cct | College Dublin
Computing • IT • Business

Big Data Storage and Processing

MSc in Data Analytics

CCT College Dublin

YCSB and Mongo DB

Week 10

Lecturer: Dr. Muhammad Iqbal*

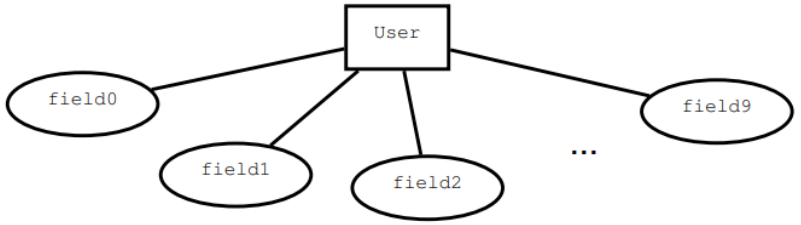
Email: miqbal@cct.ie

Agenda

cct College Dublin
Computing • IT • Business

- Introduction to YCSB and Motivation
- YCSB Workloads and output Report
- YCSB Client Architecture
- MongoDB vs Relational databases
- Scaling and Characteristics of MongoDB
- MongoDB Working
- Key Components of MongoDB Architecture
- MongoDB Documents and Collections
- MongoDB Document Structure
- Query Format in MongoDB

Introduction to YCSB



Conceptional data model of YCSB's database

- Cooper et al. suggested the Yahoo Cloud Serving Benchmark (YCSB) in 2010 as a new benchmark suite to address freshly emerging databases.
- **YCSB** is an open-source specification and software package for benchmarking SQL and NoSQL database management solutions' relative performance.
- While it was possible to compare these new databases qualitatively with each other or existing RDBMS, it was hard to do this comparison quantitatively.
- The benchmark established a simple data model that functions as a **key-value store**. This model has one entity, **User**, which has **ten variables** by default.
- **YCSB** also provided workloads to determine how the benchmark should be run.
- The amount of fields per record (fieldcount) or the proportion of **read**, **write**, and **update operations** to complete are all properties of a workload (**readproportion, insertproportion and updateproportion** respectively).

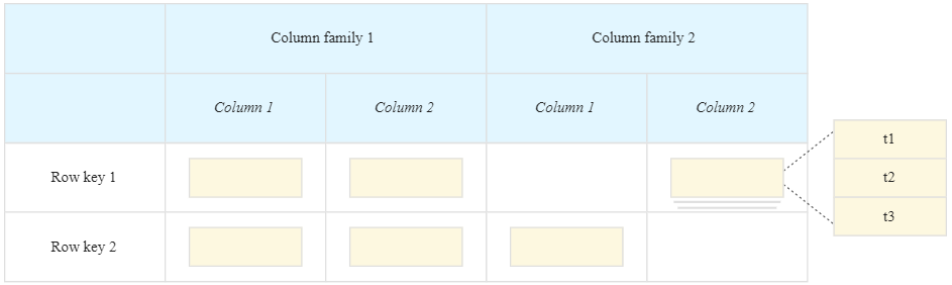
Introduction to YCSB

cct | College Dublin
Computing • IT • Business

- Most NoSQL databases make trade-offs like optimising for reads vs. writes, latency vs. durability, and synchronous vs. asynchronous replication, among others.
- The workloads they created for **YCSB** were created to "directly explore these trade-offs."
- These ***trade-offs*** are shown in Table below for a small number of databases.
- ***BigTable (Google)*** is designed for quick writes, long-term durability, and synchronous replication. It has a column-oriented data model.

System	Read/Write optimized	Latency/durability	Sync/async replication	Row/column
PNUTS	Read	Durability	Async	Row
BigTable	Write	Durability	Sync	Column
HBase	Write	Latency	Async	Column
Cassandra	Write	Tunable	Tunable	Column
Sharded MySQL	Read	Tunable	Async	Row

YCSB: Design decisions of various systems



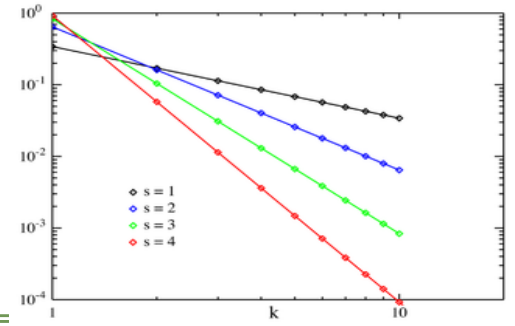
https://cloud.google.com/bigtable/docs/overview

