Audio should start immediately when you log into the event via Audio Broadcast.

If you are having issues connecting, please dial 1-877-668-4493 or +1-408-600-3600 Access code: 667 326 336

Global dial-in numbers can be found on the Event Info tab of your WebEx Event Center screen. There is a Q&A following the talk. Please enter in all questions in the WebEx chat box. A recording of the webinar will be available 24 hours after the event is complete.
Agenda

• MongoDB Data Model
• Blog Posts & Comments
• Geospatial Check-Ins
• Food For Thought
MongoDB Data Model: Rich Documents

```json
{
  title: 'Who Needs Rows?',
  reasons: [
    { name: 'scalability',
      desc: 'no more joins!' },
    { name: 'human readable',
      desc: 'ah this is nice...' }
  ],
  model: {
    relational: false,
    awesome: true
  }
}
```
Embedded Documents

{  
  _id: ObjectId("4c4ba5c0672c685e5e8aabf3"),
  author: "roger",
  date: "Sat Jul 24 2010 19:47:11 GMT-0700 (PDT)",
  text: "Spirited Away",
  tags: ["Tezuka", "Manga"],
  comments: [
    
    {  
      author: "Fred",
      date: "Sat Jul 24 2010 20:51:03 GMT-0700 (PDT)",
      text: "Best Movie Ever"
    }
  ]
}
<table>
<thead>
<tr>
<th>RDBMS</th>
<th>MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Collection</td>
</tr>
<tr>
<td>Row</td>
<td>Document</td>
</tr>
<tr>
<td>Column</td>
<td>Field</td>
</tr>
<tr>
<td>Index</td>
<td>Index</td>
</tr>
<tr>
<td>Join</td>
<td>Embedding &amp; Linking</td>
</tr>
<tr>
<td>Schema Object</td>
<td></td>
</tr>
</tbody>
</table>
Relational

- **Category**
  - Name
  - Url

- **User**
  - Name
  - Email Address

- **Article**
  - Name
  - Slug
  - Publish date
  - Text

- **Comment**
  - Comment
  - Date
  - Author

- **Tag**
  - Name
  - Url
MongoDB

User
- Name
- Email Address

Article
- Name
- Slug
- Publish date
- Text
- Author

Comment[]
- Comment
- Date
- Author

Tag[]
- Value

Category[]
- Value
Blog Posts and Comments
How Should the Documents Look?

What Are We Going to Do with the Data?

To embed or to link... That is the question!
1) Fully Embedded

{  
  blog-title: 'Commuting to Work',
  blog-text: [
    'This section is about airplanes',
    'this section is about trains'
  ],
  comments: [
    { author: 'Kevin Hanson',
      comment: 'dude, what about driving?' },
    { author: 'John Smith',
      comment: 'this blog is aWful!!11!!!!' }
  ]
}
1) Fully Embedded

Pros
• Can query the comments or the blog for results
• Cleanly encapsulated

Cons
• What if we get too many comments? (16MB MongoDB doc size)
• What if we want our results to be comments, not blog posts?
2) Separating Blog & Comments

```json
{
  _id: ObjectId("4c4ba5c0672c685e5e8aabf3")
  comment-ref: ObjectId("4c4ba5c0672c685e5e8aabf4")
  blog-title: 'Commuting to Work',
  blog-text: [
    'This section is about airplanes',
    'this section is about trains'
  ]
}

{
  _id: ObjectId("4c4ba5c0672c685e5e8aabf4")
  blog-ref: ObjectId("4c4ba5c0672c685e5e8aabf3")
  comments: [
    { author: 'Kevin Hanson',
      comment: 'dude, what about driving?' },
    { author: 'John Smith',
      comment: 'this blog is aWful!!11!!!!' }
  ],
}
```
2) Separating Blog & Comments

Pros
• Blog Post Size Stays Constant
• Can Search Sets of Comments

Cons
• Too Many Comments? (same problem)
• Managing Document Links
3) Each Comment Gets Own Doc

```json
{
    blog-title: 'Commuting to Work',
    blog-text: [
        'This section is about airplanes',
        'this section is about trains'
    ]
}

{
    commenter: 'Kevin Hanson',
    comment: 'dude, what about driving?'
}

{
    commenter: 'John Smith',
    comment: 'this blog is aWful!!11!!!!'
}
```
3) Each Comment Gets Own Doc

**Pros**
- Can Query Individual Comments
- Never Need to Worry About Doc Size

**Cons**
- Many Documents
- Standard Use Cases Become Complicated
Managing Arrays

Pushing to an Array Infinitely...
- Document Will Grow Larger than Allocated Space
- Document May Increase Max Doc Size of 16MB

Can this be avoided??
- Yes!
- A Hybrid of Linking and Embedding
Geospatial Check-Ins
We Need 3 Things

Places

Check-Ins

Users
Q: Current location
A: Places near location

User Generated Content
Inserting a Place

```javascript
var p = {
  name: "10gen HQ",
  address: "578 Broadway, 7th Floor",
  city: "New York",
  zip: "10012"
}

> db.places.save(p)
```
Tags, Geo Coordinates, and Tips

```
{name: "10gen HQ",
address: "578 Broadway, 7th Floor",
city: "New York",
zip: 10012,
tags: ["MongoDB", "business"],
latlong: [40.0, 72.0],
tips: [{user: "kevin", time: "3/15/2012", tip: "Make sure to stop by for office hours!"}],}
```
Updating Tips

db.places.update({name: "10gen HQ"},
    {
        $push: {
            tips: {
                user: "nosh",
                time: 3/15/2012,
                tip: "stop by for office hours on Wednesdays from 4-6"
            }
        }
    })
Querying Places

★ Creating

Indexes

\[ \text{db.places.ensureIndex(\{tags: 1\})} \]
\[ \text{db.places.ensureIndex(\{name: 1\})} \]
\[ \text{db.places.ensureIndex(\{latlong:"2d"\})} \]

Finding Places

\[ \text{db.places.find(\{latlong: \{\$near: [40,70]\}\})} \]

Regular Expressions

\[ \text{db.places.find(\{name: /\^typeaheadstring/\})} \]

Using Tags

\[ \text{db.places.find(\{tags: "business"\})} \]
User Check-Ins

Record User Check-Ins → Stats

Users → Stats

Check-Ins

Users
Users

user1 = { name: "Kevin Hanson" e-mail: "kevin@10gen.com", check-ins: [4b97e62bf1d8c7152c9ccb74, 5a20e62bf1d8c736ab] }

checkins [] = ObjectId reference to Check-Ins Collection
Check-Ins

checkin = {
  place: "10gen 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: "10gen HQ"})
db.checkins.find({place: "10gen HQ"})
       .sort({ts: -1}).limit(10)
d(db.checkins.find({place: "10gen 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: ”10gen HQ”, value: 17}, ......, ....]
Food For Thought
Think About How the Application Wants the Data, Not How it is most “Normalized”

Example: Our Business Cards

```javascript
{ name : "Kevin Hanson",
title : "Solutions Architect",
phone : "415-347-5384",
email : "kevin@10gen.com",
twitter : "@hungarianho" }
```

Kevin Hanson
Solutions Architect, 10gen
twitter: @hungarianhc
[kevin@10gen.com](mailto:kevin@10gen.com)

| Twitter [@mongodb](https://twitter.com/mongodb)
| LinkedIn [http://linkd.in/joinmongo](http://linkd.in/joinmongo)