JAIN DEEMED-TO-BE UNIVERSITY | SCHOOL OF COMPUTER APPLICATIONS & INFORMATION TECHNOLOGY

Department of Master of Information Technology

18MSIT2H04 - Database Management System

Unit 5

ADVANCE DATABASE AND NOSQL DATABASES

Semi-Structured Data

- Semi-structured data is basically a structured data that is unorganized.
-
- Due to unorganized information, the semi-structured is difficult to retrieve, analyze and store as compared to structured data.

- Semi-structured data is data that does not conform to the standards of traditional structured data, but it contains tags or other types of mark-up that identify individual, distinct entities within the data.

- Two of the key attributes that distinguish semi-structured data from structured data are nested data structures and the lack of a fixed schema:

Semi-Structured Data

- Structured data requires a fixed schema that is defined before the data can be loaded and queried in a relational database system.
- Semi-structured data does not require a prior definition of a schema and can constantly evolve, i.e. new attributes can be added at any time.
- In addition, entities within the same class may have different attributes even though they are grouped together, and the order of the attributes is not important.
- Unlike structured data, which represents data as a flat table, semi-structured data can contain n-level hierarchies of nested information.

Web database

- The Web-based database management system is one of the essential parts of DBMS and is used to store web application data.
- Web-based Database management system is used to handle those databases that are having data regarding E-commerce, E-business, Blog, e-mail and other online applications.
- A web database is a wide term for managing data online.
- A web database gives you the ability to build your own databases/data storage without you being a database guru or even a technical person.
- Examples: banks, airline and rental car reservations, university course registration and so on
- The Web is a distributed information system base on hypertext.
- Most Web documents are hypertext documents formatted via HTML
- HTML Documents contain, Text along with font specifications, and other formatting instructions, Hypertext links to other documents , which can be associated with region of the text.

Features

- Save money
- Flexible use
- Technical support
- Access

Mobile database

- Mobile databases are separate from the main database and can easily be transported to various places.
- Even though they are not connected to the main database, they can still communicate with the database to share and exchange data.
- The mobile database includes the following components:
- The main system database that stores all the data and is linked to the mobile database.
- The mobile database that allows users to view information even while on the move. It shares information with the main database.
- The device that uses the mobile database to access data. This device can be a mobile phone, laptop etc.
- A communication link that allows the transfer of data between the mobile database and the main database.

Advantages & Disadvantages

Some advantages of mobile databases are:

- The data in a database can be accessed from anywhere using a mobile database. It provides wireless database access.
- The database systems are synchronized using mobile databases and multiple users can access the data with seamless delivery process.
- Mobile databases require very little support and maintenance.
- The mobile database can be synchronized with multiple devices such as mobiles, computer devices, laptops etc.

Some disadvantages of mobile databases are:

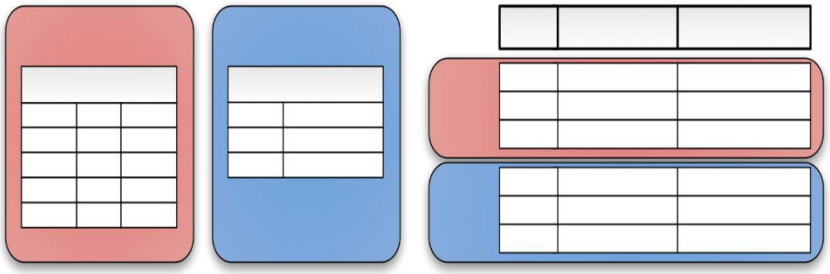
- The mobile data is less secure than data that is stored in a conventional stationary database. This presents a security hazard.
- The mobile unit that houses a mobile database may frequently lose power because of limited battery. This should not lead to loss of data in database.

Sharding Database

- A database shard ("sharding") is the phrase used to describe a horizontal partition in a database or search engine.
- The idea behind Sharding is to split data among multiple machines while ensuring that the data is always accessed from the correct place.
- Since sharding spreads the database across multiple machines, the database programmer specifies explicit sharding rules to determine which machines any piece of data will be stored on.
- Sharding may also be referred to as *horizontal scaling* or *horizontal partitioning*.

Sharding Database

- Sharding is also referred as **horizontal partitioning**. The distinction of **horizontal** vs **vertical** comes from the traditional tabular view of a database. A database can be split vertically — storing different tables & columns in a separate database, or horizontally — storing rows of a same table in multiple database nodes.



Vertical Horizontal

- Vertical partitioning is very domain specific.
- You draw a logical split within your application data, storing them in different databases.
- It is almost always implemented at the **application level** — a piece of code routing reads and writes to a designated database.

