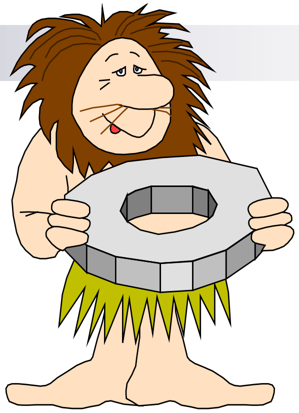
Introduction to MongoDB

Shanshan Zhang
tuf14438@temple.edu

Outline

- General Introduction to Database Management Systems
- MongoDB In Action
  - MongoDB Basics
  - MongoDB Operations with Python
  - MongoDB Analysis
- MongoDB Demo with Large-scale Data

What Is a DBMS?



- A very large, integrated collection of data.
- Models real-world *enterprise.*
  - Entities (e.g., students, courses)
  - Relationships (e.g., Madonna is taking CS564)
- A *Database Management System (DBMS)* is a software package designed to **store** and manage **data.**

What if We Don’t Want to Use DBMS?

- Alternatives
  – Store data in files (traditional OS file system)
  – Write application-specific code to manage it.
- What’s bad about it?
  – Special program for every scenario
  – Must protect the data from inconsistent changes

Files vs. DBMS

- Application must stage large datasets between main memory and secondary storage (e.g., buffering, page-oriented access, 32-bit addressing, etc.)
- Special code for different queries
- Must protect data from inconsistency due to multiple concurrent users
- Crash recovery
- Security and access control

Why Use a DBMS?

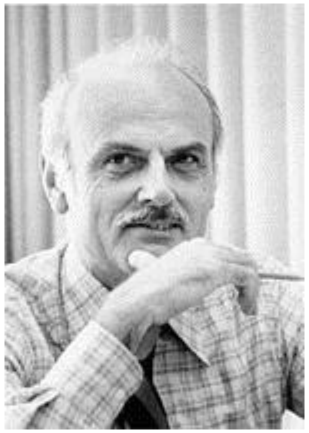
- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
- Concurrent access, recovery from crashes.

Why Study Databases??

- Shift from *computation* to *information*
  - at the “low end”: scramble to webspace (a mess!)
  - at the “high end”: scientific applications
- Datasets increasing in diversity and volume.
  - Digital libraries, interactive video, Human Genome project, EOS project
- DBMS encompasses most of CS
  - OS, languages, theory, AI, multimedia, logic

A Brief DB History

- Early 1970s
  - Many database systems
  - Incompatible, exposing many implementation details
- Then **Ted Codd** came along
  - Relational model
  - Structured Query Language (SQL)
  - Implementation differences became irrelevant
  - A few major DB systems dominated the market



Then Web 2.0 & 3.0, Big Data Happen

- What do you think happen?
  - Semi-structured data happen.
    - A lot of it and in many forms…

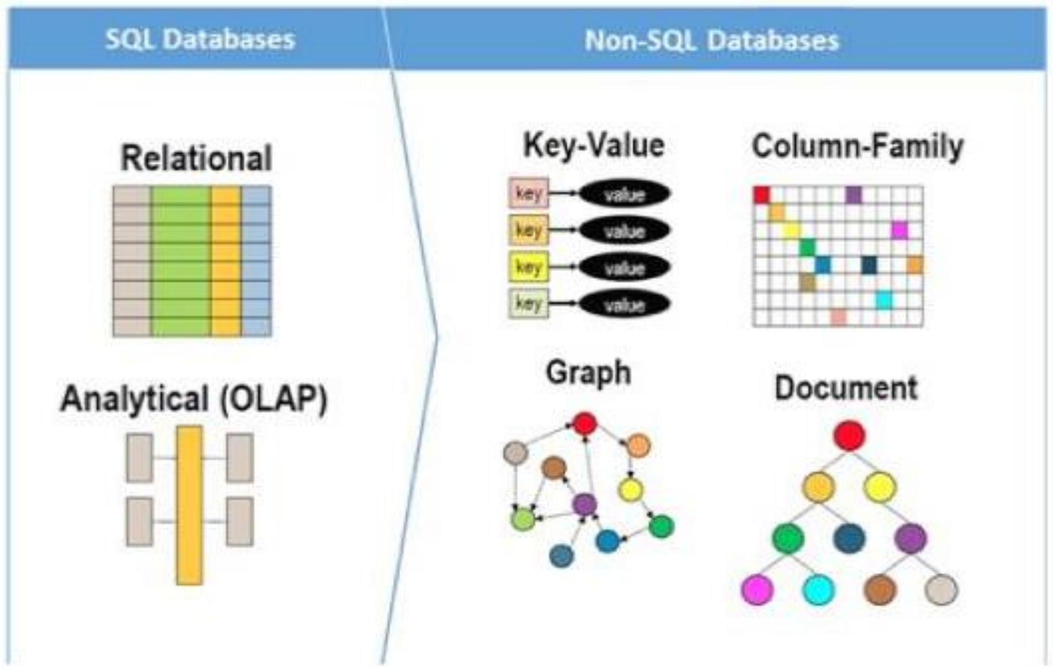
Some Facts about Web x.0 and Big Data

- Twitter: 255 million monthly active users and 500 million Tweets are sent per day,
- Facebook: over 1 billion monthly users and faces 3 million message per 20 minute
- Instagram: 200 Million Monthly Active Users and 1.6 Billion Likes and 60 Million Photos shared every day
- Linkedin: 2 new profiles are generated per second.

