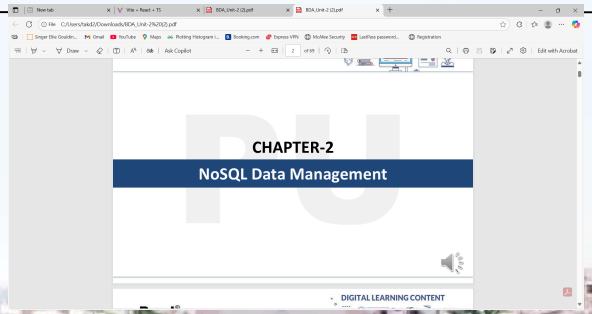
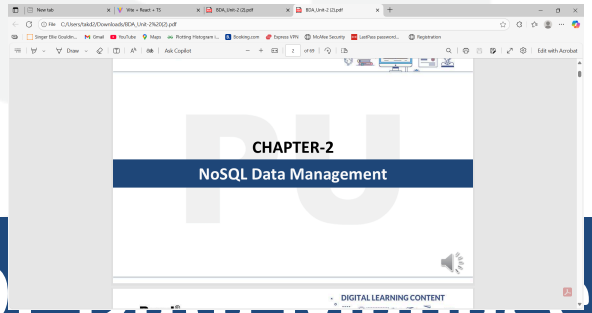
Parul® University

Nosql Data Management



Parul®
University

DIGITAL LEARNING CONTENT



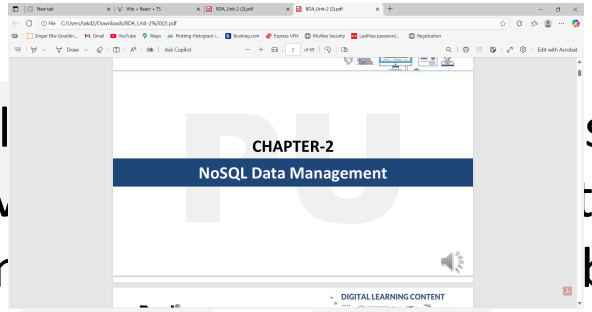
NoSQL Data Management

Parul® University

DIGITAL LEARNING CONTENT

Introduction to NoSQL

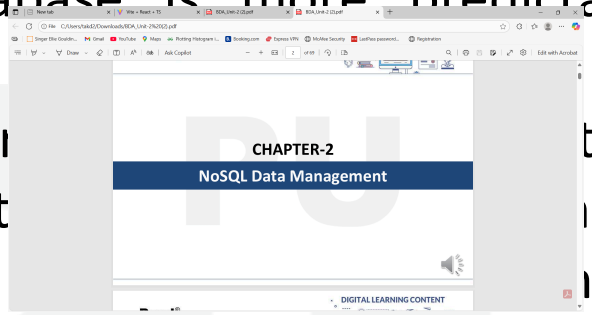
What is NoSQL?



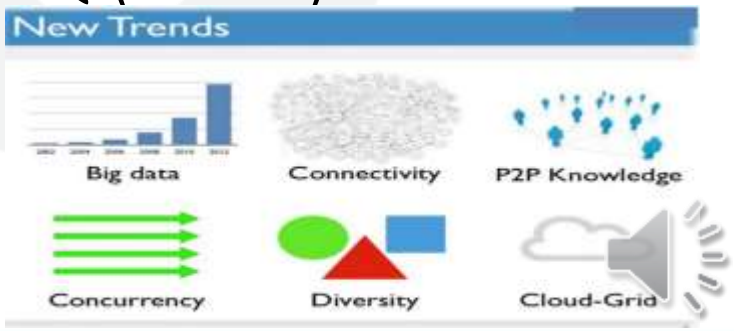
• NoSQL is non-relational system.
• A NoSQL database prov torage and retrieval of data that is modeled in r bular relations used in relational databases.
•It is a type of Non-Relational Database system.
•It follows CAP theorem rather than ACID properties of RDBMS.

Why and when NoSql should be used ?

- The Traditional database is more predictable and preferable for Structured data.
- But, NoSQL is prefer tructured type of data . Unstructured data is t o proper format this type of data is generated s in the form of satellite image, social media, mobile data, web-content etc
- NoSQL has high performance with high availability, and offers rich query language and easy scalability compared to SQL(RDBMS)

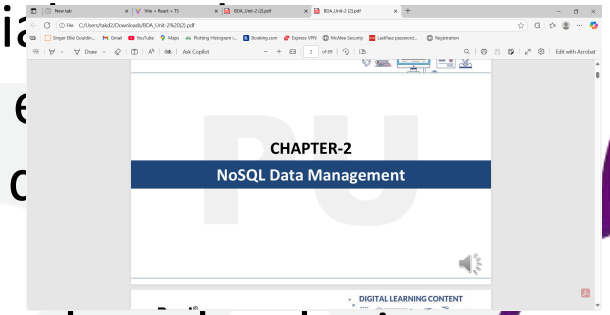


Why NoSQL is needed today most ???



Need of NoSQL

•Explosion of Soci
(FaceBook, Twitter e
generation of large d
formats and size.



•Rise of Cloud based technologies like Amazon Web Services (AWS), Google cloud and Microsoft Azure

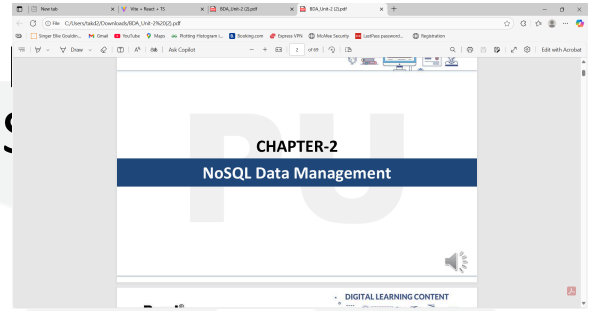
•Expansion of Open source communities.

Parul® University

DIGITAL LEARNING CONTENT

Characteristics of NoSQL

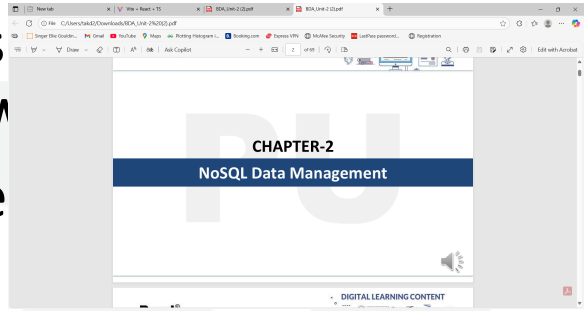
- Does not follow ACID
- Less complexity as of S
- Schema less
- Fast development
- Large data volumes
- Easy and frequent DB changes



Parul® University

When to use NoSQL

- When Traditional RDBMS model provides restriction
- When ACID properties
- Temporary data(Like W session data etc)
- When joins are expe uct cost, hardware, maintenance)

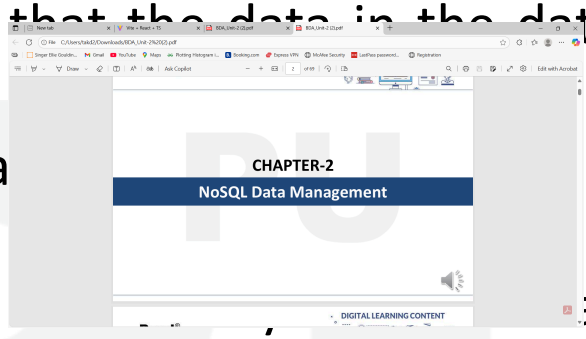


When not to use NoSQL

- Financial Data, Data requiring strict ACID properties, Buisness critical data

CAP Theorem

•**Consistency**-This means that the data in the database remains consistent after the execution of an after an update operation all clients see the same da



•**Availability**-This means ays on (service guarantee availability), no downtime.

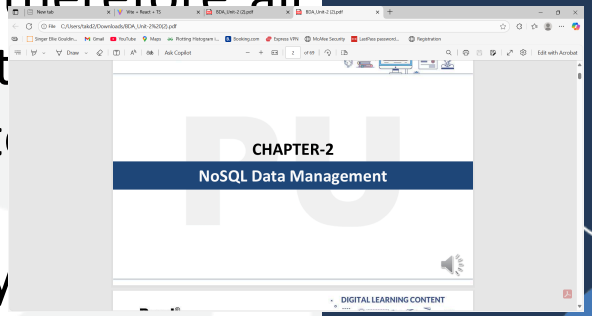
•**Partition Tolerance**-This means that the system continues to function even the communication among the servers is unreliable, i.e. the servers may be partitioned into multiple groups that cannot communicate with one another. Here, if part of the database is unavailable, other parts are always unaffected.

