGE SCHEMA FROM  
CONSOLE OR IN CODE

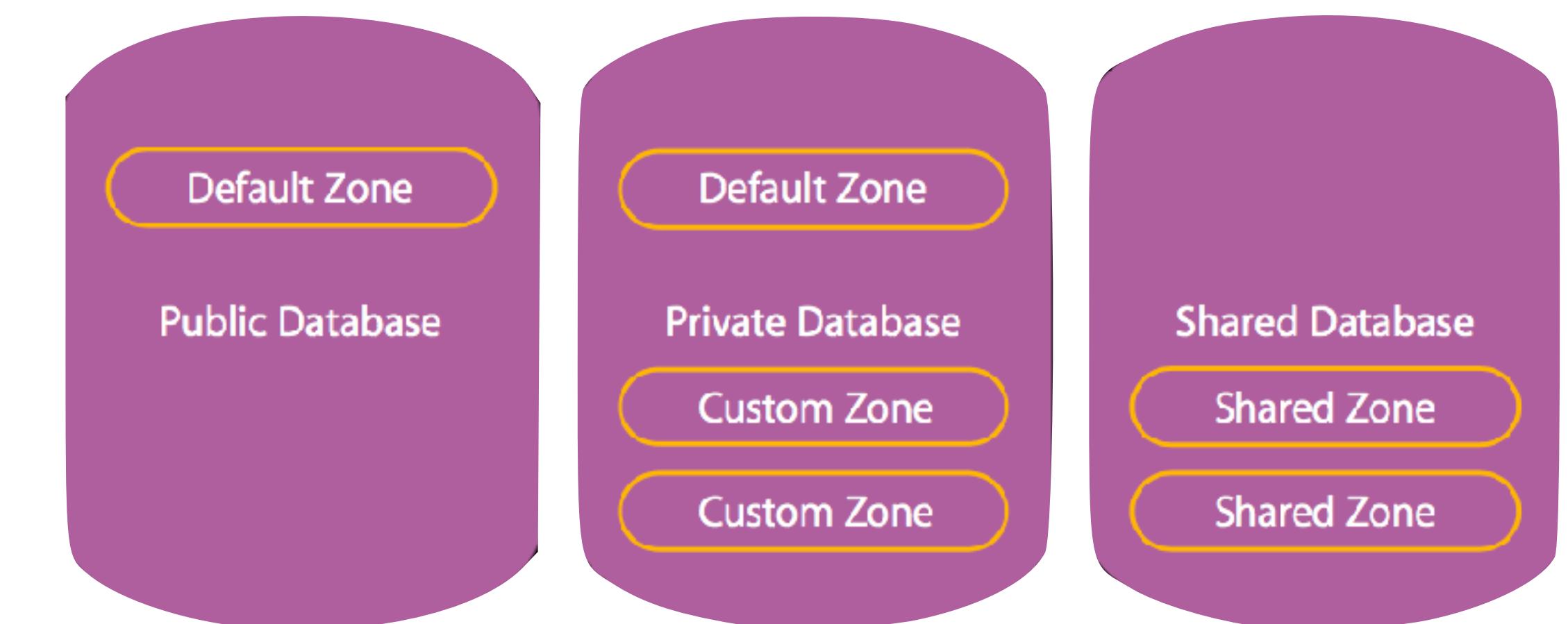
# RECORD ZONES

# CLOUDKIT OBJECTS

## RECORD ZONES

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

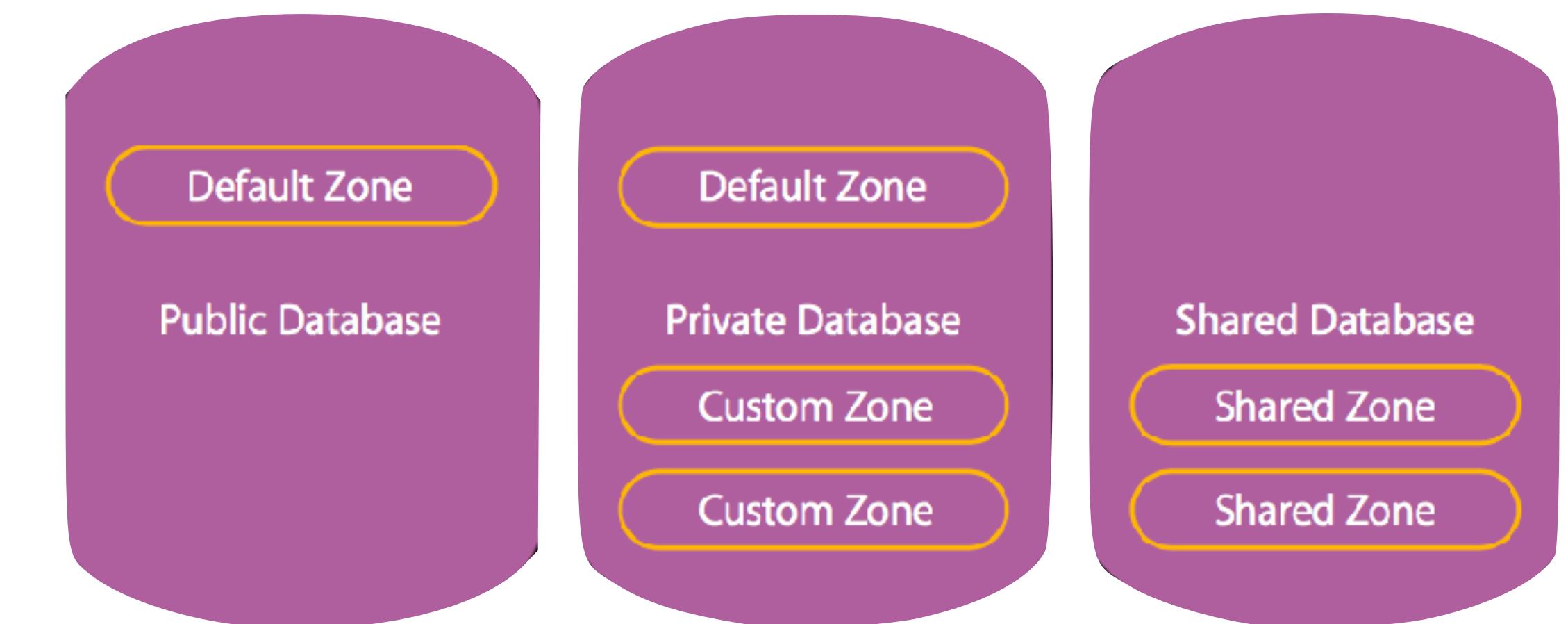
PUBLIC HAS A DEFAULT ZONE



# CLOUDKIT OBJECTS

## RECORD ZONES

- 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/Foundation.h>  
  
@class CKRecordZoneID;  
  
NS_CLASS_AVAILABLE(10_10, 8_0)  
@interface CKRecordID : NSObject <NSSecureCoding>  
  
- (instancetype)init NS_UNAVAILABLE;  
  
/* Record names must be 255 characters or less */  
/* This creates a record ID in the default zone */  
- (instancetype)initWithRecordName:(NSString *)  
- (instancetype)initWithRecordName:(NSString *)  
  
@property (nonatomic, readonly, strong) NSString *  
@property (nonatomic, readonly, strong) CKRecordZoneID  
  