YCSB Motivation

cct | College Dublin
Computing • IT • Business

- As there are several new systems for data storage and management
  - **Open source and Free systems:** Cassandra, Hbase, MongoDB, MySQL, …
  - **Cloud services:** Google cloud, Microsoft Azure, Amazon web service (S3, SimpleDB) …
- It is always challening to compare the performance of different databases
  - **Data Models**
    - Cassandra and HBase used BigTable model
    - MongoDB and CouchDB used Document model
  - **Design Options**
    - Read or write optimized
    - Synchronous or Asynchronous replication
    - Latency or Durability: synchronous writes to disk at write time or not
    - Data partitioning: row-based or column-based storage
  - **Workloads Evaluation**

YCSB Workloads



- **YCSB** has five predefined workloads (**A** to **E**).
- Table shows these workloads and their most important settings.
- **Workload A** for instance performs **50% read** and **50% update** operations.
- The key chosen to identify a record comes from a **zipfian distribution**, meaning that some keys are very likely to occur while others are not (**YCSB** supports the following distributions: uniform, zipfian, latest, multinomial).
- An application example that matches this workload could be a session store recording recent actions in a user session.

Workload	Operations	Record selection	Application example
A—Update heavy	Read: 50% Update: 50%	Zipfian	Session store recording recent actions in a user session
B—Read heavy	Read: 95% Update: 5%	Zipfian	Photo tagging; add a tag is an update, but most operations are to read tags
C—Read only	Read: 100%	Zipfian	User profile cache, where profiles are constructed elsewhere (e.g., Hadoop)
D—Read latest	Read: 95% Update: 5%	Latest	User status updates; people want to read the latest statuses
E—Short ranges	Read: 95% Update: 5%	Zipfian / Uniform	Threaded conversations, where each scan is for the posts in a given thread (assumed to be clustered by thread id)

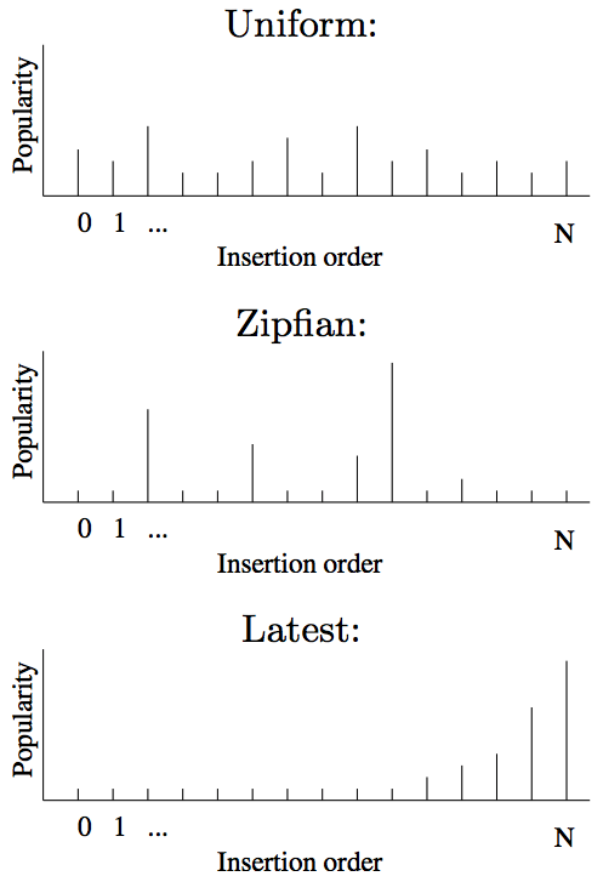
YCSB: Workloads in the core package

Zipf's Law is a statistical distribution in certain data sets, such as words in a linguistic corpus, in which the frequencies of certain words are inversely proportional to their ranks.

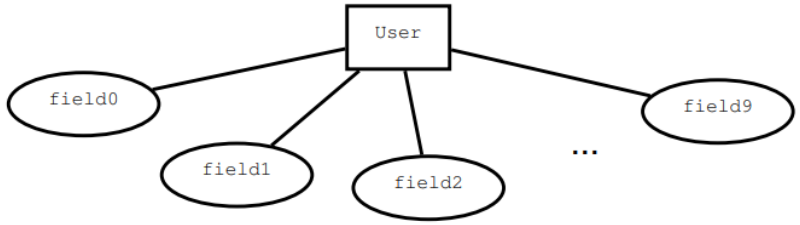
YCSB Workloads

College Dublin
Computing • IT • Business

- One **workload** is a combination of fundamental **operations** with choices of
  - Which operations to perform
  - Which record to read or write
  - How many records to scan
- Decisions are governed by random distributions
  - Uniform
  - Zipfian
  - Latest
  - Multinomial



YCSB Workloads



Conceptional data model of YCSB's database

- When a given workload is run against various databases, the **performance** and **scalability** of such systems can be compared.
- **YCSB** is an open-source project that can be extended.
- **Multithreading** is possible with the so-called **client**.
- The Stats module records performance criteria during a run and presents them later.
- Third-party components such as the **Workload Executor** and the **DB Interface Layer** can be easily extended.