Database Systems Landscape Nowadays

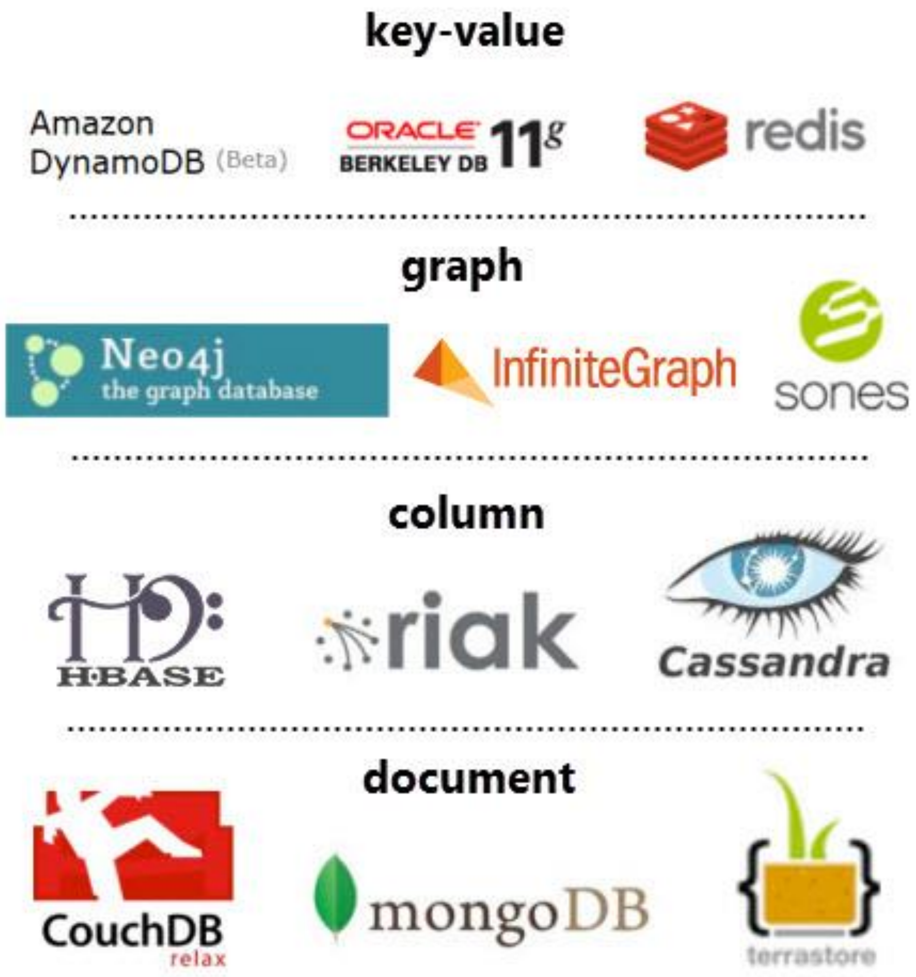
ORACLE
Microsoft SQL Server
SYBASE
PostgreSQL
DB2
TERADATA
MySQL
amazon DynamoDB
Cassandra
Redis
riak
mongoDB
MEMCACHED
HBASE
CouchDB relax
Neo4j
OrientDB

Somebody, Please, Bring Some Order to This Madness



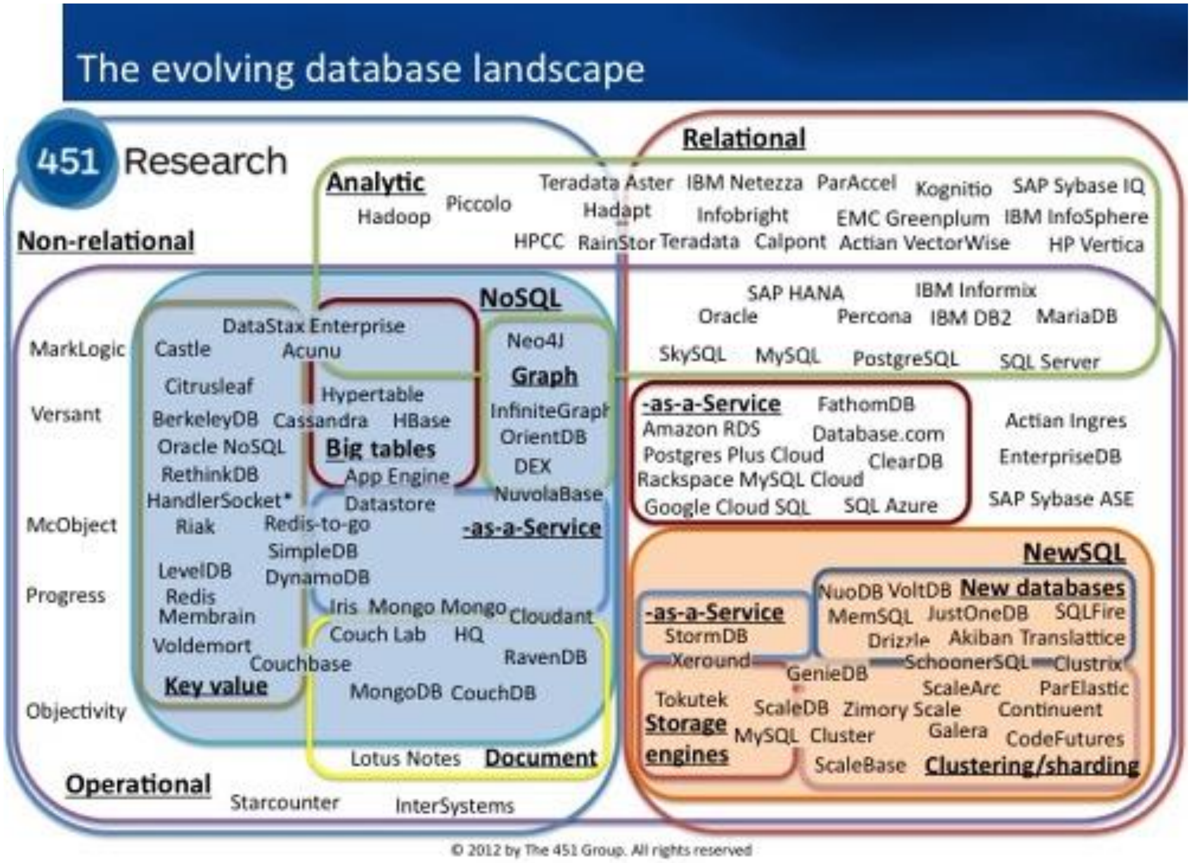
Somebody, Please, Bring Some Order to This Madness