@end
```



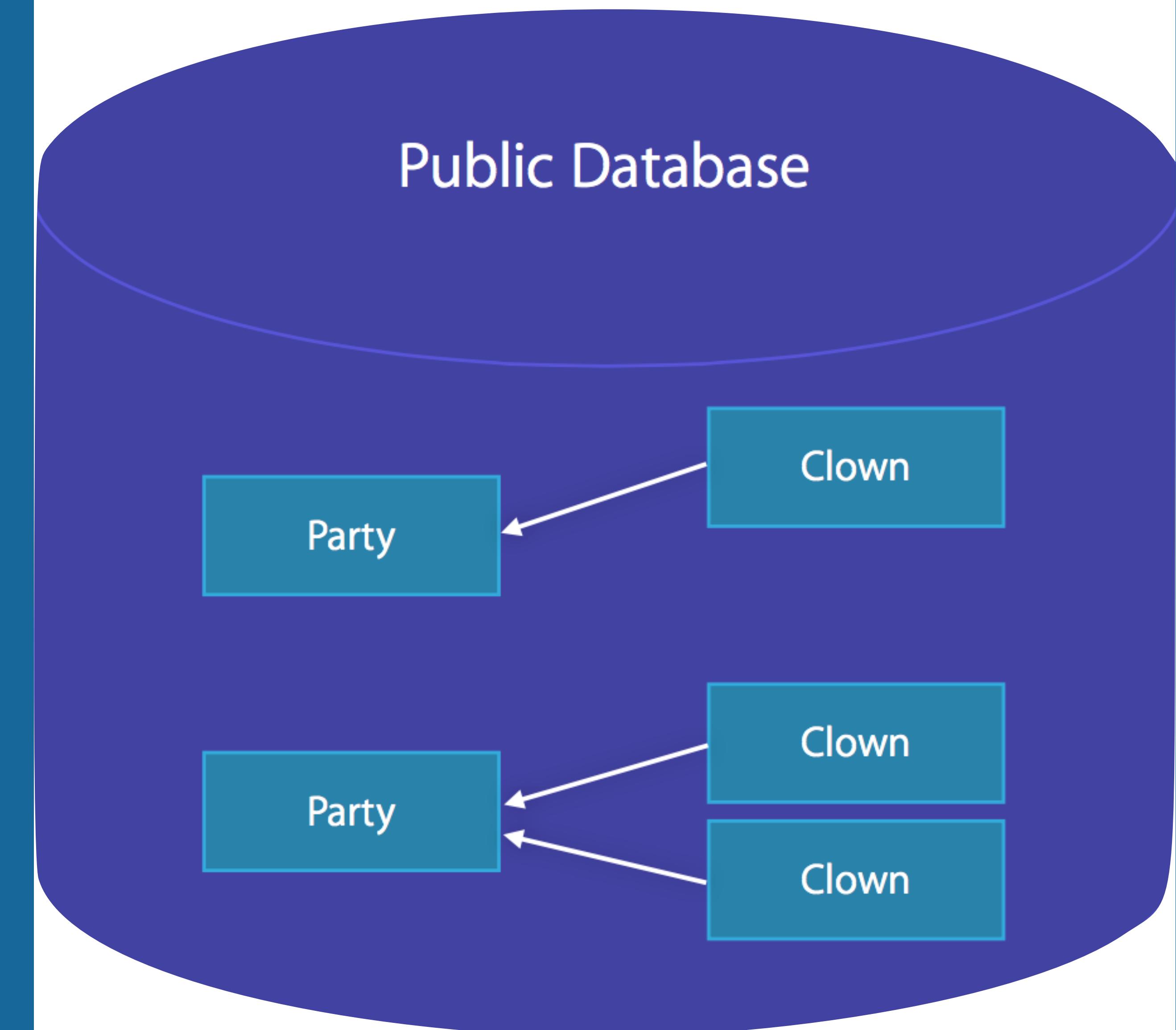
# REFERENCES

# CLOUDKIT OBJECTS

## RECORDS

SUBTITLE

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

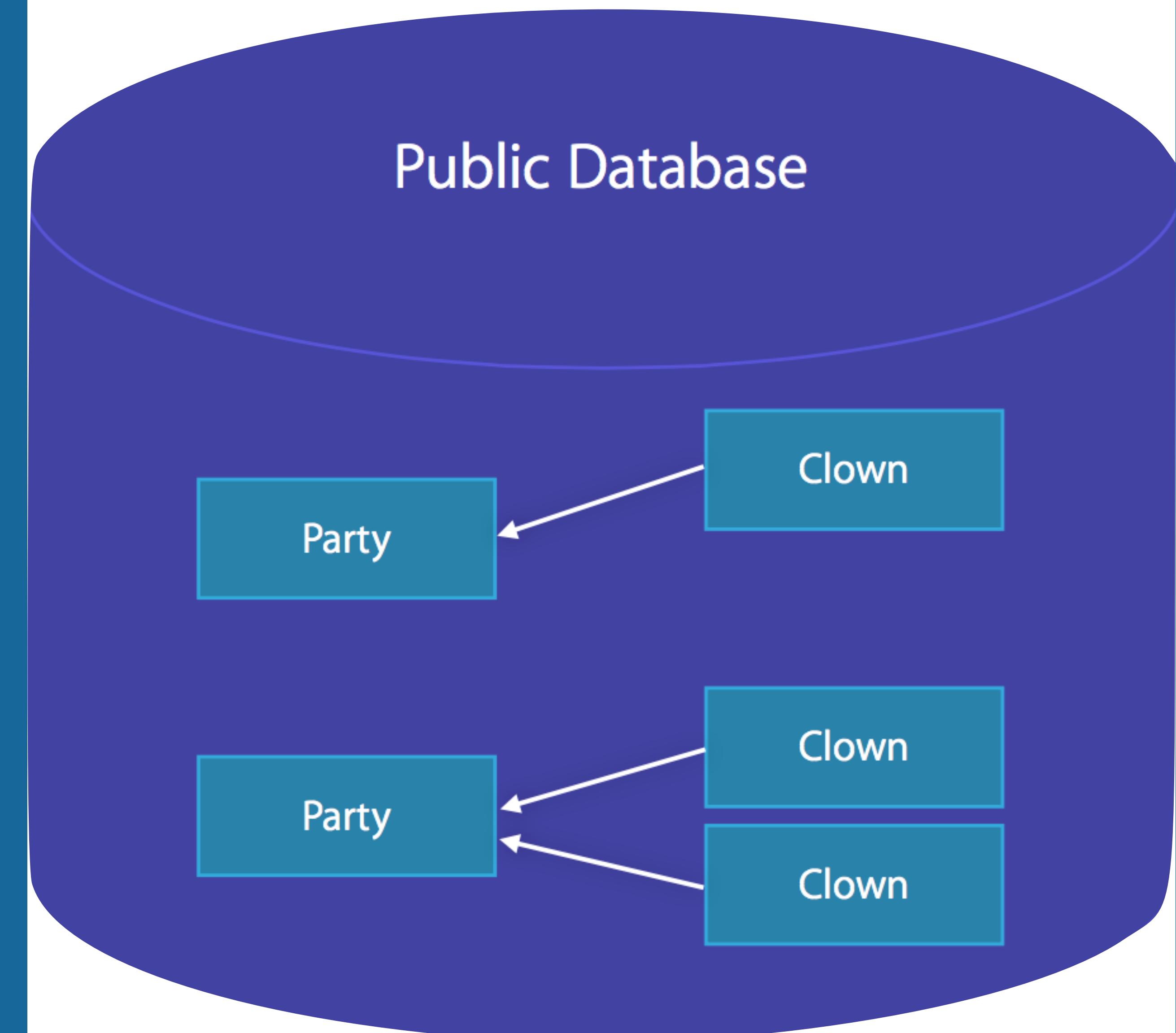


# CLOUDKIT OBJECTS

## 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

iCloud.mobi.uchicago.twenty-nineteen-cloudkit ▾ > ● Development ▾ > Schema ▾ Binkowski ▾ | ?

## Record Types ▾

- System Types
- Users
- Custom Types
- joke

**System Fields**

Field Name	Field Type	
recordName	Reference	Queryable
createdBy	Reference	Queryable
createdAt	Date/Time	Queryable
modifiedBy	Reference	None
modifiedAt	Date/Time	None
changeTag	String	None

**Custom Fields**

question	String	Queryable, Searchable, Sort...	-
rating_negative	Int(64)	Queryable, Sortable	-
rating_positive	Int(64)	Queryable, Sortable	-
response	String	Queryable, Searchable, Sort...	-

**REFERENCE**

+ New Type    - Delete Type

# CLOUDKIT API

iCloud.mobi.uchicago.twenty-nineteen-cloudkit › Development › Schema › Andrew Binkowski | ?

## Record Types ▾

- System Types
- Users
- Custom Types
- joke

**System Fields**

Field Name	Field Type	
