Data Management for Data Science

Introduction to NoSQL Databases

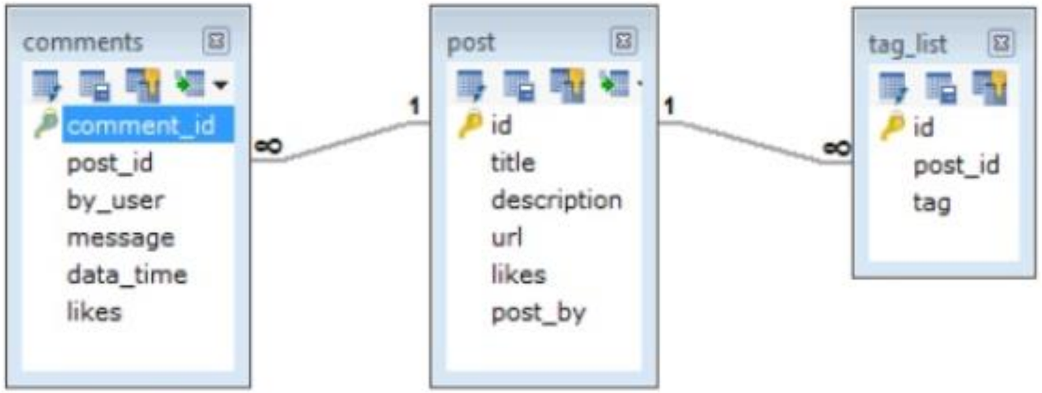
Başar Öztayşi

oztaysib@itu.edu.tr

Agenda

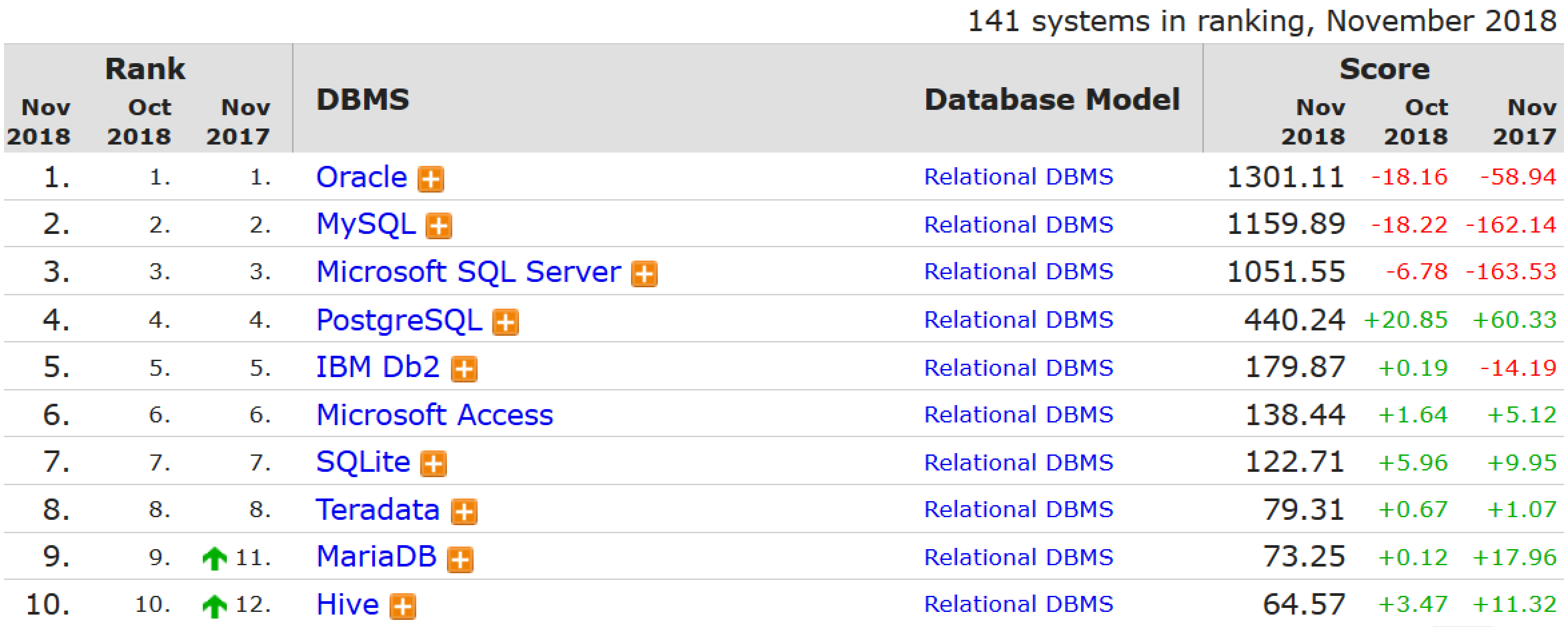
- Relational Databases
- Why NoSQL ?
- No-SQL DB
  - Key-Value
  - Wide Column
  - Document Based
  - Graph DB

Relational Data Model



Relational Data Model

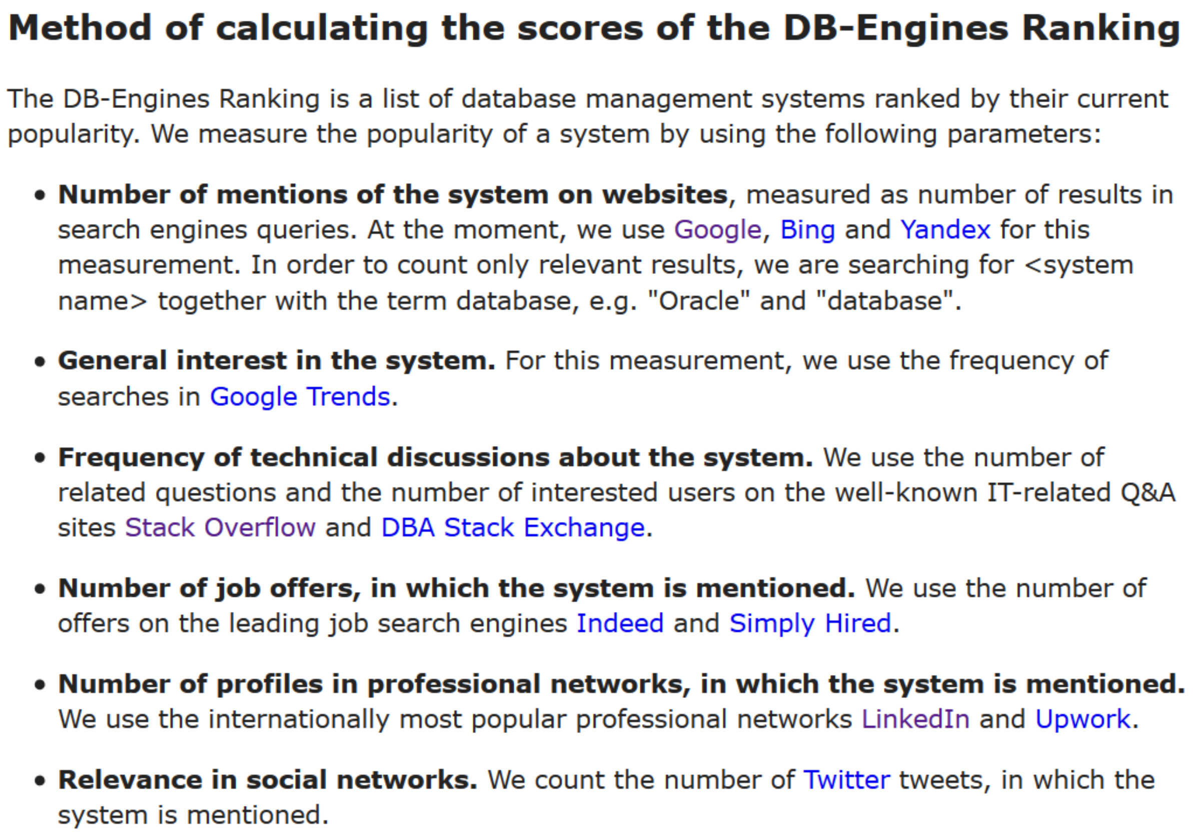
Relational Databases



141 systems in ranking, November 2018

Rank			DBMS	Database Model	Score		
Nov 2018	Oct 2018	Nov 2017			Nov 2018	Oct 2018	Nov 2017
1.	1.	1.	Oracle	Relational DBMS	1301.11	-18.16	-58.94
2.	2.	2.	MySQL	Relational DBMS	1159.89	-18.22	-162.14
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1051.55	-6.78	-163.53
4.	4.	4.	PostgreSQL	Relational DBMS	440.24	+20.85	+60.33
5.	5.	5.	IBM Db2	Relational DBMS	179.87	+0.19	-14.19
6.	6.	6.	Microsoft Access	Relational DBMS	138.44	+1.64	+5.12
7.	7.	7.	SQLite	Relational DBMS	122.71	+5.96	+9.95
8.	8.	8.	Teradata	Relational DBMS	79.31	+0.67	+1.07
9.	9.	↑ 11.	MariaDB	Relational DBMS	73.25	+0.12	+17.96
10.	10.	↑ 12.	Hive	Relational DBMS	64.57	+3.47	+11.32

Method



Method of calculating the scores of the DB-Engines Ranking

The DB-Engines Ranking is a list of database management systems ranked by their current popularity. We measure the popularity of a system by using the following parameters:

