Is the database a solved problem?

Kenneth Geisshirt
kg@realm.io

Realm Inc.
@realm
http://github.com/Realm/
http://realm.io/

Disclaimer

I work for Realm

Realm is a database vendor

↓

I am a little biased

Agenda

- What is a database?
- The history of databases
    - Relational databases, the NoSQL revolution
- Mobile platforms
- Mobile database solutions

What is a database?

- According to Wikipedia: *A **database** is an organized collection of data*.
- Common components
  - data definition (create the structure/schema)
  - insert, update, delete data
  - query/retrieve data
- Databases are often ACID compliant

The history

- First commercial database: IDS (1964) - a network database
- Relational databases were developed in 1970s
  - DB2, Oracle, Ingress
- Moving to the desktops in 1980s: dBASE were popular
- DotCom era - or the MySQL era (1995)
- ca. 2009: NoSQL movement

The Victory of The Relational Database

- Relational database model dominates the world
    - DB2 is heavily used by banks
    - Oracle is found in large enterprises
    - MySQL powers almost every web sites
- SQLite is an embedded database: initial release in 2000

SQL

- Structured Query Language is by all relational databases
- Components
  - Data definition (creating tables, …)
  - Data manipulation (inserting rows, …)
  - Querying

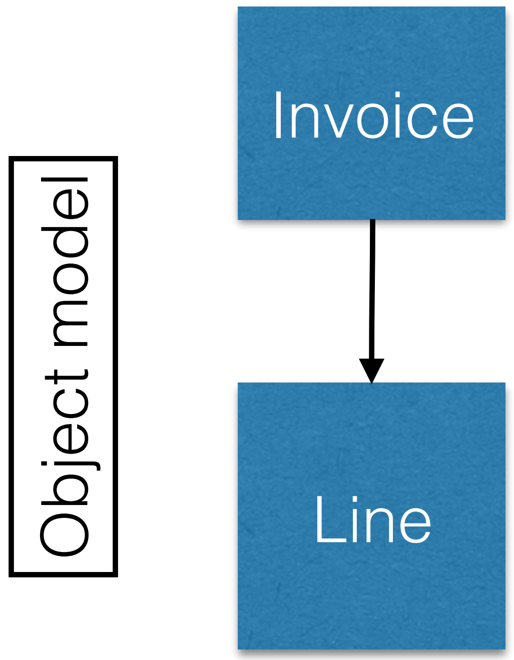
```
INSERT INTO Person VALUES('Kenneth', 46)
SELECT * FROM Person WHERE name = 'Kenneth'
```