recordName	Reference	Queryable
createdBy	Reference	Queryable
createdAt	Date/Time	Queryable
modifiedBy	Reference	None
modifiedAt	Date/Time	None
changeTag	String	None

**Custom Fields**

question	String	Queryable, Searchable, Sort...	-
rating_negative	Int(64)	Queryable, Sortable	-
rating_positive	Int(64)	Queryable, Sortable	-
response	String	Queryable, Searchable, Sort...	-

**USER REFERENCE TO RECORD IS BUILT IN**

**New Type** **Delete Type**

# DATA MODELING

# DATA MODELING

- How should 1 to many relationships be handled?

Album

PhotoArray

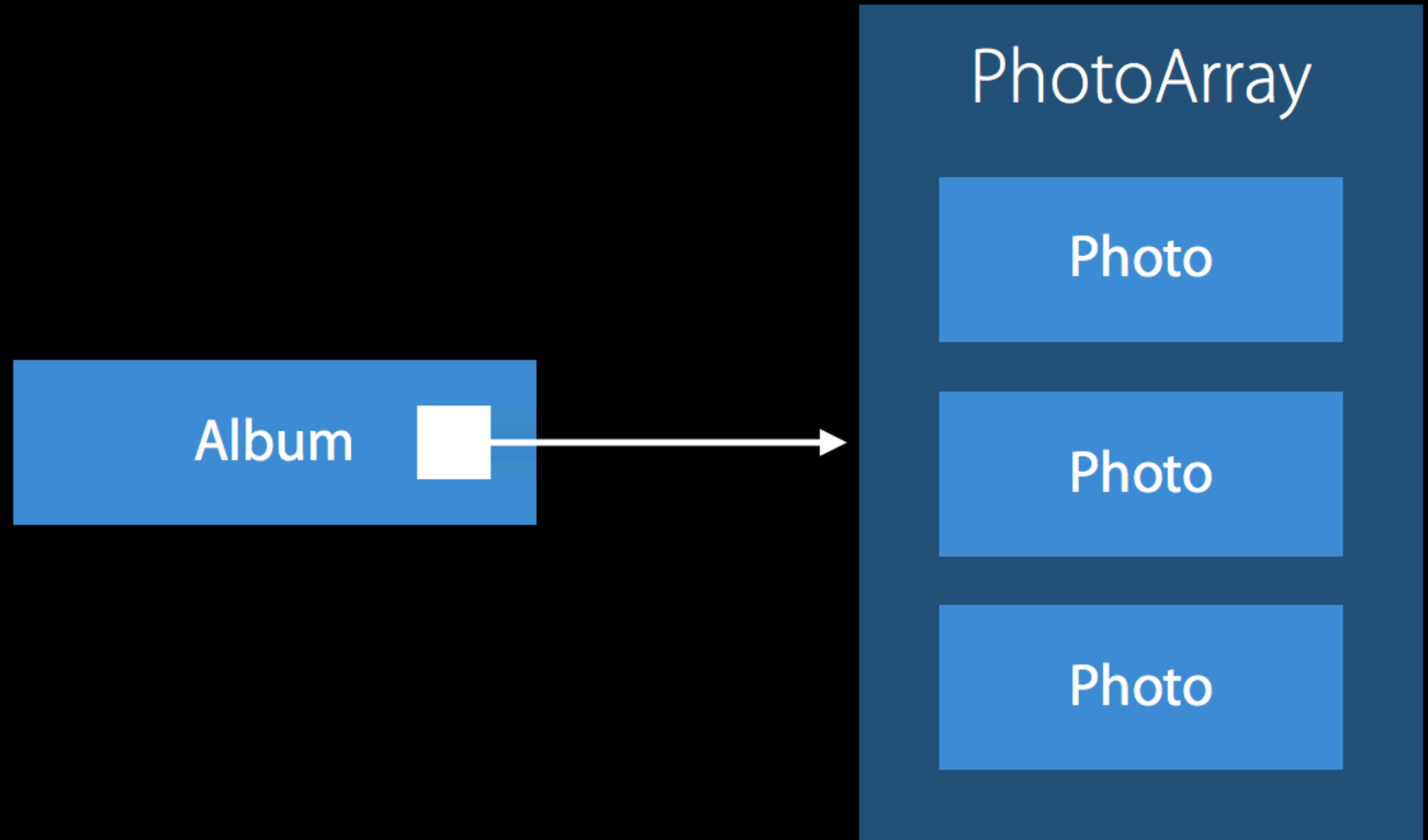
Photo

Photo

Photo

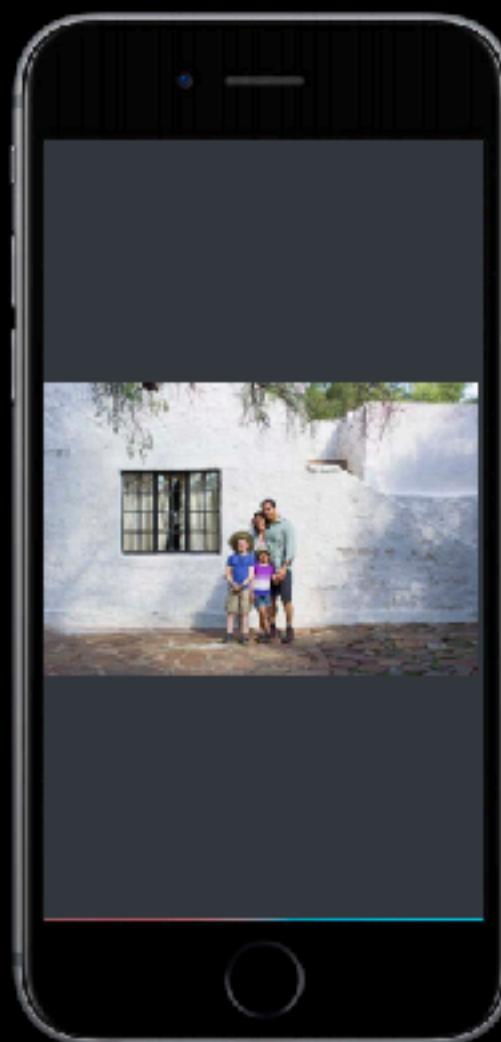
# DATA MODELING

- Typical setup to model

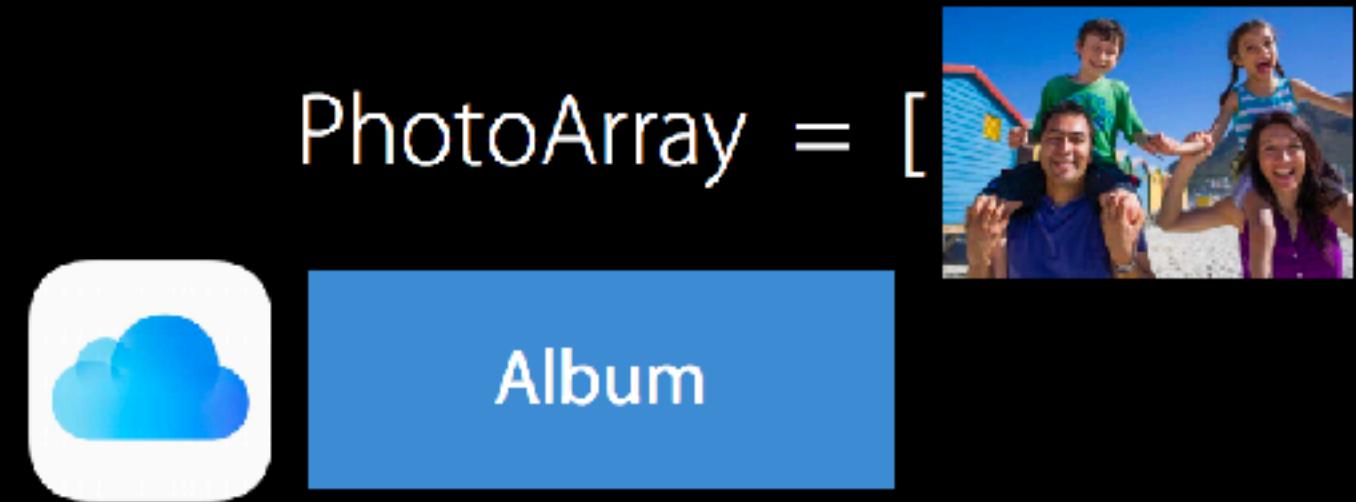


# DATA MODELING

PhotoArray = []



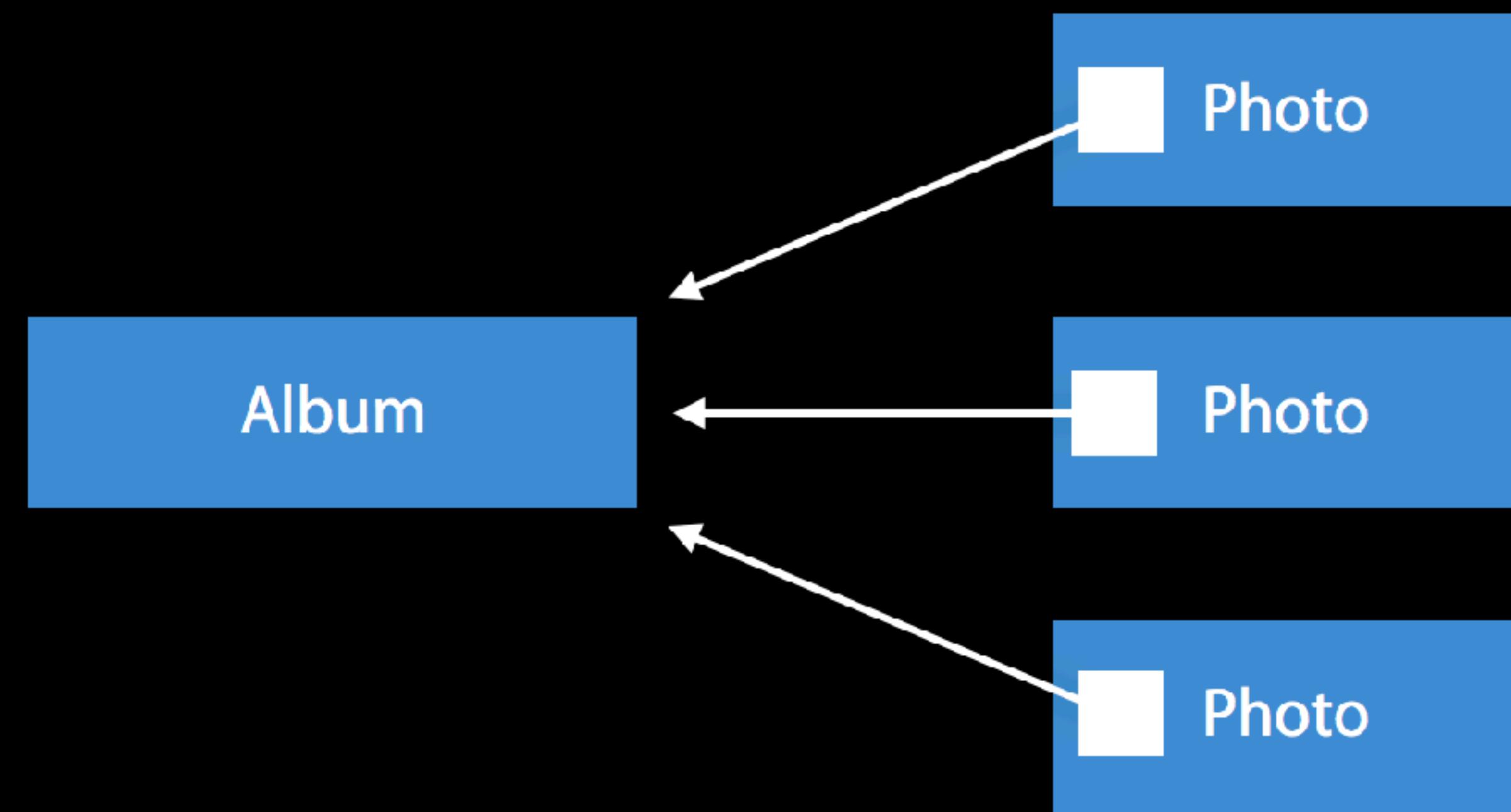
# DATA MODELING



serverRecordChanged



# DATA MODELING

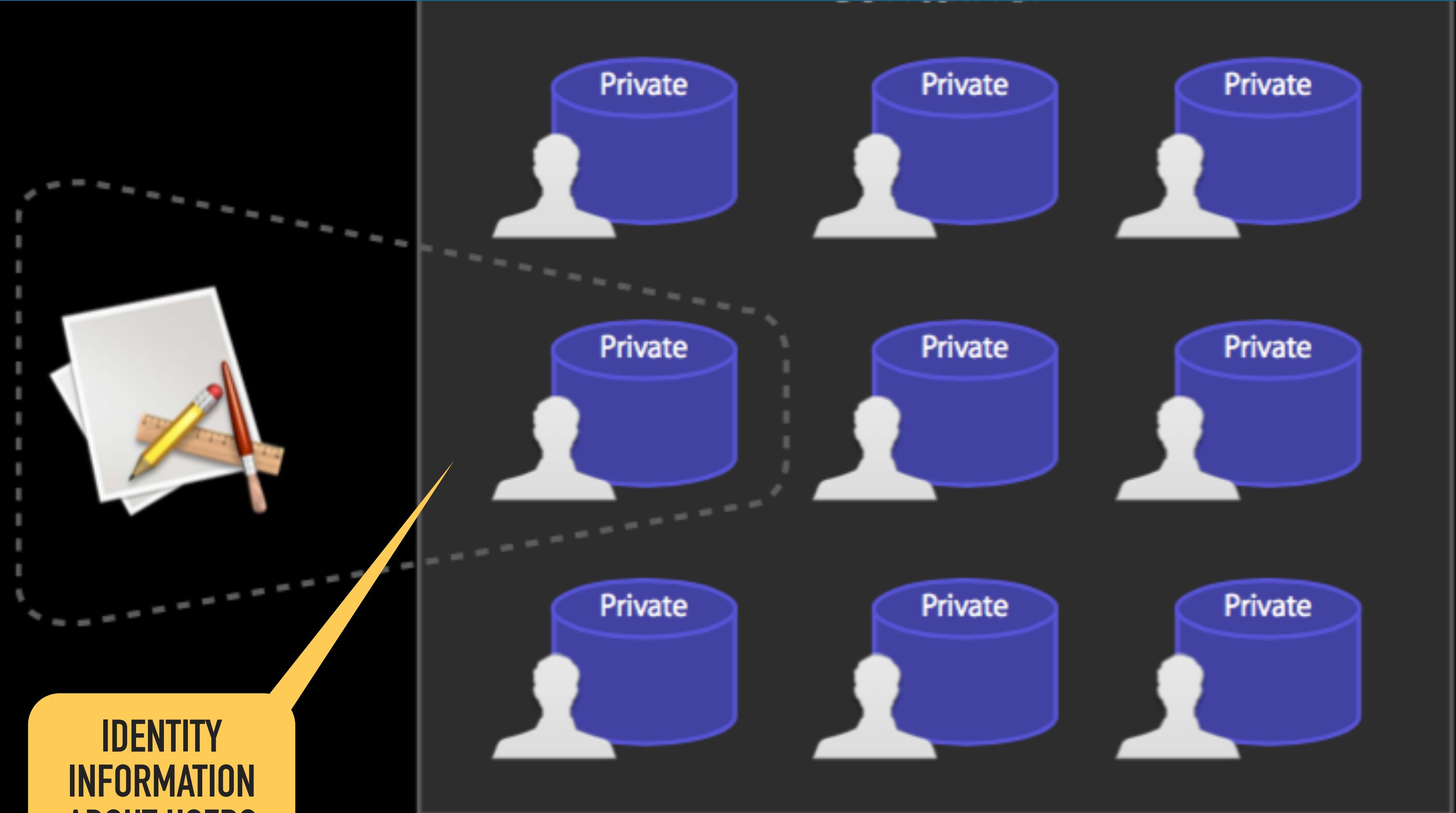


# USER RECORDS

# USER RECORDS

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

# USER RECORDS

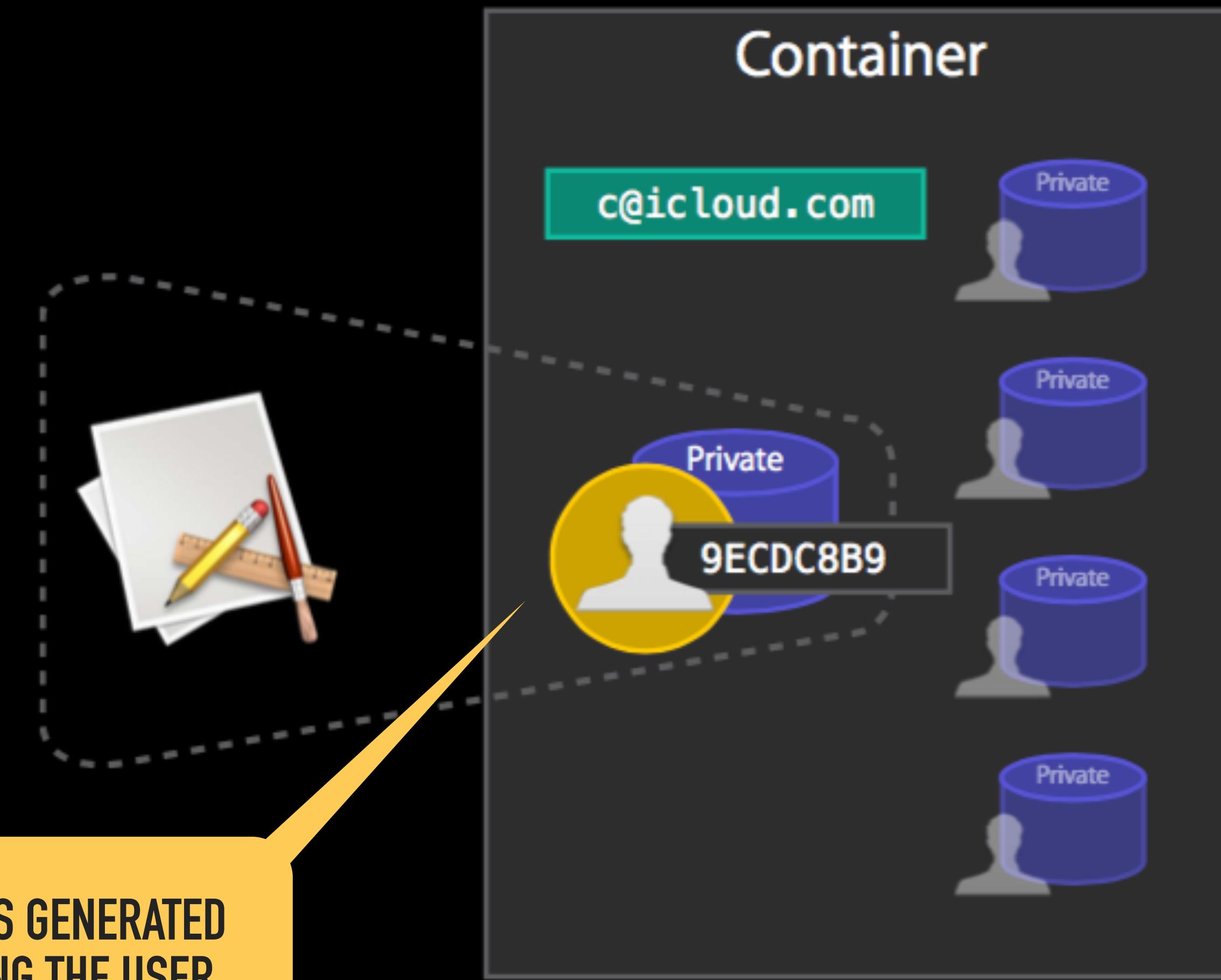


IDENTITY  
INFORMATION  
ABOUT USERS