Workload	Operations	Record selection	Application example
A—Update heavy	Read: 50% Update: 50%	Zipfian	Session store recording recent actions in a user session
B—Read heavy	Read: 95% Update: 5%	Zipfian	Photo tagging; add a tag is an update, but most operations are to read tags
C—Read only	Read: 100%	Zipfian	User profile cache, where profiles are constructed elsewhere (e.g., Hadoop)
D—Read latest	Read: 95% Update: 5%	Latest	User status updates; people want to read the latest statuses
E—Short ranges	Read: 95% Update: 5%	Zipfian / Uniform	Threaded conversations, where each scan is for the posts in a given thread (assumed to be clustered by thread id)

YCSB: Workloads in the core package

YCSB Statistics Report

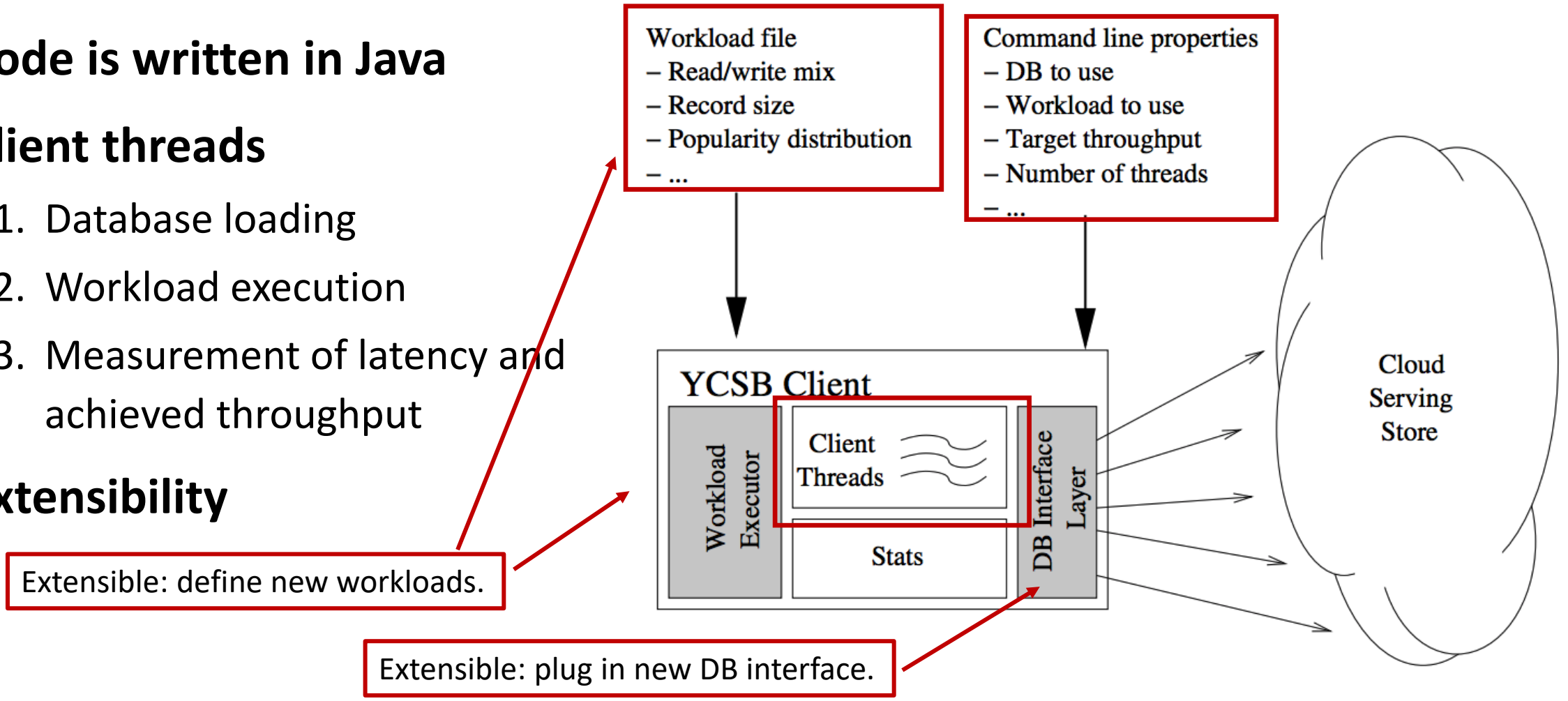
cct | College Dublin
Computing • IT • Business

