Slide 1

Mongodb by 10gen
NOSQL
I'm going to assume you are familiar with SQL for most of this discussion, and that storing information in a database is a familiar process for you. Hopefully if you need a primer on databases this gets you excited to go investigate.
BSON storage - binary formatted JSON
Collections rather than tables
2 Documents within the same collection may have different attributes
JSON query interface oop syntax

Add feature to existing table

SQL vs mongo
We're going to add a discount property on customers Martial artist website.
We sell the foo in Kung-fu
Some martial artists have dojo and resell our foo to their students
We want to incentivize these customers with a percent off
Not everyone who buys lots of foo should get the discount, so there should be a boolean
Percent off which we expect to be calculated later, today we are going to set this value so we need it in the db.
How to extend schema in SQL?
Alter table
-consume the disk space for the new field
-time to complete for large number of rows
-Any code touching that table referencing that column

Slide 2

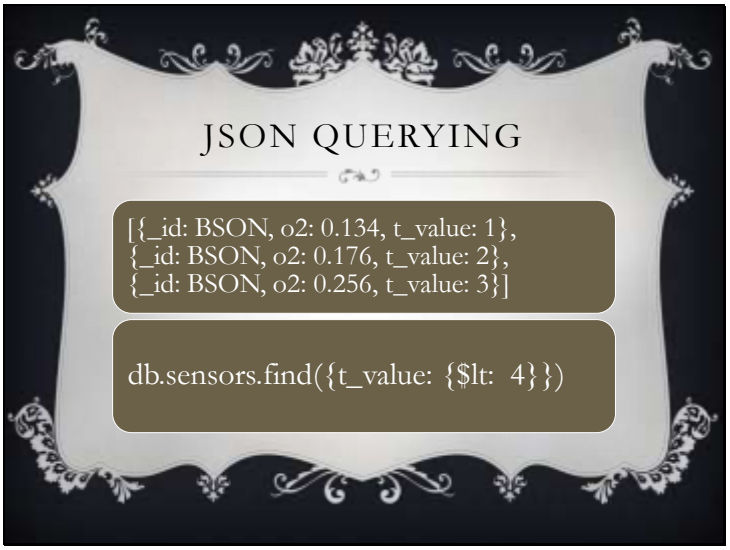
Mongo db in an implementation prepared to log data that doesn't necessarily have a schema. English speech is a good example of schemaless information.
JSON is a standard that defines data at a state in time. A sample user of a forum example.
Hi my name is Nathan Sparks and I have 42 posts on the diy_efi board.
Honestly I see the memberships embedded document being deeper but this is for a sample, and I'm trying to keep this simple.

Slide 3

In use in engine management this could be used in combination with the network of sensors standard known as CAN, the network communication protocol.
It provides banks of sensors so it would look something like this:
These could be appended a t value that defines where in the sequence of the logging and that value would be indexed.

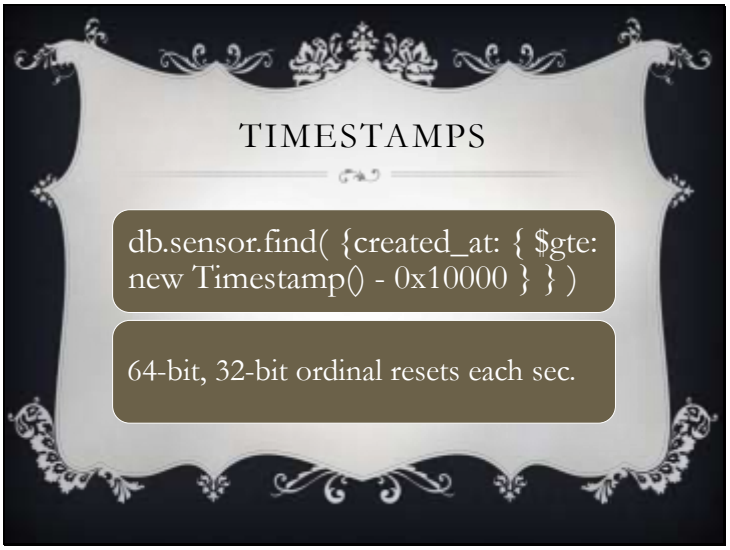
My personal contact information presented as a json document.
There would be quotes around the strings in the strict convention