# USER RECORDS



A UNIQUE ID IS GENERATED  
REPRESENTING THE USER



# USER RECORDS

iCloud.mobi.uchicago.twenty-nineteen-cloudkit › Development › Schema › Andrew Binkowski | ?

## Record Types ▾

System Types

Users

Custom Types

joke

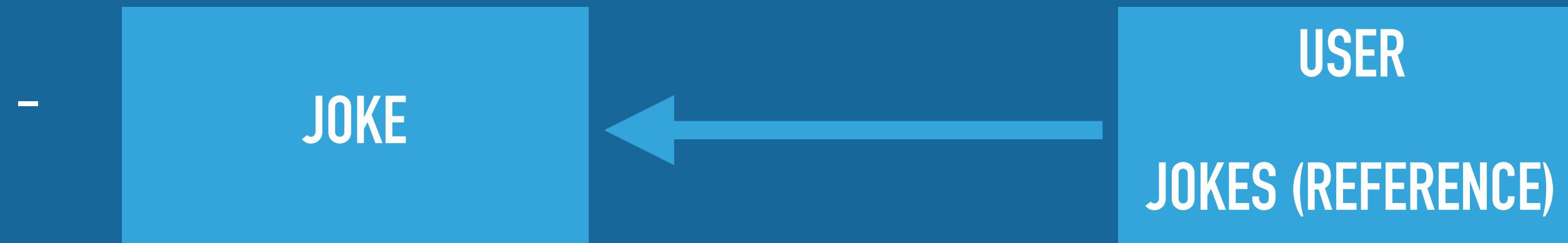
USER RECORD IS ALREADY CREATED

Field Name	Field Type	Indexes
<b>System Fields</b>		
recordName	Reference	Queryable
createdBy	Reference	Queryable
createdAt	Date/Time	Queryable
modifiedBy	Reference	
modifiedAt	Date/Time	
changeTag	String	None
<b>Custom Fields</b>		
question	String	Queryable, Searchable, Sort...
rating_negative	Int(64)	Queryable, Sortable
rating_positive	Int(64)	Queryable, Sortable
response	String	Queryable, Searchable, Sort...

ADD ADDITIONAL FIELDS TO CUSTOMIZE YOUR DATA; NOT RECOMMENDED BY APPLE

# 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



A screenshot of the iCloud cloud.uchicago.CloudyWithAChan interface, specifically the Record Types section. The top navigation bar includes icons for Zones, Records, Record Types, and Info. The 'Record Types' tab is selected. Below it, there are sections for