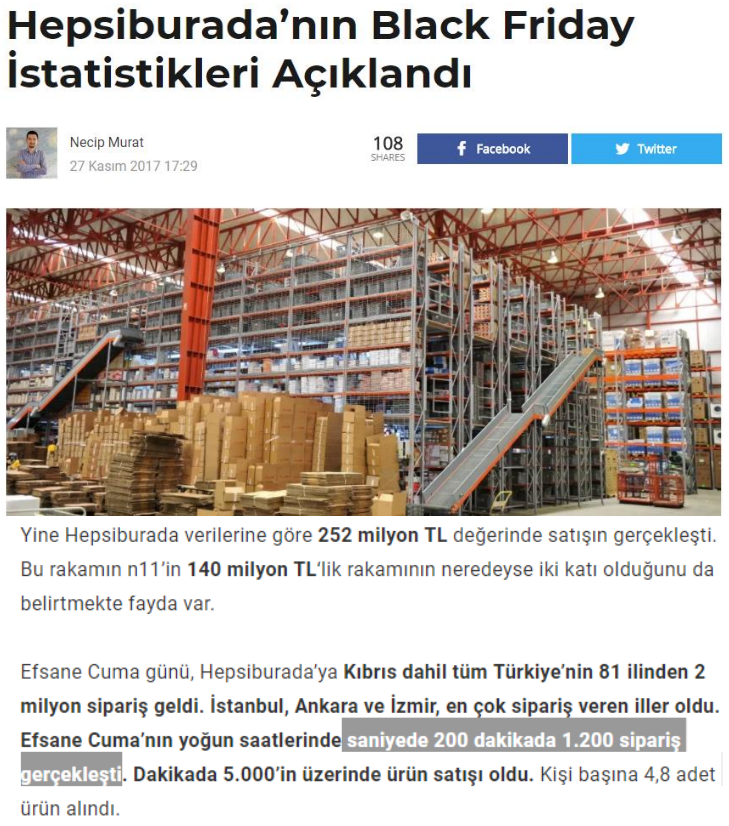
- **Number of mentions of the system on websites**, measured as number of results in search engines queries. At the moment, we use Google, Bing and Yandex for this measurement. In order to count only relevant results, we are searching for <system name> together with the term database, e.g. "Oracle" and "database".
- **General interest in the system.** For this measurement, we use the frequency of searches in Google Trends.
- **Frequency of technical discussions about the system.** We use the number of related questions and the number of interested users on the well-known IT-related Q&A sites Stack Overflow and DBA Stack Exchange.
- **Number of job offers, in which the system is mentioned.** We use the number of offers on the leading job search engines Indeed and Simply Hired.
- **Number of profiles in professional networks, in which the system is mentioned.** We use the internationally most popular professional networks LinkedIn and Upwork.
- **Relevance in social networks.** We count the number of Twitter tweets, in which the system is mentioned.

Motivations for NoSQL DBs

- Four characteristics of data management systems that are particularly important for large-scale data management tasks are
  - Scalability
  - Cost
  - Flexibility
  - Availability
- Depending on the needs of a particular application, some of these characteristics may be more important than others

Scalability

- Scalability is the ability to efficiently meet the needs for varying workloads.
  - Scale Out: Adding servers as needed
  - Scale Up: upgrading an existing database server to add additional processors, memory, network bandwidth etc.



Hepsiburada'nın Black Friday İstatistikleri Açıklandı

Necip Murat
27 Kasım 2017 17:29

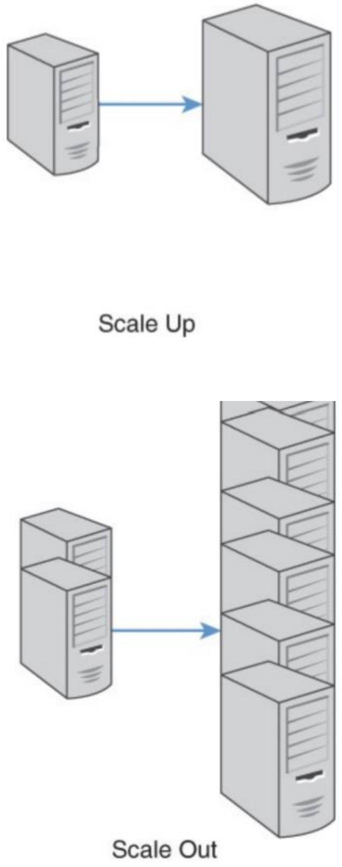
108 SHARES | Facebook | Twitter

Yine Hepsiburada verilerine göre **252 milyon TL** değerinde satışın gerçekleşti. Bu rakamın n11'in **140 milyon TL**'lik rakamının neredeyse iki katı olduğunu da belirtmekte fayda var.

Efsane Cuma günü, Hepsiburada'ya **Kıbrıs dahil tüm Türkiye'nin 81 ilinden 2 milyon sipariş geldi. İstanbul, Ankara ve İzmir, en çok sipariş veren iller oldu. Efsane Cuma'nın yoğun saatlerinde saniyede 200 dakikada 1.200 sipariş gerçekleşti. Dakikada 5.000'in üzerinde ürün satışı oldu.** Kişi başına 4,8 adet ürün alındı.

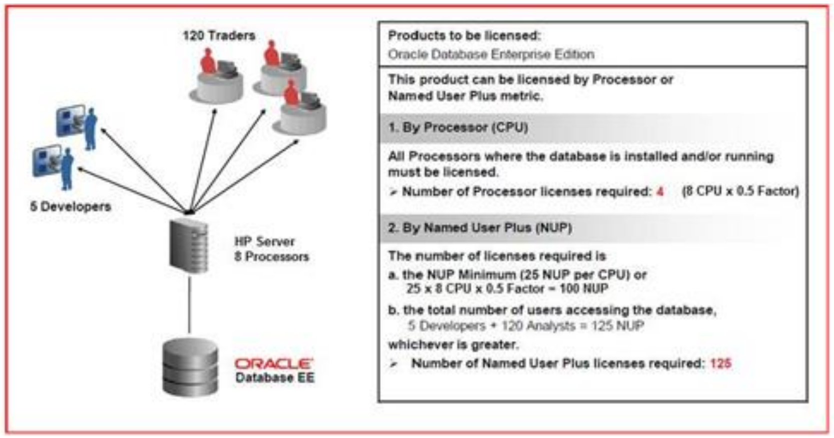
Scalability

- Scaling out is more flexible than scaling up. Servers can be added or removed as needed.
- Scaling up by replacing a server requires migrating the database management to a new server. -> require some downtime
- In relational databases, it is often challenging to scale out, generally scale up is preferred.
- NoSQL databases are designed to utilize servers available in a cluster with minimal intervention by database administrator.



Cost

- Commercial software vendors employ a variety of licensing models that include charging
  - by the size of the server running the RDBMS,
  - by the number of concurrent users on the database,
  - by the number of named users allowed to use the software.
- Web applications may have spikes in demand that increase the number of users utilizing a database at any time.
  - How should they budget for RDBMS licenses when it is difficult to know how many users will be using the system six months or a year from now?



Products to be licensed:
Oracle Database Enterprise Edition

This product can be licensed by Processor or Named User Plus metric.

1. By Processor (CPU)

All Processors where the database is installed and/or running must be licensed.

➢ Number of Processor licenses required: 4 (8 CPU x 0.5 Factor)

2. By Named User Plus (NUP)

The number of licenses required is

a. the NUP Minimum (25 NUP per CPU) or
25 x 8 CPU x 0.5 Factor = 100 NUP

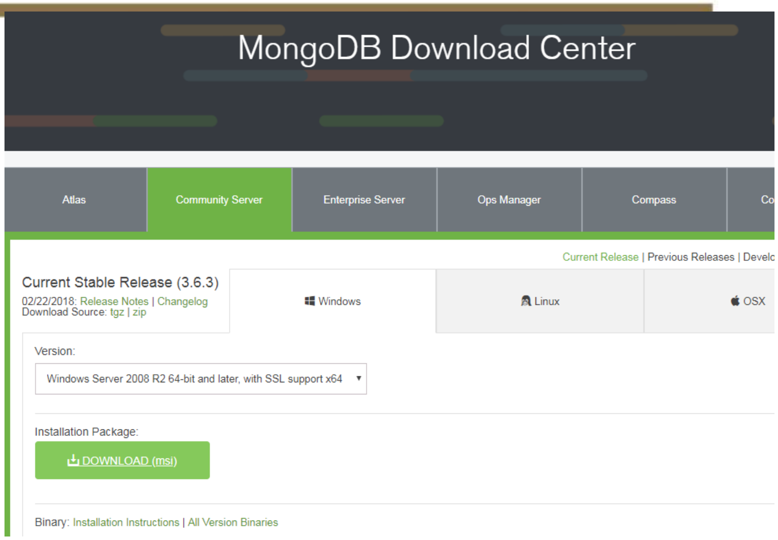
b. the total number of users accessing the database,
5 Developers + 120 Analysts = 125 NUP

whichever is greater.

➢ Number of Named User Plus licenses required: 125

Cost

- Users of open source software avoid this issues.
- The software is free to use on as many servers.
- Fortunately the major NoSQL databases are available as open source.
- Third-party companies provide commercial support services for open source NoSQL databases so businesses can have software support as they do with commercial relational databases.