Slide 4



This is an example of storing the detected value of an O2 sensor. We need to allow the sensor to warm up.
_max and _avg can be filtered until the register stars responding within the range like an O2 sensor heated to 600 degrees.
So the engine is at the mercy of the ecu until the feedback from the o2 sensor can keep the engine running. We need to be able to detect this. Before that time, the ecu needs to deal with throttle position, air bypass injection and spark.
It would be better to not log this data so the o2 return null since these values are invalid

Slide 5



In this example we have put a created_at property onto our collection and we want to have an event handler that parses the last second worth of data.
$gte is used when you want to do conditionals. This example is for greater than or equal to but there are plenty conditional that are part of the spec. $ne is one I use a lot.
Event handlers using computed values for the last second
if we need to evaluate if a sensor has been offline for 150 cycles we need listeners
timestamps would allow for analysis by time
This event handler can be used to determine a sensor is out of range.

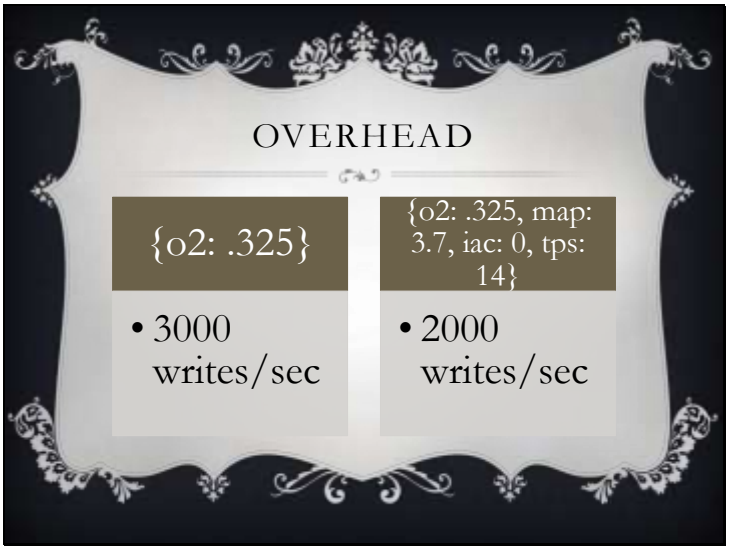
Slide 6

Since we don't know many documents per second that we are going to receive we set the capped collection to be tuned. Create a collection and log a single piece of data for as long as you want the event handler to be able to log and set the capped collection to that many documents
Since documents performance are based on size of the document, the best implementation can be set as a value based on the test.

You can then gauge your performance when you do a query on the t value for the time range the test was for. If our isolated test revealed maximized performance tooks 3000 documents

and in implementation there were 2000 documents returned from

Slide 7



This also leads to analysis of overhead and performance degredation.

Slide 8

use it as an extraction layer for an existing SQL instance by issuing sql commands and interpreting the results into a JSON log output and output to targit {name to be determined, I know it starts with the letter x and released in a later version of their bi tool; the whole reason I'm implementing it in a data center with the upgraded release version so all the new features unlock as part of the migration] which is designed to show multi dimensional views of predefined unputs.

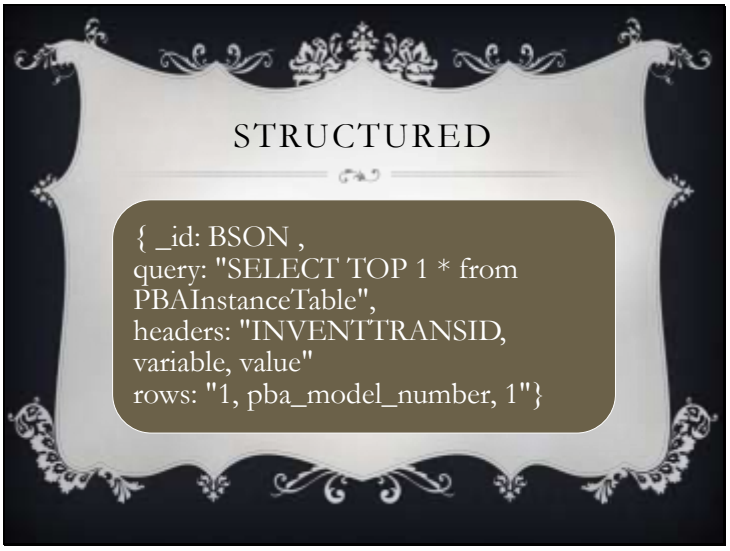
In troubleshooting large sql tables I often start by query
REF1
What is a PBAInstanceTable?
Options like vaulted, or finish, length, shade are records in sql

using reference number. I would want another SQL statement to return the collection of these configurations.