Default Types and Custom Types. Under Default Types, 'Users' is highlighted. Under Custom Types, 'Alert', 'Daily', and 'joke' are listed. A note states: "Record types will be automatically created in the development environment when you use the native API to create records of that type." At the bottom right is a 'Create New Type' button.

Field Name
recordName
createdBy
createdAt
modifiedBy
modifiedAt
roles
changeTag

DEFAULT TYPES

Users

CUSTOM TYPES

Alert

Daily

joke

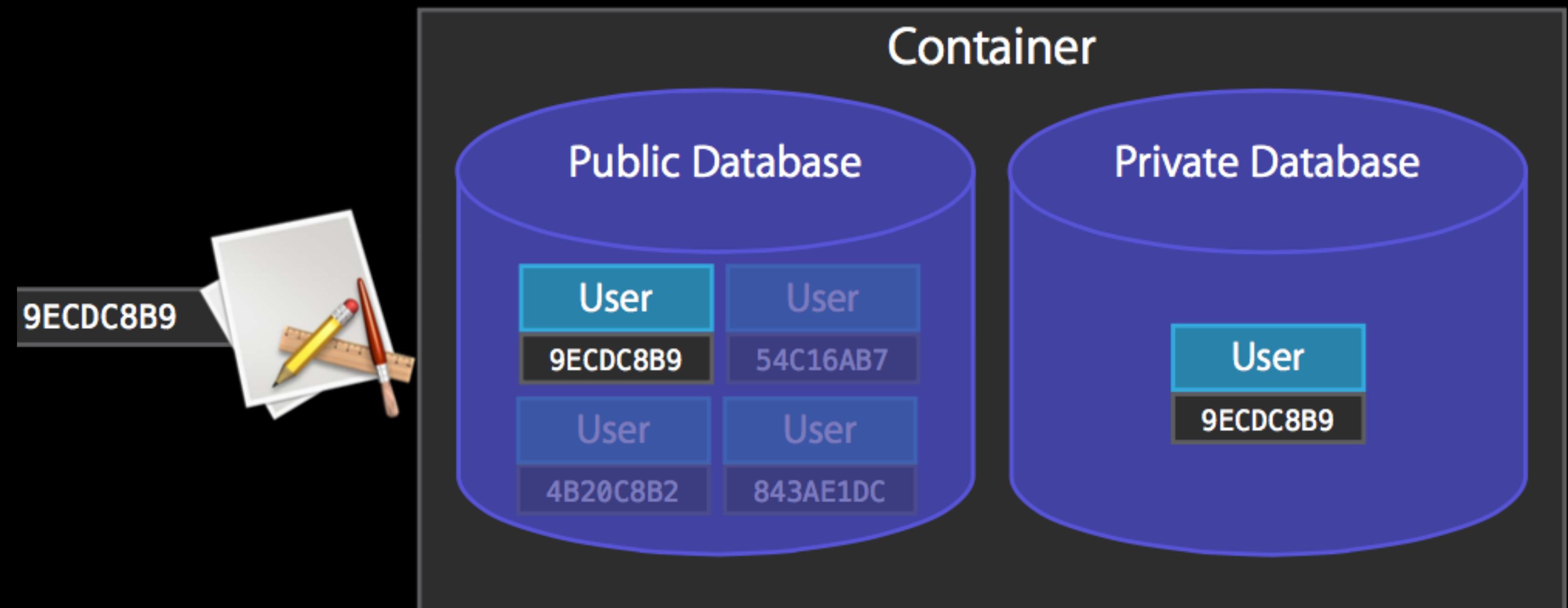
Record types will be automatically created in the development environment when you use the native API to create records of that type.

Create New Type

Add Fi

# 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.fetchUserRecord(recordID: "23D865322080D4185AD95DB121C80663") { record, error in
    if let error = error {
        print("Error fetching record: \(error.localizedDescription)")
        complete(nil, error as NSError?)
    } else {
        // We have access to the user's record
        print("fetched ID \(record?.recordName ?? "")")
        complete(record, nil)
    }
}
```