- **YCSB** has a workload characterization file that you can use to make results repeatable and to Target against different systems for comparisons.
- **Type of Statistics**
  - Percentile latency, e.g., 95th percentile and 99th percentile
  - Histogram buckets
  - Time series

```
[OVERALL], RunTime(ms), 2685
[OVERALL], Throughput(ops/sec), 372.43947858472995
[TOTAL_GCS_Copy], Count, 3
[TOTAL_GC_TIME_Copy], Time(ms), 16
[TOTAL_GC_TIME_%_Copy], Time(%), 0.5959031657355679
[TOTAL_GCS_MarkSweepCompact], Count, 0
[TOTAL_GC_TIME_MarkSweepCompact], Time(ms), 0
[TOTAL_GC_TIME_%_MarkSweepCompact], Time(%), 0.0
[TOTAL_GCs], Count, 3
[TOTAL_GC_TIME], Time(ms), 16
[TOTAL_GC_TIME_%], Time(%), 0.5959031657355679
[CLEANUP], Operations, 1
[CLEANUP], AverageLatency(us), 852.0
[CLEANUP], MinLatency(us), 852
[CLEANUP], MaxLatency(us), 852
[CLEANUP], 95thPercentileLatency(us), 852
[CLEANUP], 99thPercentileLatency(us), 852
[INSERT], Operations, 1000
[INSERT], AverageLatency(us), 1878.889
[INSERT], MinLatency(us), 779
[INSERT], MaxLatency(us), 69823
[INSERT], 95thPercentileLatency(us), 3277
[INSERT], 99thPercentileLatency(us), 6059
[INSERT], Return=OK, 1000
```

YCSB Client Architecture

- **Code is written in Java**
- **Client threads**
  1. Database loading
  2. Workload execution
  3. Measurement of latency and achieved throughput
- **Extensibility**

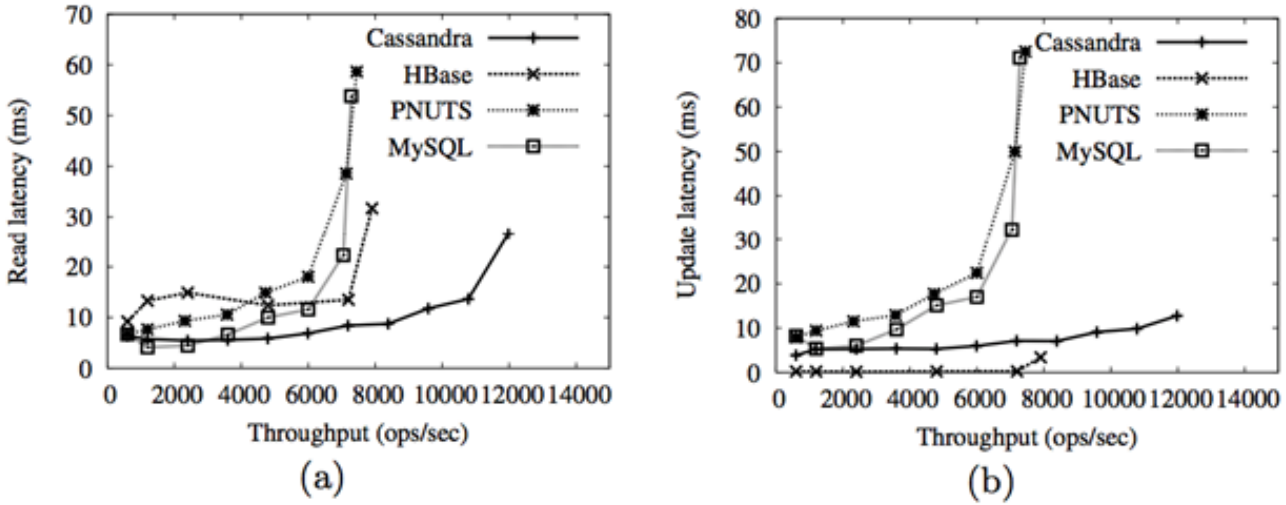


YCSB client architecture

Workload A: update heavy

College Dublin
Computing • IT • Business

- 50% reads and 50% updates.

