#MongoDB

User Data Management with MongoDB

Heather Kirksey

Solutions Architect, MongoDB

mongoDB

Agenda

Agenda

- High Level Overview
  - MongoDB
  - User Data
- Modeling & Querying User Data
  - Insurance Company User Data
  - User Check-Ins
- Extending the Data Model for Future Use Cases
  - Tracking User Activity
  - Social Media

MongoDB is a(n) _____ database

- Document
- Open source
- High performance
- Horizontally scalable
- Full featured

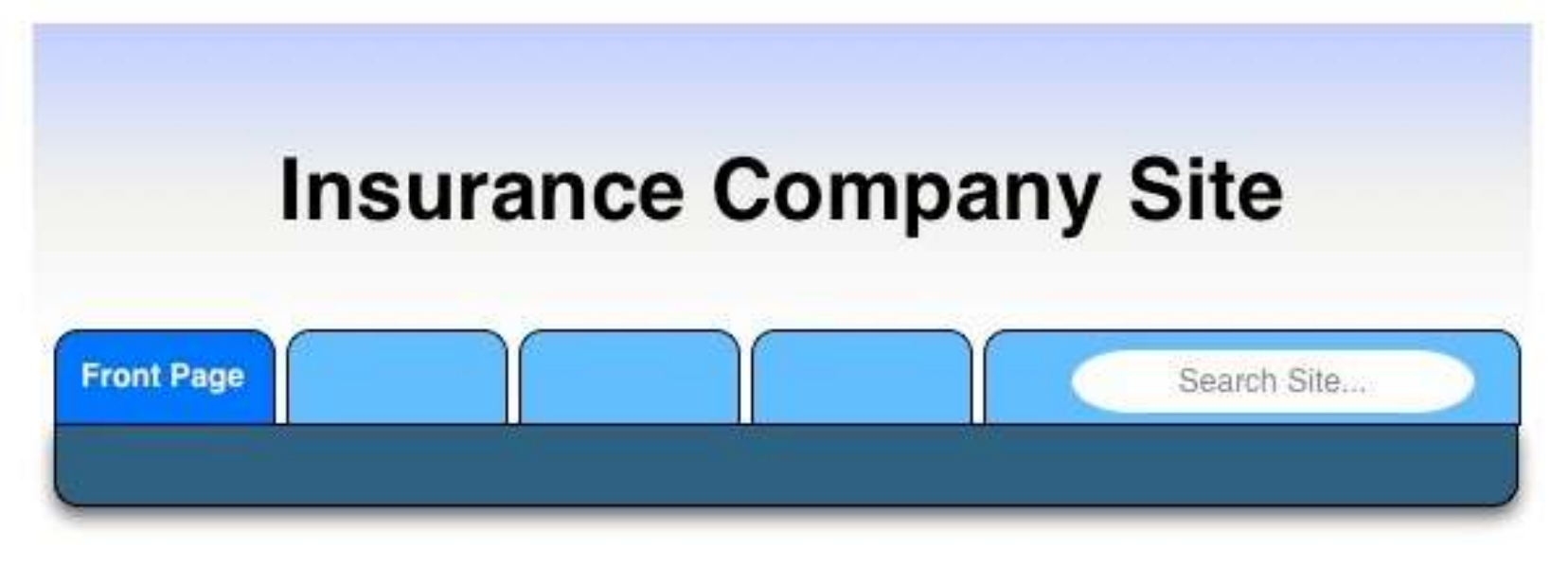
User Data

- Account Information
  - Name, address, etc.
  - Account status
  - Notes
- Activity Streams
  - Posts, tweets, likes, check-ins
  - Recording user actions
- Social Networks
  - Friends, connections
  - Groups, tags