# USER RECORDS

"\_23D865322080D4185AD95DB121C80663"

ZONES	RECORDS	RECORD TYPES	INDEXES	SUBSCRIPTIONS	SUBSCRIPTION TYPES	SECURITY R	EDITING RECORD	X
LOAD RECORDS FROM:		Record Name	Record T...	Fields	Ch. T...	Created	Modified	
Public Database		_f84642634df...	Users		j29c...	Wed May 03 ...	Wed May 03 ...	
_defaultZone		_23d865322080	Users	d4185ad95db12	j28i9s	Tue May 02 201	Tue May 02 201	X
		1c80663			gv	7 23:51:41 GMT	7 23:51:41 GMT	
					-0500 (CDT)	-0500 (CDT)		
USING:								
<a href="#">Query</a>		<a href="#">Fetch</a>	<a href="#">Changes</a>					
QUERY FOR RECORDS OF TYPE:		Users						
Filter by:		<a href="#">Add filters...</a>						
Sort by:		<a href="#">Add sorts...</a>						
<a href="#">Query Records</a>								
Query for records of the type "Users" that are in the "_defaultZone" zone of the "public" database.								
<a href="#">Create New Record...</a>								

USER RECORDS

Name: \_23d865322080d4185ad95db121c80663

Type: Users

Database: public

Zone: \_defaultZone

Created: May 2 2017 11:51 PM  
by \_23d865322080d4185ad95db121c80663

Modified: May 2 2017 11:51 PM  
by \_23d865322080d4185ad95db121c80663

Change Tag: j28i9sgv

References: 0

There are no records referenced by this record.

Security Roles: This environment has no custom security roles.