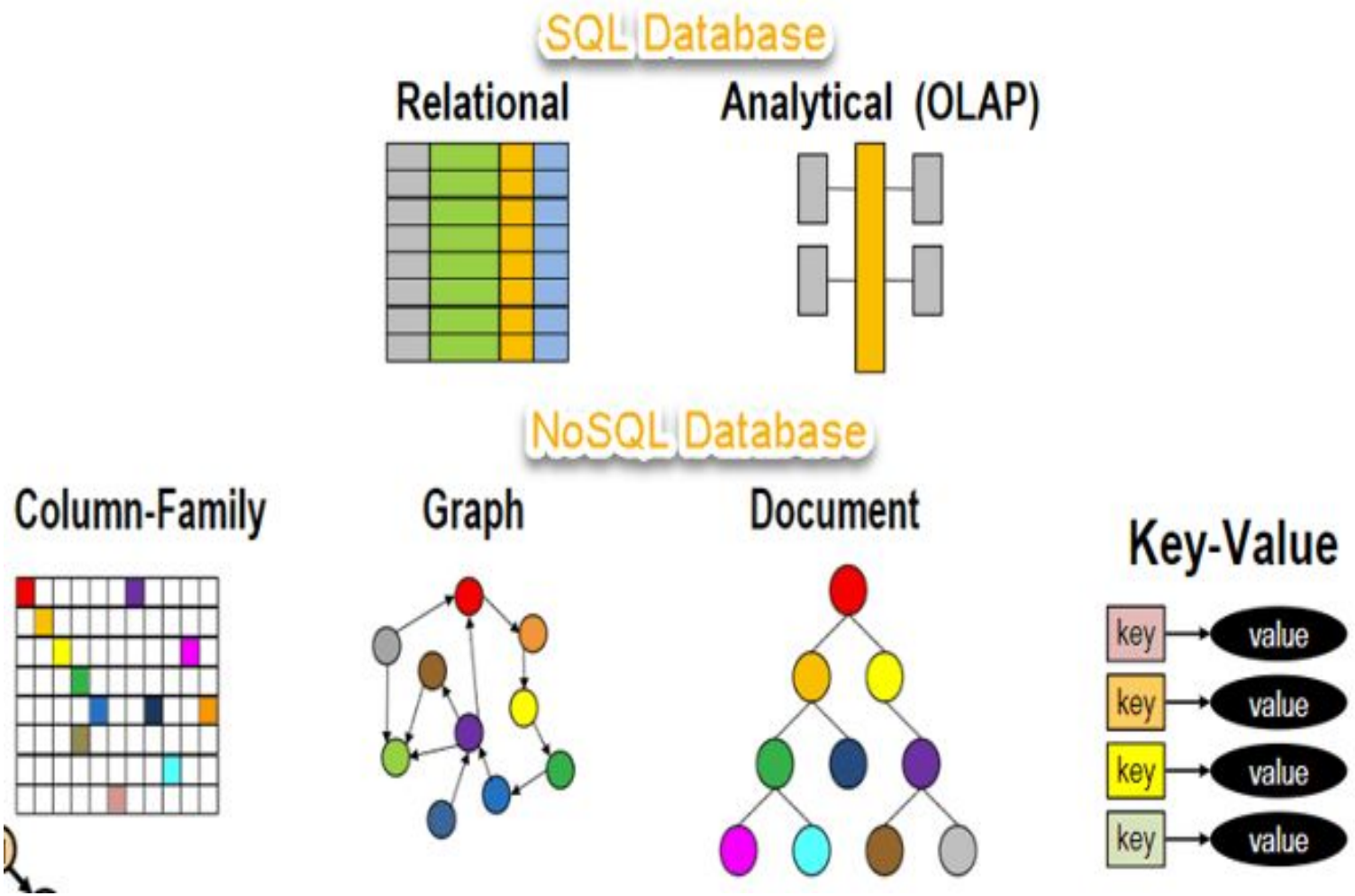
Sharding Database

- In contrast, sharding splits a homogeneous type of data into multiple databases. You can see that such an algorithm is easily generalizable.
- That's why sharding can be implemented at either the application or **database level**.
- In many databases, sharding is a first-class concept, and the database knows how to store and retrieve data within a cluster.
- Almost all modern databases are natively sharded.
- Cassandra, HBase, HDFS, and MongoDB are popular distributed databases.
- Notable examples of non-sharded modern databases are Sqlite, Redis, Memcached, and Zookeeper.

NoSQL

- NoSQL is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale.
- NoSQL database is used for distributed data stores with humongous data storage needs.
- NoSQL is used for Big data and real-time web apps. For example companies like Twitter, Facebook, Google that collect terabytes of user data every single day.
- NoSQL database stands for "Not Only SQL" or "Not SQL."
- Though a better term would NoREL NoSQL caught on.
- Carl Strozz introduced the NoSQL concept in 1998.
- Traditional RDBMS uses SQL syntax to store and retrieve data for further insights.
- Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.

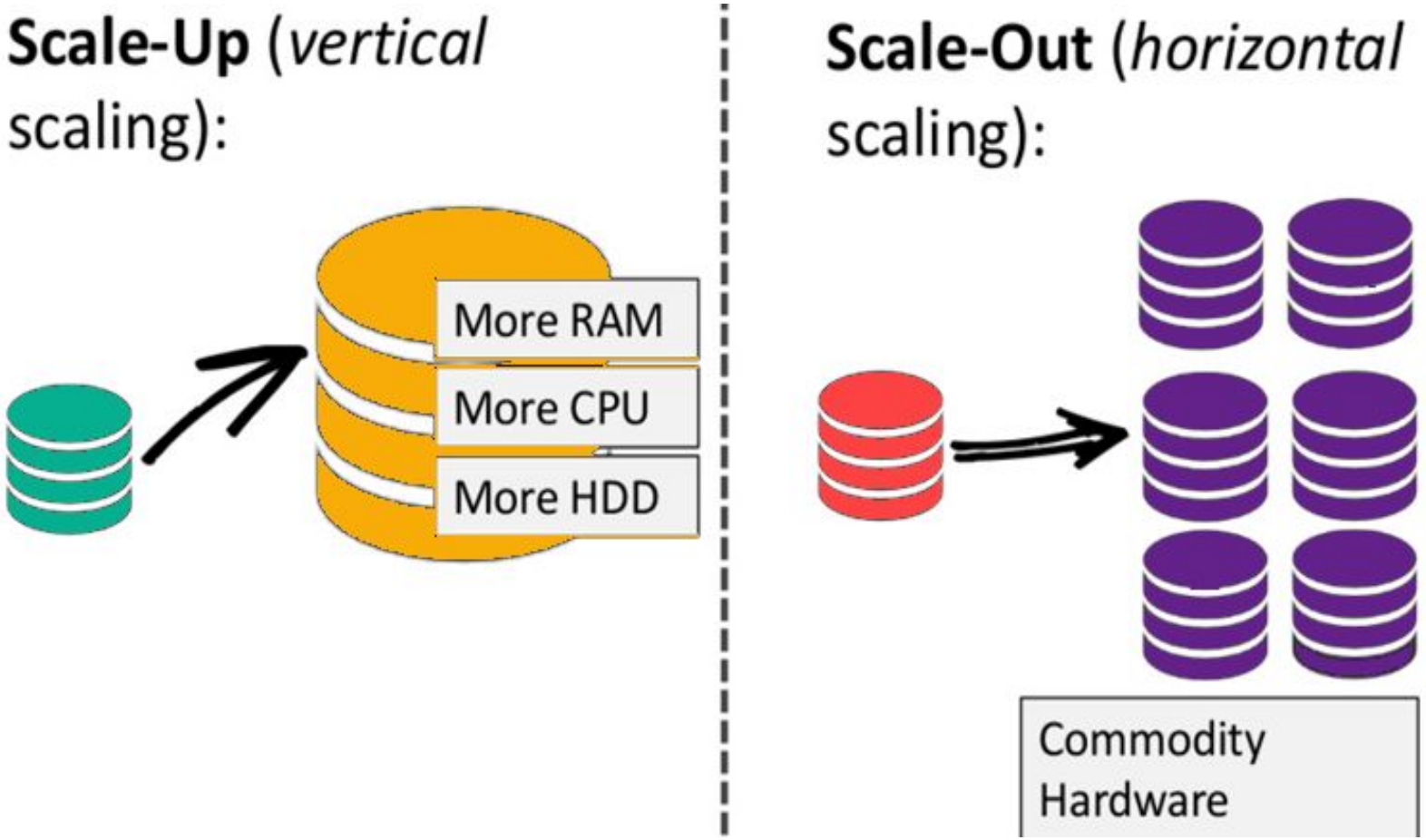
NoSQL


SQL Database
Relational
Analytical (OLAP)
NoSQL Database
Column-Family
Graph
Document
Key-Value
key
value
key
value
key
value
key
value

Why NoSQL

- The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc.
- who deal with huge volumes of data.
- The system response time becomes slow when you use RDBMS for massive volumes of data.
- To resolve this problem, we could "scale up" our systems by upgrading our existing hardware. This process is expensive.
- The alternative for this issue is to distribute database load on multiple hosts whenever the load increases. This method is known as "scaling out."