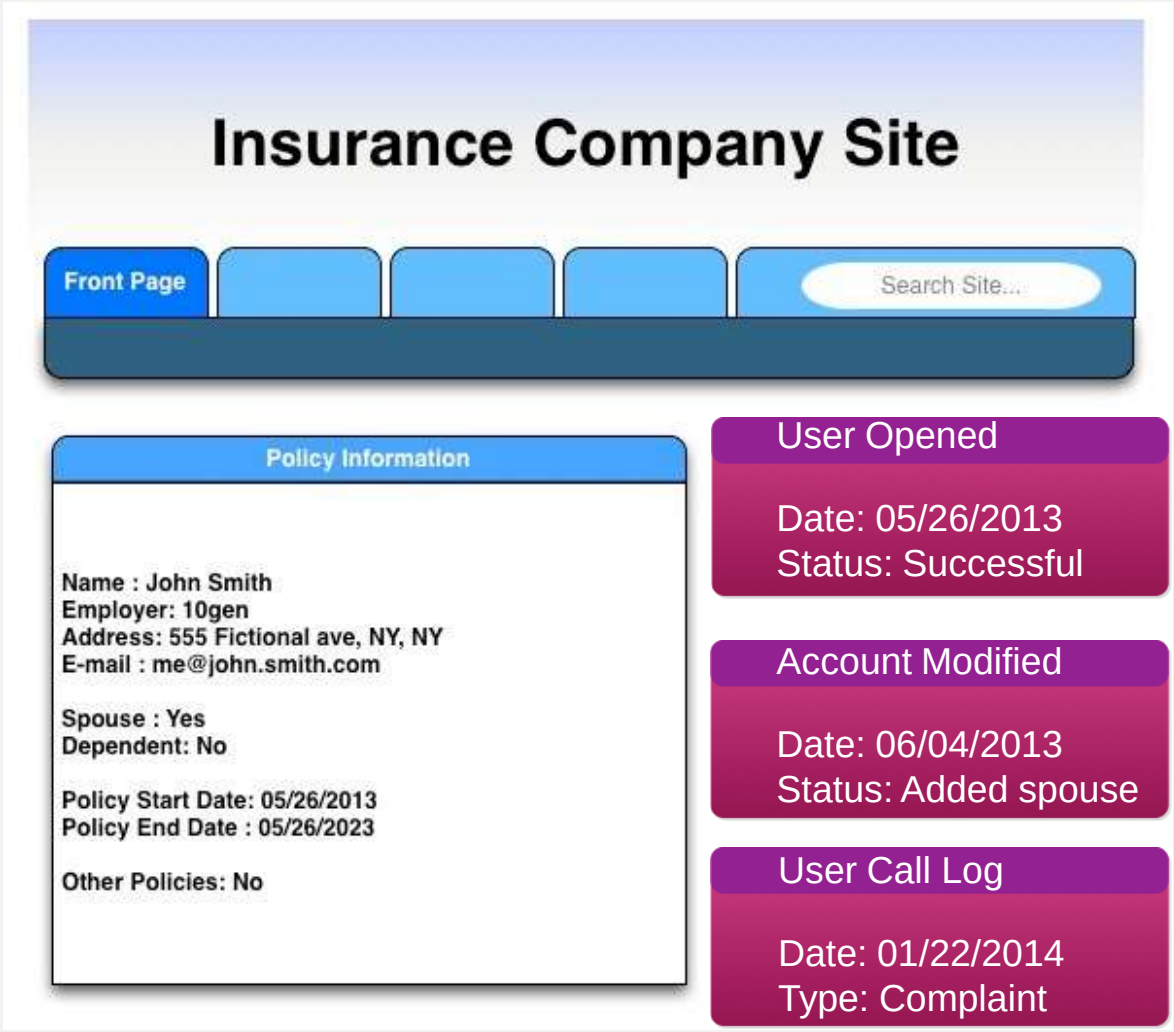
Data Modeling Exercise

- Insurance Company Data
- Account information
  - Name, address, etc
  - Account status
  - Notes