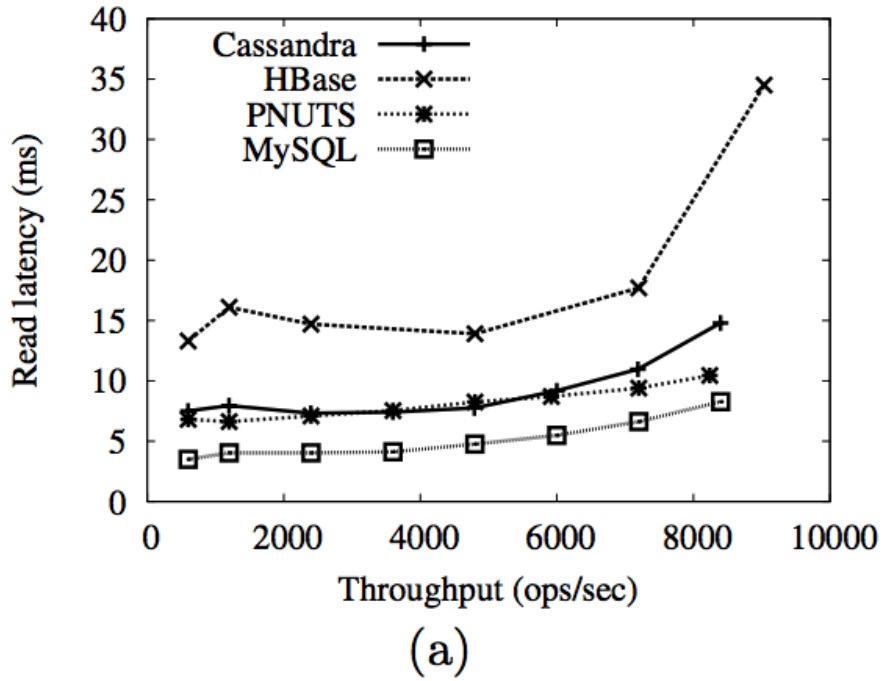



```
hduser@muhammad-VM: ~/ycsb-0.17.0/workloads
 GNU nano 4.8                                  workload_template
# The name of the workload class to use
workload=site.ycsb.workloads.CoreWorkload

# There is no default setting for recordcount but it is
# required to be set.
# The number of records in the table to be inserted in
# the load phase or the number of records already in the
# table before the run phase.
recordcount=1000000

# There is no default setting for operationcount but it is
# required to be set.
# The number of operations to use during the run phase.
operationcount=3000000

# The number of insertions to do, if different from recordcount.
# Used with insertstart to grow an existing table.
#insertcount=

# The offset of the first insertion
insertstart=0

# The number of fields in a record
fieldcount=10

# The size of each field (in bytes)
fieldlength=100


^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo
```

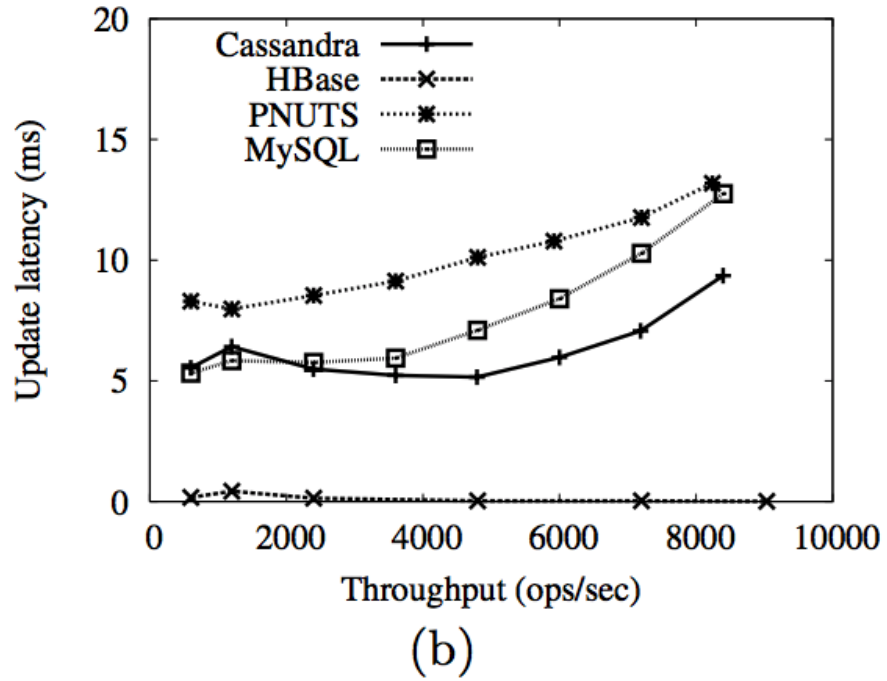
```
hduser@muhammad-VM:~/ycsb-0.17.0/workloads$ ls
tsworkloada          workloada  workloadc  workloade  workload_template
tsworkload_template  workloadb  workloadd  workloadf
hduser@muhammad-VM:~/ycsb-0.17.0/workloads$
```

Workload B: Read heavy

- 95% reads and 5% updates



(a)



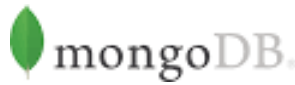
(b)

```
hduser@muhammad-VM:~/ycsb-0.17.0/workloads$ ls
tsworkloada          workloada  workloadc  workloade  workload_template
tsworkload_template  workloadb  workloadd  workloadf
hduser@muhammad-VM:~/ycsb-0.17.0/workloads$
```

Introduction to MongoDB

cct | College Dublin
Computing • IT • Business

- Imagine a world where operating a database is so simple that scalability and performance are automatic, and there is no need for time-consuming configuration or setup.