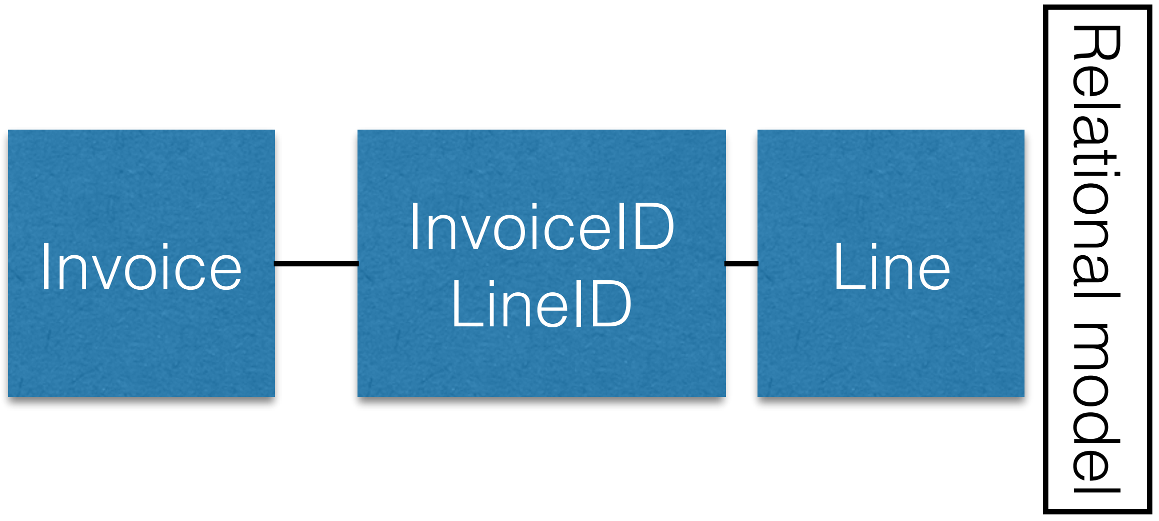
Object-Relation Mappers

- Relational modelling requires normalisation
- Object modelling is often a more direct
- ORMs try to bridge the gap