MongoDB Download Center

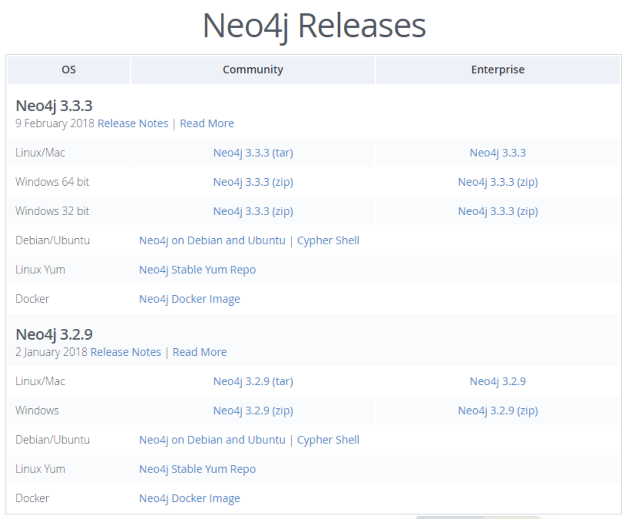
Current Release | Previous Releases | Develo

Current Stable Release (3.6.3)
02/22/2018: Release Notes | Changelog
Download Source: tgz | zip

Version:
Windows Server 2008 R2 64-bit and later, with SSL support x64

Installation Package:

Binary: Installation Instructions | All Version Binaries



Neo4j Releases

OS	Community	Enterprise
Neo4j 3.3.3 9 February 2018 Release Notes | Read More		
Linux/Mac	Neo4j 3.3.3 (tar)	Neo4j 3.3.3
Windows 64 bit	Neo4j 3.3.3 (zip)	Neo4j 3.3.3 (zip)
Windows 32 bit	Neo4j 3.3.3 (zip)	Neo4j 3.3.3 (zip)
Debian/Ubuntu	Neo4j on Debian and Ubuntu | Cypher Shell	
Linux Yum	Neo4j Stable Yum Repo	
Docker	Neo4j Docker Image	
Neo4j 3.2.9 2 January 2018 Release Notes | Read More		
Linux/Mac	Neo4j 3.2.9 (tar)	Neo4j 3.2.9
Windows	Neo4j 3.2.9 (zip)	Neo4j 3.2.9 (zip)
Debian/Ubuntu	Neo4j on Debian and Ubuntu | Cypher Shell	
Linux Yum	Neo4j Stable Yum Repo	
Docker	Neo4j Docker Image	

Flexibility

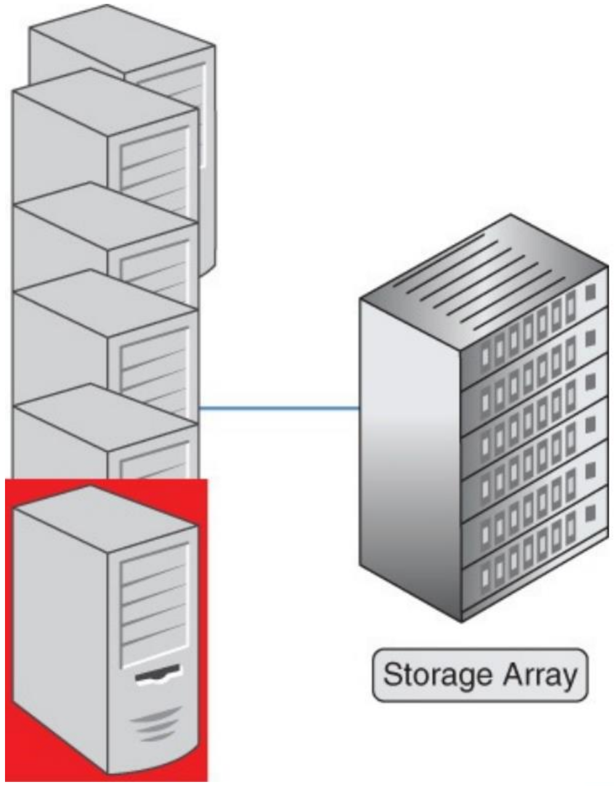
- Relational database management systems are flexible in the range of problems that can be addressed using relational data models.
- However, another aspect of relational databases that is less flexible.
- Database designers expect to know at the start of a project all the tables and columns that will be needed to support an application.

  → What if you need to add a new feature to data on the fly ?
- It is also commonly assumed that most of the columns in a table will be needed by most of the rows. For example, all employees will have names and employee Ids.

  -> What if most of the columns are empty ? (i.e. Ecommerce website properties of the products)
- NoSQL databases do not require a fixed table structure
- A program could dynamically add new attributes as needed without having to have a database designer alter the database design

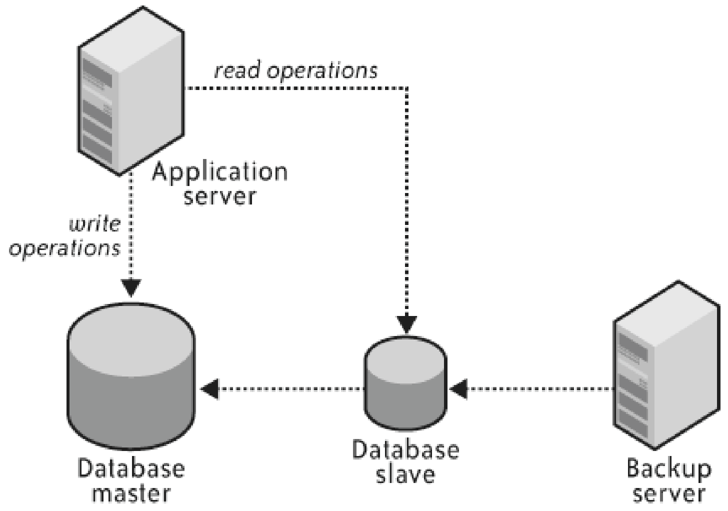
Availability

- NoSQL databases are designed to take advantage of multiple, low-cost servers.
  - When one server fails or is taken out of service for maintenance, the other servers in the cluster can take on the entire workload.
  - Performance may be somewhat less, but the application will still be available.



Availability

- Single Point of Failure:
  - If a database is running on a single server and it fails, the application will become unavailable unless there is a backup server.
  - Backup servers keep replicated copies of data from the primary server in case the primary server fails. If that happens, the backup can take on the workload that the primary server had been processing.
  - This can be an inefficient configuration because a server is kept in reserve in the event of a failure but otherwise is not helping to process the workload.



Why NoSQL ?

- Relational databases provide various successful applications!!
- However, the exponential growth of e-commerce and social media led to the need for data management systems that were scalable, low cost, flexible, and highly available.
- NoSQL databases were created to address these limitations of relational database management systems.
- NoSQL databases are unlikely to displace relational databases the way RDBMSs displaced flat file, hierarchical, and network databases.
- The two will likely complement each other and adapt features from each other…

Types of NoSQL Databases

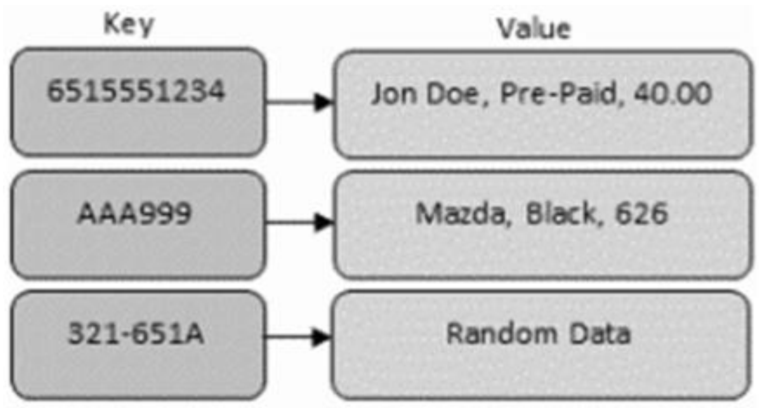
- Key-value stores are the simplest. Every item in the database is stored as an attribute name (or "key") together with its value.
  - Exp. Redis, Riak, Voldemort
- Wide-column stores store data together as columns instead of rows and are optimized for queries over large datasets.
  - Exp. Cassandra and HBase.
- Document databases pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.
  - Exp. MongoDB, DynamoDb, Cauchbase
- Graph databases are used to store information about networks, such as social connections.
  - Exp. Neo4j, OrientDB, Titan

Relational Data Model

Key-Value Store

A key-value Database

- A key-value database, or key-value store, is a data storage paradigm designed for storing, retrieving, and managing a dictionary or hash.
- Dictionaries contain a collection of objects, or records, which in turn have many different fields within them, each containing data.
- These records are stored and retrieved using a key that uniquely identifies the record, and is used to quickly find the data within the database.