Fields: 0 of 0

# USER RECORDS

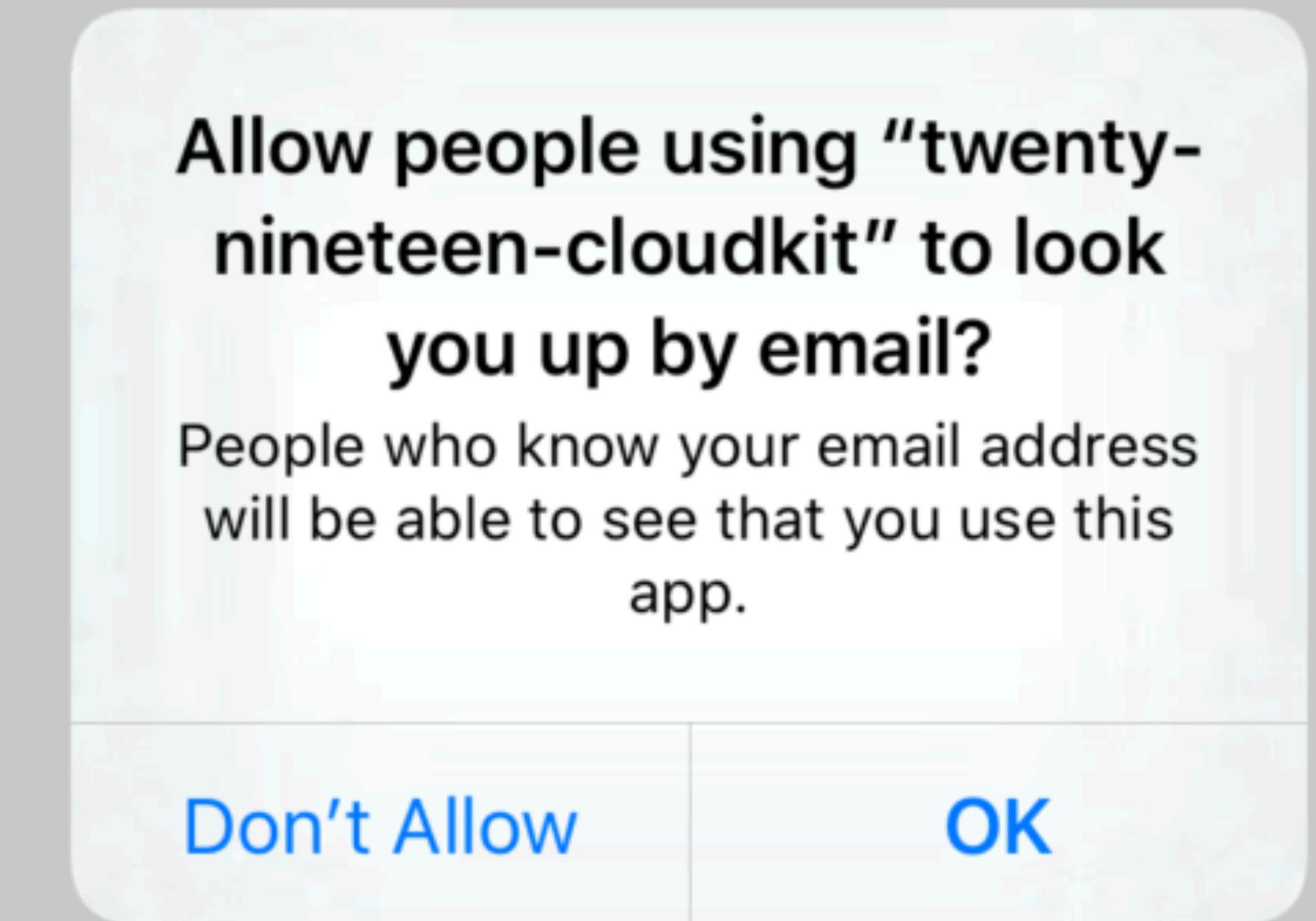
```
/// Get the users `RecordID`  
/// - parameters complete: A completion block passing two parameters  
func getUserRecordId(complete: @escaping (CKRecordID?, NSError?) -> ()) {  
  
    let container = CKContainer.default()  
    container.fetchUserRecordID() {  
        recordID, error in  
  
        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

**Allow people using “twenty-nineteen-cloudkit” to look you up by email?**

People who know your email address will be able to see that you use this app.

Don't Allow

OK

# 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)
    }
}
}

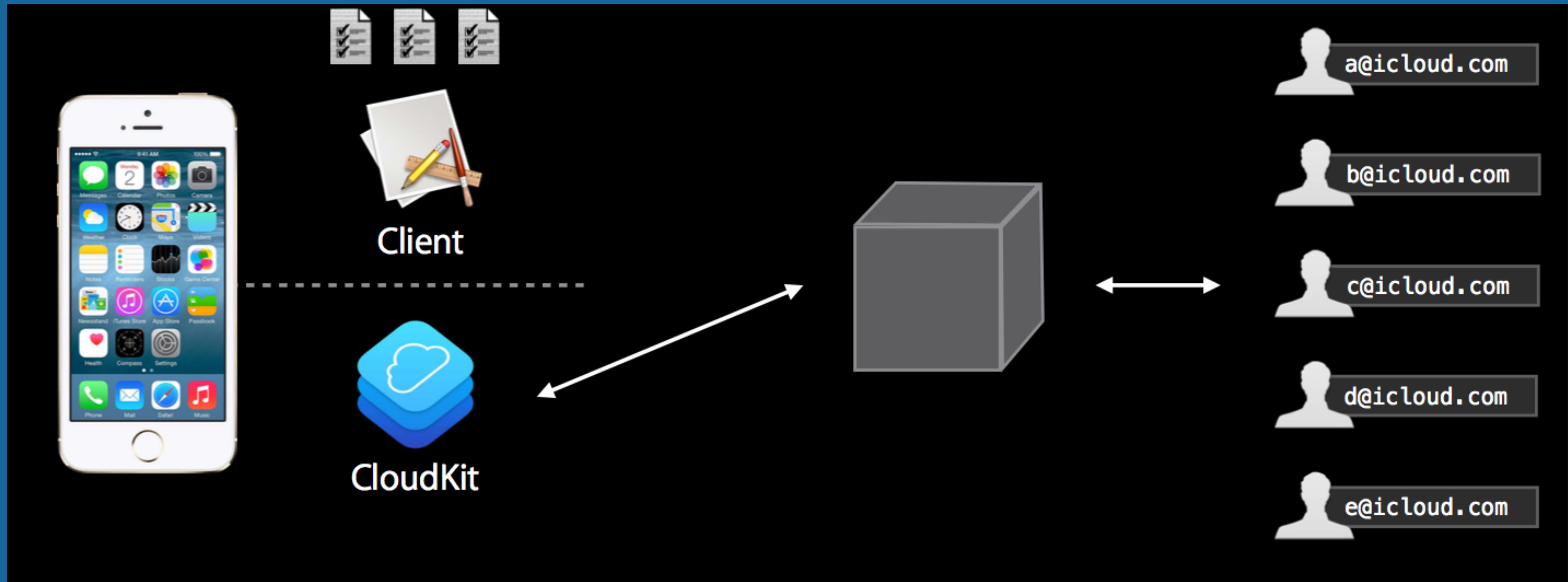
CKContainer.default().requestApplicationPermission(.userDiscoverability) { (status, error) in
    CKContainer.default().fetchUserRecordID { (record, error) in
        CKContainer.default().discoverUserIdentity(withUserRecordID: record!, completionHandler: { (userID, error) in
            userName = (userID?.nameComponents?.givenName)!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

CloudyWithAChanceOfErrors

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

# USER RECORDS

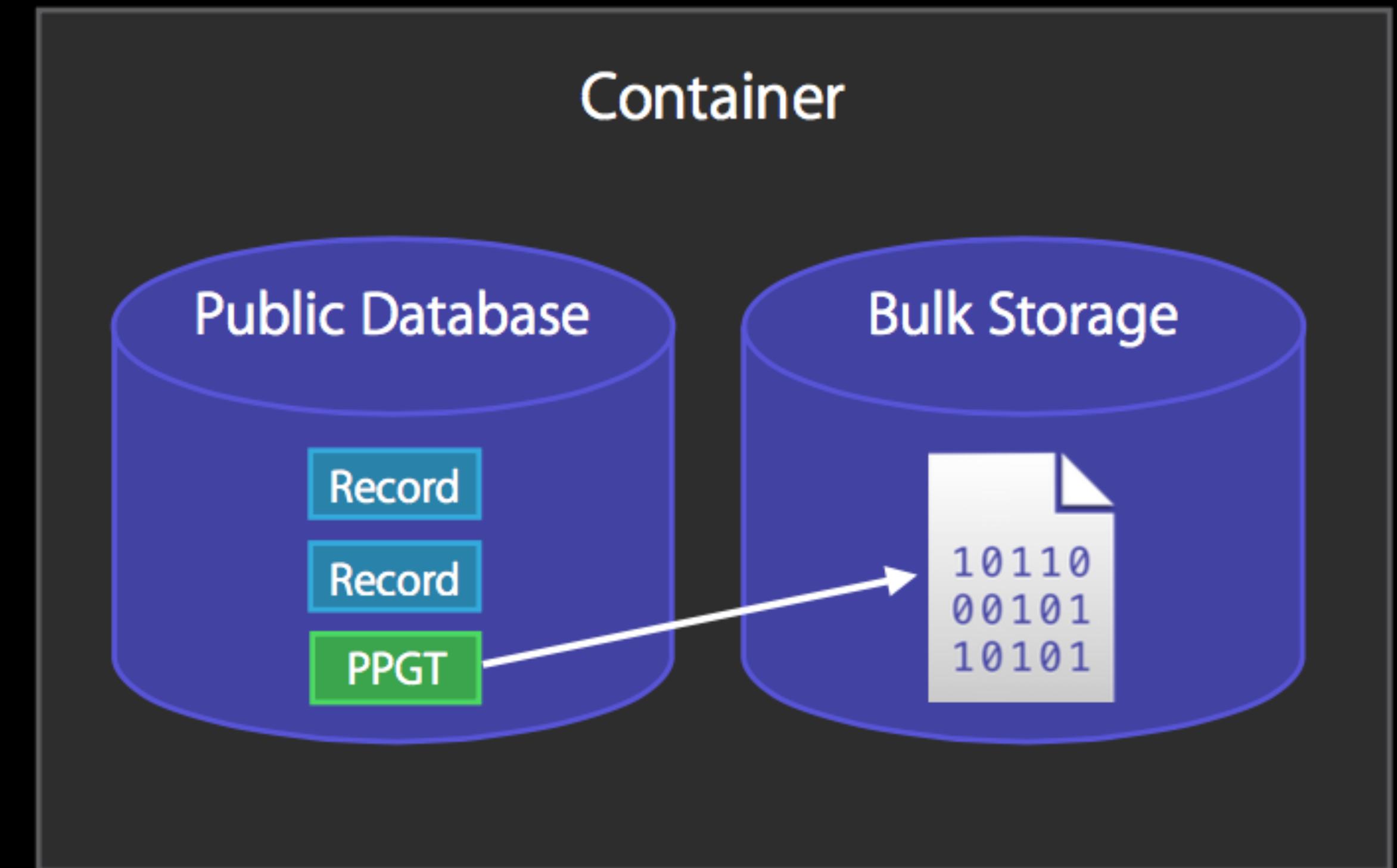


- You can get all friends who use the same app (discover users api)

# ASSETS

# ASSETS

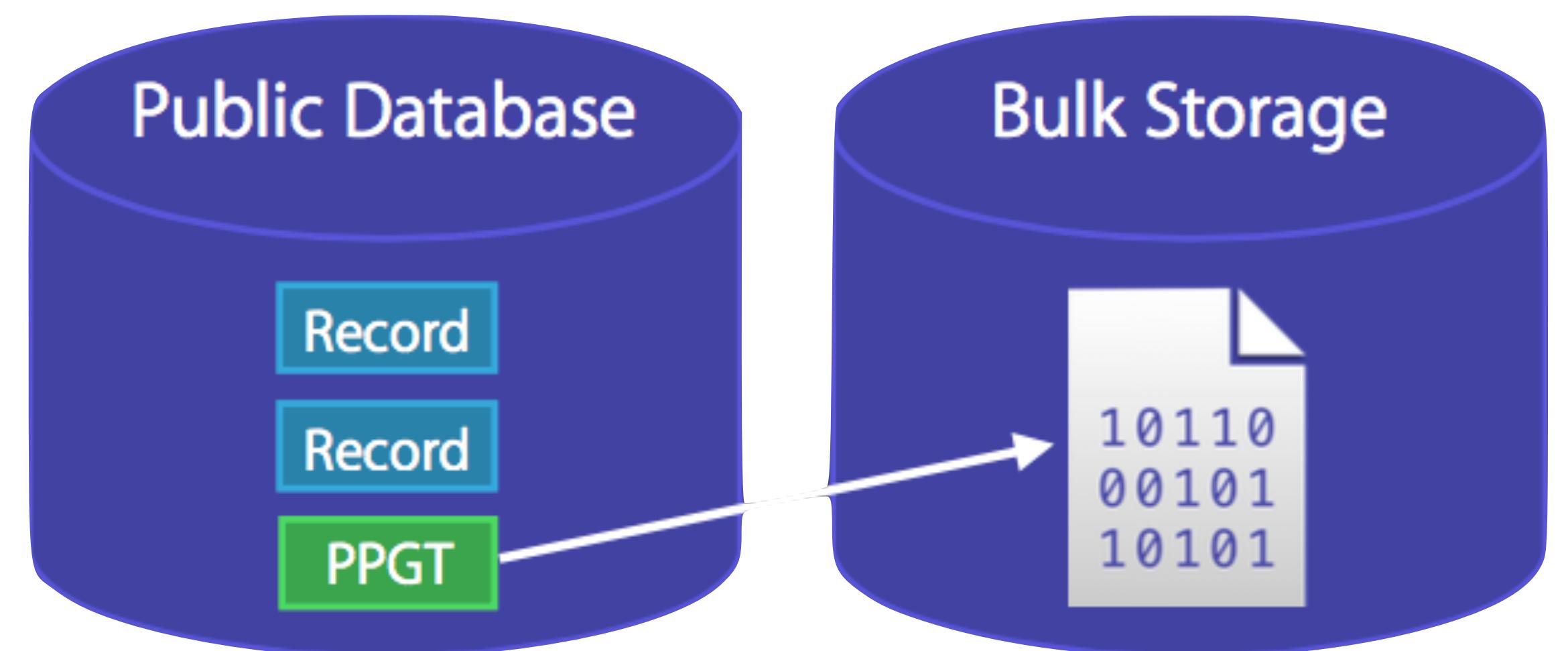