Parul® University

DIGITAL LEARNING CONTENT

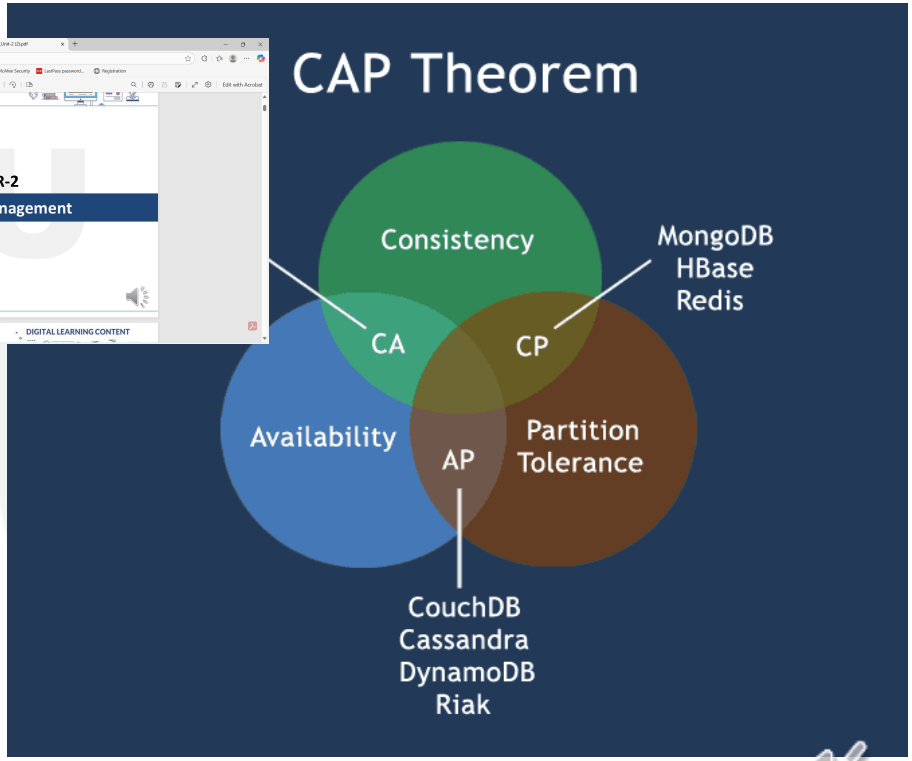
CAP Theorem

•**CA** -Single site cluster, therefore all nodes are always in cont
partition occurs, the syst



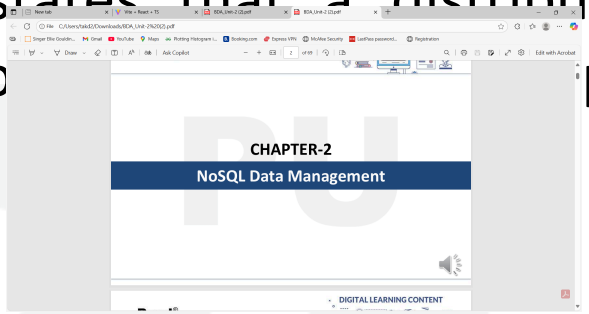
•**CP**-Some data may accessible, but the rest is still consistent/accurate.

•**AP**-System is still available under partitioning, but some of the data returned may be inaccurate.



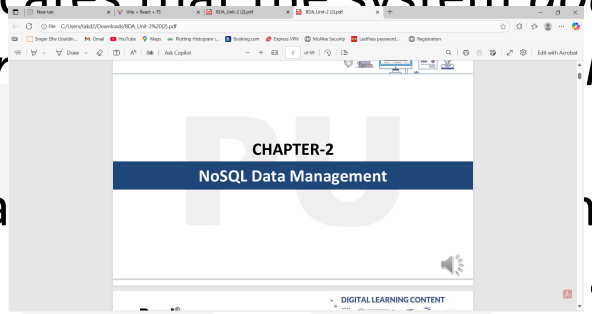
The BASE

- The CAP theorem states that a distributed computer system cannot guarantee all of the properties at the same time:
  - Consistency
  - Availability
  - Partition tolerance
- A BASE system gives up on consistency.



The BASE

- **Basically Available** indicates that the system *does guarantee availability, in terms of the CAP theor* *ll the time.*

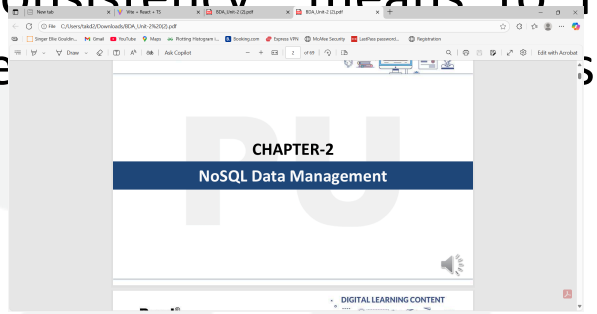


- **Soft state** indicates tha n may change over time, even without input. This al consistency model.

- **Eventual consistency** indicates that the system will become consistent over time, given that the system doesn't receive input during that time.

Eventual Consistency

•The term "eventual consistency" means to have copies of data on multiple machines to ge scalability. Thus, changes made to any data item be propagated to other replicas.



•Data replication may not be instantaneous as some copies will be updated immediately while others in due course of time. These copies may be mutually, but in due course of time, they become consistent. Hence, the name eventual consistency.

Types of NoSQL

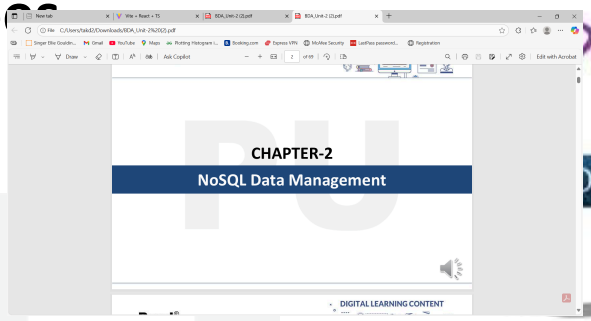
- There are 4 main types of NoSQL databases:

  -Key value

  -Column Family

  -Document

  -Graph



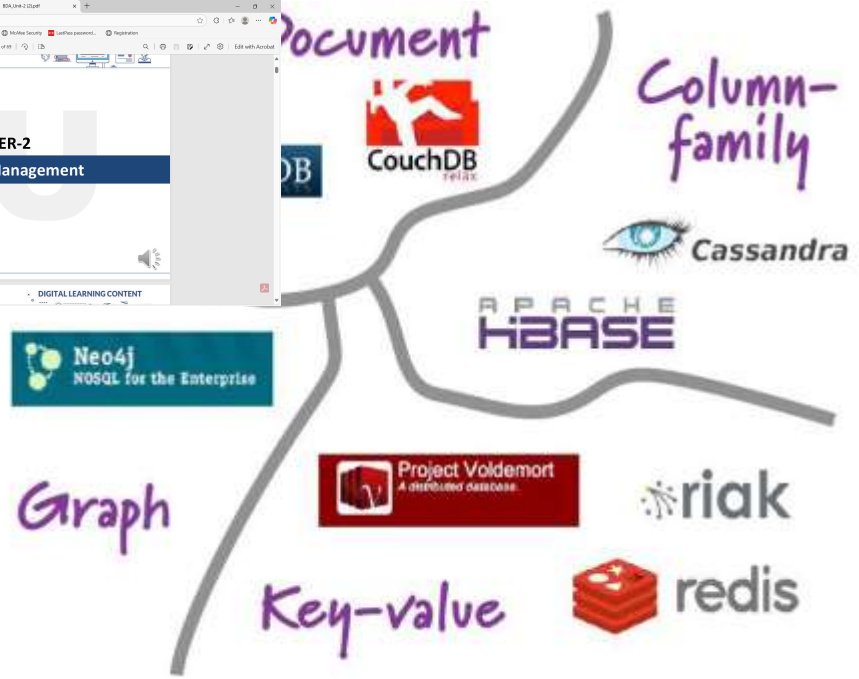


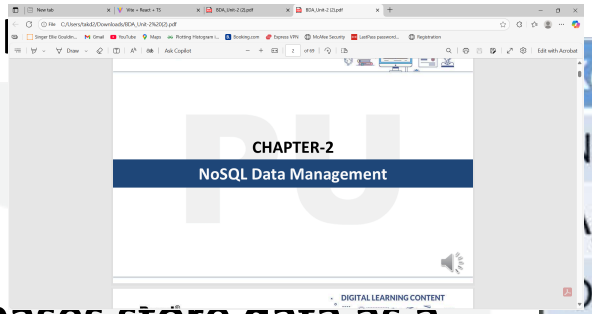
Image source : Google

Parul®
University

DIGITAL LEARNING CONTENT

Key Value

- Data is stored in Key/value pairs in such a way that it can handle load.

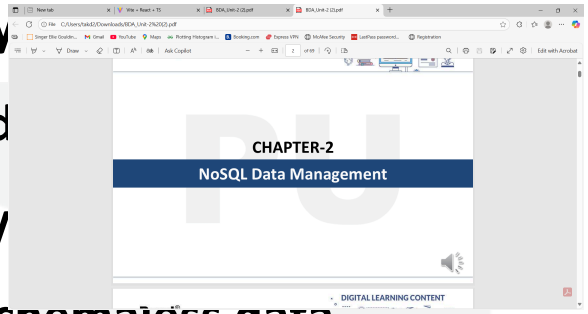


Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

- Key-value pair storage databases store data as a hash table where each key is unique and the value can be JSON, BLOB(Binary Large Object), String etc can be accessed by string called keys
- For EX: A key value pair may contain key like "University" associated with value like "Parul".

Key Value

- It is one of the most basic ty



- This kind of database is used

dictionaries, associative array

help the developer to store schemaless data

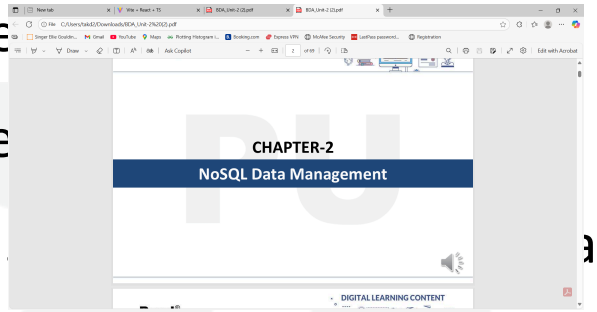
- They work best for Shopping cart contents

Basic Operations are

Insert(Key,value), Fetch(key), Update(key), delete(Key).

