Post-relational: NoSQL Modeling

Physical Data Modeling

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Kevin C.C. Chang, Professor

Computer Science @ Illinois

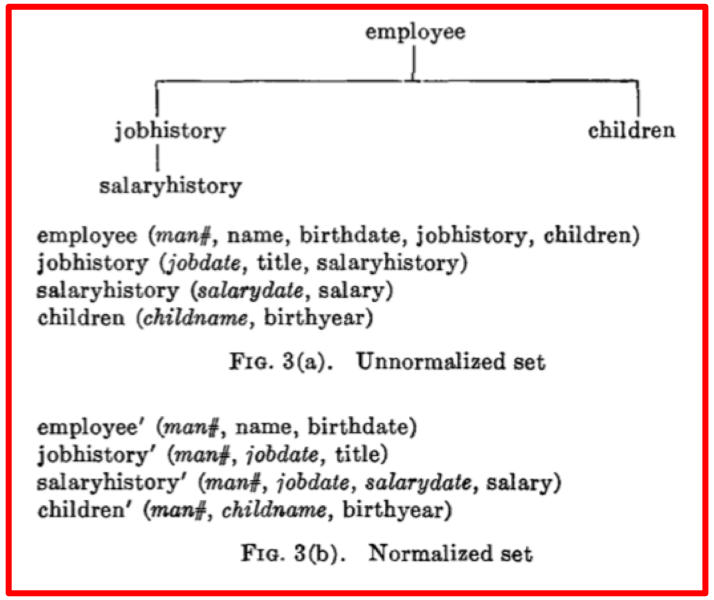
Learning Objectives

By the end of this video, you will be able to:

- Describe the “NoSQL” data models after the relational model.
- Identify the motivations behind these models.
- Give examples of such models.
- Define what JSON model is and how JSON schema works.

Post-Relational Models: Two Driving Forces

- **Meeting programming paradigms**
  - Driven by the ”impedance mismatch” with object-oriented programming.
  - (1980s) Object-Oriented.
  - (1980s) Object-Relational Model.
- **Dealing with data in various new settings**
  - Driven by applications beyond enterprise data management.
  - (1990s) Document Model.
  - (1990s) Key-Value Model.
  - (2000s) Graph Model.


employee
jobhistory
children
salaryhistory

employee (*man#*, name, birthdate, jobhistory, children)
jobhistory (*jobdate*, title, salaryhistory)
salaryhistory (*salarydate*, salary)
children (*childname*, birthyear)

Fig. 3(a). Unnormalized set

employee′ (*man#*, name, birthdate)
jobhistory′ (*man#*, *jobdate*, title)
salaryhistory′ (*man#*, *jobdate*, *salarydate*, salary)
children′ (*man#*, *childname*, birthyear)

Fig. 3(b). Normalized set

Example enterprise data management scenario from Codd’s 1970 relational model paper (Codd 1972)

Dealing with Data in Various New Settings → NoSQL

- New kinds of data: Web data, social networks, scientific data.
- New requirements
  - Volume → Scalability
    - Handling extremely large data.
    - Handling extremely many users.
  - Variety → One model may not fit all
    - Handling very simple to very complex data.
- NoSQL databases
  - Originally “non SQL” or “non relational”.
  - Now “not only SQL”.

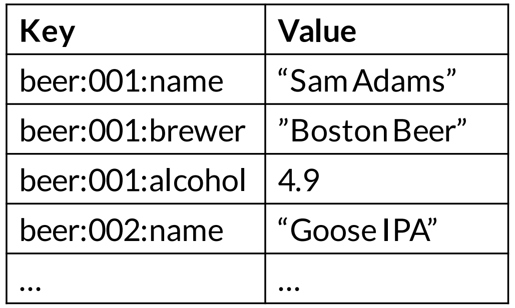
NoSQL Data Models

- Key-Value Model
  - Berkeley DB, Redis
- Document Model
  - MongoDB, CouchDB
  - JSON is a popular document model.
- Graph Model
  - Neo4j, OrientDB