- NoSQL Databases



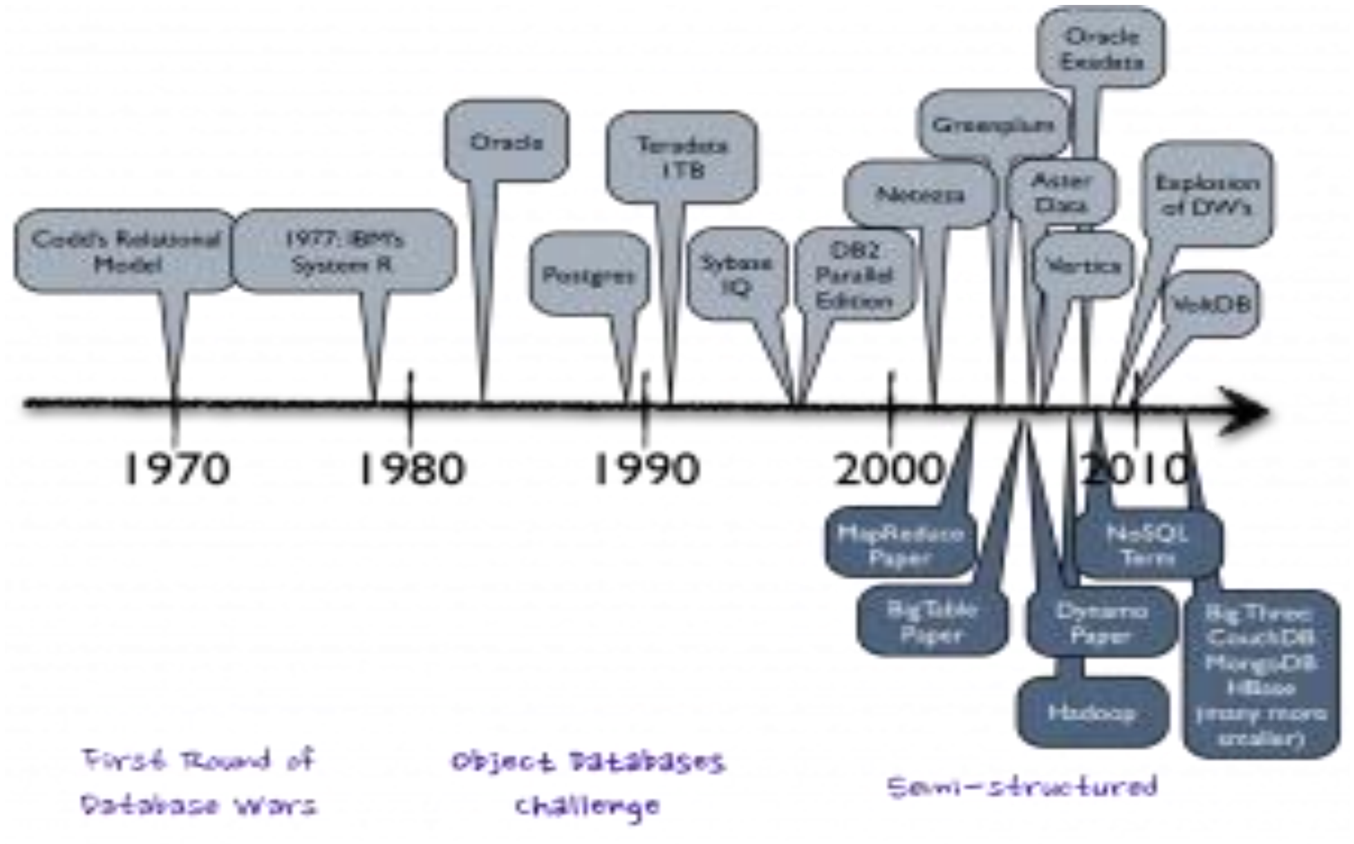
Somebody, Please, Bring Some Order to This Madness

- Different Interfaces
- Different hardware su
- Different application support
- Lack of Uniformity