Column- based

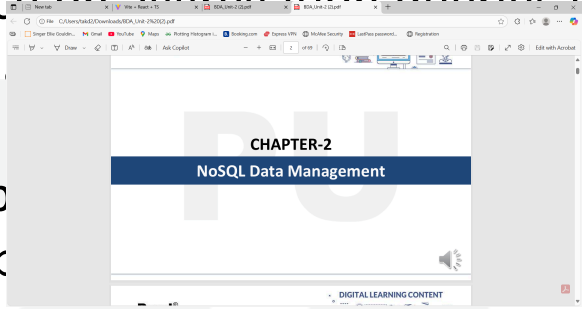
- Column oriented database
- are based on Big table pape
- The column is the lowest/ .
- It is a tuple that contains a name, value and a timestamp
- Every column is treated separately.
- Values of single column databases are stored contiguously.



ColumnFamily			
Row Key	Column Name		
	Key	Key	Key
	Value	Value	Value
	Column Name		
	Key	Key	Key
	Value	Value	Value

Column based

•They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily avail



•Column-based NoSQL datab manage data warehouses, business intelligence, CRM, Lib

•HBase, Cassandra, HBase, Hypertableare examples of column based database

Some statistics about Facebook search(Using Cassandra).

MySQL>50gb data	Rewritten in Cassandra<50gb data
Writes average – 300 ms	Writes average: 0.12 ms
Reads average _ 350 ms	reads average: 15ms

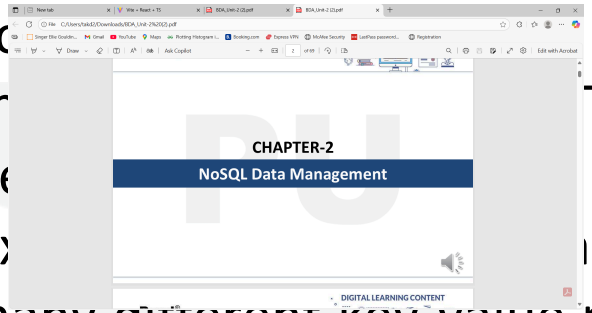
Document based

•Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is sto

•The document is stored ir The value is understood by the DB and can be querie

•Pair each key with complex s data structure.

•Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.



Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data

Document 1

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 2

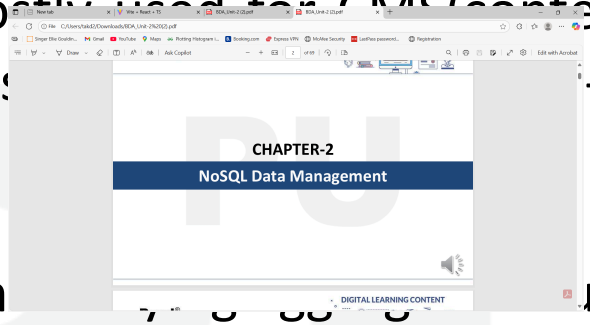
```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 3

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document based

•The document type is mostly used for CMS(content management system) systems, blogging platforms -commerce applications.



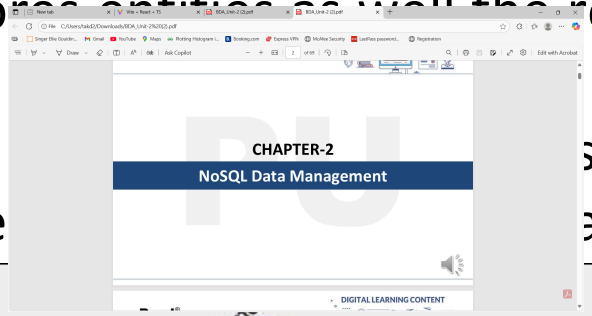
•It should not use for which require multiple operations or queries again uctures.

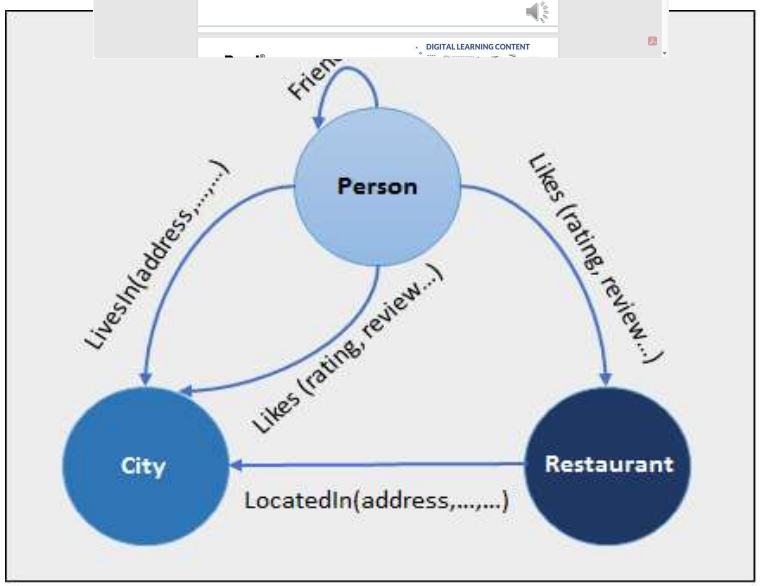
•Amazon SimpleDB, CouchDB, MongoDB, Riak, Lotus Notes,are popular Document originated DBMS systems.

Graph based

•A graph type database stores entities as well the relations amongst those entities.

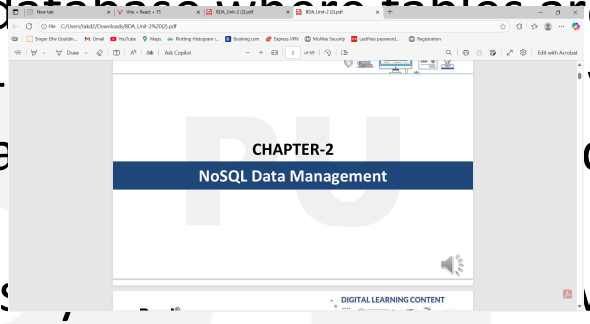
•The entity is stored as a ship as edges. An edge gives a relationship betwe and edge has a unique identifier.





Graph based

•Compared to a relational database where tables are loosely connected, a Graph database is a multi- versing relationship is fast as they are already ca d there is no need to calculate them.



•Graph base database mos vorks, logistics, spatial data.

•Neo4J, Infinite Graph, OrientDB, FlockDBare some popular graph-based databases.

Parul® University

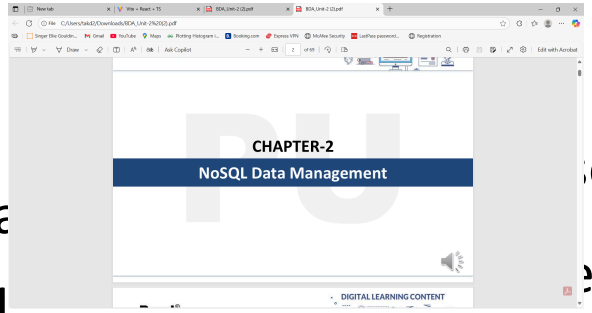
DIGITAL LEARNING CONTENT

Comparison of SQL and NoSQL

1.Relational Database Management System

2. SQL is also know as Structured query langua

3.They have predefined Schema



1.Non relational management

o known as Not

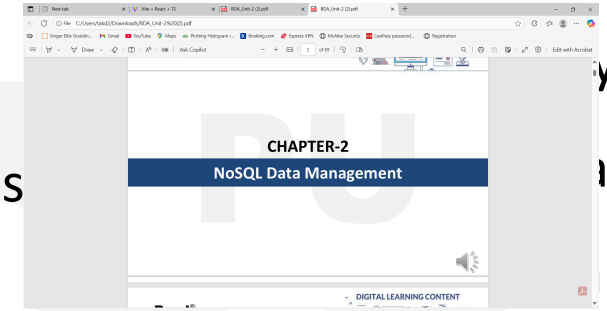
ed query language

3.They have dynamic schema for unstructured data

Comparison of SQL and NoSQL

4.Vertically scalable

5. RDBMS are Table bas



y scalable

abases are document,

ph or wide-column

based

6.SQL are better for multi-row transactions.

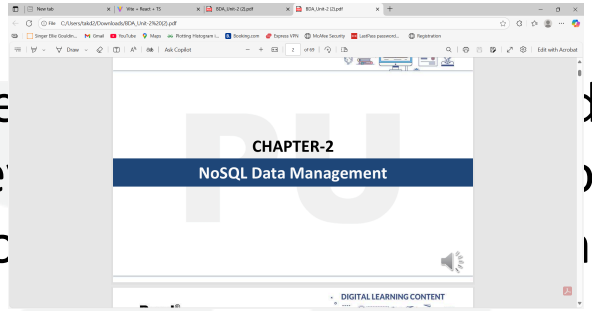
7. Example: Oracle, MySQL

6. NoSQL is better for unstructured data like documents or JSON

7. Example: MongoDB, Cassandra, CouchDB etc.

What is NewSQL

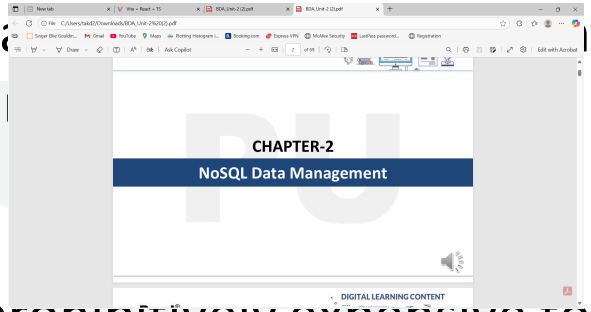
•The term NewSQL is not exactly as wide as NoSQL.



•NewSQL systems all be data model and the SQL query language and the ortion of similar types of scalability, flexibility c as driven the NoSQL development.

Why NewSQL ?

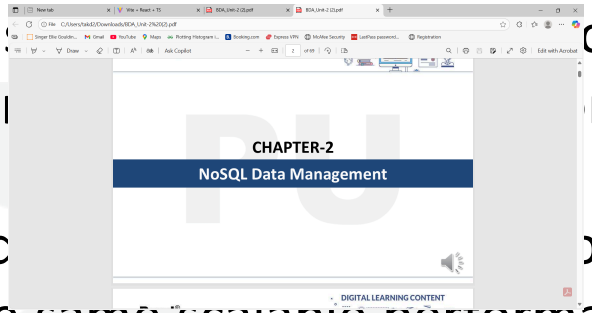
•The early counter-measures to the above approaches were a powerful single node machine tha[illegible]e transaction, a custom built middleware syste[illegible] over traditional DBMS nodes.



•But both of them are prohibitively expensive to be carried out.

Why NewSQL ?

•As a result, there was a need of an intermediate database system which combines the di of NoSQL systems with multiple node concurre rage mechanism.

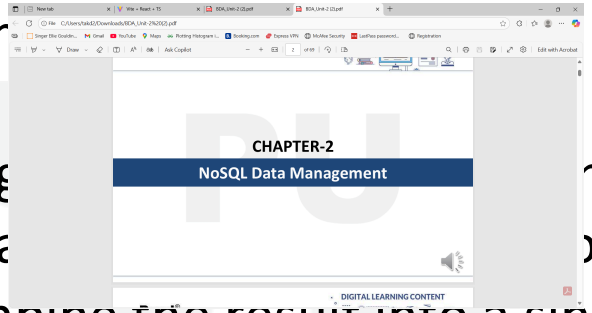


•Thus Newsql can be d odern relational DBMSs that seek to provide the same scalable performance of NoSQL for OLTP workloads and simultaneously guaranteeing ACID compliance for transactions as in RDBMS.

•these systems want to achieve the scalability of NoSQL without having to discard the relational model with SQL and transaction support of the legacy DBMS.

Concepts of NewSQL

•Main Memory storage of OLTP(Online transition process) databases enables in-memory com



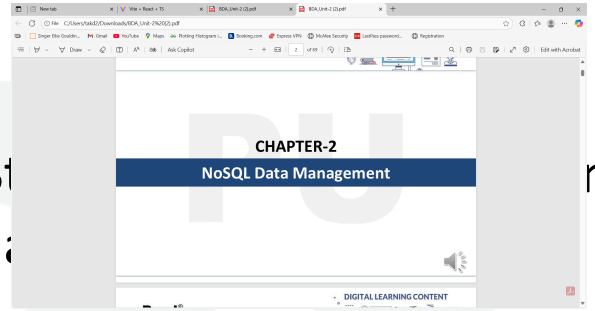
•Scaling out by splitting nt subsets called either partitions or shards, lea of a query into multiple partitions and then combine the result into a single result.

•NewSQL systems preserve the ACID properties of databases.

•Enhanced concurrency control system benefits upon traditional ones.

Concepts of NewSQL

•Presence of secondary index allowing NewSQL to support faster query processing times.



•High availability and St made possible with the use of replication mecha

•NewSQL systems can be configured to provide synchronous updates of data over the WAN.

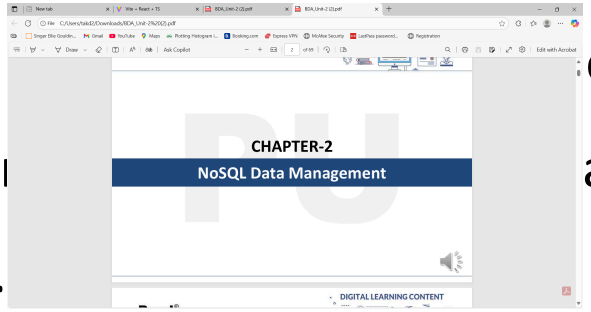
•Minimizes Downtime, provides fault tolerance with its crash recovery mechanism.

Parul® University

DIGITAL LEARNING CONTENT

Comparison of NoSQL and NewSQL

1. It doesn't follow a relational model, it designed to be enti different from that.	1. Since the relational model essential for analytics.
2. Supports CAP Theorem	2. Supports ACID property
3. Supports old SQL	3. Supports old SQL and even enhanced functionality o f old SQL



Parul® University

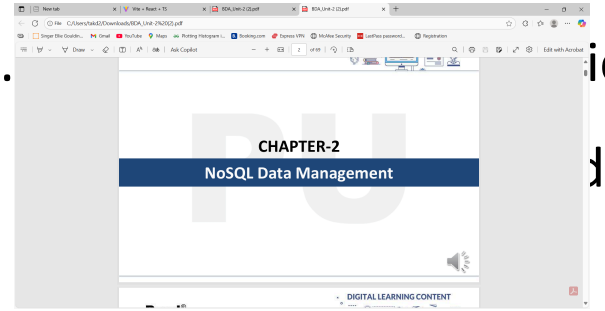
Comparison of NoSQL and NewSQL

4. Supports OLTP database but it is not best suited.

5. Vertical Scaling

6. Better than SQL for processing complex queries

7. Supports distributed system



4. Supports OLTP database icient.

d horizontal

6. Highly efficient to process complex queries and smaller queries

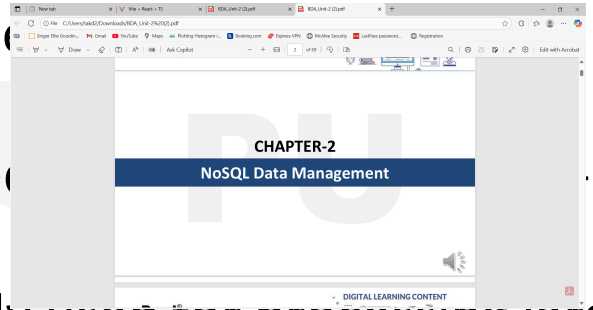
7. Supports distributed system

Parul® University

DIGITAL LEARNING CONTENT

Map reduce

•Map-reduce is a data processing paradigm for condensing large volumes of data into use



•MongoDB uses mapRe reduce operations.

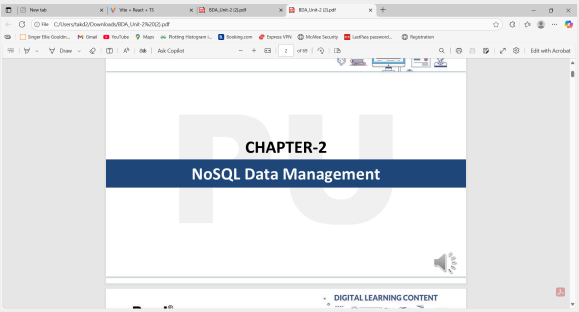
•MapReduce is generally used for processing large data sets.

Parul® University

DIGITAL LEARNING CONTENT

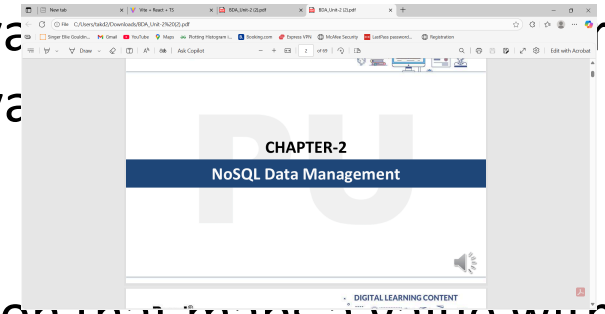
MapReduce Command