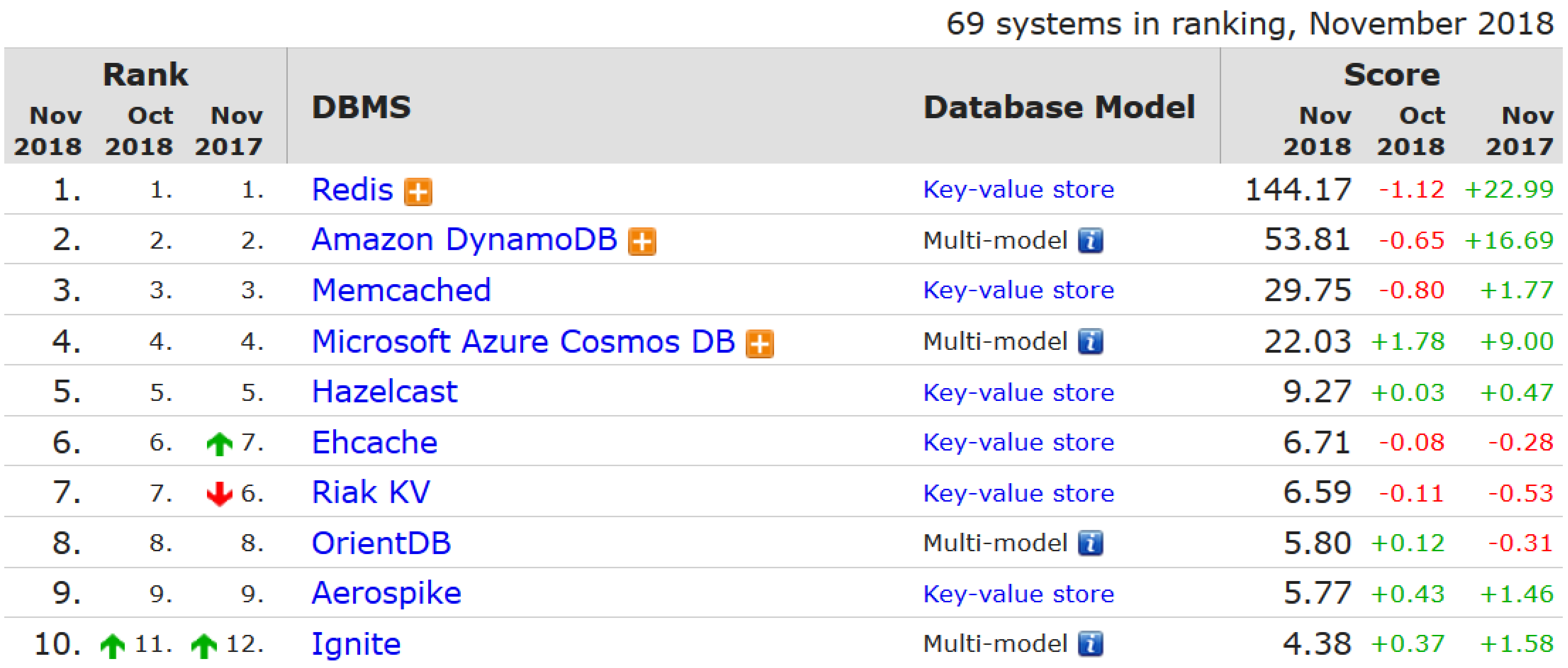
Use Cases

- **Session Store**: A session store collects information about a particular user or customer who visits a website. A session is created when someone first visits a site. The session has a key, which is passed to the user in the form of a cookie (the value). Each time the user makes a request to the site, the session information is retrieved and updated. In addition to session information, session stores can capture user ids, URL clicks and the contents of a shopping cart to be used for future interactions.
- **URL Shortening**: URL shortening is a common practice on the web. To shorten a URL a short URL is linked to a web page with a long URL using an HTTP Redirect on a domain name. For example, the URL "http://en.wikipedia.org/wiki/URL_shortening" can be shortened to "http://tinyurl.com/urlwiki".

Advantages

- **Flexible data modeling:** Because a key-value store does not enforce any structure on the data, it offers tremendous flexibility for modeling data to match the requirements of the application.
- **High performance:** Key-value architecture can be more performant than relational databases in many scenarios because there is no need to perform lock, join, union, or other operations when working with objects.
- **Scalability**: Most key-value databases make it easy to scale out on demand using commodity hardware.
- **High availability**: Key-value databases may make it easier and less complex to provide high availability than can be achieved with relational database
- **Operational simplicity**: Some key-value databases are specifically designed to simplify operations by ensuring that it is as easy as possible to add and remove capacity as needed and that any hardware or network failures within the environment do not create downtime.

Key-Value Stores Rankings



69 systems in ranking, November 2018

Rank			DBMS	Database Model	Score		
Nov 2018	Oct 2018	Nov 2017			Nov 2018	Oct 2018	Nov 2017
1.	1.	1.	Redis ➕	Key-value store	144.17	-1.12	+22.99
2.	2.	2.	Amazon DynamoDB ➕	Multi-model ℹ	53.81	-0.65	+16.69
3.	3.	3.	Memcached	Key-value store	29.75	-0.80	+1.77
4.	4.	4.	Microsoft Azure Cosmos DB ➕	Multi-model ℹ	22.03	+1.78	+9.00
5.	5.	5.	Hazelcast	Key-value store	9.27	+0.03	+0.47
6.	6.	↑ 7.	Ehcache	Key-value store	6.71	-0.08	-0.28
7.	7.	↓ 6.	Riak KV	Key-value store	6.59	-0.11	-0.53
8.	8.	8.	OrientDB	Multi-model ℹ	5.80	+0.12	-0.31
9.	9.	9.	Aerospike	Key-value store	5.77	+0.43	+1.46
10.	↑ 11.	↑ 12.	Ignite	Multi-model ℹ	4.38	+0.37	+1.58

Source: https://db-engines.com/en/ranking/key-value+st

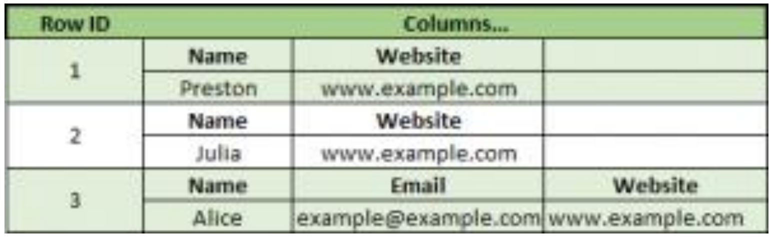
Wide Column Stores

Wide Column Stores

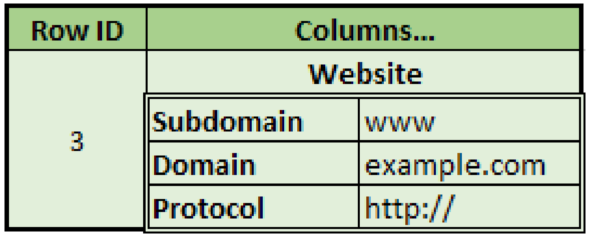
- Wide column stores have tables just like RDMBs, but these tables do not have relationship with other tables.
- Complex data is gouped into a single table.
- Tables have Column Families and Row Key
- Row Key → Primary Key in RDMBS
- Row Keys are also used to distribute records between servers

Wide Column Stores

- A *column* is the basic unit in a wide-column database and consists of a key and value pair.
- a *column family* = table
- each row in the column family has a unique identifier key, and then as many columns as it needs to hold the information.
- There is no requirement for every row to have the same columns,
- But the family itself must be declared before it’s used as column families typically represent how data is actually saved to disk.
- A *super column* contains many key-value pairs – as many as are needed to adequately hold the data.



Row ID	Columns...		
1	Name	Website	
	Preston	www.example.com	
2	Name	Website	
	Julia	www.example.com	
3	Name	Email	Website
	Alice	example@example.com	www.example.com



Row ID	Columns...	
3	Website	
	Subdomain	www
	Domain	example.com
	Protocol	http://

CQL

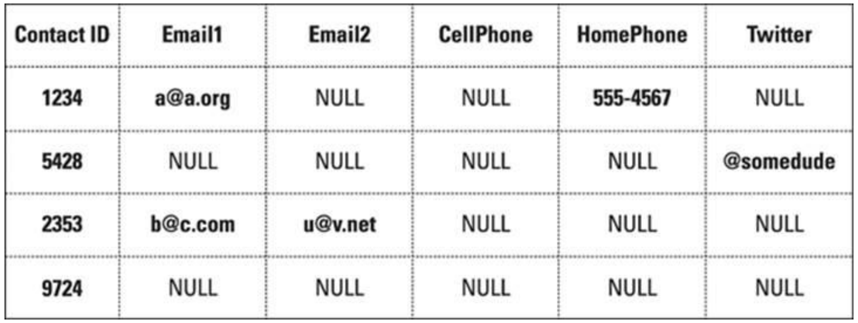
```
SELECT select_expression FROM
keyspace_name.table_name WHERE relation
AND relation ... ORDER BY ( clustering_column
ASC | DESC ...) LIMIT n ALLOW FILTERING
```

```
CREATE TABLE emp ( empID int, deptID
int, first_name varchar, last_name varchar,
PRIMARY KEY (empID, deptID))
```

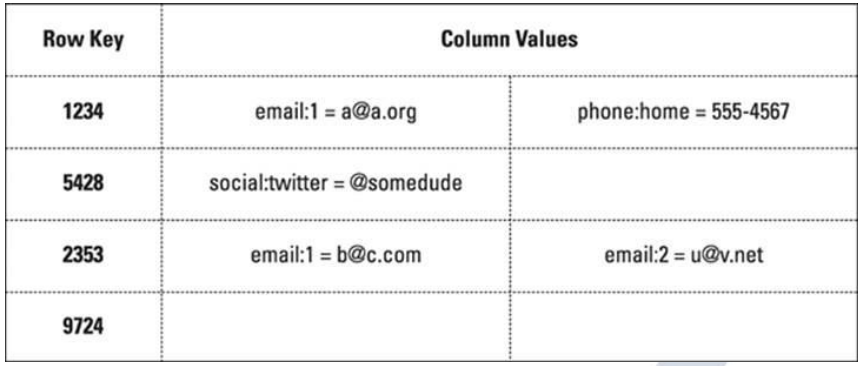
```
Select * FROM ruling_stewards WHERE king =
'none' AND reign_start >= 1500 AND
reign_start < 3000 LIMIT 10 ALLOW
FILTERING;
```