Why NoSQL


Scale-Up (vertical scaling):
More RAM
More CPU
More HDD
Scale-Out (horizontal scaling):
Commodity Hardware

Brief History of NoSQL Databases

- 1998- Carlo Strozzi use the term NoSQL for his lightweight, open-source relational database
- 2000- Graph database Neo4j is launched
- 2004- Google BigTable is launched
- 2005- CouchDB is launched
- 2007- The research paper on Amazon Dynamo is released
- 2008- Facebooks open sources the Cassandra project
- 2009- The term NoSQL was reintroduced

Features of NoSQL

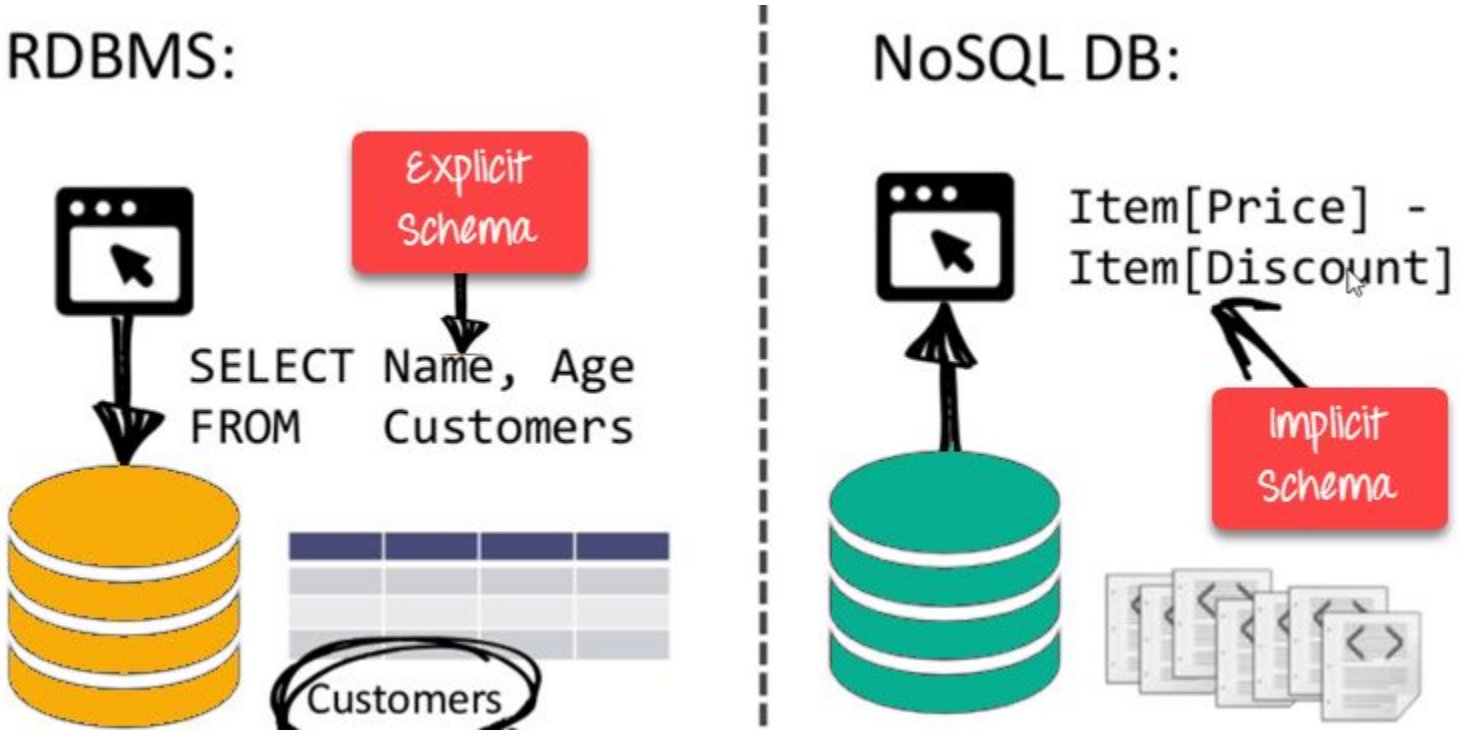
Non-relational

- NoSQL databases never follow the relational model
- Never provide tables with flat fixed-column records
- Work with self-contained aggregates or BLOBs
- Doesn't require object-relational mapping and data normalization
- No complex features like query languages, query planners,
- referential integrity joins, ACID

Schema-free

- NoSQL databases are either schema-free or have relaxed schemas
- Do not require any sort of definition of the schema of the data
- Offers heterogeneous structures of data in the same domain