```
>db.collection.mapReduce(
   function() {emit(key,v
   function(key,values) {                    //reduce function
      out: collection,
      query: document,
      sort: document,
      limit: number
   }
)
```



MapReduce Command

The map-reduce function first queries the collection, then maps the result documents to emit key-va reduced based on the keys that have multiple va



In the above syntax −

map is a javascript function that maps a value with a key and emits a key-value pair

reduce is a javascript function that reduces or groups all the documents having the same key

out specifies the location of the map-reduce query result

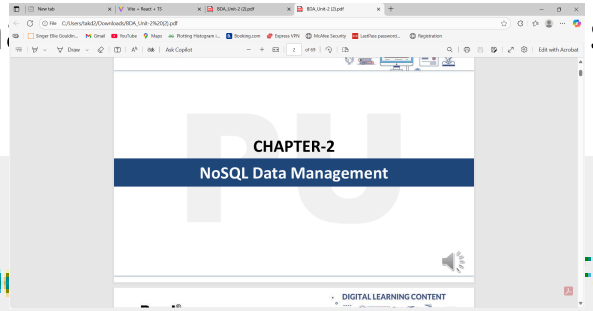
query specifies the optional selection criteria for selecting documents

sort specifies the optional sort criteria

limit specifies the optional maximum number of documents to be returned

Using MapReduce

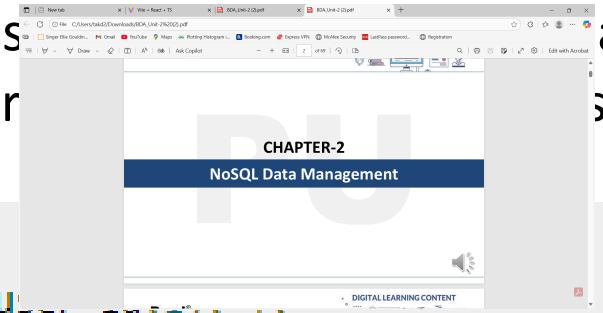
Consider the following document structure storing user posts. The document stores user_n status of post.