Key-Value Model

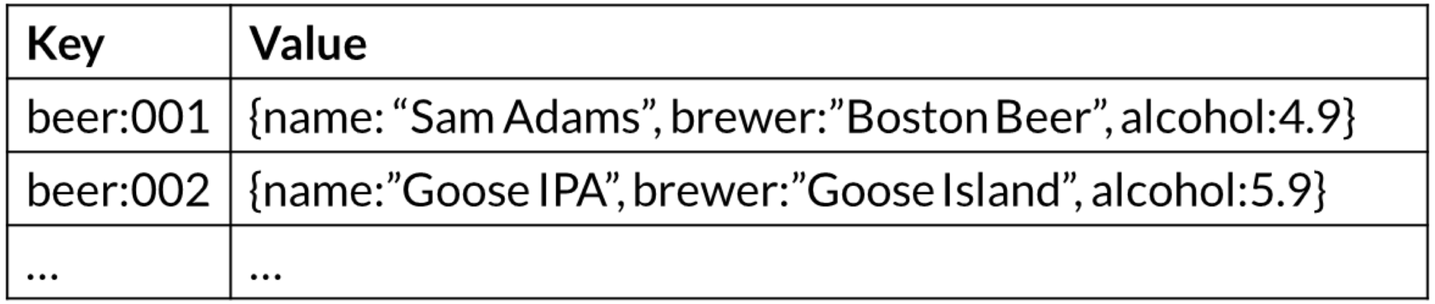
- DB = key-value pairs.
- Key: Any binary sequence given/named by programmers.
- Value: String, or more complex types list, hash, set (as in redis).
- Data model is effectively managed at applications.
- Good for simple data with mostly simple look-up queries.
  - e.g., collection of users, look-up password for logging in.

Example key-value database



Key	Value
beer:001:name	“Sam Adams”
beer:001:brewer	”Boston Beer”
beer:001:alcohol	4.9
beer:002:name	“Goose IPA”
…	…

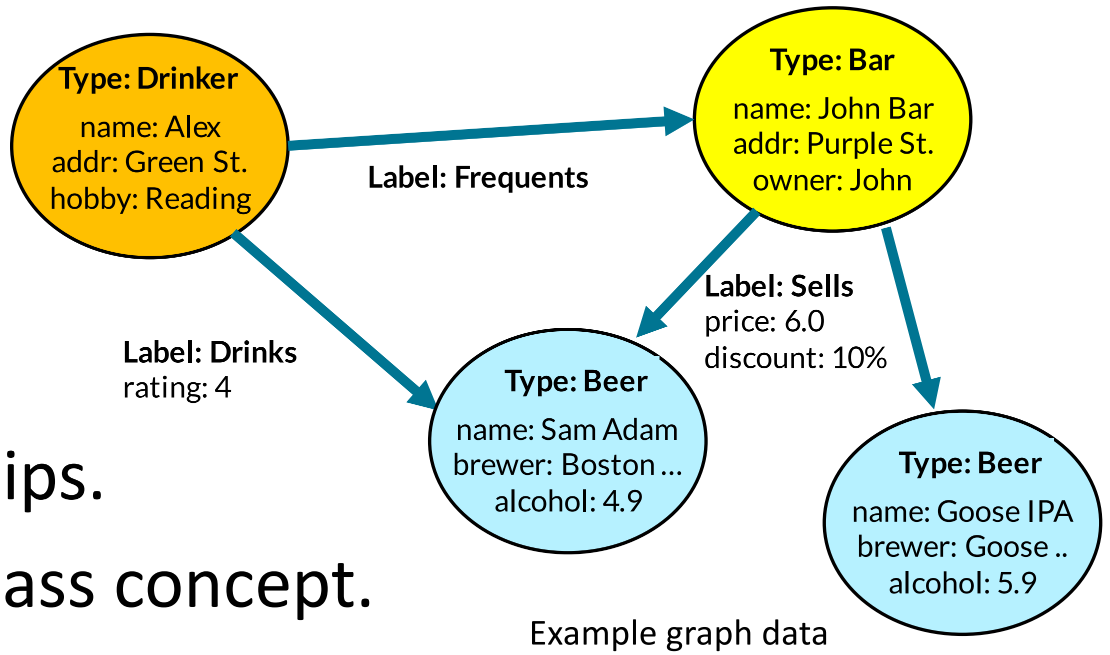
Example key-value store



Key	Value
beer:001	{name: “Sam Adams”, brewer:”Boston Beer”, alcohol:4.9}
beer:002	{name:”Goose IPA”, brewer:”Goose Island”, alcohol:5.9}
…	…

Example key-value store, with complex values

Graph Model



Example graph data

- Database = Graph.
- Node = Entities. Edges = Relationships.
- Key concept: Relationship as first class concept.
- *"Think of the ease and beauty of a well-done, normalized entity-relationship diagram: a simple, easy to understand model you can quickly whiteboard with your colleagues and domain experts. A graph is exactly that: a clear model of the domain, focused on the use cases you want to efficiently support."* Neo4j Blog, 2016.
- Network data model coming back?
  - Subtle differences, but similar in spirit.

neo4j

Example graph database

Document Model

- DB = Collections of Documents.
- Document encoded in nested structure, e.g., XML, JSON.
- Documents do not necessarily share the same schema.
  - But you can enforce a schema (or document validation rules).