Slide 9

This table holds the variable to value relationship but instead of rows in a SQL table, I want to be able to look at the data as if it were a single instance of a product, since that is what it is.

Slide 10



Here is the json representation of that query. There are properties including
Query – the original code input into the sql query window
Headers: SQL always returns headers so we can capture this and log it
Rows: This data is just the string representation returned from the sql engine. This is how it would look if we were in MySQL command line or parsing responses from sqlcmd

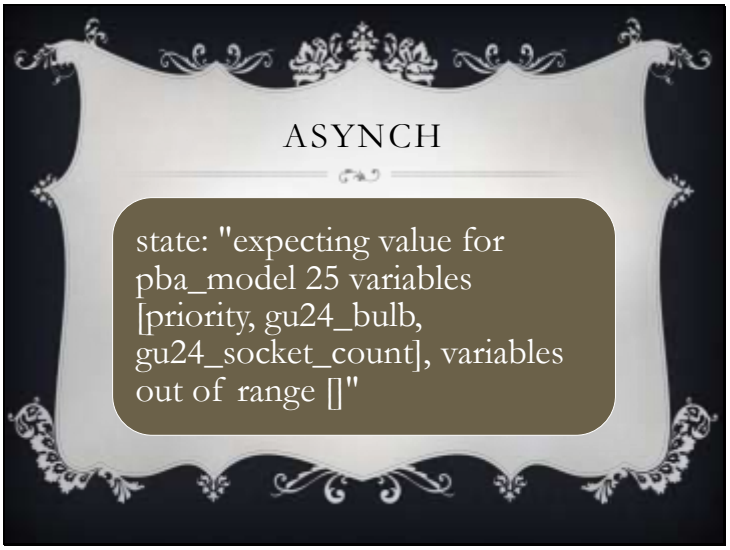
Slide 11

Again just logging the output of the example from earlier. Our output of sqlcmd uses commas and line breaks to format the rows, and all that fits nicely into the document without having to parse the output of the command to store the data.

Slide 12

Asynchronous process at a higher level language can parse this document and run an update statement that appends to the model embedded document.
We're using simple text parsing to pull the variable=value properties and attaching them directly to the schema of the database. The state property is to make it easier for the asynchronous process to know what work it has left to do.

Slide 13



The state is the addition of statemachine which allows the document processing state to follow the document. It allows built in error handling that happens within the document itself. MongoDB is a great thing to add features like statemachine for some documents.

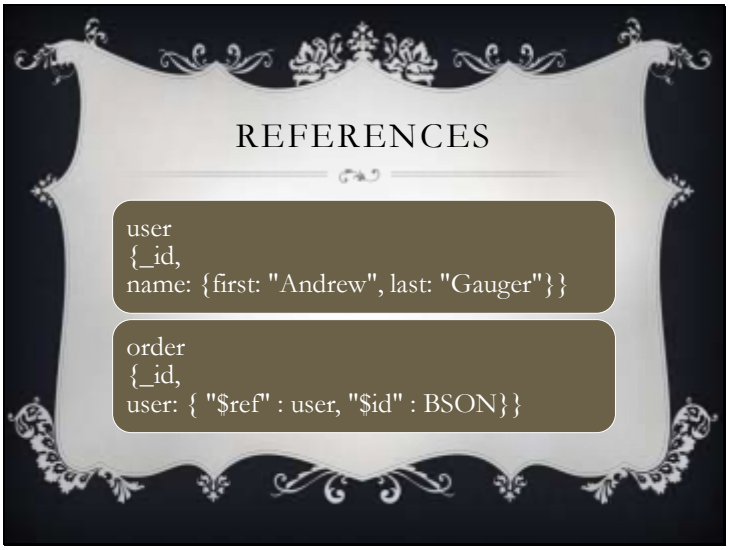
For example your state machine might be inspecting the model for the product and validating that all of the required configuration variables exist and if it detects an anomaly against the model it can set the state to something like this so the processing engine doesn't consume it again, and so the output of validation can be recognized by the user.

Slide 14

State machine often uses a cleaned model by keeping to predefined states. This would be a more likely implementation of the statemachine

As you can see here, mongodb has built in support for arrays embedded in the document.

Slide 15



db.order.find({ ... }).user.name.first
References
DBRef - $ref = collection, $id = document
Embedded documents an attribute on a document data can be a document
Solid built in referencing, and oop syntax of join is much easier to model and concpetualize.