- MongoDB (derived from the word humongous) is a relatively new breed of database that has no concept of tables, schemas, SQL, or rows.
- Transactions, **ACID** compliance, joins, foreign keys, and many more things that create difficulties in the wee hours of the morning are absent.
- **MongoDB** isn't like any other database you have used before, especially if you have worked with a relational database management system (**RDBMS**).
- Indeed, the lack of so-called "normal" features may have you scratching your head in disbelief.

MongoDB

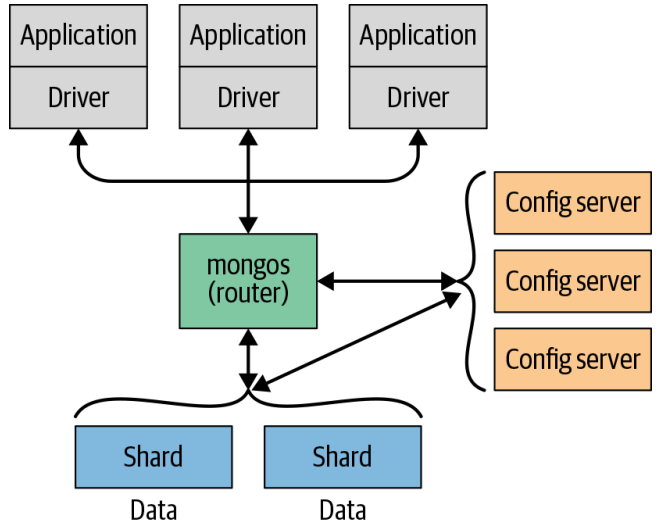
cct College Dublin
Computing • IT • Business

- **MongoDB** is a powerful, flexible, and scalable general-purpose database.
- **MongoDB** has the ability to scale out with features such as secondary indexes, range queries, sorting, aggregations, and geospatial indexes.
- **User-Friendliness**

- **MongoDB** is a **document-oriented** database, not a relational one. The primary reason for moving away from the relational model is to make scaling out easier, but there are some other advantages as well.
- The concept of a "row" is replaced by a more flexible model called "document" in a **document-oriented database, like MongoDB**.
- **MongoDB** allows complicated hierarchical relationships to be represented with just a single record by allowing embedded documents and arrays. This is how current object-oriented language developers think about their data.

Scaling of MongoDB

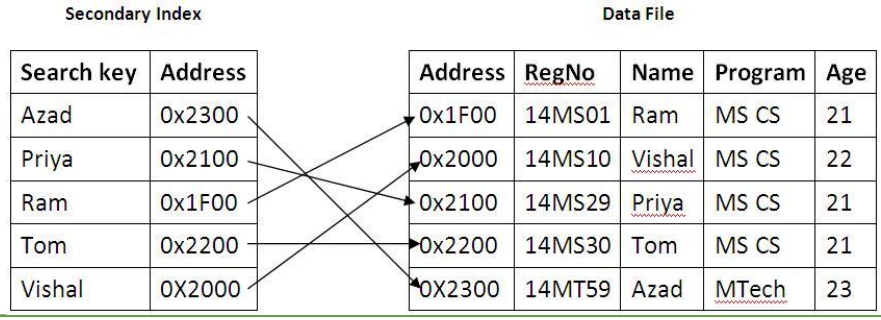
- The size of datasets for applications is rapidly increasing.
- Increased bandwidth and low-cost storage have created an environment where even small-scale applications require more data storage than many databases were designed to manage.
- Developers are faced with a difficult decision as the amount of data they need to store expands.
- **How should they scale their databases rapidly?**



Scaling out MongoDB using sharding across multiple servers

- MongoDB was created with scalability in mind. Splitting data across different servers is easy using the **document-oriented data paradigm.**
- As demonstrated in Figure, **MongoDB** dynamically balances data and load throughout a cluster, redistributing documents as needed, and routing reads and writes to the appropriate servers.

Characteristics of MongoDB



Search key	Address
Azad	0x2300
Priya	0x2100
Ram	0x1F00
Tom	0x2200
Vishal	0X2000

Address	RegNo	Name	Program	Age
0x1F00	14MS01	Ram	MS CS	21
0x2000	14MS10	Vishal	MS CS	22
0x2100	14MS29	Priya	MS CS	21
0x2200	14MS30	Tom	MS CS	21
0X2300	14MT59	Azad	MTech	23

- MongoDB is a general-purpose database, which means that in addition to **creating, reading, updating, and deleting** data, it has most of the functionality you'd expect from a database management system, as well as a few unique characteristics.
- **Indexing:** MongoDB supports both generic and unique secondary indexes, as well as compound, geographic, and full-text indexing.
- **Aggregation:** MongoDB has an aggregation system that is built on data processing pipelines. Aggregation pipelines enable you to create complicated analytics engines by processing data through a succession of relatively simple server-side stages while taking advantage of database improvements.
- **Special Collection and Index Types:** MongoDB has time-to-live (TTL) collections for data that should expire after a given amount of time, such as sessions, and fixed-size (capped) collections for data that should be kept for a long time, such as logs.
- **File Storage:** MongoDB supports an easy-to-use protocol for storing large files and file metadata.