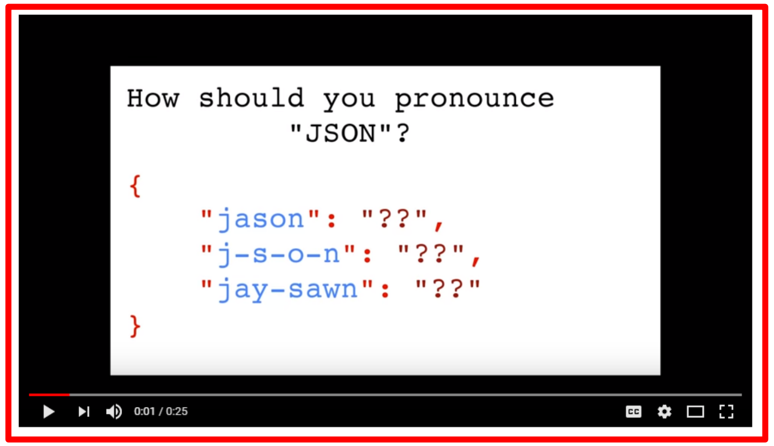
mongoDB

CouchDB relax

Example document databases

JSON: JavaScript Object Notation

- Standard for serializing data object in human-readable format.
- Popularized as common format for browser server data exchange.
- Then backend databases start to store data as JSON too.
  - And the idea of document databases emerged.
- Used by different languages/systems beyond JavaScript.
- How to pronounce ”JSON”?

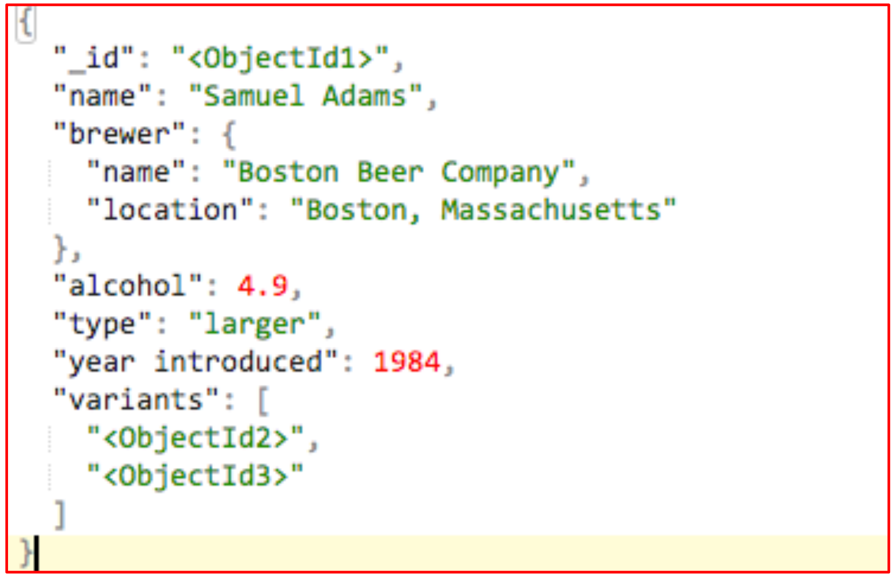

How should you pronounce
"JSON"?

```
{
    "jason": "??",
    "j-s-o-n": "??",
    "jay-sawn": "??"
}
```