- CKAssets is bulk storage for CloudKit



# ASSETS

SUBTITLE

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



# ASSETS

Cloud > iCloud.cloud.uchicago.CloudyWithAChanceOfErrors > Development Data ANDREW BINKOWSKI

ZONES	RECORDS	RECORD TYPES	INDEXES	SUBSCRIPTIONS	SUBSCRIPTION TYPES	SECURITY R	◀ ▶	EDITING RECORD	X
LOAD RECORDS FROM:	Public Database	Record Name	Record T...	Fields	Ch. T...	Created	Modified		
	_defaultZone	977D8136-0A78-430E-93C2-513B7B0674ED	joke	question Why did the student throw a clock out the window?	j9pri30j	Tue Nov 07 2017 09:20:06 GM	Tue Nov 07 2017 09:20:06 GM	X	
USING:				response She wanted to see time fly.		T-0600 (CST)	T-0600 (CST)		
				rating_positive 0					
				rating_negative 0					
QUERY FOR RECORDS OF TYPE:	joke	be9916ef-e31...	joke	question	j29q...	Wed May 03 ...	Wed May 03 ...		
Filter by:	Add filters...								
Sort by:	Add sorts...								
<b>Query Records</b>									
Query for records of the type "joke" that are in the "_defaultZone" zone of the "public" database.									
<b>ASSETS</b>									
There are no records referenced by this record.									
<b>Fields</b>									
audioFile ASSET									
Drag file here to upload.									
Choose File									

# 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



# ADVANCED iOS APPLICATION DEVELOPMENT

---

MPCS 51032 • SPRING 2020 • SESSION 3