Slide 16

now lets say we want to tie in a shutter from an external web cam.
we need to tie in an embedded document within the logger with the image captured from the camera so it gets tied to the sensors
We will also reference the image across multiple collections easily.

BSON id begins with the timestamp so just by logging data we get the timestamp and we can log any type of data, since the nature of binary storage is to save raw data. GridFS is also available if you intend to save very large dumps. GridFS would be applicable if you wanted to locally save images of computers

under version control. Replication would come in handy but these machines should have a network control task to limit their bandwidth, monitored, reported.

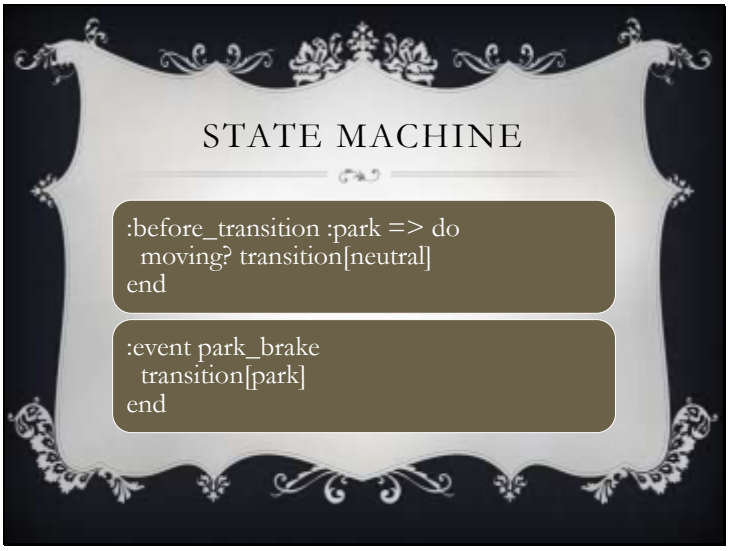
binary data storage makes storing any types of documents. UTF-8 seems to work well for me and file storage, and sensor data are all well optimized

The problem comes in with the overhead of a large database engine on top of small memory footprints. Transactional data being tied directly to point in time better suited application. User registrations, orders, inventory with relationships are far better suited. Mongo facilitates this through doc ref

Slide 17

sensor from laser readings nightly to apollo 1 Gw laser measuring the distance to a trillinth. This example is data stored as a kilometer
This is used to detect at what rate and acceleration the moon is distancing itself from earth.
Further precision can be created by continually updating thought the millisecond. These inserts can then be averaged.

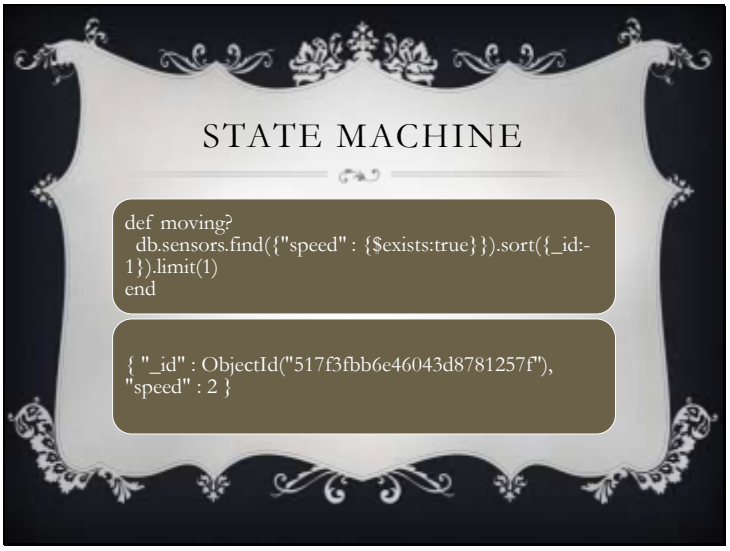
Slide 18



So we're heading back to our car/computer example. We want to set up a system of control for the transmission. We're going to set up an event handler to respond to pulling on the parking break to shift to park. We want to ensure that the sensor data for the vehicle in motion is compared before putting the car in park.

Optimizing state machine to do simplest, most critical validation just before the transition.

Slide 19



Here's the abstracted rubyish implementation of moving?

If you don't know ruby just know that def and end are the beginning and end of the method
You can see many interesting features implemented in this example
We're filtering our query by documents that have a property. This is for good reason since a schemaless database is designed to return when there isn't a property enabled for the document. Therefore, it returns a null. You can filter based on null by using $ne: null but that returns both documents where the property is set to null and for those that it doesn't exist.