Source: http://www.infoq.com/articles/State-of-NoSQL

Database Evolution Timeline

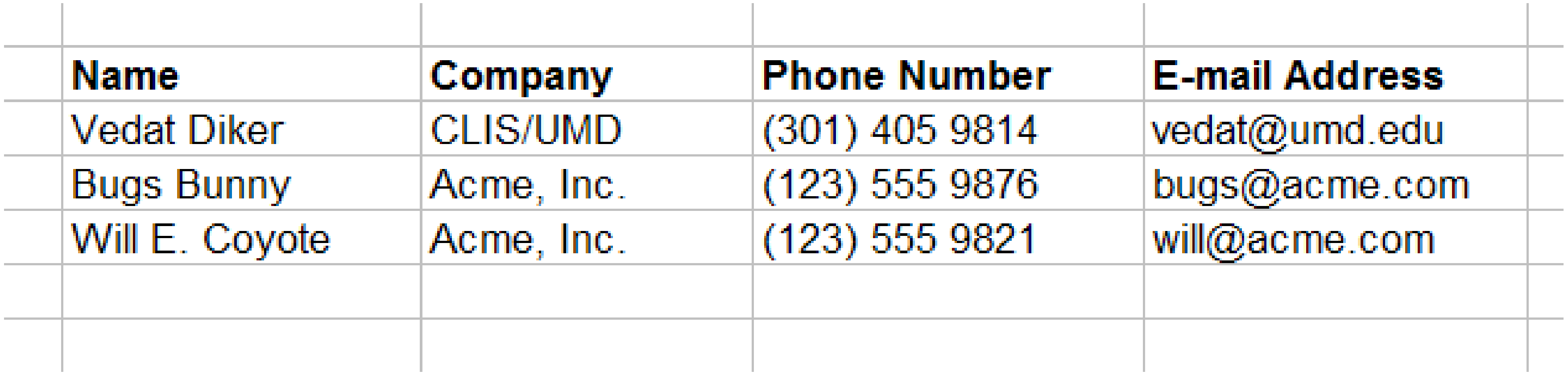


Additional Resources

- Tutorial by C. Mohan, An In-Depth Look at Modern Database Systems
- https://docs.google.com/file/d/0B7lNUaak0bK1encwYnBVUWZSWjA/edit

Relational DBMS

- Tables or Relations



Name	Company	Phone Number	E-mail Address
Vedat Diker	CLIS/UMD	(301) 405 9814	vedat@umd.edu
Bugs Bunny	Acme, Inc.	(123) 555 9876	bugs@acme.com
Will E. Coyote	Acme, Inc.	(123) 555 9821	will@acme.com

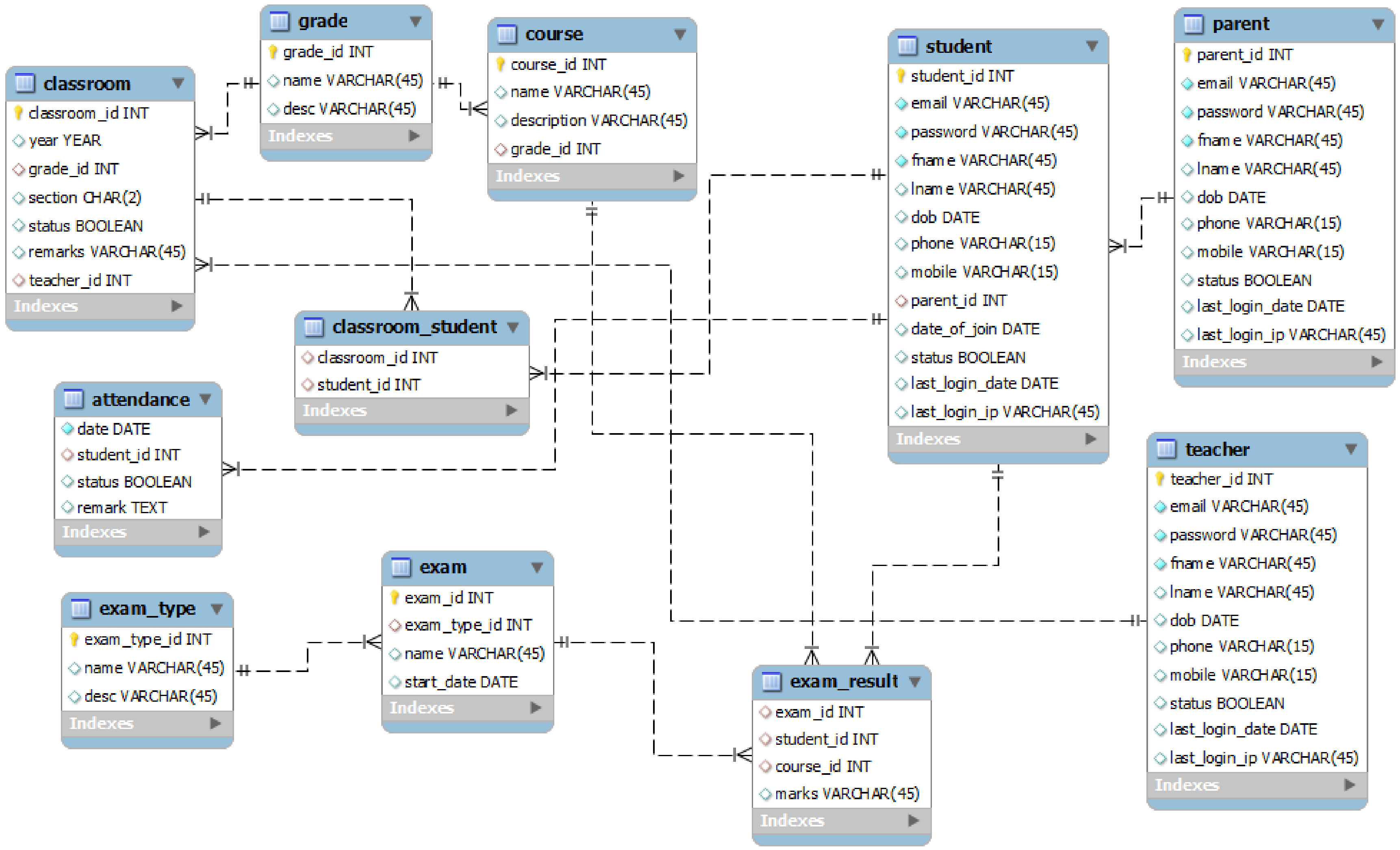
- Suppose you a person belongs to multiple companies?
- Companies have their own properties?

Example: University Database

Conceptual model:

- *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
- *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:string)*

Relational Database:Schemas



Beyond the data modeling

- SQL language for query. Define *what* you want, not *how* you want.

  eg. SELECT Student.name

  WHERE Student.grade = A

  FROM StudentFile Organization and **Indexing**
- Transaction Management
- Concurrency Control
- Crash Recovery
- Data Partition

What is MongoDB?