MongoDB Working

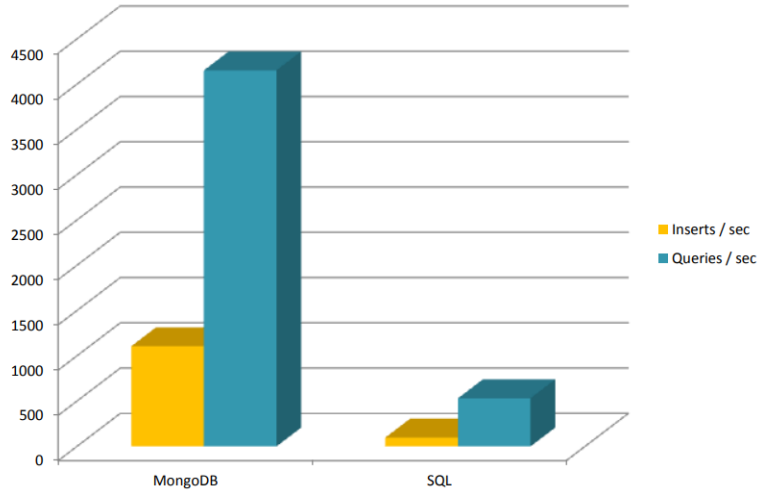
cct | College Dublin
Computing • IT • Business

- **MongoDB** is powerful and its functionality is mentioned as
- A **document** is the basic unit of data for **MongoDB** and is roughly equivalent to a **row** in a relational database management system (RDBMS).
- A **collection** can be thought of as a **table** with a dynamic schema.
- A single instance of **MongoDB** can host multiple independent databases, each of which contains its own collections.
- Every document has a special key, "**_id**", that is unique within a collection.
- **MongoDB** is distributed with a simple but powerful tool called the **mongo shell**.
    - The **mongo shell** provides built-in support for administering MongoDB instances and manipulating data using the MongoDB query language.
    - It is also a fully functional JavaScript interpreter that enables users to create and load their own scripts for a variety of purposes.

MongoDB vs Relational DBMS

cct | College Dublin
Computing • IT • Business

- **Collection** vs **Table**
- **Document** vs **Row**
- **Field** vs **Column**
- **Schema-less** vs **Schema-oriented**
- **Example: Mongo Document**
- **user** = { name: "Z", occupation: "A scientist", location: “New York" }




```
{ "_id": ObjectId("4efa8d2b7d284dad101e4bc9"),
  "Last Name": "DUMONT",
  "First Name": "Jean",
  "Date of Birth": "01-22-1963" },

{ "_id": ObjectId("4efa8d2b7d284dad101e4bc7"),
  "Last Name": "PELLERIN",
  "First Name": "Franck",
  "Date of Birth": "09-19-1983",
  "Address": "1 chemin des Loges",
  "City": "VERSAILLES" }
```

Obligatory, and automatically generated by MongoDB

MongoDB
Documents

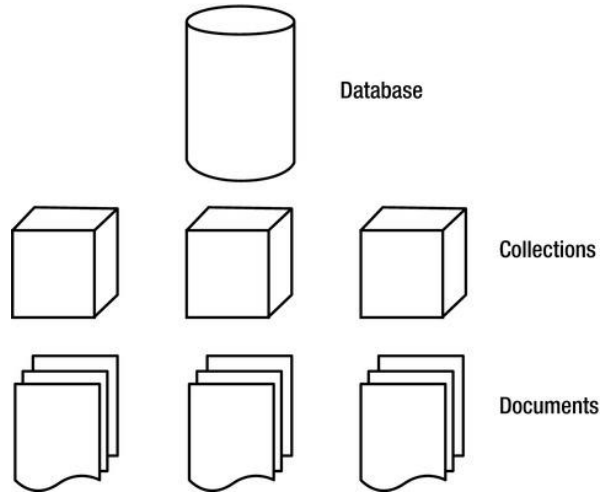
cct College Dublin
Computing • IT • Business

- The document, which is an ordered set of keys with associated values, is at the centre of MongoDB.
- The representation of a document varies by programming language, but most languages have a data structure that is a natural fit, such as a map, hash, or dictionary.
- MongoDB is type-sensitive and case-sensitive. For example, these documents are distinct

```
{"count" : 5}
{"count" : "5"}
```

- Duplicate keys are not allowed in MongoDB documents. For example, the following is not a legal document

```
{"greeting" : "Hello, world!", "greeting" : "Hello, MongoDB!"}
```