Z Corporation
64 Hard Drive
1024 Machineville

Item	Number	Price
CPU	1	2000
RAM	4	4000
Disk	2	6000





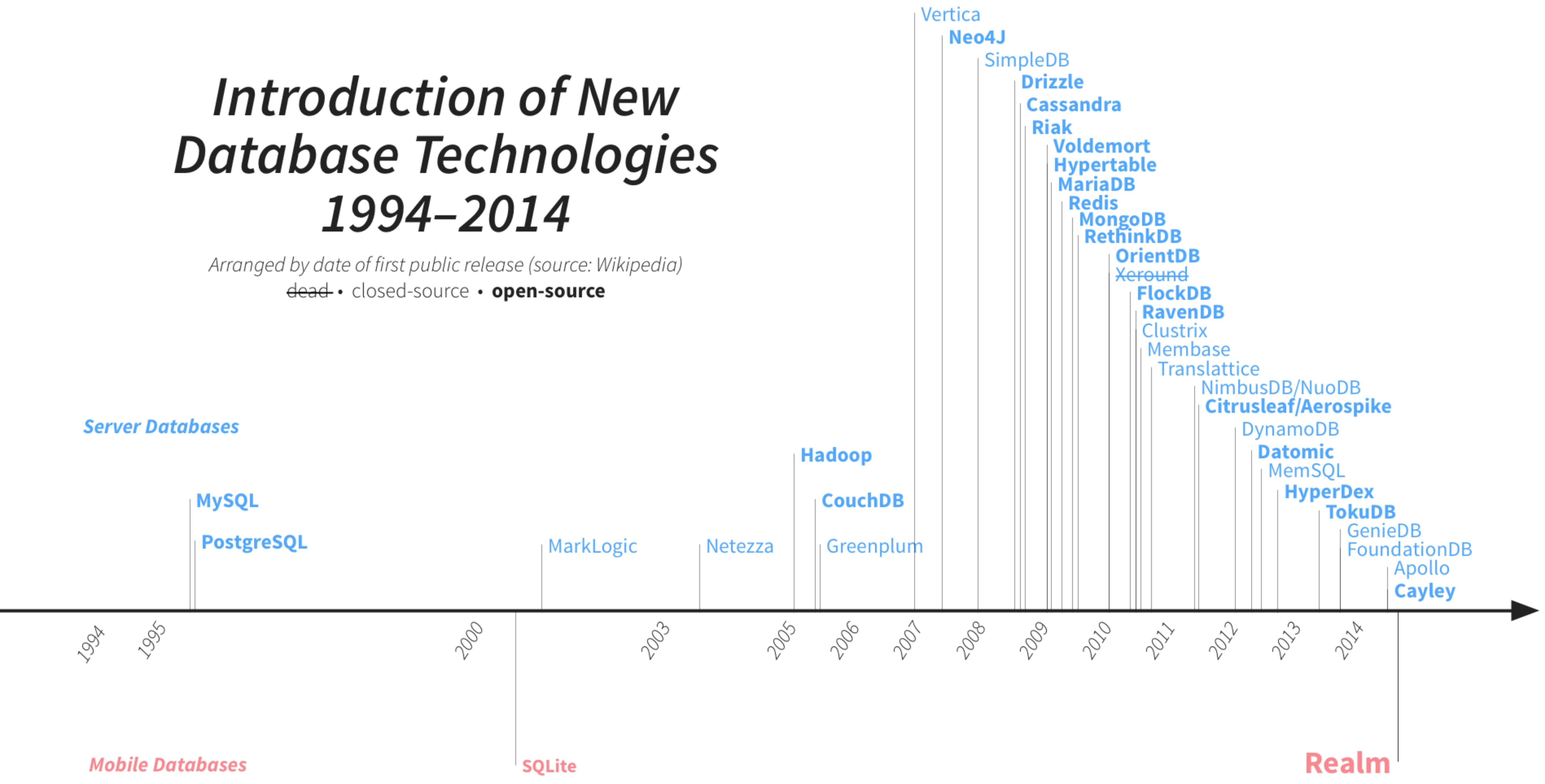
End of The Relational Era

- Relational databases don’t scale well
    - popular web sites are very popular (think Twitter)
- Schemas are not flexible
    - normalisation leads to complex architecture
- ACID is not always a requirement

Introduction of New Database Technologies 1994–2014

Arranged by date of first public release (source: Wikipedia)

~~dead~~ • closed-source • **open-source**



Mobile platforms

- Nokia N95 (2007)
    - dual-core @ 322 MHz, 160 MB ram
- OnePlus One (2014)
    - quad-core @ 2.5 GHz, 3 GB ram, 64 GB storage
- iOS and Android dominate the market
    - UNIX like kernels + libraries
    - Java (version 6 + some version 7), Objective C, and Swift

Mobile databases

- Three types of mobile data solutions:
  - Real databases
  - Data storages using SQLite as store engine
  - Object Relation Mappers (ORMs) on top of SQLite

SQLite

- Library providing a SQL interface to data
    - Most of SQL-92, simplified type system
- Preinstalled on iOS, Android, Windows Phone 8, Blackberry 10
    - 1.8+ billion active devices[1]
- Liberal license: public domain

[1]http://www.engadget.com/2014/06/25/google-io-2014-by-the-numbers/

SQLite on Mobile

iOS

- Rarely used directly
- Popular ORMs: Core Data, Magical Record, FMDB

Android

- The Java community is highly object oriented
- Can be used directly
- Popular ORMs: ORMLite, GreenDAO, SugarORM, DBFlow

Core Data

- Apple’s object graph and persistent framework
- Supported under iOS and OS X
- Many storage formats: XML, binary files, SQLite
- Highly integrated into Xcode
- Part of the iOS/Cocoa SDKs

棒 Magical Record

- Inspired by Fowler's *active record pattern* and Ruby on Rails' Active Record
- Build on-top of Core Data
- Managed Object
- https://github.com/magicalpanda/MagicalRecord

FMDB

- Objective C wrapper around SQLite
- Major classes
  - `FMDatabase` - “connection” to SQLite database
  - `FMDatabaseQueue` - queue of queries and operations
  - `FMResultSet` - results from a query
- License: MIT
- http://ccgus.github.io/fmdb/

FMDB - query

```
FMDatabase *db = [FMDatabase databaseWithPath:@“/tmp/tmp.db”];

[db open];

FMResultSet *s =
    [db executeQuery:@"SELECT * FROM myTable”];
while ([s next]) {
    // …
}

[db close];
```

YAP

- Key/value store
- Supports iOS (and OS X)
- SQLite is used as storage engine
- License: BSD
- https://github.com/yapstudios/YapDatabase

LevelDB

- Embedded key/value store (in C++)
    - License: BSD
    - https://github.com/google/leveldb
- iOS: https://github.com/matehat/Objective-LevelDB
- Android: SnappyDB uses LevelDB + additional compression

Realm

- Realm is an object store
    - Data model = classes (inheriting from a Realm object)
- Supports iOS, Android and OS X
- Core is written in C++ and highly portable
- Custom bindings to give "native touch"
- License: Apache 2.0 (binding) + closed (core)