Features of NoSQL



Features of NoSQL

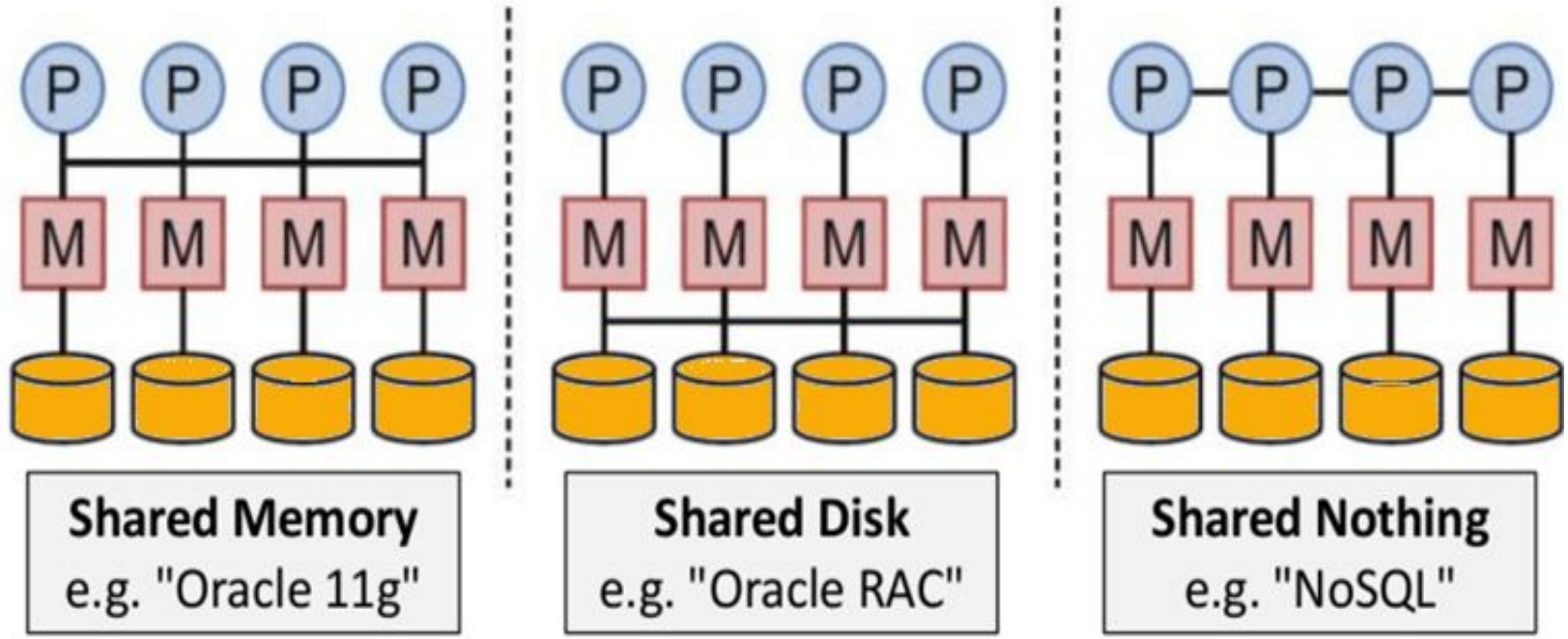
Simple API

- Offers easy to use interfaces for storage and querying data provided
- APIs allow low-level data manipulation & selection methods
- Text-based protocols mostly used with HTTP REST with JSON
- Mostly used no standard based query language
- Web-enabled databases running as internet-facing services

Distributed

- Multiple NoSQL databases can be executed in a distributed fashion
- Offers auto-scaling and fail-over capabilities
- Often ACID concept can be sacrificed for scalability and throughput
- Mostly no synchronous replication between distributed nodes Asynchronous Multi-Master Replication, peer-to-peer, HDFS Replication
- Only providing eventual consistency
- Shared Nothing Architecture. This enables less coordination and higher distribution.

Features of NoSQL



Types of NoSQL Databases

Key Value

Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris

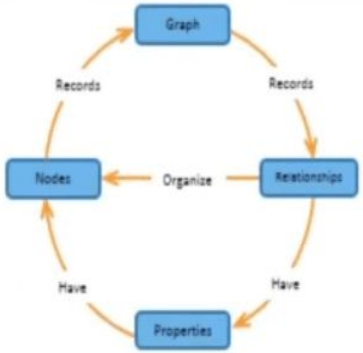
Document-Based

Example:
MongoDB, CouchDB, OrientDB, RavenDB

Column-Based

Example:
BigTable, Cassandra, Hbase, Hypertable

Graph-Based



Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

Key Value Pair Based

- Data is stored in key/value pairs. It is designed in such a way to handle lots of data and heavy load.
- Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB(Binary Large Objects), string, etc.
- For example, a key-value pair may contain a key like "Website" associated with a value like "Guru99".

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

- It is one of the most basic types of NoSQL databases. This kind of NoSQL database is used as a collection, dictionaries, associative arrays, etc. Key value stores help the developer to store schema-less data. They work best for shopping cart contents.
- Redis, Dynamo, Riak are some examples of key-value store DataBases. They are all based on Amazon's Dynamo paper.

Column-based

- Column-oriented databases work on columns and are based on BigTable paper by Google.
- 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 NoSQL database
- They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.
- Column-based NoSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs,
- HBase, Cassandra, HBase, Hypertable are examples of column based database.

Document-Oriented

- Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document.
- The document is stored in JSON or XML formats. The value is understood by the DB and can be queried.