- Developed by 10gen
- It is a NoSQL database
- A document-oriented database
  - Not .pdf or .doc
  - Is associative array
  - Document == JSON Object
  - Document == PHP array
  - Document == PYTHON dict
  - Document == Ruby Hash
- It uses BSON format: Binary Json

The Basics

- A MongoDB instance may have zero or more databases
- A database may have zero or more ‘collections’.
- A collection may have zero or more ‘documents’.
- A document may have one or more ‘fields’.
- MongoDB ‘Indexes’ function much like their RDBMS counterparts.

MongoDB vs. RDBMS

RDBMS	Mongo
Table, View	Collection
Row(s)	JSON Document
Index	Index
Join	Embedded Document
Partition	Shard
Partition Key	Shard Key

Why we want MongoDB

Sharding and Load-Balancing. When you have extremely large amounts of data or you need to distribute your database traffic across multiple machines for load-balancing purposes, MongoDB has heavy advantages over many classic relational databases such as MySQL.

Speed. When you data is truly document!

Flexibility. It doesn't require a unified data structure across all objects

Data Modeling in MongoDB

A linkedin Page

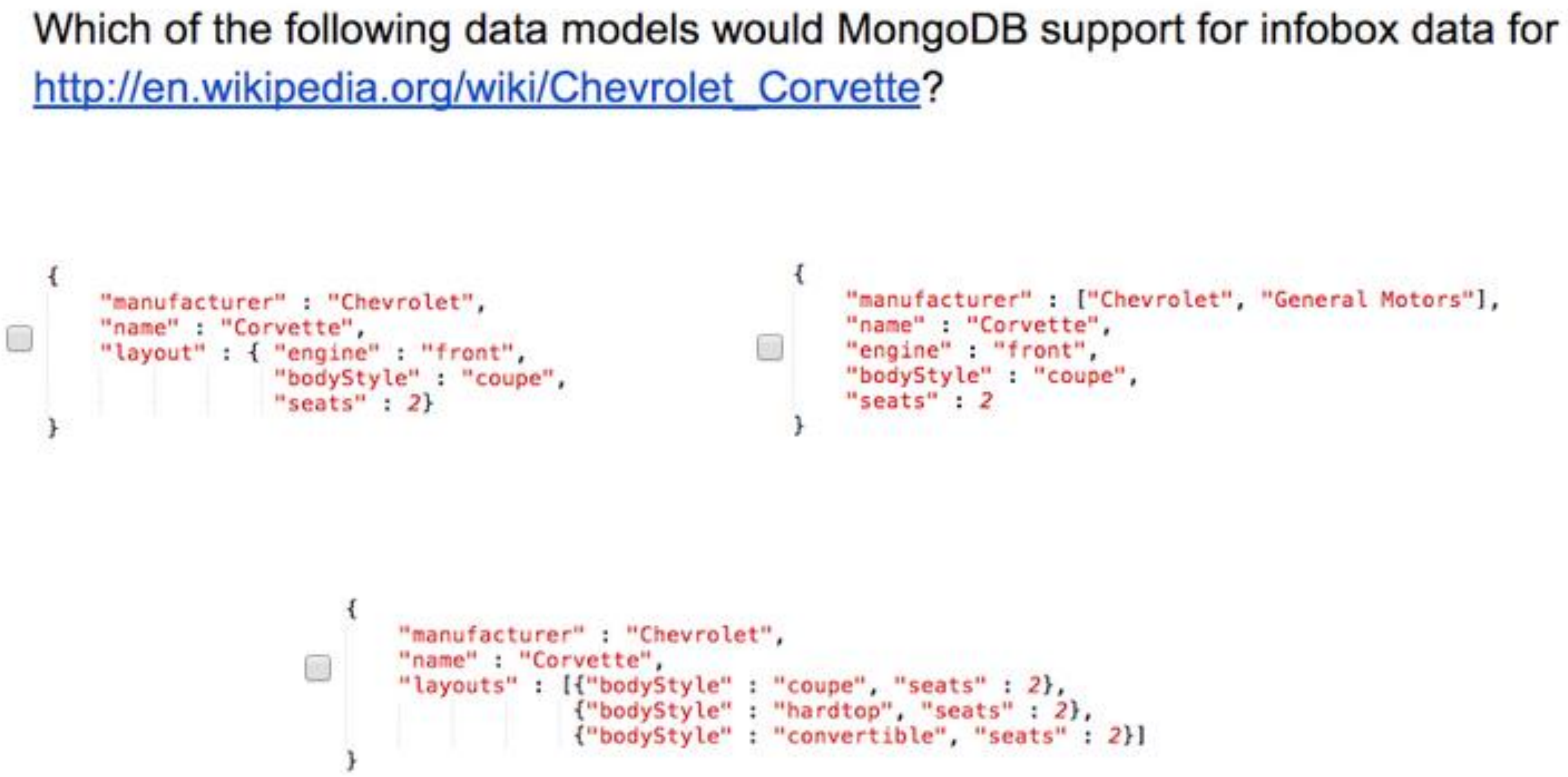
https://www.linkedin.com/in/shanewillis

- ❑ A page can be treated as a document.
- ❑ Info of interest can be stored in key/value pairs of a JSON Object
- ❑ The data type of value can be very flexible.
- ❑ Even the fields can be flexible.

```
{  "_id" : "extrapub_166/1",
   "fullname" : "Shane Willis",
   "edu_size" : 3,
   "exp_size" : 5
}
```

```
{  "_id" : "extrapub_166/1",
   "fullname" : "Shane Willis",
   "edu_size" : 3,
   "exp_size" : 5,
   "universities": ["University of Tasmania"]
}
```

Question



Which of the following data models would MongoDB support for infobox data for http://en.wikipedia.org/wiki/Chevrolet_Corvette?