Data Modeling Example

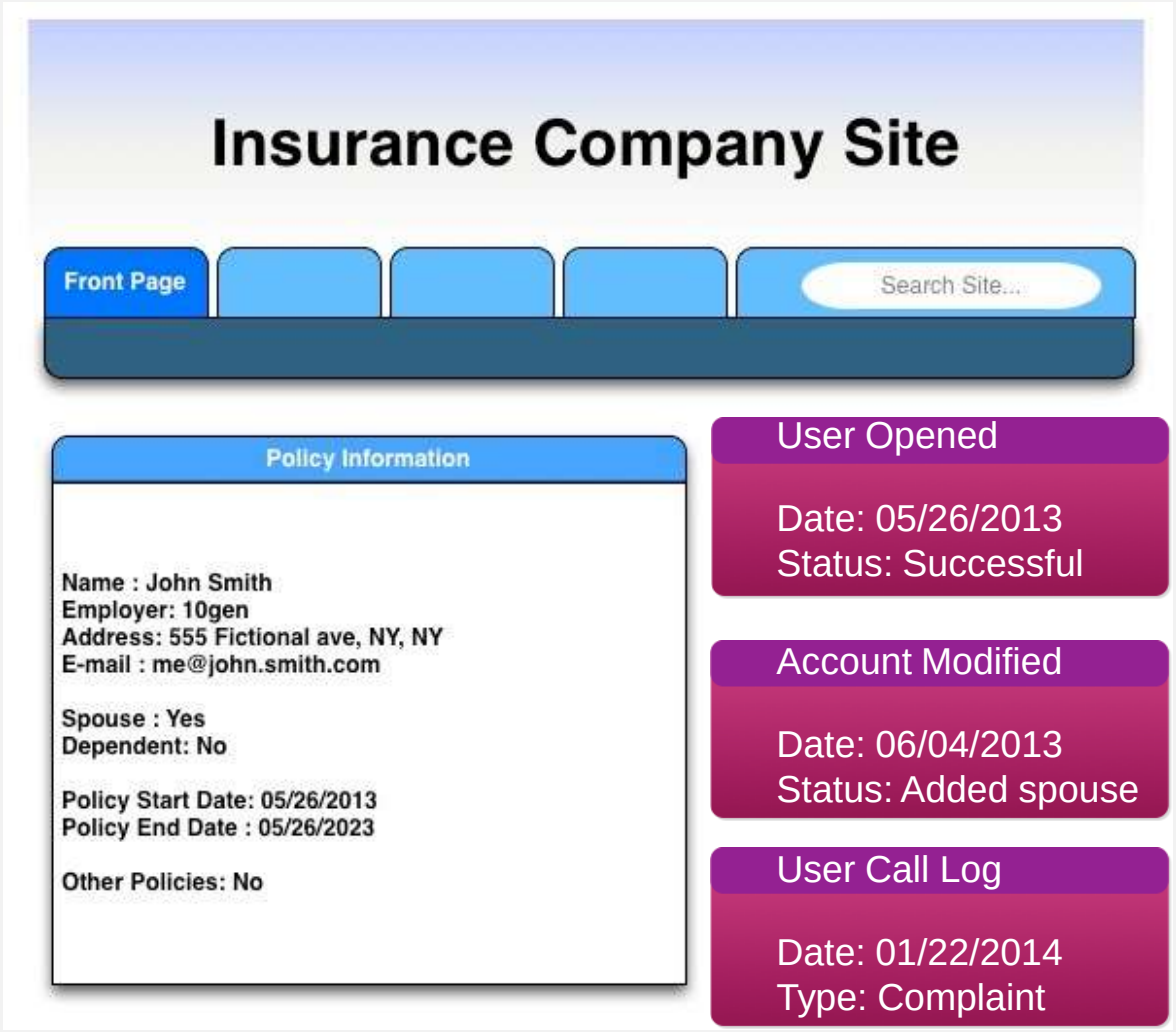


Data Modeling Example



Insurance Company Site

2 types of data



2 types of data

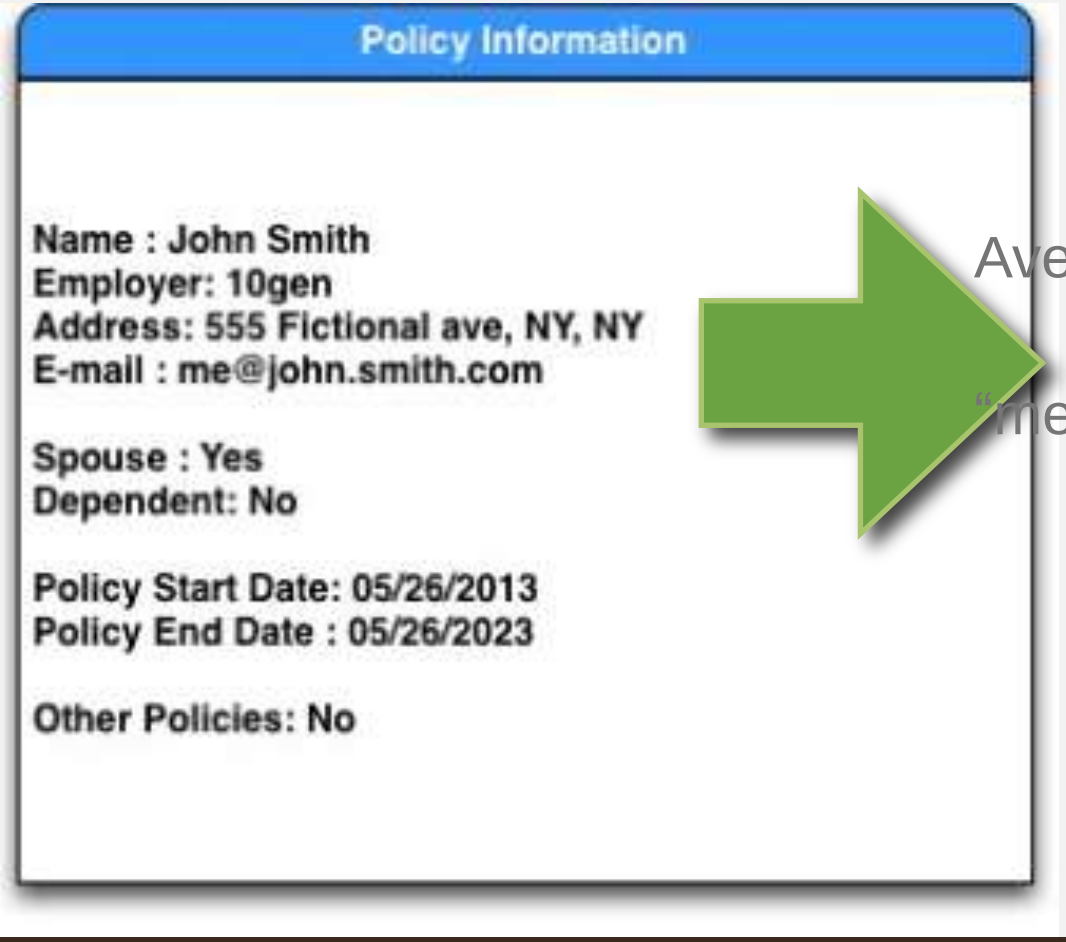
Rule of Thumb

- Categories of data map well to MongoDB Collections

Policies

Activities

Policies



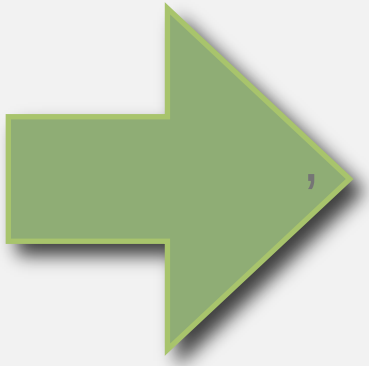

```
policy = {
        name: “John Smith”
        employer: “10gen”,
        address:  “555 Fictional
Ave”,
        e-mail:
“me@john.smith.com”,
        spouse: “Yes”  ,
        dependents:  “No”,
        dates: [
          {start:  5/26/2013
          10:12:00},
           end:  5/26/2023
          10:12:00}],
others: “No”
```