```
{
   "post_text": "tutorialsp                    or tutorials",
   "user_name": "mark",
   "status":"active"
}
```

Using MapReduce

Now, we will use a mapReduce function on our posts collection to select all the active pos asis of user_name and then count the number sing the following code –

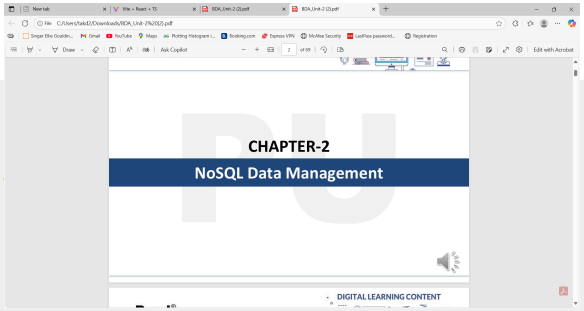



```
>db.posts.mapReduce(
   function() { emit(this.u_  _  }, );

   function(key, values) {return Array.sum(values)}, {
      query:{status:"active"},
      out:"post_total"
   }
)
```

Using MapReduce

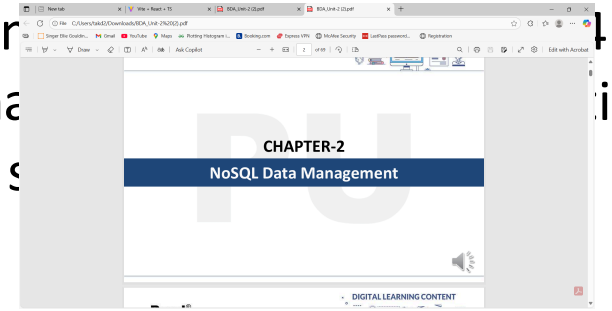
The above mapReduce query outputs the following result –




```
{
   "result" : "post_total"
   "timeMillis" : 9,
   "counts" : {
      "input" : 4,
      "emit" : 4,
      "reduce" : 2,
      "output" : 2
   },
   "ok" : 1,
}
```

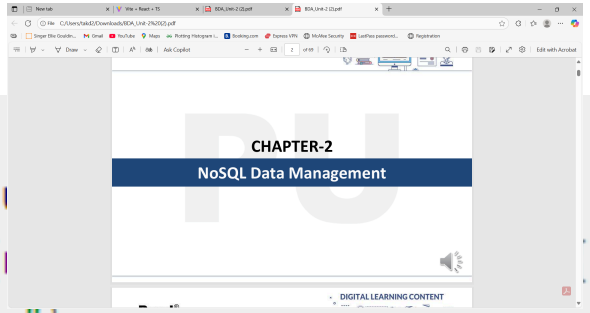
Using MapReduce

The result shows that a total of 4 documents matched the query (status:"active"), the r[illegible]l documents with key-value pairs and fina[illegible]ion grouped mapped documents having the s[illegible]



Using MapReduce

To see the result of this mapReduce query, use the find operator –



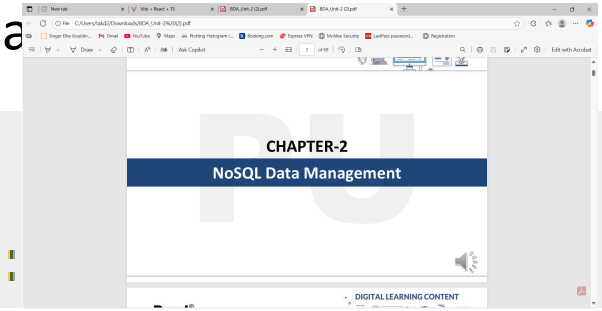

```
>db.posts.mapReduce(
   function() { emit(this.
   function(key, values) {
      query:{status:"active"},
      out:"post_total"
   }

).find()
```

Using MapReduce

The above query gives the following result which indicates that both users tom and mark ha ates –

```
{ "_id" : "tom", "value" :
{ "_id" : "mark", "value" :
```



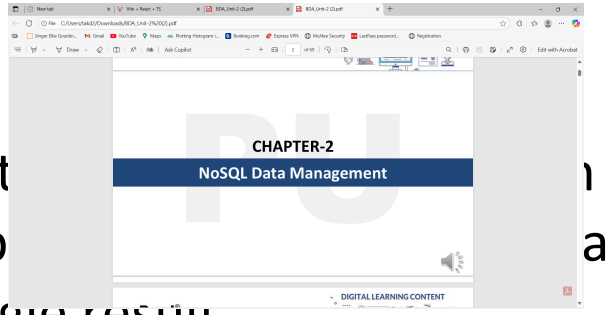
In a similar manner, MapReduce queries can be used to construct large complex aggregation queries. The use of custom Javascript functions make use of MapReduce which is very flexible and powerful.

Parul® University

DIGITAL LEARNING CONTENT

Aggregations

- Aggregations operations process data records and return computed results.



- Aggregation operat … multiple documents together, and can p … ations on the grouped data to return a single result.

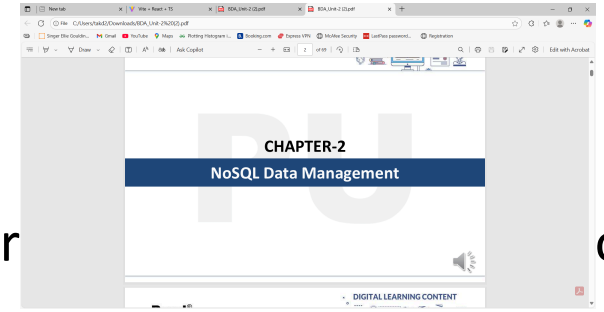
- In SQL count(*) and with group by is an equivalent of MongoDB aggregation.

Parul® University

DIGITAL LEARNING CONTENT

Aggregations

- For the aggregation in MongoDB, you should use aggregate() method.



- Syntax
- Basic syntax of aggr ows –

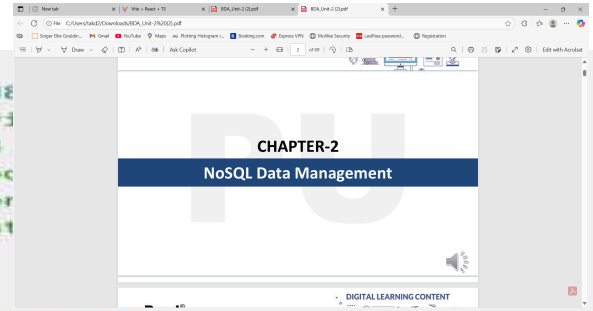
```
>db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)
```

Parul® University

DIGITAL LEARNING CONTENT

Aggregations

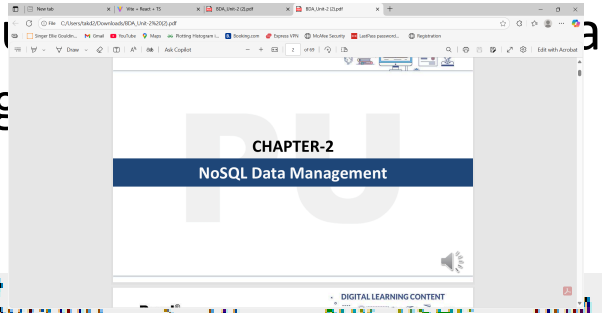
- In the collection you have the following data −




```
{
    _id: ObjectId(7df78ad8
    title: 'MongoDB Overvi
    description: 'MongoDB
    by_user: 'tutorials po
    url: 'http://www.tutor
    tags: ['mongodb', 'dat
    likes: 100
},
{
    _id: ObjectId(7df78ad8902d)
    title: 'NoSQL Overview',
    description: 'No sql database is very fast',
    by_user: 'tutorials point',
    url: 'http://www.tutorialspoint.com',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 10
},
{
    _id: ObjectId(7df78ad8902e)
    title: 'Neo4j Overview',
    description: 'Neo4j is no sql database',
    by_user: 'Neo4j',
    url: 'http://www.neo4j.com',
    tags: ['neo4j', 'database', 'NoSQL'],
    likes: 750
},
```

Aggregations

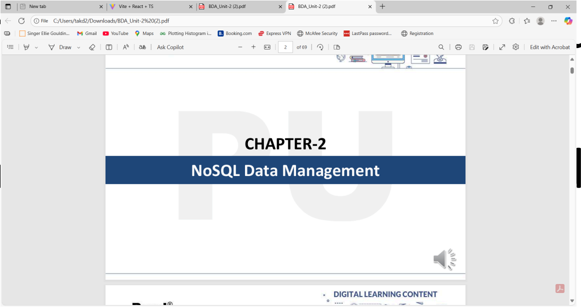
- Now from the above collection, if you want to display a list stating how many tu ach user, then you will use the following ag