☐
```
{
    "manufacturer" : "Chevrolet",
    "name" : "Corvette",
    "layout" : { "engine" : "front",
                 "bodyStyle" : "coupe",
                 "seats" : 2}
}
```

☐
```
{
    "manufacturer" : ["Chevrolet", "General Motors"],
    "name" : "Corvette",
    "engine" : "front",
    "bodyStyle" : "coupe",
    "seats" : 2
}
```

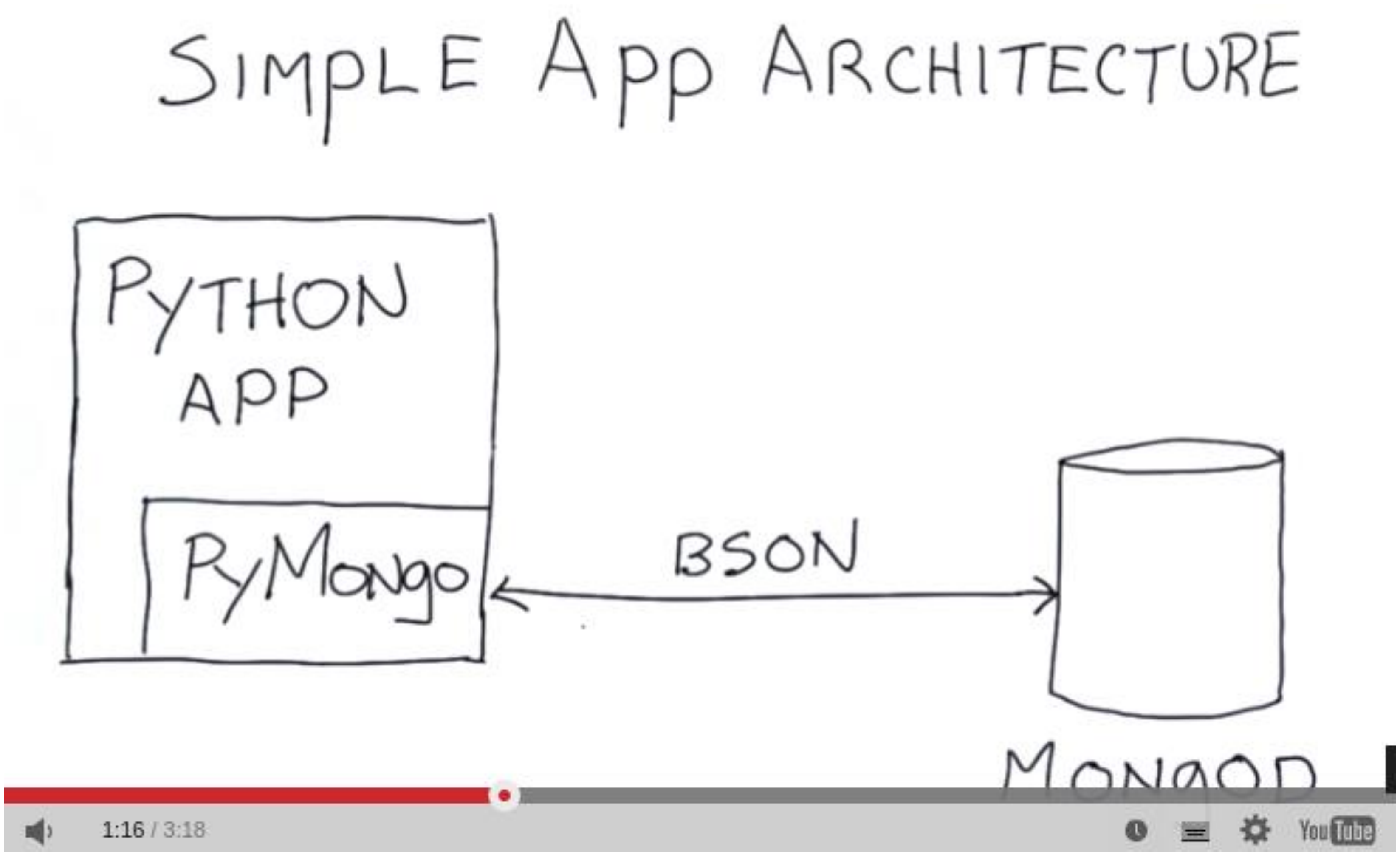
☐
```
{
    "manufacturer" : "Chevrolet",
    "name" : "Corvette",
    "layouts" : [{"bodyStyle" : "coupe", "seats" : 2},
                 {"bodyStyle" : "hardtop", "seats" : 2},
                 {"bodyStyle" : "convertible", "seats" : 2}]
}
```

Intro to PyMongo

- Two modes of working with MongoDB
  1 From the MongoShell (JavaScript)
  2 From MongoDB drivers in other languages, e.g., Python
- PyMongo is the Python driver for MongoDB.So we use PyMongo

```
> sudo pip install PyMongo
```

- Before you can use PyMongo, **MongoDB** should have been installed in your system.


SIMPLE App ARCHITECTURE
PYTHON APP
PyMongo
BSON
MONGOD
1:16 / 3:18

MongoDB Operations

- Insertion
- Querying

  e.g. db.linkedin.find({“fullname”: “CIS4340”})
- Updates
- Index
- Analysis

Examples are included in the scripts

Insert

- *Definition*: Store your modeled data (JSON objects) into the database.

- Command in Python and terminal:

```
In[]: db.collection.insert()
$ mongoimport --db db --collection c --file
```

Field query

- *Definition*: query in a database is to select a set of documents that meet the customized conditions.
- Commands in Python

```
In[]: db.collection.find({QUERY})
In[]: db.collection.find_one({QUERY})
```

- The query can be simple query, but it also can be nested document query

Projection

- *Definition*: Specify the fields to be shown in the query results.