Activities

User Opened Account
Date: 05/26/2013
Status: Success

Account Modified
Date: 06/04/2013
Action: Added Spouse

User Call Log
Date: 01/22/2014
Type: Complaint

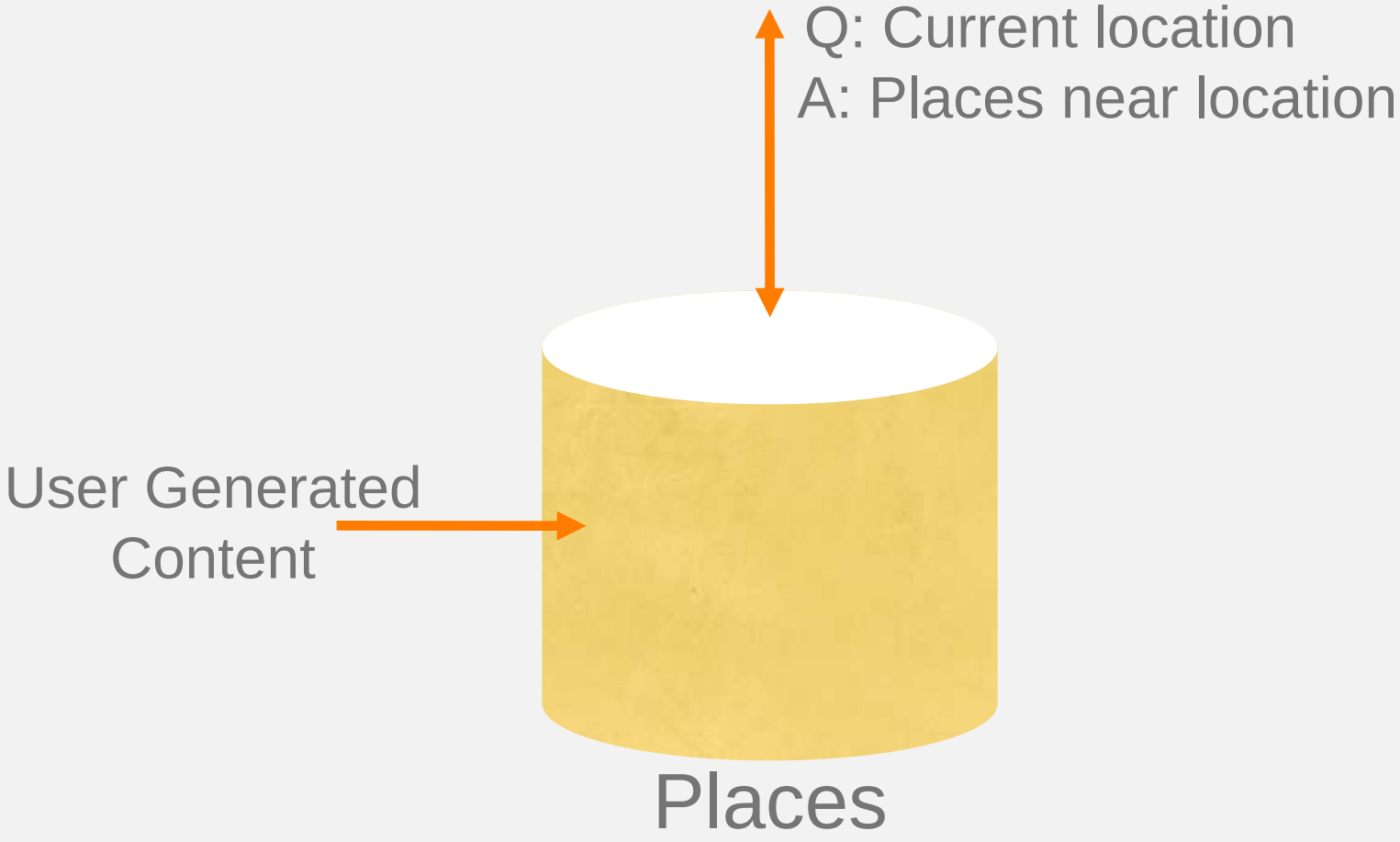


```
activity = {
        user-id:  “JohnSmith421”
        type:  “account-opening”
,
        status:  “Success”,
        dates: 5/26/2013
10:12:00,
        related-doc:
“/customer/JohnSmith421/open
.pdf”
        }
```

User Check-Ins

- Activity Streams
  - Posts, tweets, check-ins
  - Recording user actions

Places


Q: Current location
A: Places near location
User Generated Content
Places

Inserting a Place

```
var p =  { name: “MongoDB HQ”,
           address: “229 W 43rd St”,
           city: “New York”,
           zip: “10036”,
           tags: [“mongoDB”, “business”],
           latlong: [40.0, 72.0],
           tips: [{user: “John Smith”, time: “3/15/2013”,tip: “Make sure to stop by for office hours!”}]}

> db.posts.save(p)
```