Realm - iOS - store

```
RLMRealm *realm = [RLMRealm defaultRealm];

[realm beginWriteTransaction];

Person *person = [[Person alloc] init];
person.name = @"Kenneth";
person.age = 46;
[realm addObject:person];

[realm commitWriteTransaction];
```

Realm - iOS - query

```
RLMResults *persons =
  [Person objectsWhere:@"name = ‘Kenneth'"];

for (Person *person in persons) {
  NSLog(@"Age:", person.age);
}
```

Realm - Android - store

```
Realm realm = Realm.getInstance(context);

realm.beginTransaction();

Person person =
     realm.createObject(Person.class);
person.setName(“Kenneth”);
person.setAge(46);

realm.commitTransaction();
```

Realm - Android - query

```
RealmResults<Person> persons =
   realm.where(Person.class)
      .equalTo("name", “Kenneth").findAll();

for (Person person : persons) {
  Log.d("REALM", "Age: " + person.getAge());
}
```

CouchBase Mobile

- Three components:
  - CouchBase Lite: embedded database
  - CouchBase Server: backend storage
  - CouchBase Sync: synchronization
- Supports iOS, Android and desktop/server platforms
- Local storage is based on SQLite
- http://www.couchbase.com/nosql-databases/couchbase-mobile

CouchBase - iOS

```
CBLManager *manager = CBLManager sharedInstance];
CBLDatabase *database = [manager databaseNamed: dbname
error: &error];

NSDictiorary *person =  @{
  @"name": @"Kenneth",
  @"age":  @46
 };

CBLDocument* doc = [database createDocument];
NSString *docID = doc.documentID;
CBLRevision *newRevision = [doc putProperties:
myDictionary error: &error];
```

CouchBase - Android

```
Manager manager = new Manager(new AndroidContext(this),
Manager.DEFAULT_OPTIONS);
Database database = manager.getDatabase("database");

Map<String, Object> docContent = new HashMap<String, Object>();
docContent.put("name", "Kenneth");
docContent.put("age", 46);

Document document = database.createDocument();
document.putProperties(docContent);
String docID = document.getId();
```

Parse

- Cloud database (and local storage)
- Supports iOS, Android, and Windows Phone (and desktop/server platforms)
- Store and retrieve objects in the background
- Payment = requests per second
- https://www.parse.com

Parse - iOS - store

```
PFObject *person = [PFObject
objectWithClassName:@“Person”];

[person setObject:@“Kenneth”
forKey:@“name”];

[person setObject:@“46” forKey:@“age”];

[person saveInBackground];
```

Parse - iOS - query

```
PFQuery *query = [PFQuery
queryWithClassName:@“Person”];

[query whereKey:@“name” equalTo:@“Kenneth”];

[query findObjectsInBackgroundWithBlock:^(NSArray
*objects, NSError *error) {

  for (PFObject *object in objects) {

    NSLog(@“%@“, object[@“age”]);

  }

];
```

Parse - Android - store

```
ParseObject person = new
ParseObject(“Person”);

person.put(“name”, “Kenneth”);

person.put(“age”, 46);

person.saveInBackground();
```

Parse - Android - query

```
ParseQuery<ParseObject> query = ParseQuery.getQuery(“Person”);

query.whereEqualTo(“name”, “Kenneth”);

query.findInBackground(new FindCallback<ParseObject>() {

  public void done(List<ParseObject> list, ParseException e) {

    for (int i=0; i<list.size(); i++) {

      Log.d(“age”, “Age: “ + list.get(i).getInt(“age”));

    }

  }

});
```

Hot topics

- Synchronisation between devices and back-end
    - Often ReST services are used (with JSON)
    - Parse and CouchBase Lite try to solve it
- Reactive programming
    - RxJava (for Android)
    - ReactiveCocoa (iOS)

Dan Strange, http://bit.ly/1QqQfEe

The database is **not** a solved problem