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
}
```

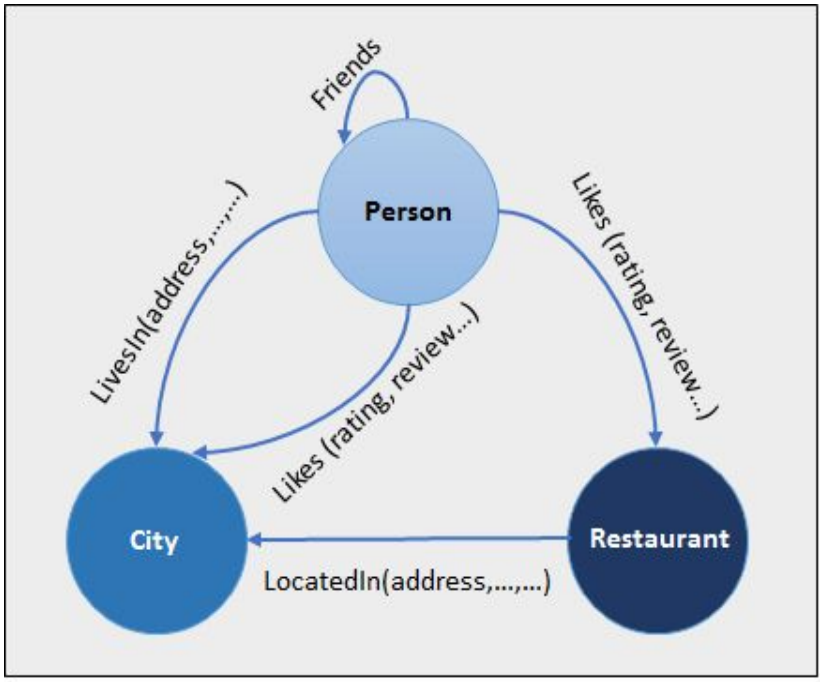
Document 3

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

Graph-Based

- A graph type database stores entities as well the relations amongst those entities.
- The entity is stored as a node with the relationship as edges.
- An edge gives a relationship between nodes.
- Every node and edge has a unique identifier.

- Compared to a relational database where tables are loosely connected, a Graph database is a multi-relational in nature.
- Traversing relationship is fast as they are already captured into the DB, and there is no need to calculate them.
- Graph base database mostly used for social networks, logistics, spatial data.
- Neo4J, Infinite Graph, OrientDB, FlockDB are some popular graph-based databases.



CAP Theorem

- CAP theorem is also called brewer's theorem.
- It states that is impossible for a distributed data store to offer more than two out of three guarantees
- Consistency
- Availability
- Partition Tolerance

Consistency:

- The data should remain consistent even after the execution of an operation.
- This means once data is written, any future read request should contain that data.
- For example, after updating the order status, all the clients should be able to see the same data.