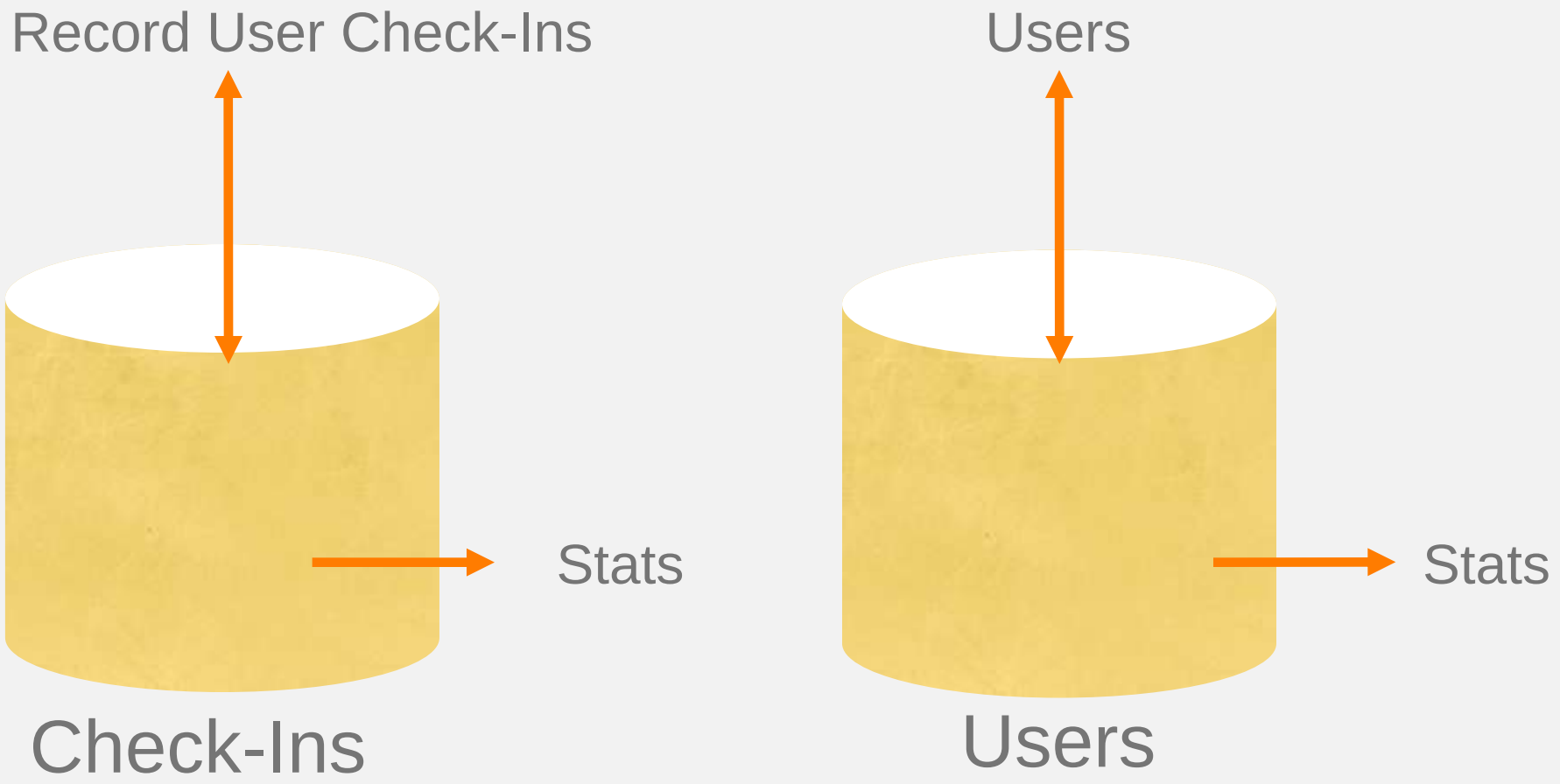
Updating Tips

```
db.places.update({name: “MongoDB HQ”},
        {$push :{tips:
                {user: “John”, time:3/10/2014,
                tip: “stop by for office hours on
                Wednesdays from 4-6”}}})
```

Querying Our Places

- Creating Indexes
  - db.places.ensureIndex({tags:1})
  - db.places.ensureIndex({name:1})
  - db.places.ensureIndex({latlong:”2d”})
- Finding Places
  - db.places.find({latlong:{$near:[40,70]}})
- Regular Expressions
  - db.places.find({name: /^typeaheadstring/)
- Using Tags
  - db.places.find({tags: “business”})

User Check Ins



Users

user1 = {

name: “John Smith”

e-mail: “me@john.smith.com”,

check-ins: [4b97e62bf1d8c7152c9ccb74, 5a20e62bf1d8c736ab]

}

checkins [] = ObjectId reference to Check-Ins Collection

Check-Ins

```
user1 = {
        place:  “MongoDB HQ”,
        ts:  9/20/2010 10:12:00,
        userId: <object id of user>
}
```

Every Check-In is Two Operations

- Insert a Check-In Object (check-ins collection)
- Update ($push) user object with check-in ID (users collection)