Wide Column Stores – Use cases

- Handling Sparse Data:



Contact ID	Email1	Email2	CellPhone	HomePhone	Twitter
1234	a@a.org	NULL	NULL	555-4567	NULL
5428	NULL	NULL	NULL	NULL	@somedude
2353	b@c.com	u@v.net	NULL	NULL	NULL
9724	NULL	NULL	NULL	NULL	NULL

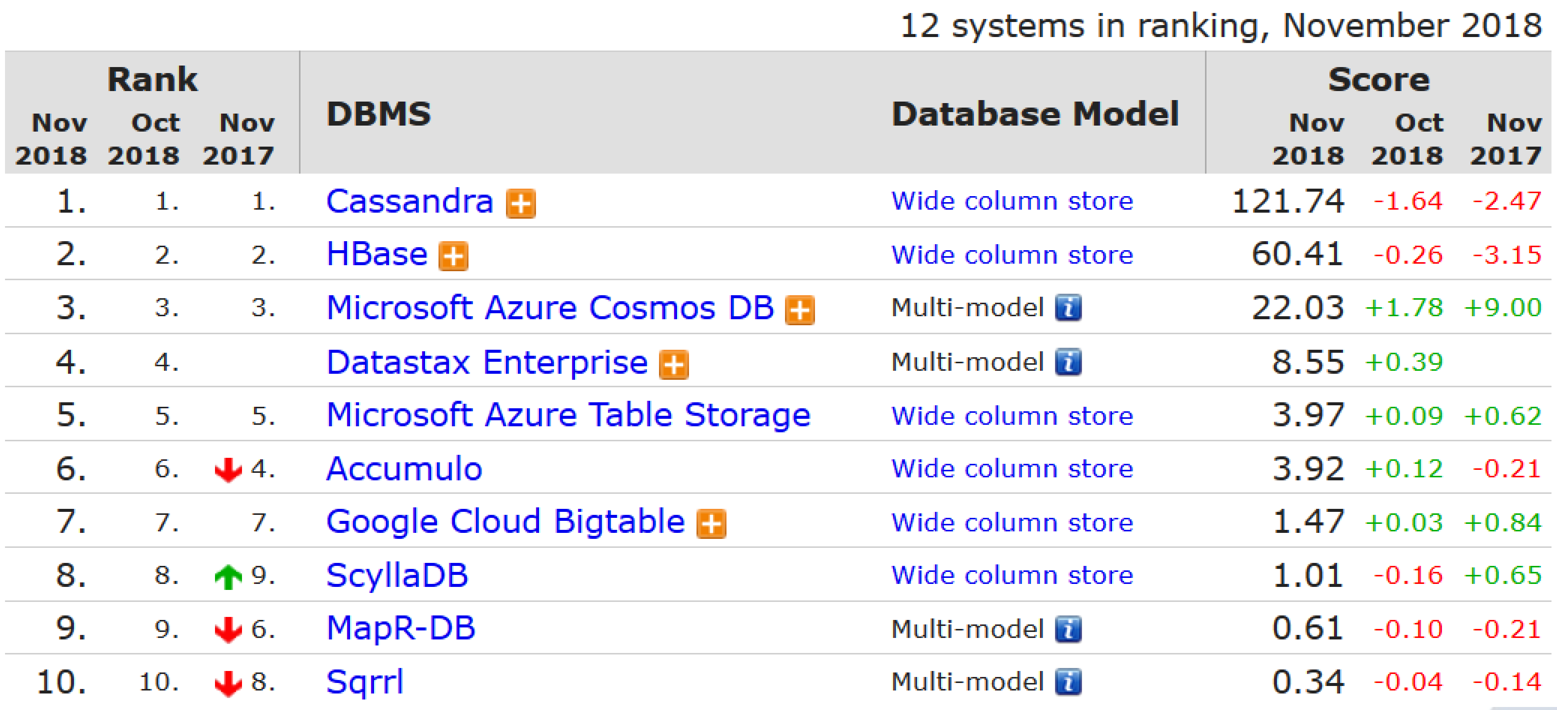


Row Key	Column Values	
1234	email:1 = a@a.org	phone:home = 555-4567
5428	social:twitter = @somedude	
2353	email:1 = b@c.com	email:2 = u@v.net
9724		

Wide Column Stores - Advantages

- Highly scalable – well-suited for distribution across multiple data stores and computing nodes.
- Sorting and some data manipulation – directly from the database rather than relying solely on the application.
- Indexing – some databases index different levels of columns to help with data processing.
- Increased granularity – being able to update an individual column rather than having to replace all of the data when something changes.

Wide-Column Stores - Rankings



12 systems in ranking, November 2018

Rank			DBMS	Database Model	Score		
Nov 2018	Oct 2018	Nov 2017			Nov 2018	Oct 2018	Nov 2017
1.	1.	1.	Cassandra	Wide column store	121.74	-1.64	-2.47
2.	2.	2.	HBase	Wide column store	60.41	-0.26	-3.15
3.	3.	3.	Microsoft Azure Cosmos DB	Multi-model	22.03	+1.78	+9.00
4.	4.		Datastax Enterprise	Multi-model	8.55	+0.39	
5.	5.	5.	Microsoft Azure Table Storage	Wide column store	3.97	+0.09	+0.62
6.	6.	⬇ 4.	Accumulo	Wide column store	3.92	+0.12	-0.21
7.	7.	7.	Google Cloud Bigtable	Wide column store	1.47	+0.03	+0.84
8.	8.	⬆ 9.	ScyllaDB	Wide column store	1.01	-0.16	+0.65
9.	9.	⬇ 6.	MapR-DB	Multi-model	0.61	-0.10	-0.21
10.	10.	⬇ 8.	Sqrrl	Multi-model	0.34	-0.04	-0.14

Document Based Store

Document Databases

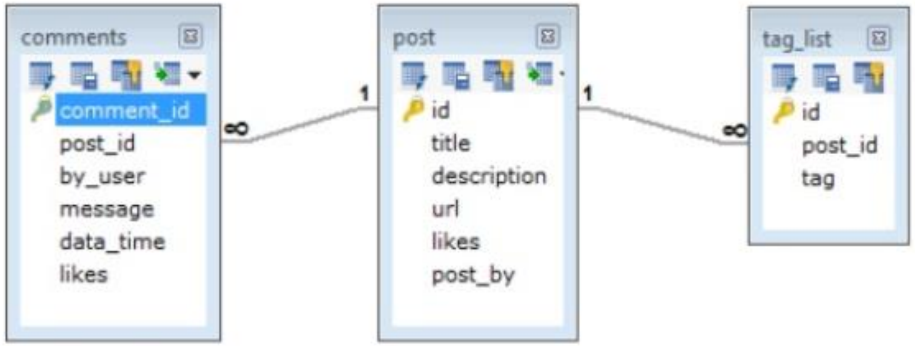
- Document-oriented databases similar to key-value store
- The difference lies in the way the data is processed; in a key-value store, the data is considered to be inherently opaque to the database, whereas a document-oriented system relies on internal structure in the document in order to extract metadata that the database engine uses for further optimization.
- Document: hierarchical structure of data that can contain substructures.
- Documents canconsist of
  - binary data
  - plain text
  - Javascript Object Notation (JSON)
  - Extensible Markup Language (XML)
  - → for well-structured documetns XML Shema Definition can be used.

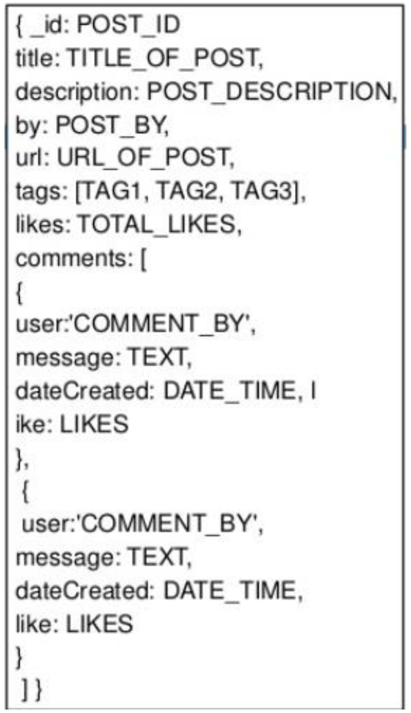
```xml
<SampleXML>
  <Colors>
    <Color1>White</Color1>
    <Color2>Blue</Color2>
    <Color3>Black</Color3>
    <Color4 Special="Light">Green</Color4>
    <Color5>Red</Color5>
  </Colors>
  <Fruits>
    <Fruits1>Apple</Fruits1>
    <Fruits2>Pineapple</Fruits2>
    <Fruits3>Grapes</Fruits3>
    <Fruits4>Melon</Fruits4>
  </Fruits>
</SampleXML>
```

```json
{
    "Rail Booking": {
        "reservation": {
            "ref_no": 1234567,
            "time_stamp": "2016-06-24T14:26:59.125",
            "confirmed": true
        },
        "train": {
            "date": "07/04/2016",
            "time": "09:30",
            "from": "New York",
            "to": "Chicago",
            "seat": "57B"
        },
        "passenger": {
            "name": "John Smith"
        },
        "price": 1234.25,
        "comments": ["Lunch & dinner incl.", "\"Have a nice day!\""]
    }
}
```