CAP Theorem

Availability:

- The database should always be available and responsive.
- It should not have any downtime.

Partition Tolerance:

- Partition Tolerance means that the system should continue to function even if the communication among the servers is not stable.
- For example, the servers can be partitioned into multiple groups which may not communicate with each other. Here, if part of the database is unavailable, other parts are always unaffected.

BASE

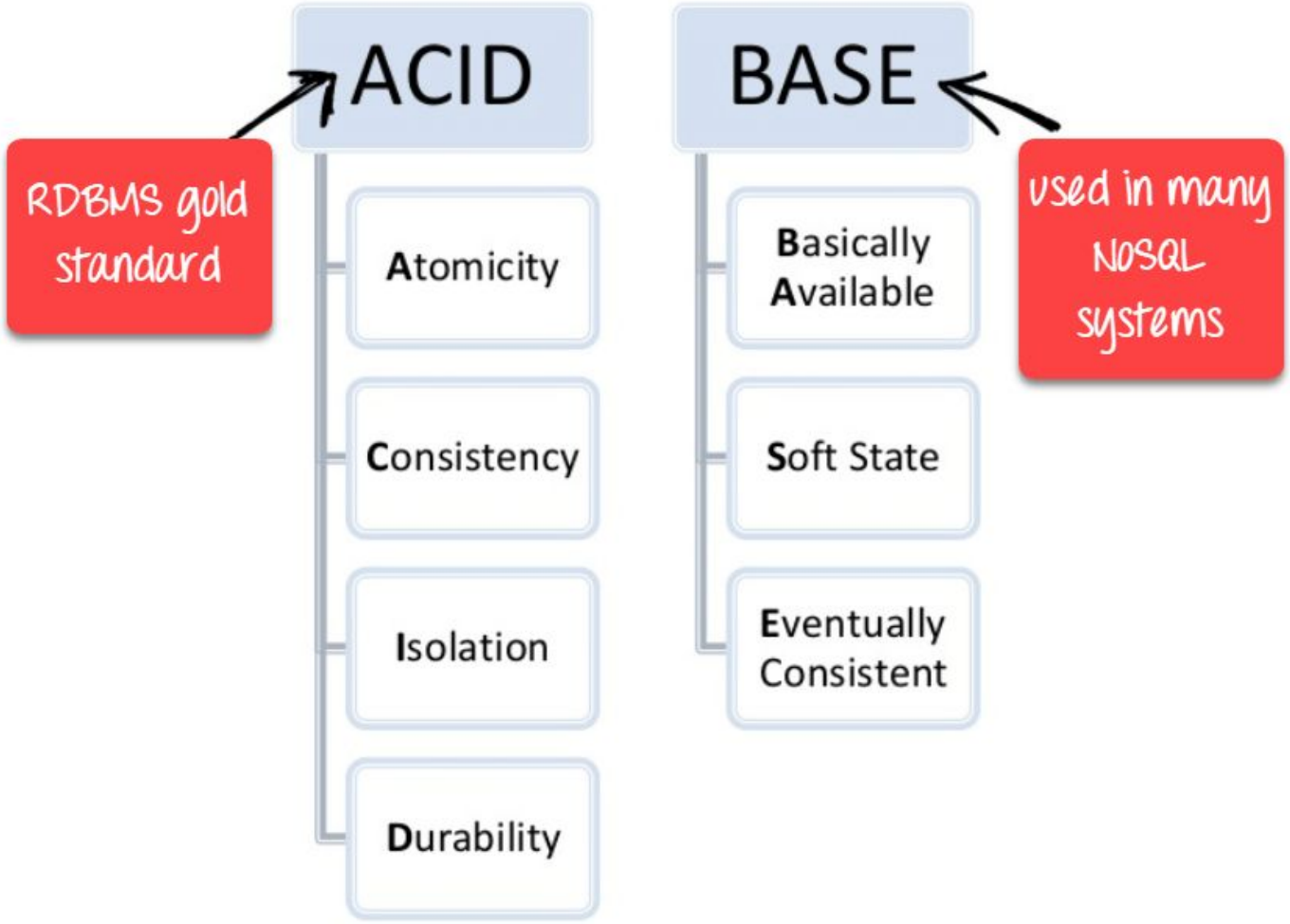
Basically Available, Soft state, Eventual consistency

- Basically, available means DB is available all the time as per CAP theorem
- Soft state means even without an input; the system state may change
- Eventual consistency means that the system will become consistent over time

- **Eventual Consistency**
- The term "eventual consistency" means to have copies of data on multiple machines to get high availability and scalability. Thus, changes made to any data item on one machine has to 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.

Unit 5

ADVANCE DATABASE AND NOSQL DATABASES



Advantages of NoSQL

- Can be used as Primary or Analytic Data Source
- Big Data Capability
- No Single Point of Failure
- Easy Replication