Simple Stats

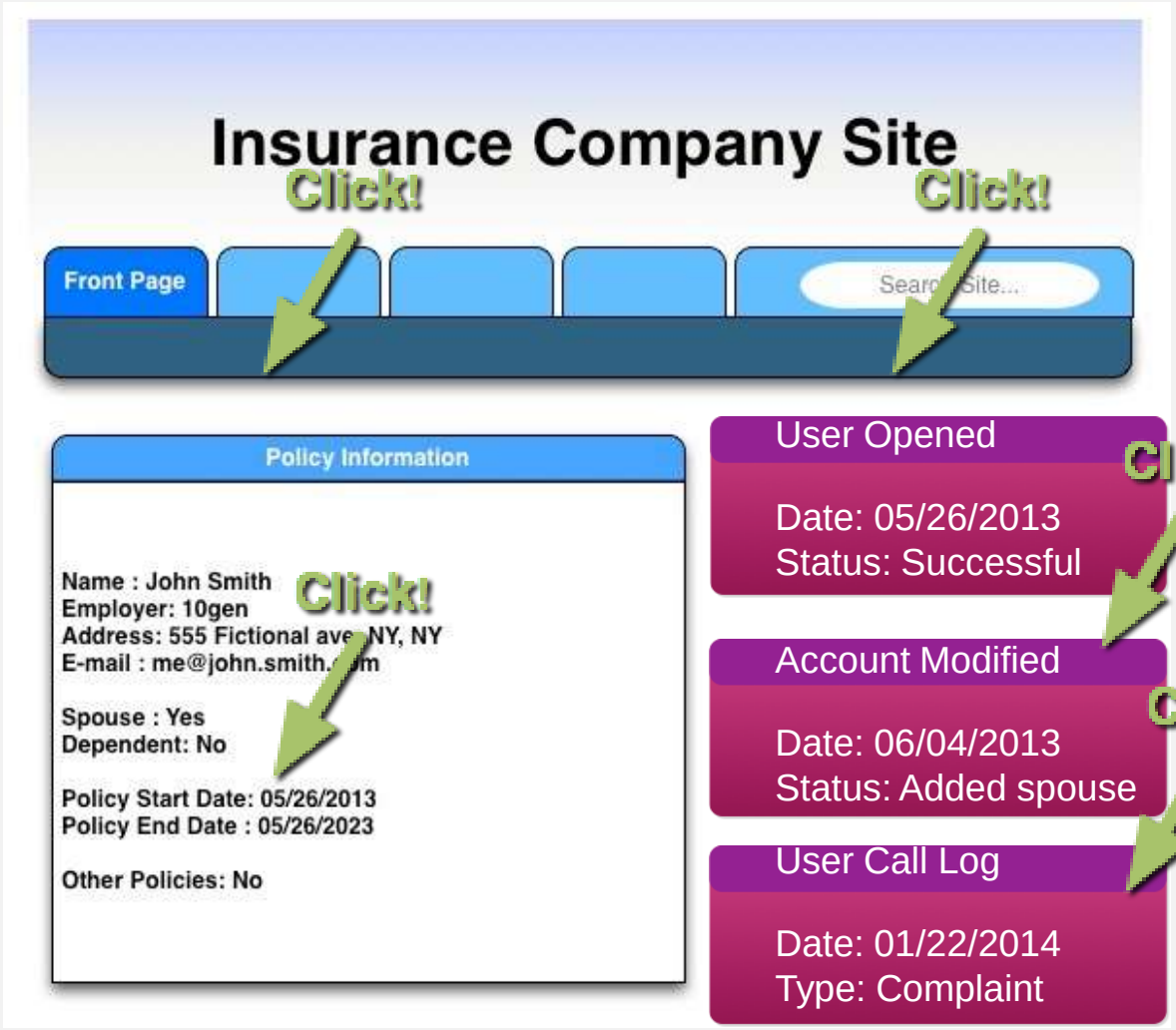
```
db.checkins.find({place: “MongoDB HQ”)

db.checkins.find({place: “MongoDB HQ”})
                                .sort({ts:-1}).limit(10)

db.checkins.find({place: “MongoDB HQ”,
                                ts: {$gt: midnight}}).count()
```

Stats w/ MapReduce

```
mapFunc = function() {emit(this.place, 1);}

reduceFunc = function(key, values) {return
Array.sum(values);}

res = db.checkins.mapReduce(mapFunc,reduceFunc,
{query: {timestamp: {$gt:nowminus3hrs}}})

res = [{_id:”MongoDB HQ”, value: 17}, … , …]
```

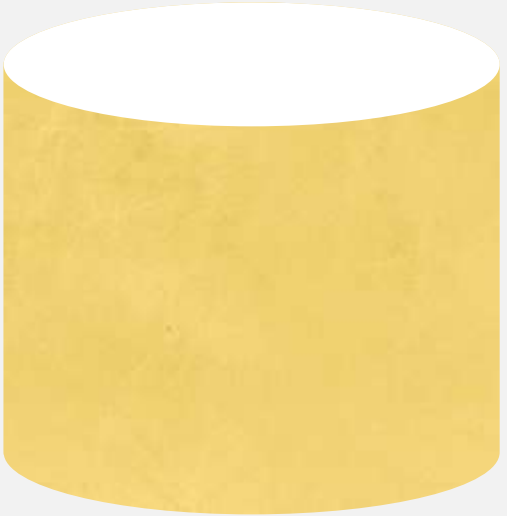
…or try using the aggregation framework!