```
> db.mycol.aggregate([{$group : {_id : "by_user", num_tutorial : {$sum : 1}}}])
{ "_id" : "tutorials point", "num_tutorial" : 2 }
{ "_id" : "Neo4j", "num_tutorial" : 1 }
>
```

Aggregations

- Sql equivalent query for the above use case will be select by_user, count(*) fr user.

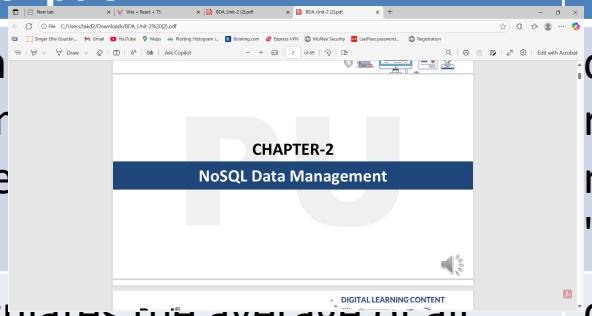


- In the above exan l documents by field by_user and on eac previous value of sum is incremented.

Parul® University

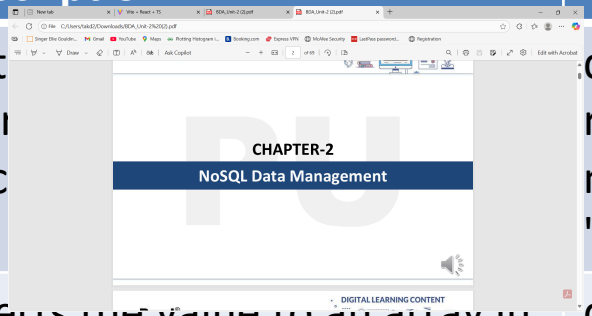
Aggregations

Expression	Description	Example
$sum	Sum from colle	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])
$avg	Calculates the average of all given values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
$min	Gets the minimum of the corresponding values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])



Aggregations

Expression	Description	Example
$max	Get cor doc	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
$push	Inserts the value to an array in the resulting document.	db.mycol.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])
$addToSet	Inserts the value to an array in the resulting document but does not create duplicates.	db.mycol.aggregate([{$group : {_id : "$by_user", url : {$addToSet : "$url"}}}])

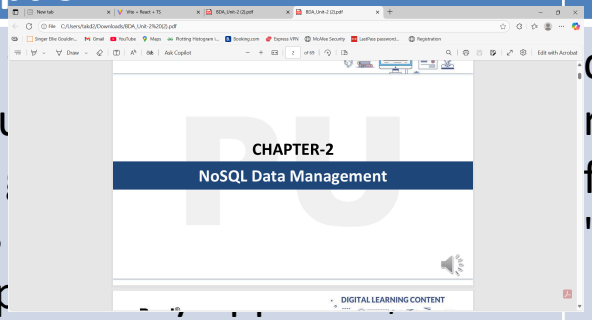


Parul® University

DIGITAL LEARNING CONTENT

Aggregations

Expression	Description	Example
$first	GGets the sou to the makes some p stage.	db.mycol.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])
$last	Gets the last document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage.	db.mycol.aggregate([{$group : {_id : "$by_user", last_url : {$last : "$url"}}}])

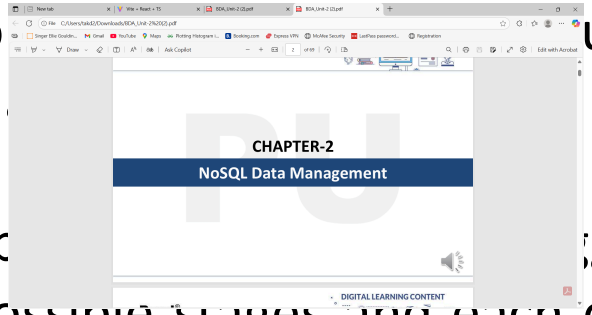


Parul® University

DIGITAL LEARNING CONTENT

Aggregations

- In UNIX command, shell pipeline means the possibility to execute an operation on so utput as the input for the next command

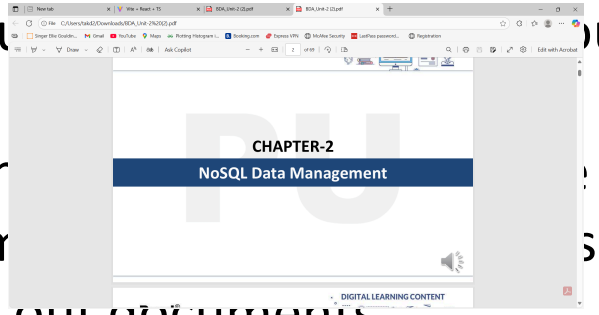


- MongoDB also supp ggregation framework. There is a set of possible stages and each of those is taken as a set of documents as an input and produces a resulting set of documents (or the final resulting JSON document at the end of the pipeline). This can then in turn be used for the next stage and so on.

Aggregations

- The MongoDB aggregation pipeline consists of stages. Each stage transforms the docu ough the pipeline.

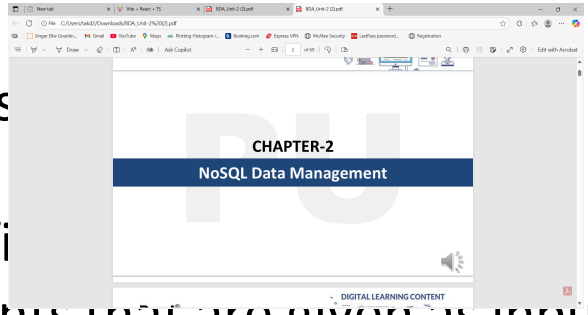
- Pipeline stages do r output document for every input docum s may generate new documents or filter out documents.



Parul® University

Aggregations

- Following are the possible stages in aggregation framework –
- $project – Used to s s from a collection.
- $match – This is a fi us this can reduce the amount of documents that are given as input to the next stage.
- $group – This does the actual aggregation as discussed above.
- $sort – Sorts the documents.
- $skip – With this, it is possible to skip forward in the list of documents for a given amount of documents.

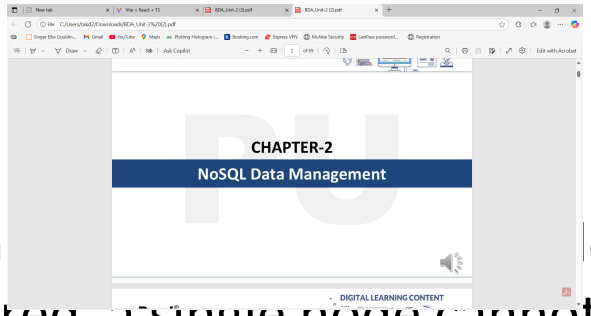


Parul® University

DIGITAL LEARNING CONTENT

Partitioning/Sharding