- Command in Python

```
In[]: db.collection.find({QUERY},
  {PROJECTION})
```

Using Operators in Queries

- Use '$' to distinguish operators and field names
- Inequality operators: $gt, $ne, $le, $gte, $eq
- Others:

$exists, $regex, $size, $in, $all …

Updates

- *Definition*: Make changes to existing documents.
- Command in Python

```
In[]: db.collection.update({QUERY},
  {UPDATE})
```

- Operator: $set, $unset

Indexes

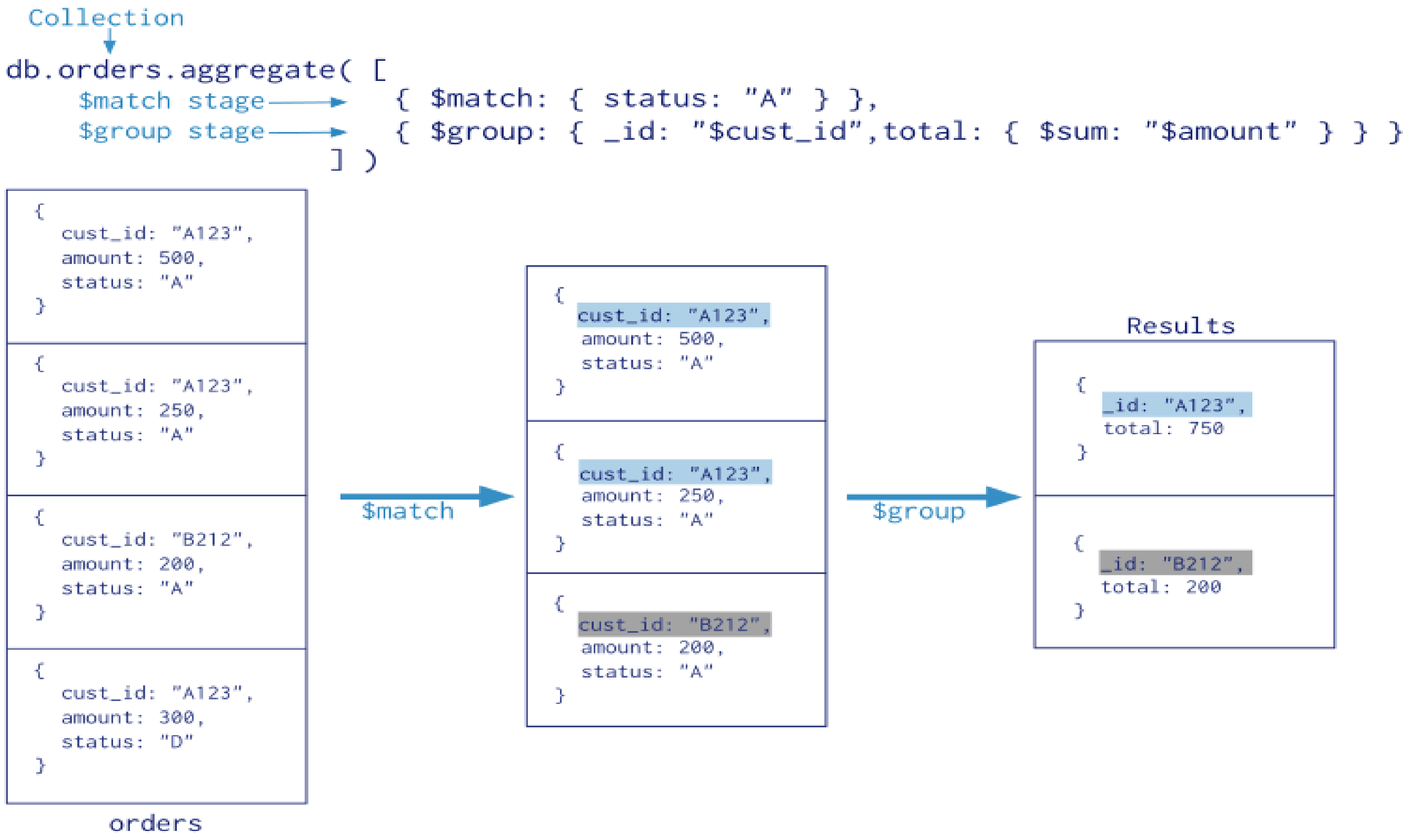
- *Definition*. A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and storage space to maintain the index data structure.
- Command in Python

```
In[]: db.collection.create_index('FIELD')
```

Advanced Analysis

- Simple analysis
  - In[]: db.collection.group()
  - In[]: db.collection.distinct()
- **Aggregation:**
  - In[]: db.collection.aggregate()
- **Map-reduce:**
  - In[]: db.collection.map_reduce()