- No Need for Separate Caching Layer
- It provides fast performance and horizontal scalability.
- Can handle structured, semi-structured, and unstructured data with equal effect
- Object-oriented programming which is easy to use and flexible
- NoSQL databases don't need a dedicated high-performance server
- Support Key Developer Languages and Platforms

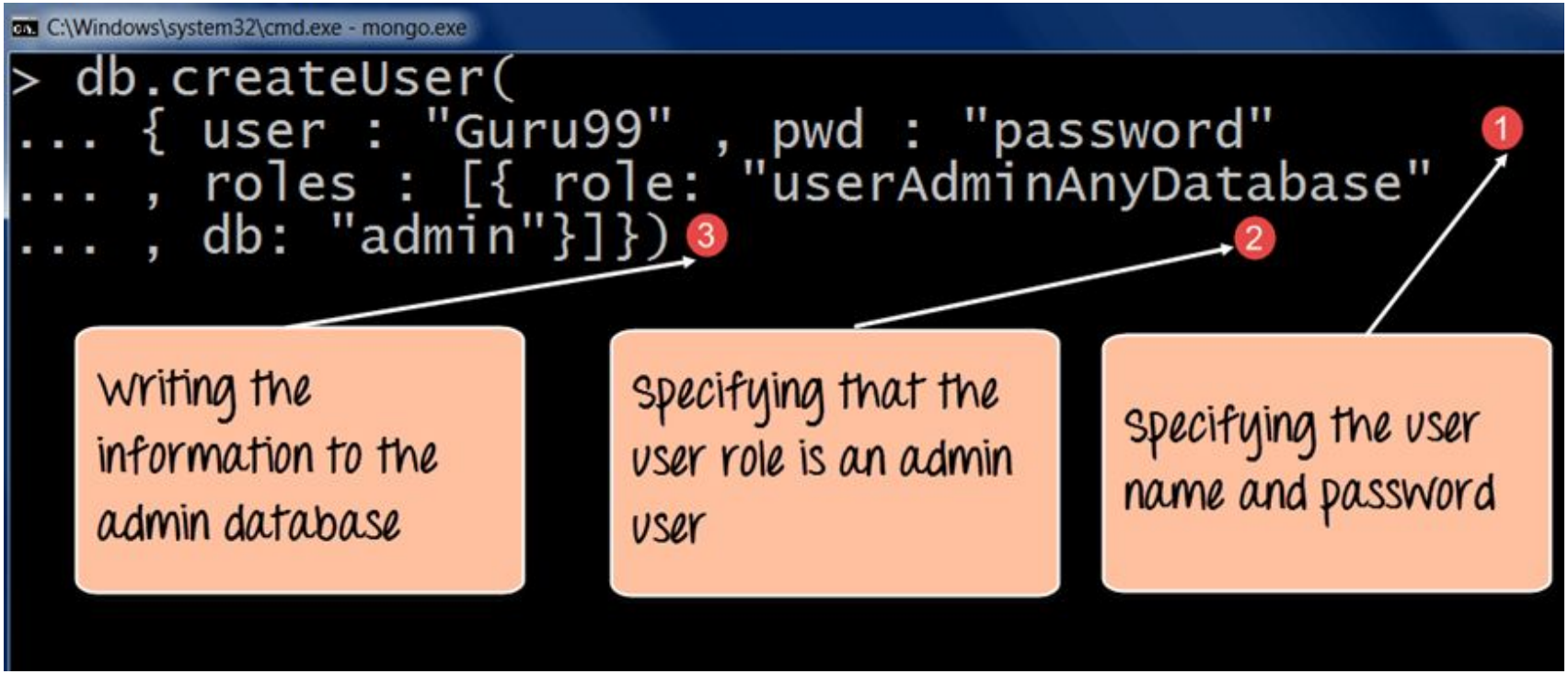
Advantages of NoSQL

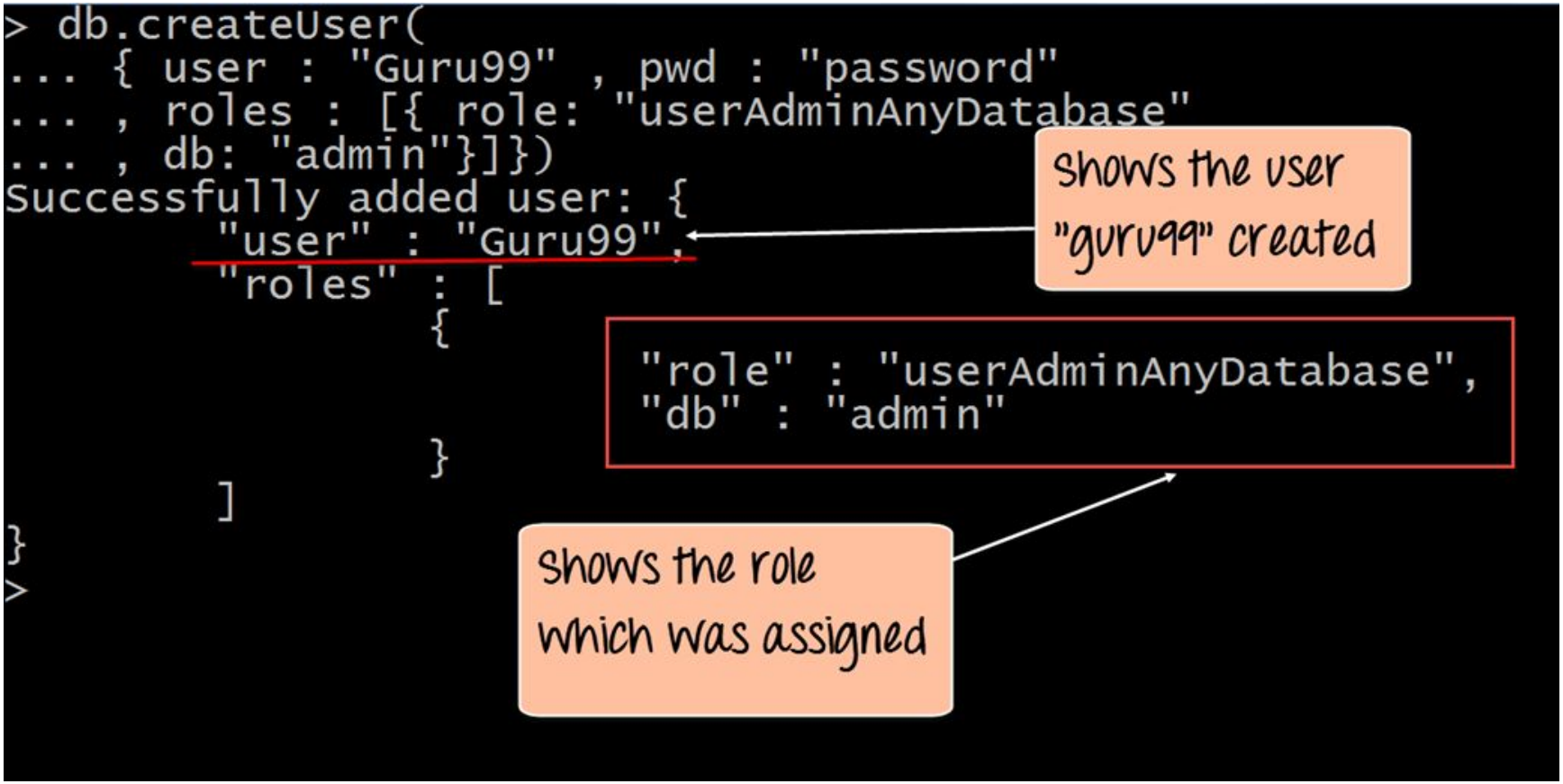
- Simple to implement than using RDBMS
- It can serve as the primary data source for online applications.
- Handles big data which manages data velocity, variety, volume, and complexity
- Excels at distributed database and multi-data center operations
- Eliminates the need for a specific caching layer to store data
- Offers a flexible schema design which can easily be altered without downtime or service disruption

Disadvantages of NoSQL

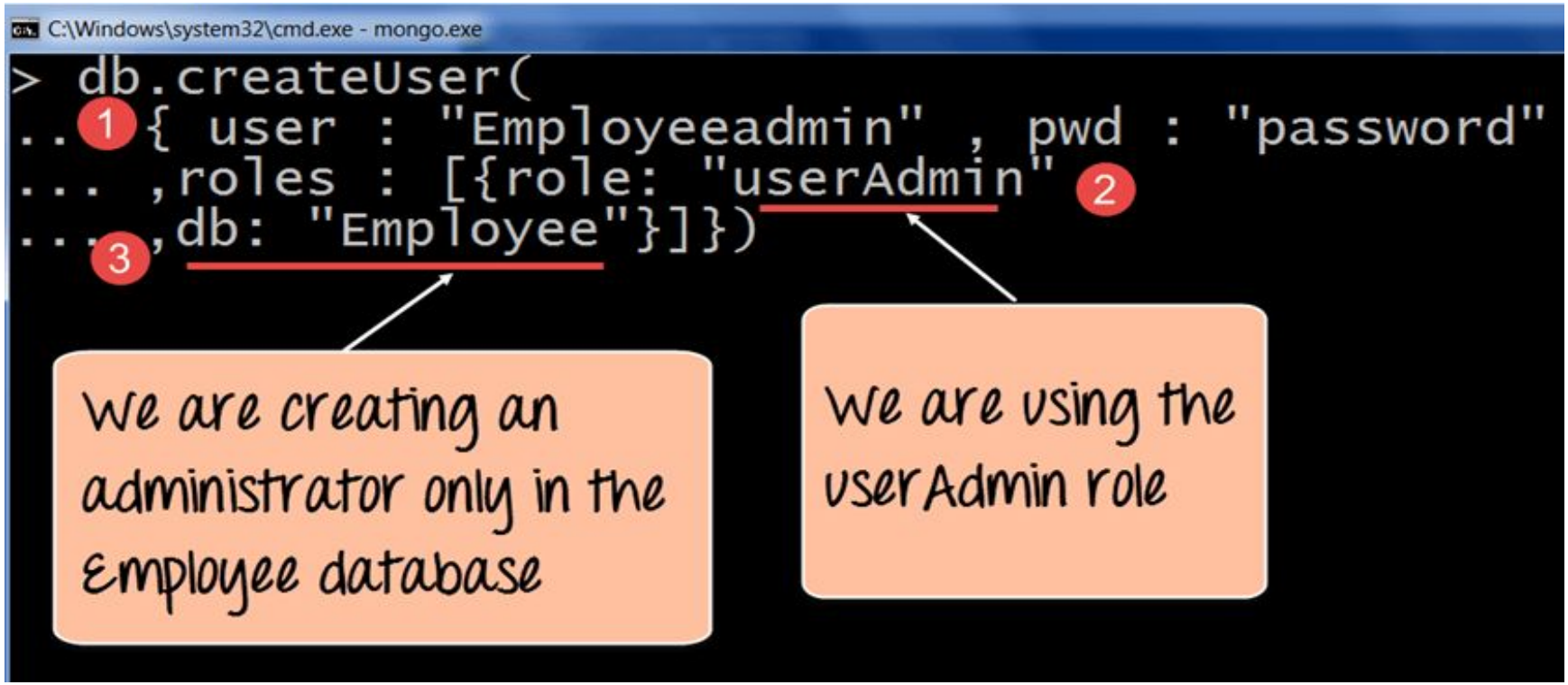
- No standardization rules
- Limited query capabilities
- RDBMS databases and tools are comparatively mature
- It does not offer any traditional database capabilities, like consistency when multiple transactions are performed simultaneously.
- When the volume of data increases it is difficult to maintain unique values as keys become difficult