Adding More User Data



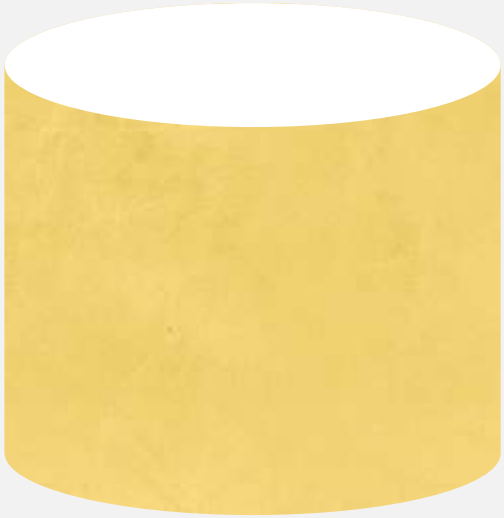
Tracking Clicks

Policies

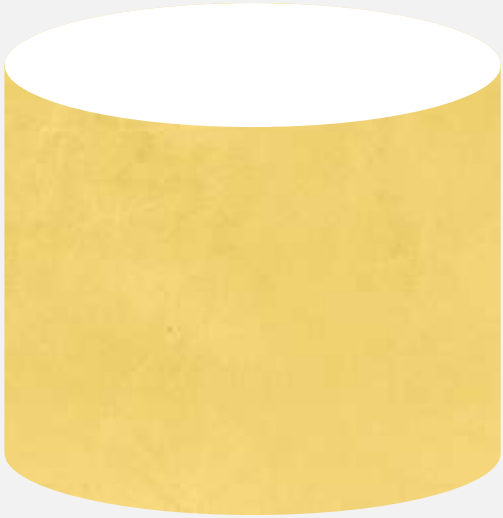


Activities

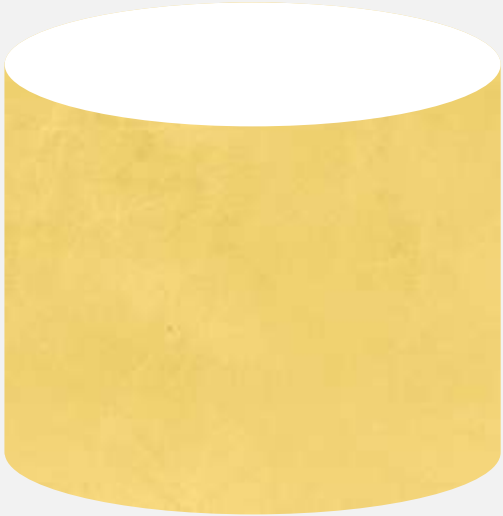
Each Click Creates a New Doc



Policies



Activities



Clicks

Clicks

```
click = {
        user:  “JohnSmith”,
        ts:  9/20/2010 10:12:00,
        link: “http://some-link-here.com/wherever”
}
```

Now we can audit user activity...

```
db.clicks.find({user:”JohnSmith”}).sort({ts:-1})
```

Show me all of John’s clicks sorted by timestamp.

Extending the Schema

```
user1 = {
        name: “John Smith”
        e-mail: “me@John.Smith.com”,
        check-ins:
[4b97e62bf1d8c7152c9ccb74,
5a20e62bf1d8c736ab]
}
```

Extending the Schema

```
user1 = {
        name: “John Smith”
        e-mail: “me@John.Smith.com”,
        check-ins:
[4b97e62bf1d8c7152c9ccb74,  5a20e62bf1d
8c736ab]
}
        friends:
[7b47j62bk1d3c5621c1icv90,  1h11p62bf1d8
c716za]
}
```

Takeaways

- User data fits well in MongoDB
  - Flexible data model
  - Stay agile; make changes
  - Many customers in production
- Application patterns drive data design
  - Optimize data model for queries
  - Primary use cases drive design
- Adding features is easy
  - Create new data structures
  - Extend existing


Questions?

#MongoDBWorld

MongoDB World
New York City, June 23-25

See what's next in MongoDB including

- MongoDB 2.6
- Sharding
- Replication
- Aggregation

http://world.mongodb.com

Save $200 with discount code THANKYOU

#MongoDB

Thank You

Heather Kirksey

Solutions Architect, MongoDB

mongoDB