- Breaking large datasets into smaller ones and distributing datasets and query are requisites to high scalability.

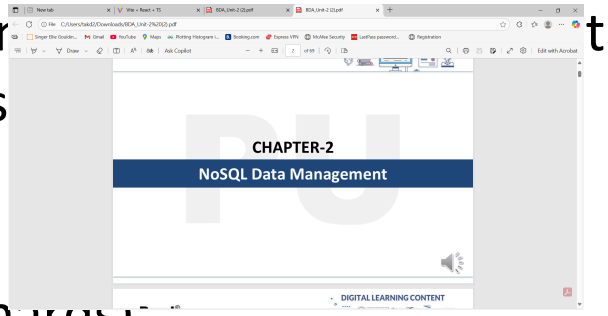


- If a dataset becom e node or when high throughput is required, a single node cannot suffice.

- We need to partition/shard such datasets into smaller chunks and then each partition can act as a database on its own.

Partitioning

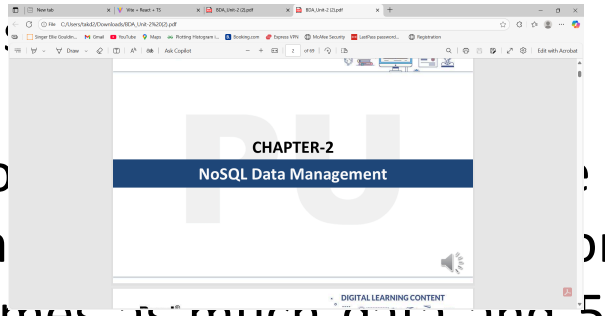
- Thus, a large dataset can be spread across many smaller partitions/shards ar tly execute queries or run some programs



- This way large parallelized across nodes(Partitions/Shards)

Why Partitioning ?

- The purpose behind partitioning is to spread data so that execution can be di

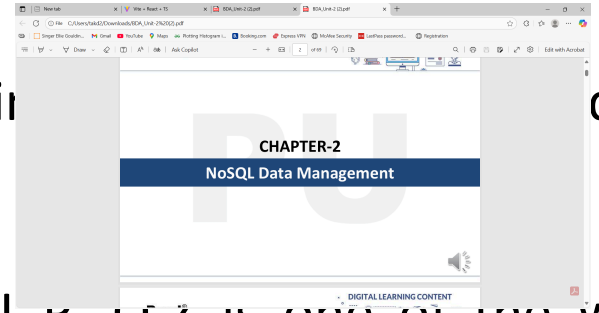


- Along with partitio e that every partition node takes a fair sh ory, 5 nodes should be able to handle 5 times as much data and 5 times as much read and write throughput of a single partition.

- If sharding is unfair, then a single node might be taking all the load and other nodes might sit idle. This defeats the purpose of sharding/partitioning. A good partition strategy should avoid Hot spots.

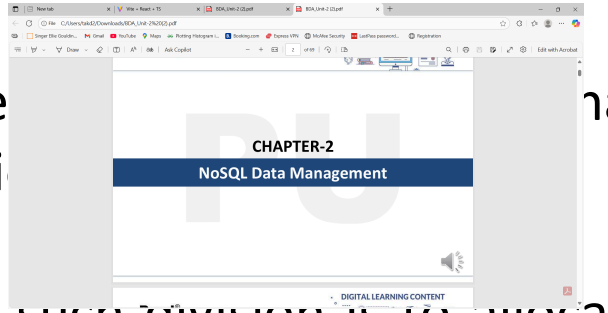
Partition by key-range

- Partition by key-range divides partitions based on certain ranges.
- For example, dividi d on the first letter of their name.
- So A-C, D-F, G-K, L-P, Q-Z is one of the ways by which whole Organization data can be partitioned in 5 parts.
- Now we know the boundaries of such partition so we can directly query a particular partition if we know the name of an employee.



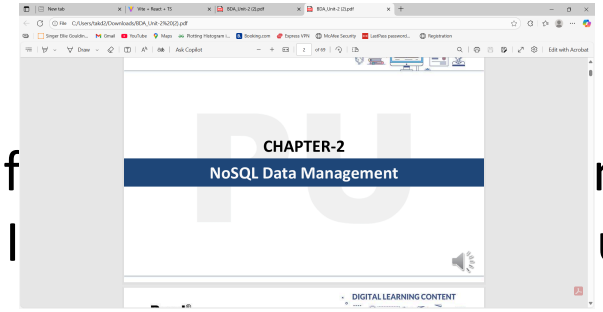
Partition by key-range

- Ranges are not necessarily evenly spaced.
- As in the above e…has 10 letters of the alphabet but partiti…
- The reason behind such division is to allocate the same amount of data to different ranges.
- Due to the fact that most people have the name starting from letters between A-C, as compared to Q-Z, so this strategy will result in near equal distribution of data across the partition.



Partition by key-range

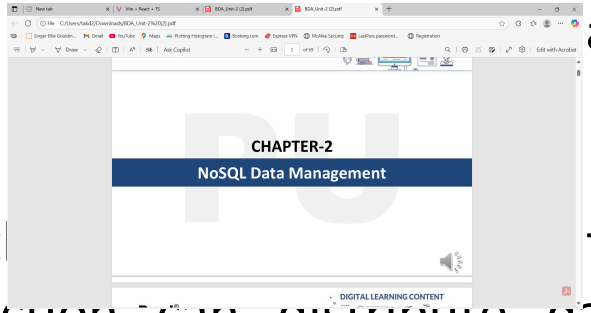
- One of the biggest benefits of such partitioning is the range queries.



- Suppose I need to f me starts from letters between R-S, then I uery to partition R-S.

Partition by key hash

- Key range partition strategy is quite prone to hot spots hence many distributed another partitioning strategy.

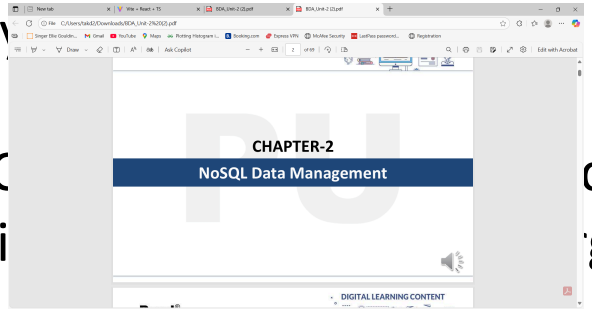


- The hash value of t find out the partition. A good hash function can distribute data uniformly across multiple partitions.

- Cassandra, MongoDB, and Voldemort are databases employing a key hash-based strategy.

Partition by key hash

- Hash functions exercised for the purpose of partitioning should be cryptographicall

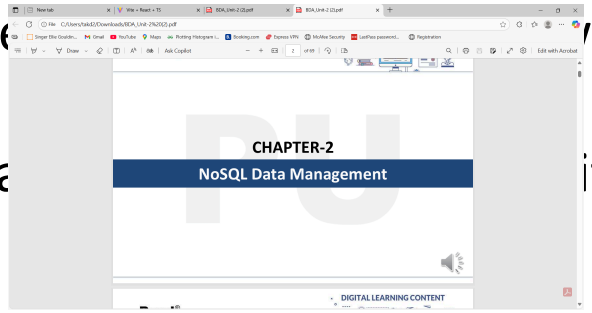


- Java's Object.hashC d for this purpose as such implementati ge number of hash collisions.

- Each partition is then assigned a range of key hashes (Rather than range of keys) and all keys which fall within the parameter of partitions range will be stored on that partition. Partition ranges can be chosen to be evenly spaced or can be chosen from a strategy like consistent hashing.