Use Cases

- Blog

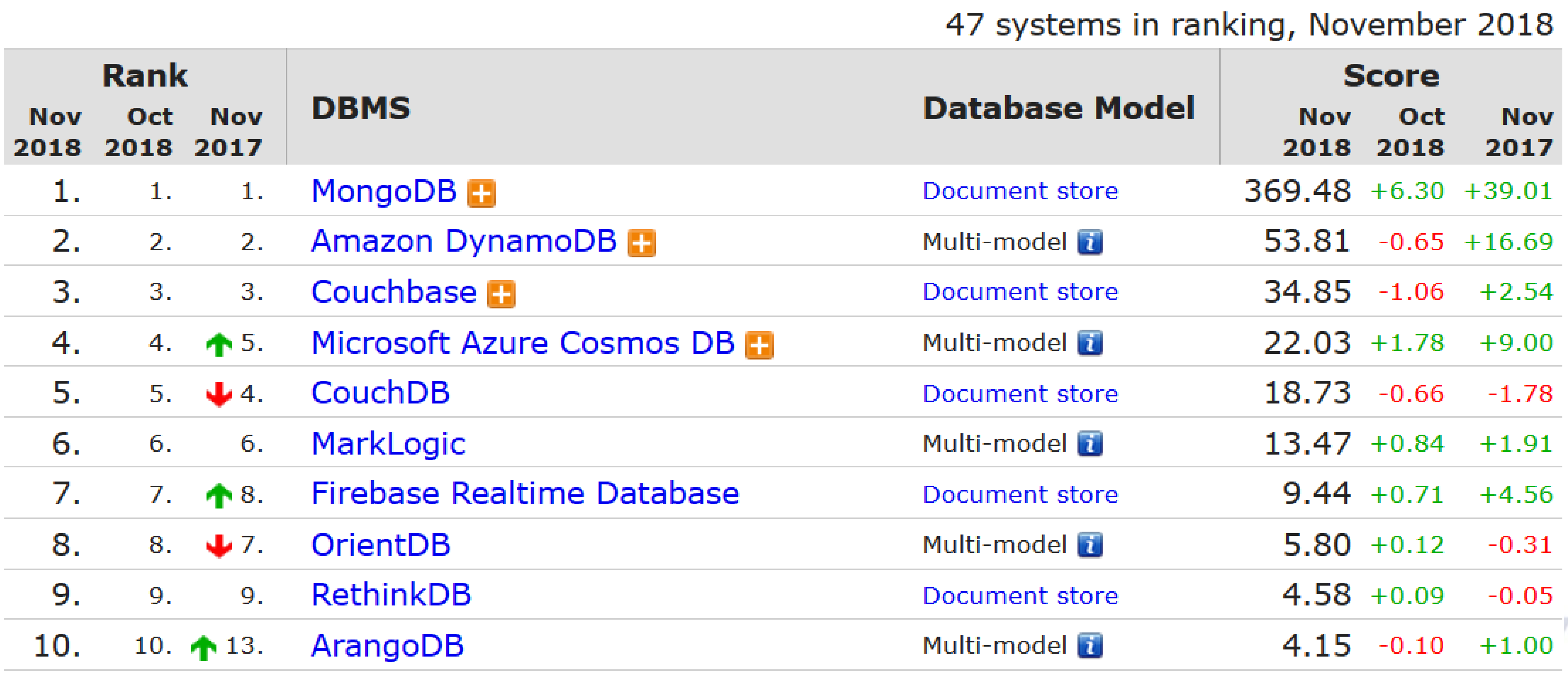




```
{ _id: POST_ID
title: TITLE_OF_POST,
description: POST_DESCRIPTION,
by: POST_BY,
url: URL_OF_POST,
tags: [TAG1, TAG2, TAG3],
likes: TOTAL_LIKES,
comments: [
{
user:'COMMENT_BY',
message: TEXT,
dateCreated: DATE_TIME, l
ike: LIKES
},
 {
 user:'COMMENT_BY',
message: TEXT,
dateCreated: DATE_TIME,
like: LIKES
}
 ] }
```

RDBMS
Document Store

Document Databases - Rankings



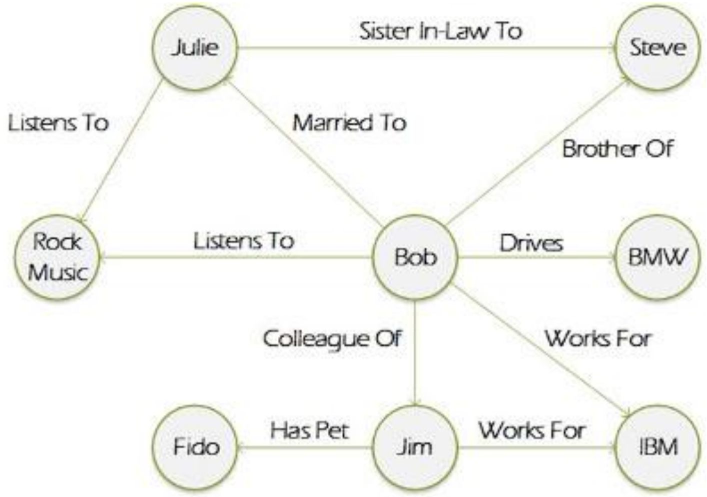
47 systems in ranking, November 2018

Rank			DBMS	Database Model	Score		
Nov 2018	Oct 2018	Nov 2017			Nov 2018	Oct 2018	Nov 2017
1.	1.	1.	MongoDB	Document store	369.48	+6.30	+39.01
2.	2.	2.	Amazon DynamoDB	Multi-model	53.81	-0.65	+16.69
3.	3.	3.	Couchbase	Document store	34.85	-1.06	+2.54
4.	4.	↑ 5.	Microsoft Azure Cosmos DB	Multi-model	22.03	+1.78	+9.00
5.	5.	↓ 4.	CouchDB	Document store	18.73	-0.66	-1.78
6.	6.	6.	MarkLogic	Multi-model	13.47	+0.84	+1.91
7.	7.	↑ 8.	Firebase Realtime Database	Document store	9.44	+0.71	+4.56
8.	8.	↓ 7.	OrientDB	Multi-model	5.80	+0.12	-0.31
9.	9.	9.	RethinkDB	Document store	4.58	+0.09	-0.05
10.	10.	↑ 13.	ArangoDB	Multi-model	4.15	-0.10	+1.00

Graph Database

Graph database

- Graph DB is a database that uses graph structures for semantic queries with nodes, edges and properties to represent and store data.
- A key concept of the system is the graph, which directly relates data items in the store.
- The relationships allow data in the store to be linked together directly, and in many cases retrieved with one operation.



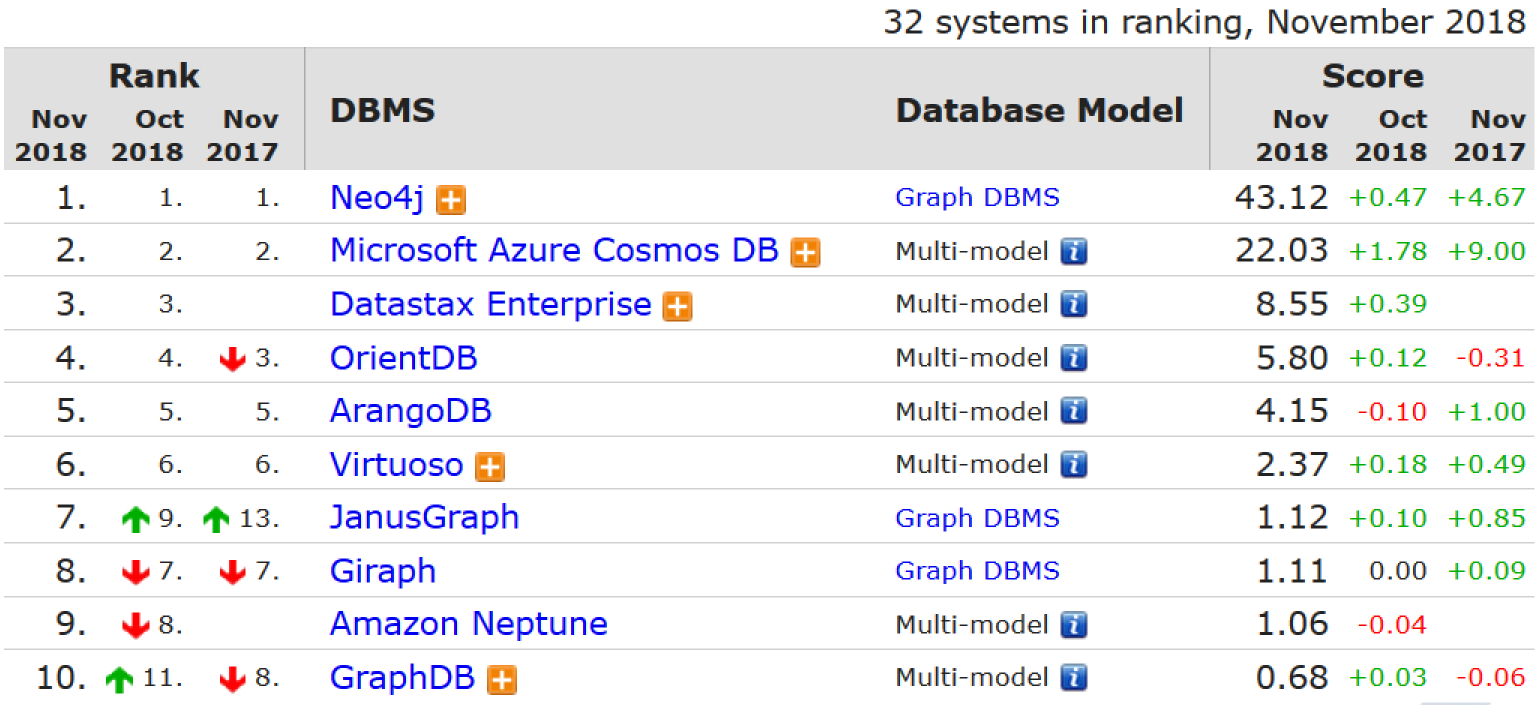
Graph DB Use Cases

- Recommendation Engines: Neo4j claims to count seven of the world's top ten retailers as customers. Rather than taking a tabular view of transactions, retailers can pull together product, customer, inventory, supplier and social sentiment data into a graph database to spot patterns and make smarter recommendations
- Fraud Detection: Graph databases are uniquely positioned to spot the connections between large data sets and identify patterns, a useful trait when it comes to spotting complex, modern fraud techniques. Neo4j says that it already counts a number of major banks using its graph services to aid fraud detection.