Pipeline of MongoDB analysis

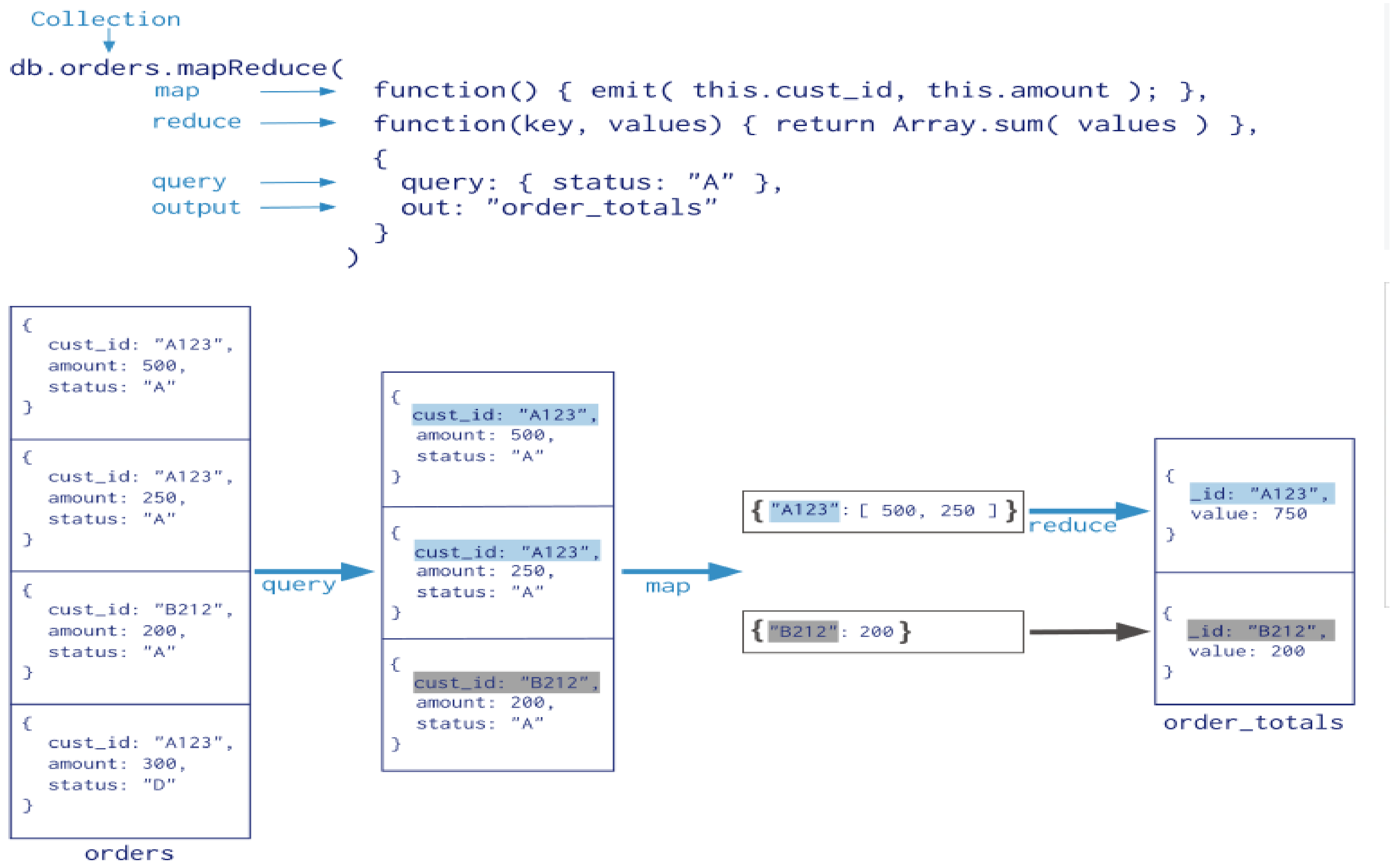


```
db.orders.aggregate( [
      { $match: { status: "A" } },
      { $group: { _id: "$cust_id",total: { $sum: "$amount" } } }
] )
```

```
orders:
{
  cust_id: "A123",
  amount: 500,
  status: "A"
}
{
  cust_id: "A123",
  amount: 250,
  status: "A"
}
{
  cust_id: "B212",
  amount: 200,
  status: "A"
}
{
  cust_id: "A123",
  amount: 300,
  status: "D"
}
```

```
{
  cust_id: "A123",
  amount: 500,
  status: "A"
}
{
  cust_id: "A123",
  amount: 250,
  status: "A"
}
{
  cust_id: "B212",
  amount: 200,
  status: "A"
}
```

```
Results:
{
  _id: "A123",
  total: 750
}
{
  _id: "B212",
  total: 200
}
```



```
  Collection
db.orders.mapReduce(
         map    →   function() { emit( this.cust_id, this.amount ); },
         reduce →   function(key, values) { return Array.sum( values ) },
                    {
         query  →     query: { status: "A" },
         output →     out: "order_totals"
                    }
                  )
```

```
{
  cust_id: "A123",
  amount: 500,
  status: "A"
}
{
  cust_id: "A123",
  amount: 250,
  status: "A"
}
{
  cust_id: "B212",
  amount: 200,
  status: "A"
}
{
  cust_id: "A123",
  amount: 300,
  status: "D"
}
```

orders

```
{
  cust_id: "A123",
  amount: 500,
  status: "A"
}
{
  cust_id: "A123",
  amount: 250,
  status: "A"
}
{
  cust_id: "B212",
  amount: 200,
  status: "A"
}
```

```
{ "A123": [ 500, 250 ] }
{ "B212": 200 }
```

```
{
  _id: "A123",
  value: 750
}
{
  _id: "B212",
  value: 200
}
```

order_totals

MongoDB Demo with Large-scale Data

Dataset:

65,000,000~ linkedin pages

Data Model:

Parse each page as JSON Object, with fields of interest.

Size:

Original : 1.3 Terabytes

Collection size in DB: 80~ GB

Tasks and Challenges

Task 1: Search for the existence of a certain page link.

Strategy	Time
Scan original files	5~ hours
Store link in DB, query without index	35~ seconds
Store link in DB, query without index	2~ seconds

Tasks and Challenges

Task 2: Get all the unique university names, and count the number of students using each name.

No easy way, used the map reduce of MongoDB to do it.

Disadvantages

No Joins

In MongoDB there exists no possibility for joins like in a relational database. This means that when you need this type of functionality, you need to make multiple queries and join the data manually within your code (which can lead to slow, ugly code, and reduced flexibility when the structure changes).

Disadvantages

Memory usage

Concurrency issues

Young software; Inexperienced User-Base; Still Under Construction; Little Documentation

transactions

What is your choice?

RDBMS or NoSQL

The point should be made that one should definitely not attempt to force a square peg through a circular hole. Don't try to force your data into a particular model just so that you can use a particular database; you will have issues in the future if you try. Don't do things just because they're cool; do them because it makes sense.

Thanks!