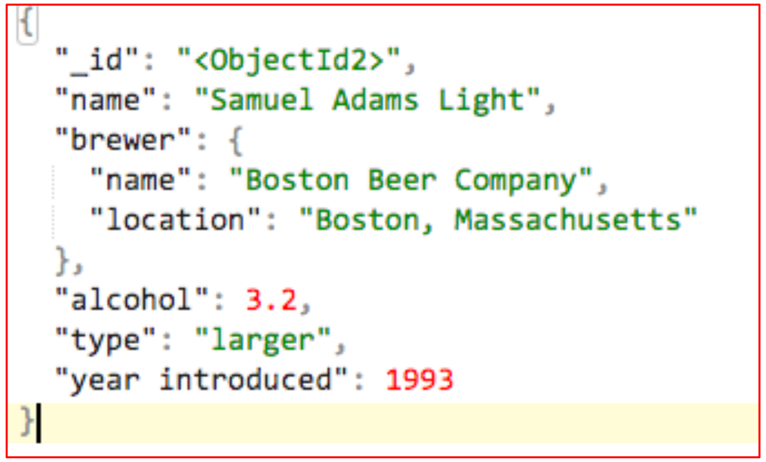
How should you pronounce “JSON”? Retrieved from https://www.youtube.com/watch?v=zhVdWQWKRqM

JSON Documents

- Human readable.
- Value: basic types are strings, numbers, booleans, null.
- Object: { field-value pair }, i.e., set of field-value pairs.
- Array: [value]
- Nesting: Value can be an embed objects or referenced objects (by object id).



```
{
  "_id": "<ObjectId1>",
  "name": "Samuel Adams",
  "brewer": {
    "name": "Boston Beer Company",
    "location": "Boston, Massachusetts"
  },
  "alcohol": 4.9,
  "type": "larger",
  "year introduced": 1984,
  "variants": [
    "<ObjectId2>",
    "<ObjectId3>"
  ]
}
```

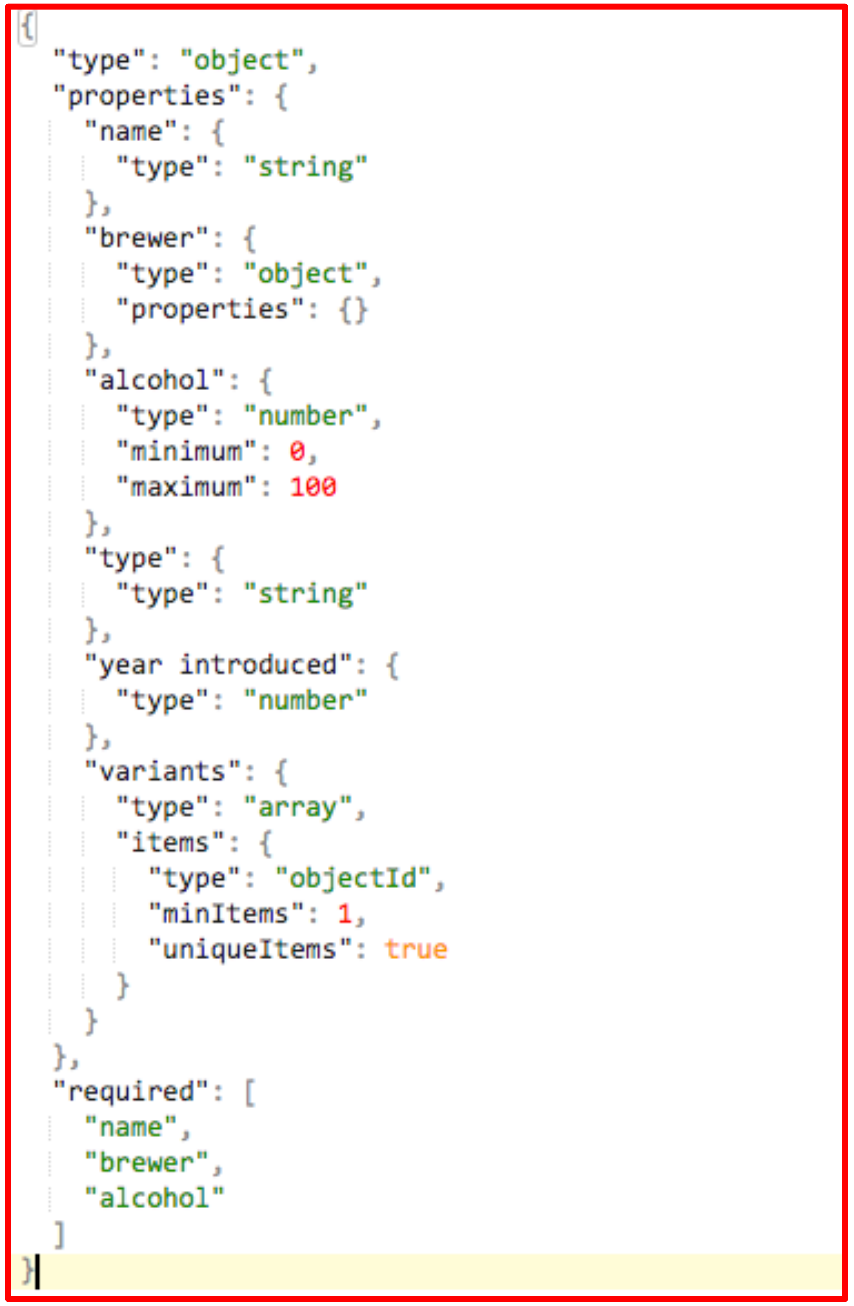


```
{
  "_id": "<ObjectId2>",
  "name": "Samuel Adams Light",
  "brewer": {
    "name": "Boston Beer Company",
    "location": "Boston, Massachusetts"
  },
  "alcohol": 3.2,
  "type": "larger",
  "year introduced": 1993
}
```