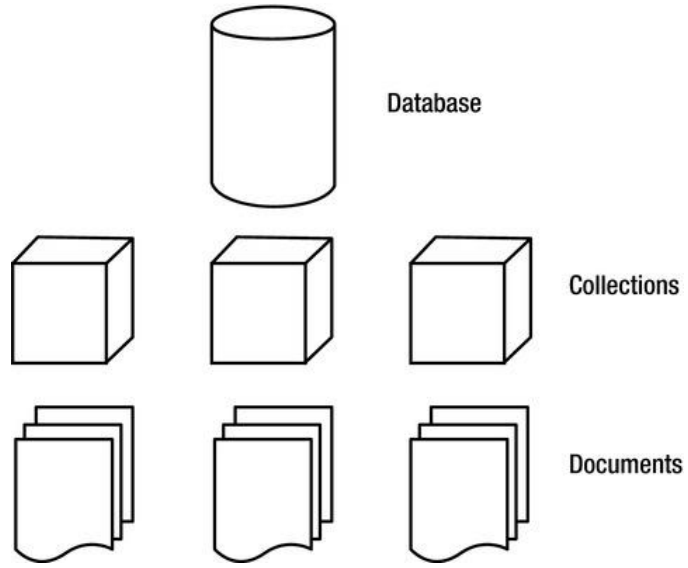
MongoDB

Collections

- A group of documents is referred to as a collection. If a document is the MongoDB similar to a row in RDBMS, then a collection can be thought of as the similar to a table.
- **Dynamic Schemas**
- **Collections** have dynamic schemas. This means that the documents within a single collection can have any number of different “shapes.” For example, both of the following documents could be stored in a single collection.

```
{"greeting" : "Hello, world!", "views": 3}
{"signoff": "Good night, and good luck"}
```

- Note that the previous documents have different keys, different numbers of keys, and values of different types.
- Why do we need separate collections at all?
- With no need for separate schemas for different kinds of documents, why should we use more than one collection?



Key Components of
MongoDB Datastorage

cct | College Dublin
Computing • IT • Business

1. **_id:** This is a field required in every MongoDB document. The **_id** field represents a unique value in the MongoDB document. The **_id** field is like the document's primary key. If you create a new document without an **_id** field, MongoDB will automatically create the field.

2. **Collection:** A collection exists within a single database. As seen from the introduction collections don't enforce any sort of structure.

3. **Database:** This is a container for collections like in RDBMS wherein it is a container for tables. Each database gets its own set of files on the file system. A MongoDB server can store multiple databases.

4. **Document:** A record in a MongoDB collection is basically called a document. The document will consist of field name and values.

5. **Field:** A name-value pair in a document. A document has zero or more fields. Fields are analogous to columns in relational databases

MongoDB Document Structure

cct College Dublin
Computing • IT • Business

- In the relational approach, your data structure might look something like this:

```
|_media
    |_cds
        |_id, artist, title, genre, releasedate
    |_ cd_tracklists
        |_cd_id, songtitle, length
```

- In the nonrelational approach, your data structure might look something like this:

```
|_media
    |_items
        |_<document>
```

- In the nonrelational approach, the document might look something like the following:

```
{
    "Type": "CD",
    "Artist": "Nirvana",
    "Title": "Nevermind",
    "Genre": "Grunge",
    "Releasedate": "1991.09.24",
    "Tracklist": [
        {
        "Track" : "1",
        "Title" : "Smells Like Teen Spirit",
        "Length" : "5:02"
        },
        {
        "Track" : "2",
        "Title" : "In Bloom",
        "Length" : "4:15"
        }
    ]
}
```

Query Format in MongoDB

cct | College Dublin
Computing • IT • Business

- Query expression objects indicate a pattern to match
  - `db.users.find( {last_name: 'Smith'} )`
- Several query objects for advanced queries
  - `db.users.find( {age: {$gte: 23} } )`
  - `db.users.find( {age: {$in: [23,25]} } )`
- Exact match an entire embedded object
  - `db.users.find( {address: {street: 'Oak Terrace', city: 'Denton'}} )`
- Dot-notation for a partial match
  - `db.users.find( {"address.city": 'Denton'} )`

Resources/ References

cct | College Dublin
Computing • IT • Business

- Some slides are modified and used from the presentation provided at this link https://people.cs.uchicago.edu/~junchenj/34702/slides/YCSB.pptx.
- MongoDB Basics, David Hows, Peter Membrey, Eelco Plugge, Apress, December 2014.
- MongoDB: The Definitive Guide, 3rd Edition, Shannon Bradshaw, Eoin Brazil, Kristina Chodorow, O'Reilly Media, December 2019.
- Learning Path: Architecting Big Data Applications: A Beginners Guide, O'Reilly Media, Inc., December 2016.
- https://isip.piconepress.com/courses/temple/ece_3822/lectures/2019_spring/lecture_34f.pdf
- Some images are used from Google search repository (https://www.google.ie/search) to enhance the level of learning.