Advantages

- Connectivity:Unlike relational databases, a graph database doesn’t make use of foreign keys and JOIN operations. Instead, all relationships are natively stored within vertices resulting in deep traversal capabilities, increased flexibility and agility.
- Scalability: graph databases are equipped to handle rapidly scaling data.
- Performance: Native graph databases that apply Index free adjacency show reduced latency on CRUD operations.

Graph DB -Rankings



32 systems in ranking, November 2018

Rank			DBMS	Database Model	Score		
Nov 2018	Oct 2018	Nov 2017			Nov 2018	Oct 2018	Nov 2017
1.	1.	1.	Neo4j	Graph DBMS	43.12	+0.47	+4.67
2.	2.	2.	Microsoft Azure Cosmos DB	Multi-model	22.03	+1.78	+9.00
3.	3.		Datastax Enterprise	Multi-model	8.55	+0.39	
4.	4.	↓ 3.	OrientDB	Multi-model	5.80	+0.12	-0.31
5.	5.	5.	ArangoDB	Multi-model	4.15	-0.10	+1.00
6.	6.	6.	Virtuoso	Multi-model	2.37	+0.18	+0.49
7.	↑ 9.	↑ 13.	JanusGraph	Graph DBMS	1.12	+0.10	+0.85
8.	↓ 7.	↓ 7.	Giraph	Graph DBMS	1.11	0.00	+0.09
9.	↓ 8.		Amazon Neptune	Multi-model	1.06	-0.04	
10.	↑ 11.	↓ 8.	GraphDB	Multi-model	0.68	+0.03	-0.06

Introduction to MongoDB

MongoDB Overview

– MongoDB is:
  - An open source and document-oriented database.
  - Data is stored in JSON-like documents.
  - Designed with both scalability and developer agility.
  - Dynamic schemas.

MongoDB Overview

- Who uses MongoDB
  - Weather Channel
  - ADP
  - Expedia
  - SourceForge
  - Bosch

MongoDB Overview

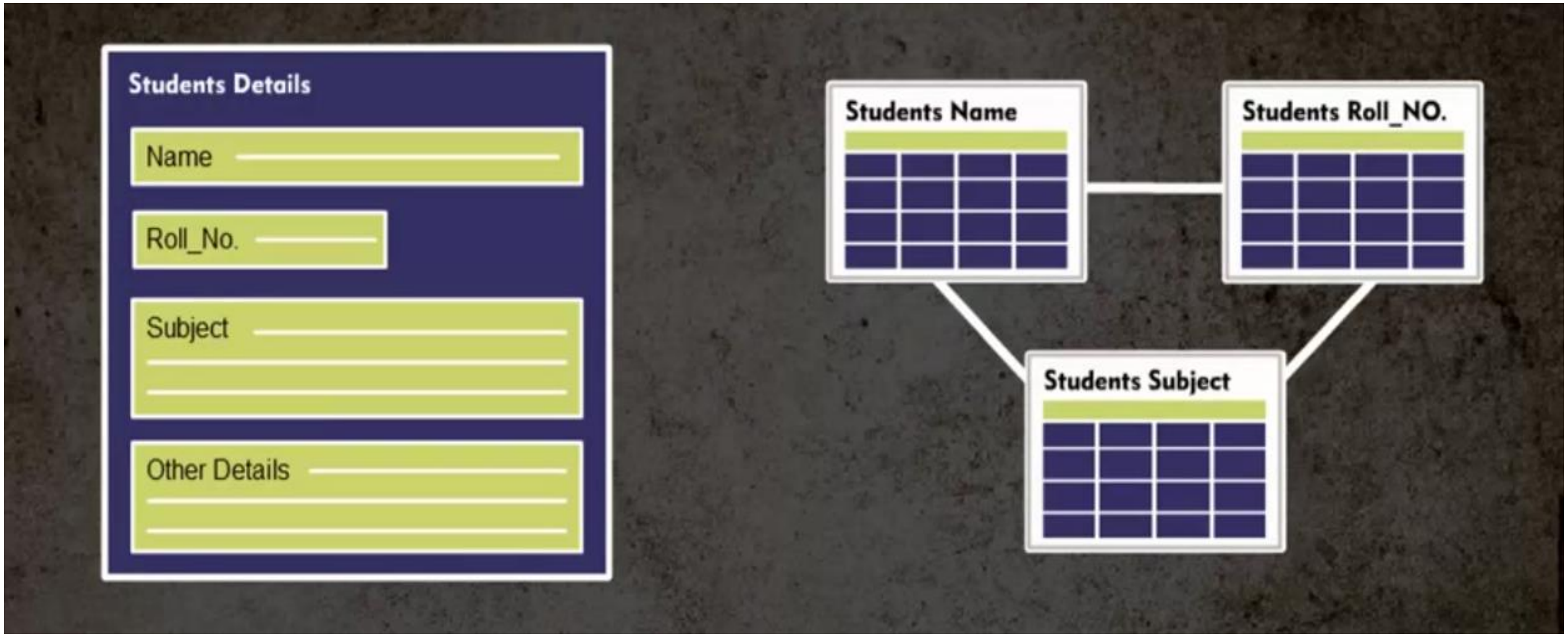
- Features of MongoDB
  - document-oriented database
  - not strongly typed (structure not enforced)
  - server-side JavaScript execution
  - interoperates nicely with applications (JSON)
  - offers load balancing (multiple servers)

MongoDB Overview

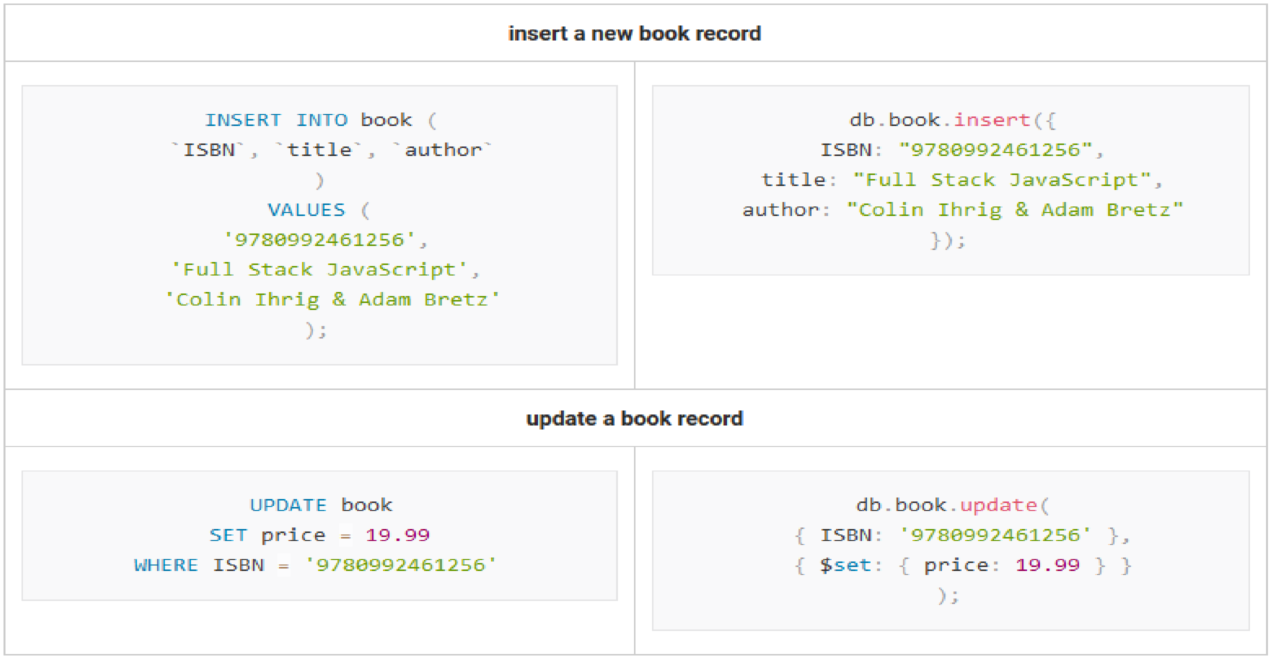
- Reasons for using MongoDB (or NoSQL)
  - High amount of data having low value
    - Social networking comments
    - Log data
  - Document Storage
  - Cached Data
  - Unstable Schema

RDBMS		MongoDB
Database	⇨	Database
Table, View	⇨	Collection
Row	⇨	Document (JSON, BSON)
Column	⇨	Field
Index	⇨	Index
Join	⇨	Embedded Document
Foreign Key	⇨	Reference
Partition	⇨	Shard

MONGODB vs RDBMS



SQL VS NOSQL QUERIES



insert a new book record

```sql
INSERT INTO book (
`ISBN`, `title`, `author`
)
VALUES (
'9780992461256',
'Full Stack JavaScript',
'Colin Ihrig & Adam Bretz'
);
```

```
db.book.insert({
ISBN: "9780992461256",
title: "Full Stack JavaScript",
author: "Colin Ihrig & Adam Bretz"
});
```

update a book record

```sql
UPDATE book
SET price = 19.99
WHERE ISBN = '9780992461256'
```

```
db.book.update(
{ ISBN: '9780992461256' },
{ $set: { price: 19.99 } }
);
```

A brief intro to JSON

- What is JSON?
  - JSON stands for **J**ava**S**cript **O**bject **N**otation
  - JSON is a lightweight data-interchange format
  - JSON is "self-describing" and easy to understand
  - JSON is language independent *****
- Why JSON
  - Since the JSON format is text only, it can easily be sent to and from a server, and used as a data format by any programming language.