Partition by key hash

- Sadly with the hash of the key, we lose a nice property of key-range partitioning: e ith some ranges.

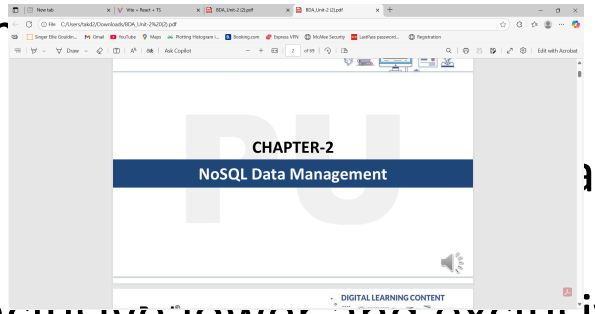


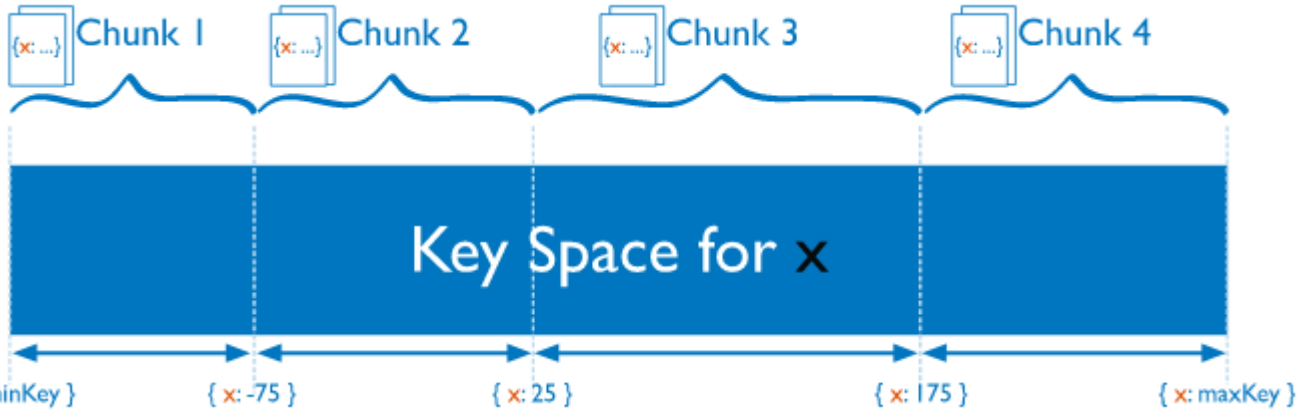
- As keys are now sca itions instead of being adjacent to a single

- Hashing on key indeed reduces hot spots, however, it doesn't eliminate it completely. In case read and writes are for the same key, all requests still end up on the same partition.

Partitioning with Chunks

- MongoDB uses the shard key associated to the collection to partition the data in
- A chunk consists of a.
- Each chunk has a inclusive lower and exclusive upper range based on the shard key.

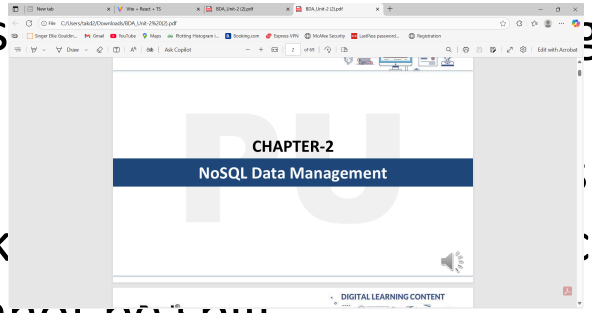




Partitioning with Chunks

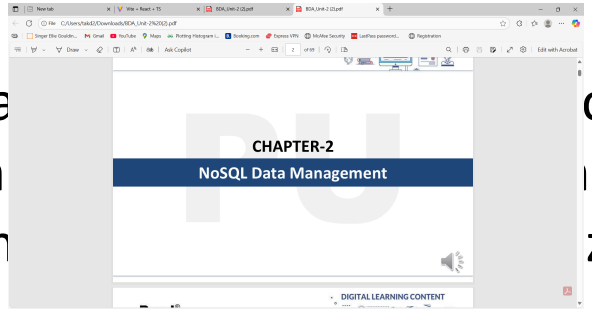
- MongoDB splits chunks when they grow beyond the configured chunk size. Both ins gger a chunk split.

- The smallest range a single unique shard key value. A chunk cuments with a single shard key value cannot be split.



Partitioning with Chunks

- Initial Chunks



- The sharding opera chunk(s) to cover the entire range of th e number of chunks created depends on ze.

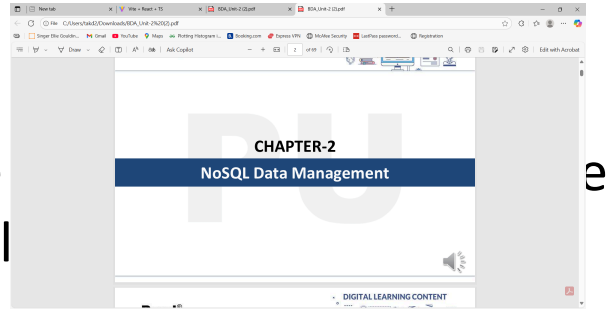
- After the initial chunk creation, the balancer migrates these initial chunks across the shards as appropriate as well as manages the chunk distribution going forward.

Parul® University

Partitioning with Chunks

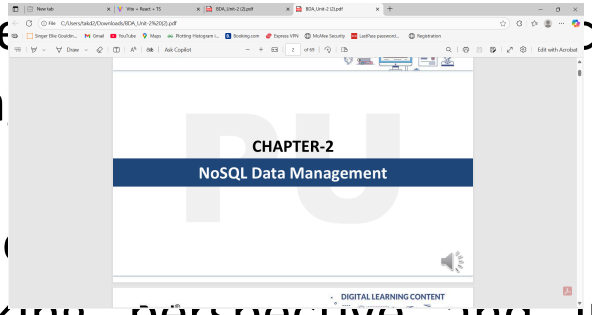
- Chunk Size

- The default chunk megabytes. You can increase or reduce er the implications of changing the defaul



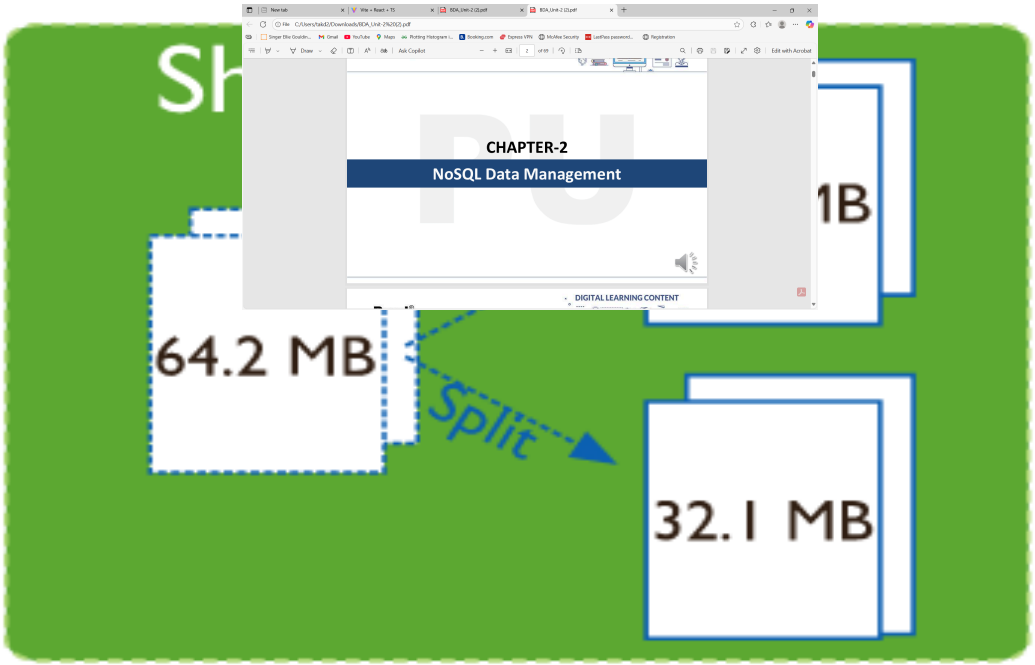
Partitioning with Chunks

- Small chunks lead to a more even distribution of data at the expense of more fre creates expense at the query routing (mon



- Large chunks lead t is more efficient both from the networking perspective and in terms of internal overhead at the query routing layer. But, these efficiencies come at the expense of a potentially uneven distribution of data.

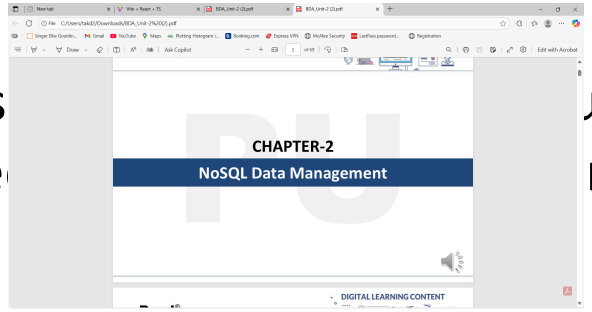
Partitioning with Chunks



Parul® University

Partitioning with Chunks

- Chunk Migration



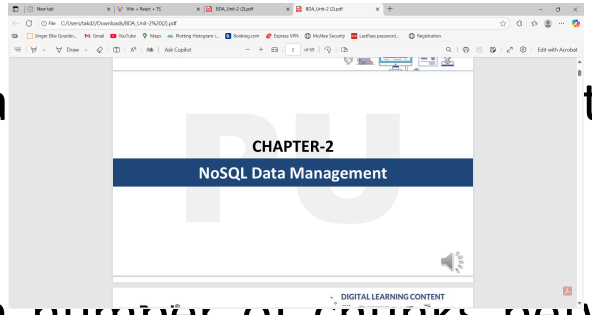
- MongoDB migrates uster to distribute the chunks of a sharde ng shards. Migrations may be either:

- Manual. Only use manual migration in limited cases, such as to distribute data during bulk inserts.

- Automatic. The balancer process automatically migrates chunks when there is an uneven distribution of a sharded collection's chunks across the shards.

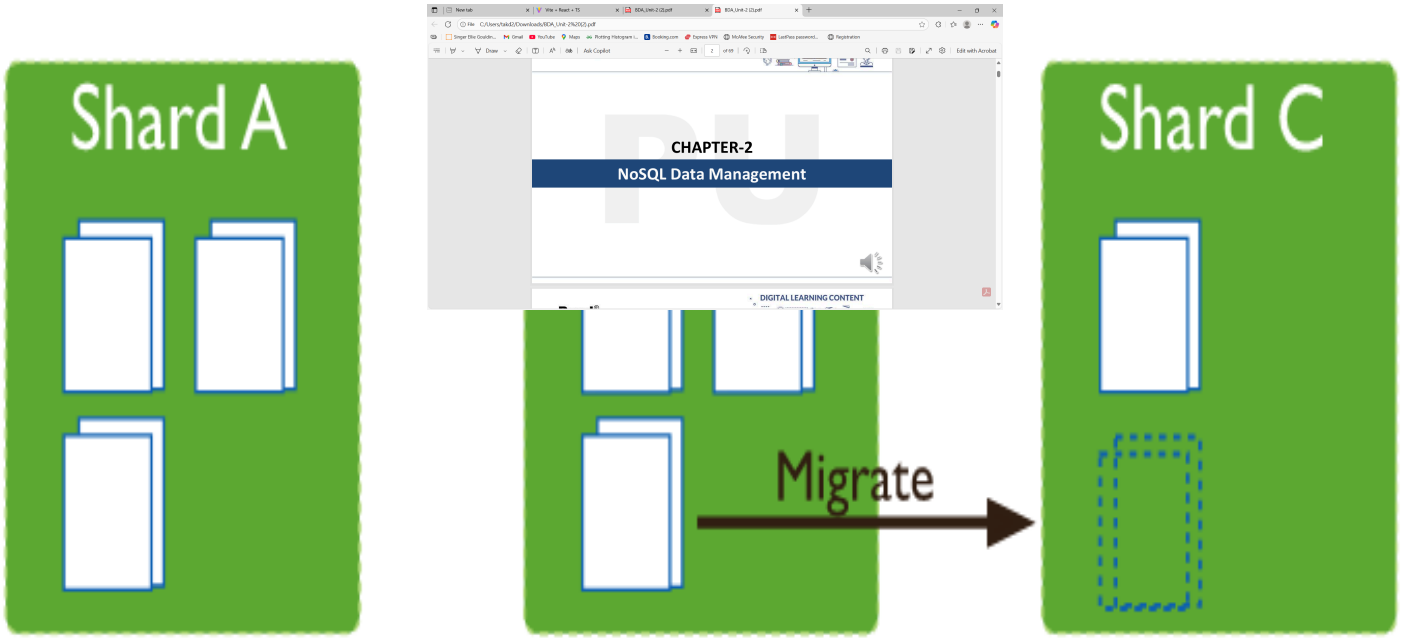
Partitioning with Chunks

- Balancing
- The balancer is a [redacted] that manages chunk migrations.

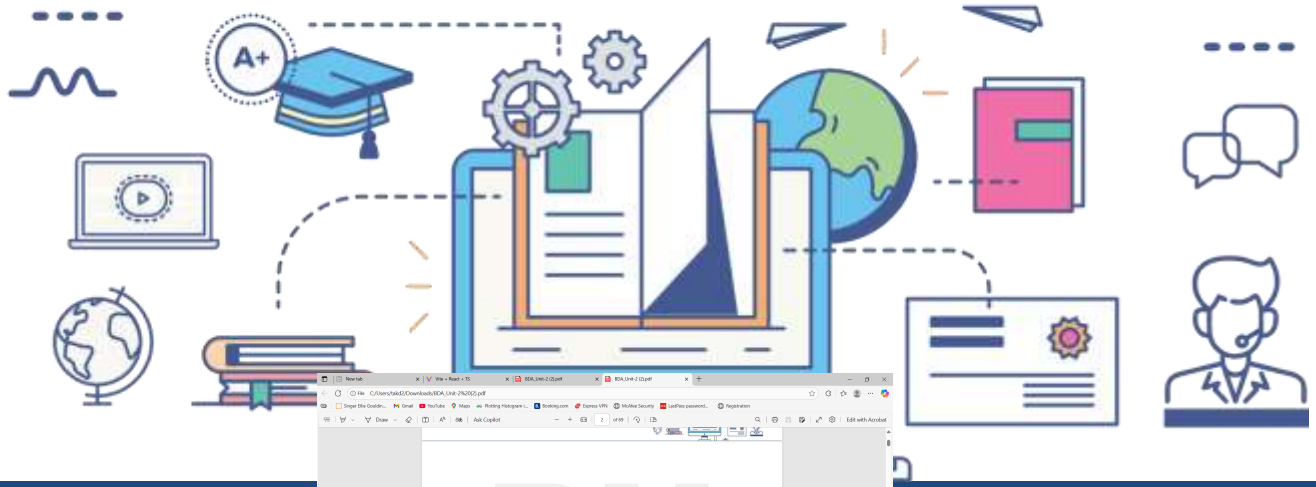


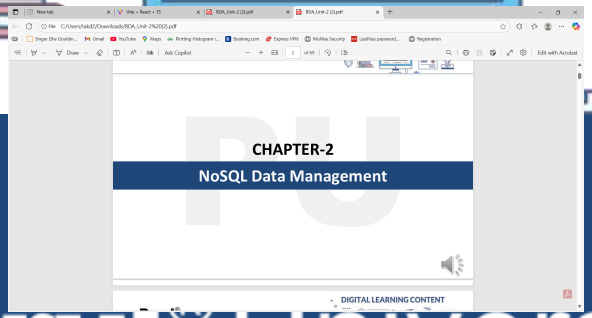
- If the difference in number of chunks between the largest and smallest shard exceed the migration thresholds, the balancer begins migrating chunks across the cluster to ensure an even distribution of data.

Partitioning with Chunks



DIGITAL LEARNING CONTENT





Parul® University

www.paruluniversity.ac.in