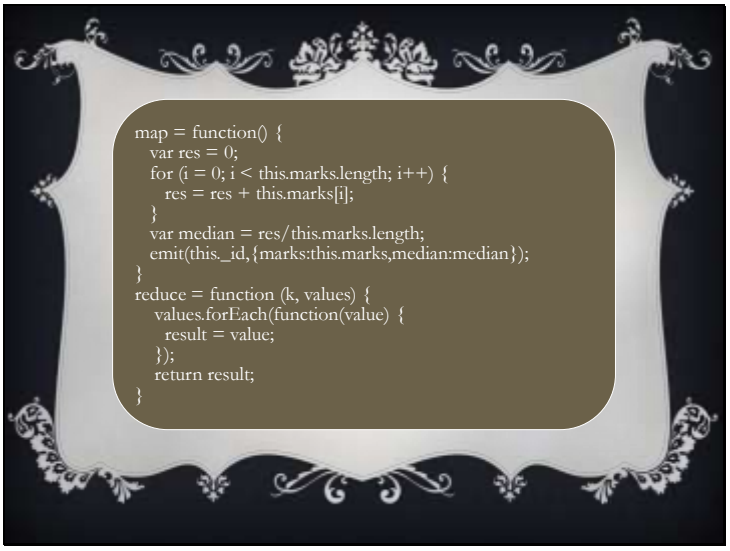
An example of the syntax for returning the most recent document. By default sorting is done on the _id field which in bson begins with the hexidecimal timestamp so these will be in transactional order.

Slide 20

aggregation provides powerful framework to do anything outside of the simple sum succinctly

But what we are looking to do is at the database be able to implement high level OOP into the database layer which is where we need functions added to our playground. a mapreduce might look like.

Slide 21

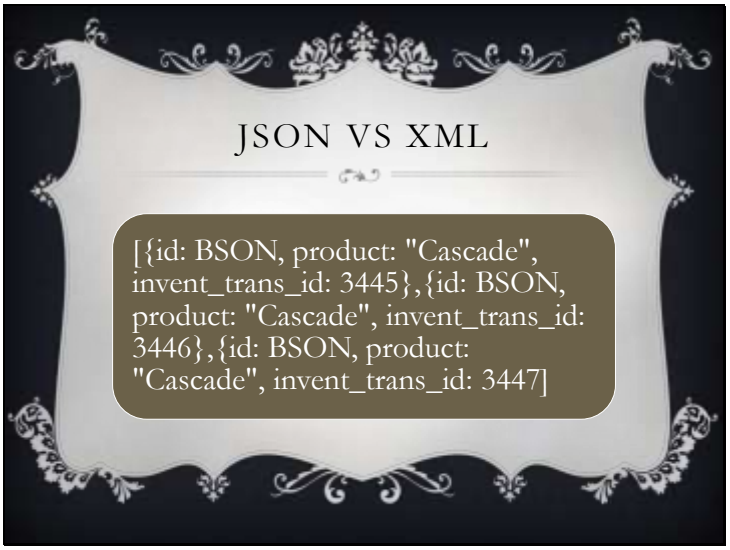


I found this map reduce example to perform statistical analysis on a data set. Map reduce provides a powerful framework to do anything you would want with data for analysis. Of the aggregation framework tools, this is likely the most powerful. Map is used to collect all the results of the query and perform a function against the data to create a total.

Slide 22

Group allows you to reduce without the map requirement. So you have this function defined. You can see here that out will have a few properties on it and within the function use simple object oriented dot notation to access properties. It is generally easier to get started with group than map reduce. However, always look at the map reduce implementation while building against group. There are almost always a way to do it either way, and I would recommend either efficiency in code because these things get difficult to read quickly, or tuned for production level performance.

Slide 23



Going to quickly compare the structure of JSON to XML. In this example, I've removed the whitespace. I've called out the id property as BSON in both examples to make this fair. Remember that BSON id takes up 96 bits.

Slide 24

Advantages of XML over HTTP was the design pattern, but many people AJAX without a problem. Special characters have to be handled a bit differently.

Slide 25

Notice the overhead of a collection holding 2 documents. 200 MB of storage used in this example as pure overhead so MongoDB is not a good implementation for the firmware that I had intended.
Within the local database references can be created to the sensors collection using an extended dbref syntax.

Slide 26

Questions and comments