- JSON Syntax Rules
  - Data is in name/value pairs
  - Data is separated by commas
  - Curly braces hold objects
  - Square brackets hold arrays
- JSON data is written as name/value pairs.
  - "name":"Ahmet"
- In JSON, *values* must be one of the following data types:
  - a string
  - a number
  - an object (JSON object)
  - an array
  - a boolean
  - null

JSON vs XML

JSON

- {"employees":[
    { "firstName":"John", "lastName":"Doe" }
  ,
    { "firstName":"Anna", "lastName":"Smith
  " },
    { "firstName":"Peter", "lastName":"Jone
  s" }
  ]}

XML

- <employees>
    <employee>
      <firstName>John</firstName>
  <lastName>Doe</lastName>
    </employee>
    <employee>
      <firstName>Anna</firstName>
  <lastName>Smith</lastName>
    </employee>
    <employee>
      <firstName>Peter</firstName>
  <lastName>Jones</lastName>
    </employee>
  </employees>

JSON vs XML

Similarities

- Both JSON and XML are "self describing" (human readable)
- Both JSON and XML are hierarchical (values within values)
- Both JSON and XML can be parsed and used by lots of programming languages
- Both JSON and XML can be fetched with an XMLHttpRequest

Differences

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays

JSON Data Types

- JSON Strings

  { "name":"John" }
- JSON Numbers

  { "age":30 }
- JSON Objects

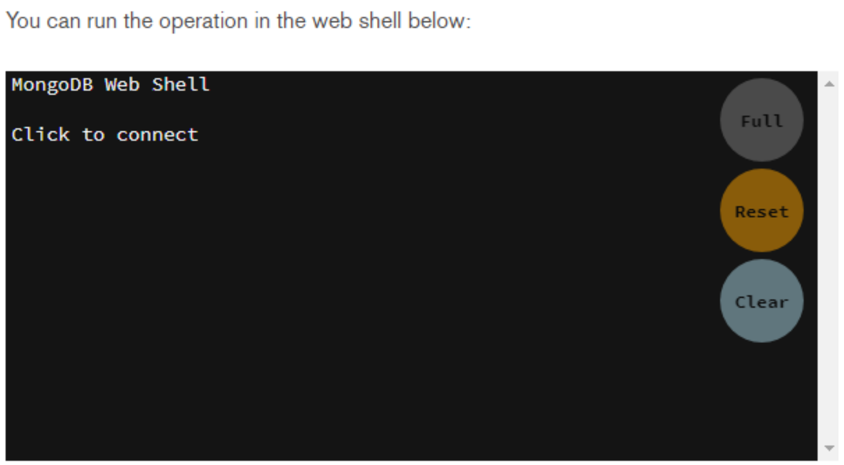
  { "employee":{ "name":"John", "age":30, "city":"New York"}   }
- JSON Arrays

  {   "employees":["John", "Anna", "Peter"] }
- JSON Booleans

  { "sale":true }
- JSON null

  { "middlename":null }

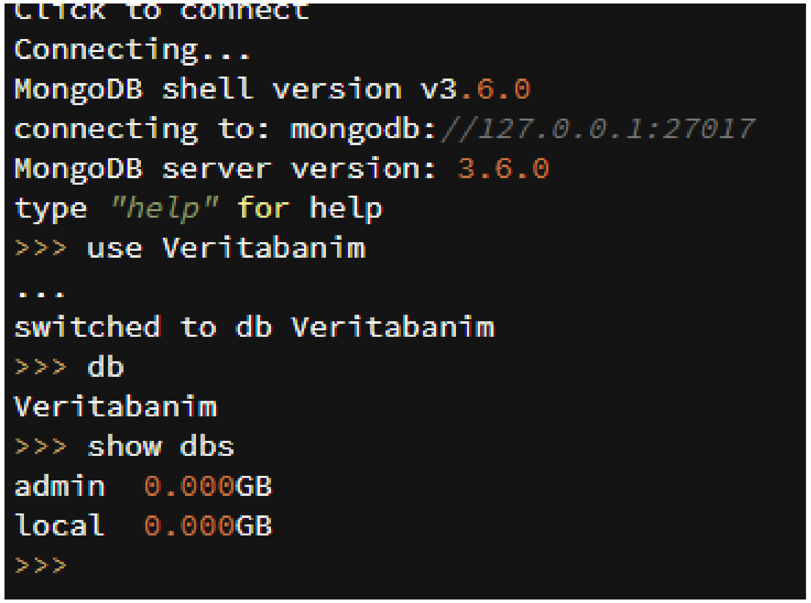
https://mws.mongodb.com/?version=3.6


You can run the operation in the web shell below:

```
MongoDB Web Shell

Click to connect
```

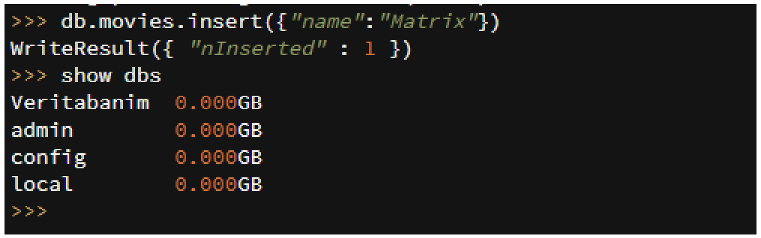
- MongoDB **use DATABASE_NAME** is used to create database.
  - use Veritabanim
- To check your currently selected database, use the command **db**
  - db
- If you want to check your databases
list, use the command **show dbs**.
  - show dbs



```
Connecting...
MongoDB shell version v3.6.0
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.0
type "help" for help
>>> use Veritabanim
...
switched to db Veritabanim
>>> db
Veritabanim
>>> show dbs
admin   0.000GB
local   0.000GB
>>>
```

Your created database (mydb) is not present in list. To display database, you need to insert at least one document into it.

- – db.movie.insert({"name":"Matrix"})
- – show dbs



```
>>> db.movies.insert({"name":"Matrix"})
WriteResult({ "nInserted" : 1 })
>>> show dbs
Veritabanim  0.000GB
admin        0.000GB
config       0.000GB
local        0.000GB
>>>
```

- **db.dropDatabase()** command is used to drop a existing database.
  - – db.dropDatabase()
  - – show dbs

CreateCollection – Show Collection

- MongoDB **db.createCollection(name, options)** is used to create collection.
  - db.createCollection("mycollection")
- You can check the created collection by using the command **show collections**.
  - show collections

Insert - Drop

- In MongoDB, you don't need to create collection. MongoDB creates collection automatically, when you insert some document.
  - db.newCollection.insert({"name" : "Istanbul Teknik"})
  - show collections
- **db.collection.drop()** is used to drop a collection from the database.
  - db.newCollection.drop()
  - show collections

MongoDB data types

- **String** − This is the most commonly used datatype to store the data. String in MongoDB must be UTF-8 valid.
- **Integer** − This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
- **Boolean** − This type is used to store a boolean (true/ false) value.
- **Double** − This type is used to store floating point values.
- **Min/ Max keys** − This type is used to compare a value against the lowest and highest BSON elements.
- **Arrays** − This type is used to store arrays or list or multiple values into one key.
- **Timestamp** − ctimestamp. This can be handy for recording when a document has been modified or added.
- **Object** − This datatype is used for embedded documents.
- **Null** − This type is used to store a Null value.
- **Symbol** − This datatype is used identically to a string; however, it's generally reserved for languages that use a specific symbol type.
- **Date** − This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
- **Object ID** − This datatype is used to store the document's ID.
- **Binary data** − This datatype is used to store binary data.
- **Code** − This datatype is used to store JavaScript code into the document.
- **Regular expression** − This datatype is used to store regular expression.

insertOne() – insertMany()

- To insert data into a collection, you need to use **insert()**or **insertMany()** method.

```
db.inventory.insertOne
(  { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } } )
-------------

db.inventory.insertMany([
   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
   { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
]);
```

Query

- SELECT * FROM inventory WHERE status = "D"
  - db.inventory.find({ status: "D" })
- SELECT * FROM inventory WHERE status in ("A", "D")
  - db.inventory.find({ status: { $in: ["A", "D"] } })
- SELECT * FROM inventory WHERE status = "A" AND qty < 30
  - db.inventory.find({ status: "A", qty: { $lt: 30 } })
- SELECT * FROM inventory WHERE status = "A" OR qty < 30
  - db.inventory.find({ $or: [{ status: "A" }, { qty: { $lt: 30 } }] })