JSON documents (for Beers Collection)

JSON Schema

- We can mix different kinds of documents in a collection.
- We can also specify the structure of JSON data– by a JSON schema.
- In JSON format. Human readable
- Define structures
  - Set of properties (fields).
  - Type for each property.
  - Constraint for each property.

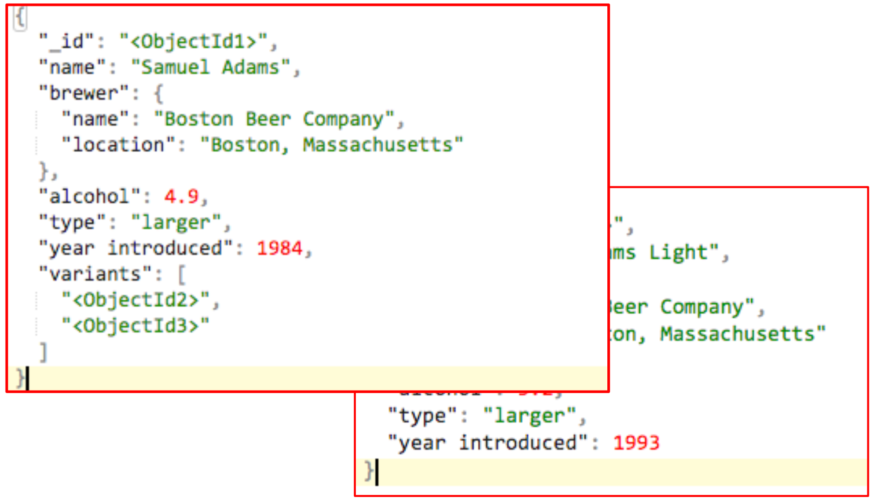


```
{
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "brewer": {
      "type": "object",
      "properties": {}
    },
    "alcohol": {
      "type": "number",
      "minimum": 0,
      "maximum": 100
    },
    "type": {
      "type": "string"
    },
    "year introduced": {
      "type": "number"
    },
    "variants": {
      "type": "array",
      "items": {
        "type": "objectId",
        "minItems": 1,
        "uniqueItems": true
      }
    }
  },
  "required": [
    "name",
    "brewer",
    "alcohol"
  ]
}
```

JSON schema

Food for Thought

How do you compare the **document model** to the **relational model**? What are major different concepts?



```
{
  "_id": "<ObjectId1>",
  "name": "Samuel Adams",
  "brewer": {
    "name": "Boston Beer Company",
    "location": "Boston, Massachusetts"
  },
  "alcohol": 4.9,
  "type": "larger",
  "year introduced": 1984,
  "variants": [
    "<ObjectId2>",
    "<ObjectId3>"
  ]
}
```

```
                  ",
                  ms Light",

                  eer Company",
                  on, Massachusetts"

  "type": "larger",
  "year introduced": 1993
}
```

vs.

name	brewer	alcohol
Sam Adams	Boston Beer	4.9
Goose IPA	Goose Island	5.9
Summer Ale	Boston Beer	5.3

References

- *Neo4j Blog, 2016*. RDBMS & Graphs: Relational vs. Graph Data Modeling. Retrieved from https://neo4j.com/blog/rdbms-vs-graph-data-modeling/.