- Doesn't work as well with relational data
- The learning curve is stiff for new developers
- Open source options so not so popular for enterprises.

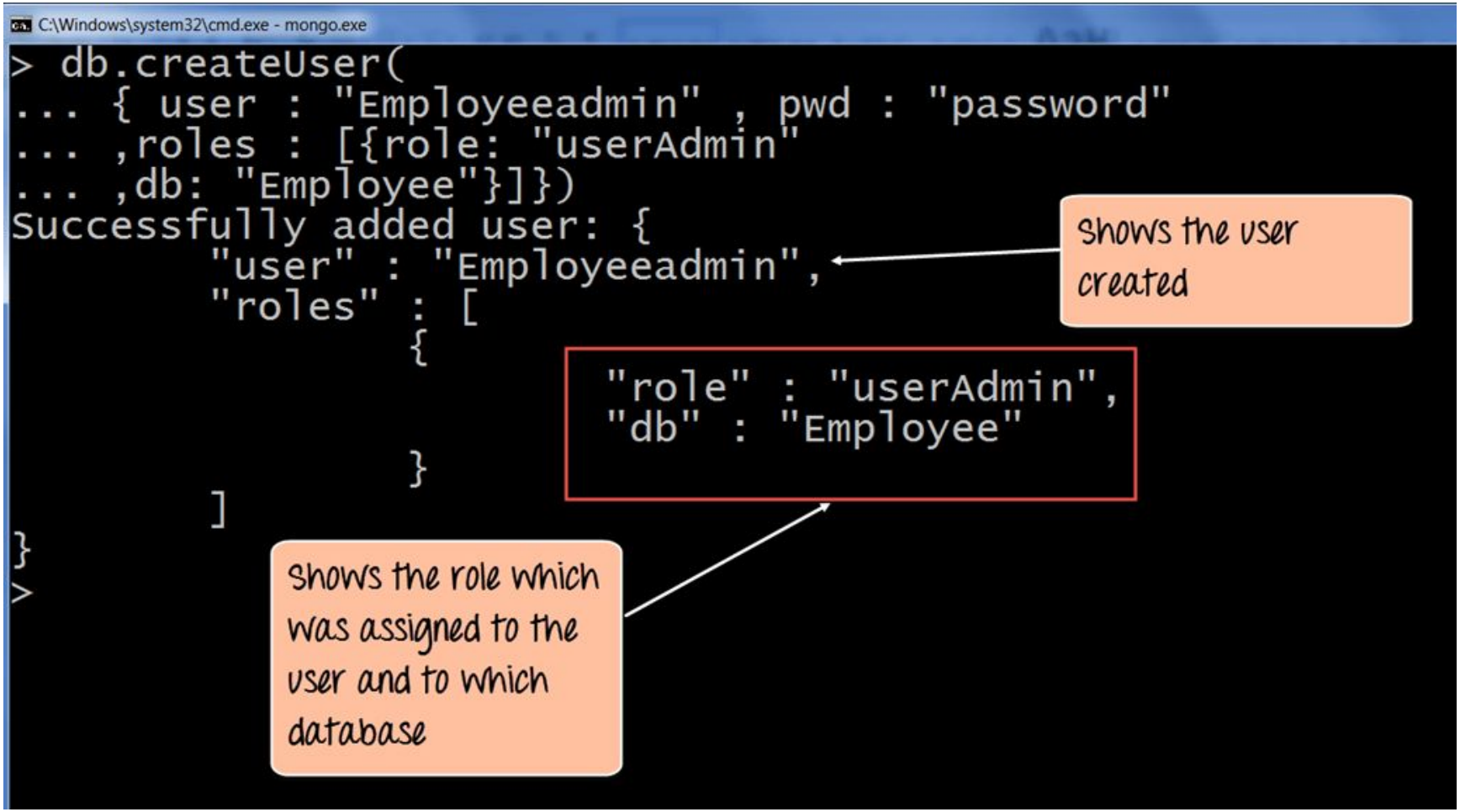
MongoDB Create Administrator User



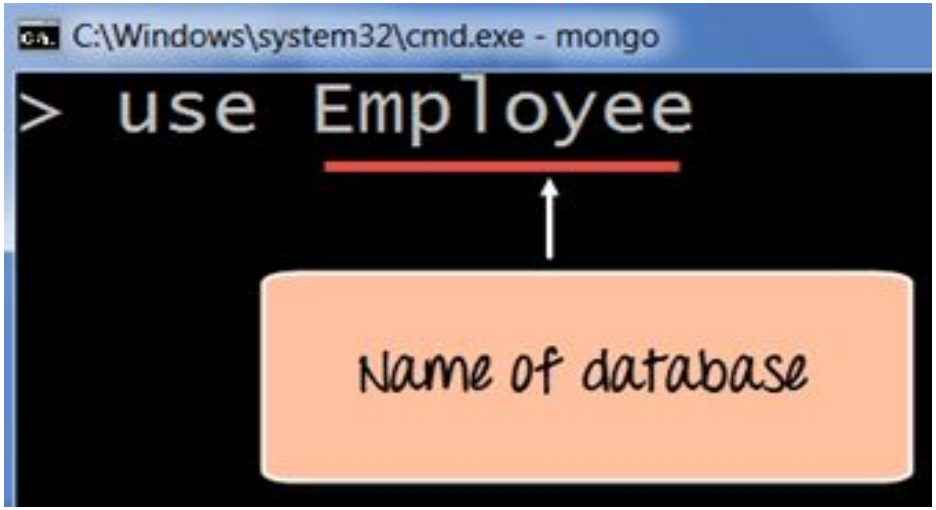

> db.createUser(
... { user : "Guru99" , pwd : "password"
... , roles : [{ role: "userAdminAnyDatabase"
... , db: "admin"}]})
Successfully added user: {
"user" : "Guru99",
"roles" : [
{
"role" : "userAdminAnyDatabase",
"db" : "admin"
}
]
}
Shows the user "guru99" created
Shows the role which was assigned

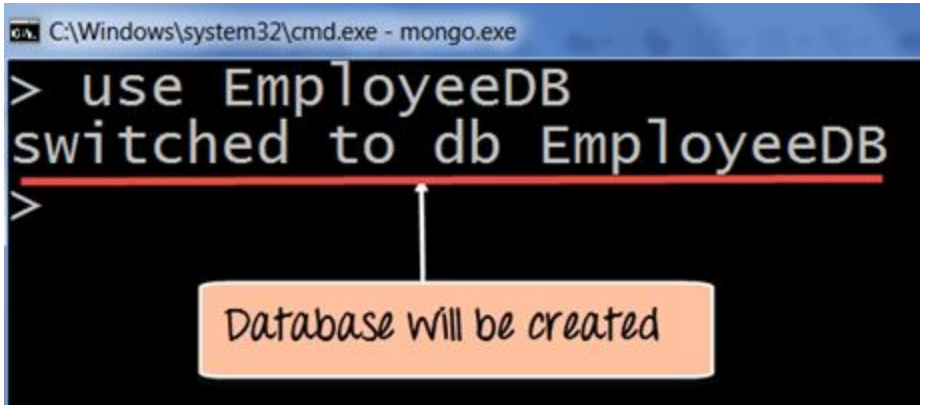
MongoDB Create User for Single Database




C:\Windows\system32\cmd.exe - mongo.exe
> db.createUser(
... { user : "Employeeadmin" , pwd : "password"
... ,roles : [{role: "userAdmin"
... ,db: "Employee"}]})
Successfully added user: {
"user" : "Employeeadmin",
"roles" : [
{
"role" : "userAdmin",
"db" : "Employee"
}
]
}
Shows the user created
Shows the role which was assigned to the user and to which database

Creating a database using "use" command





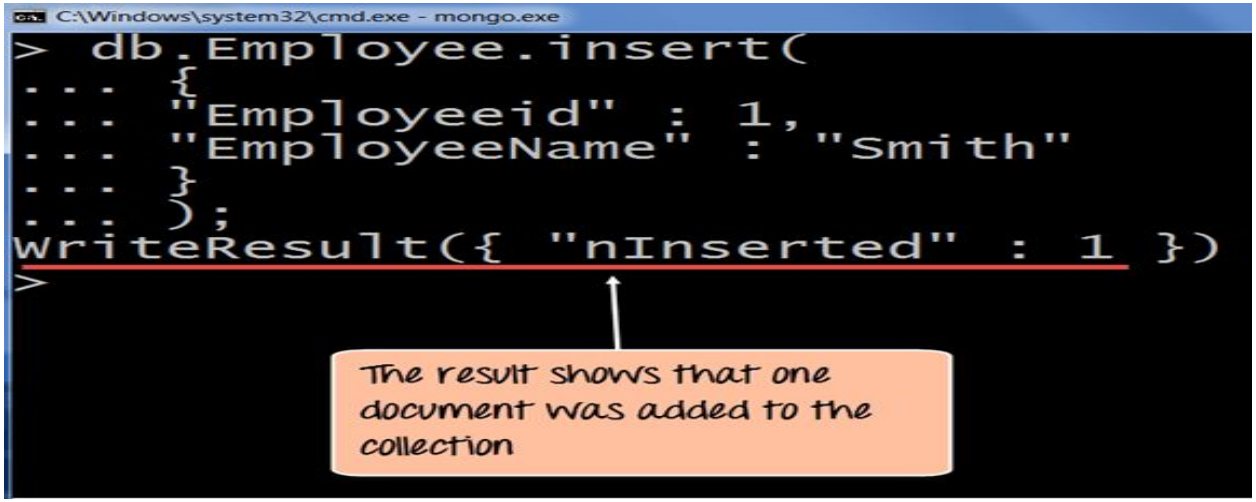
•Creating a Collection/Table using insert()

```
1 db.Employee.insert
  (
        {
            "Employeeid" : 1,
            "EmployeeName" : "Smith"
        }
  )
```

2

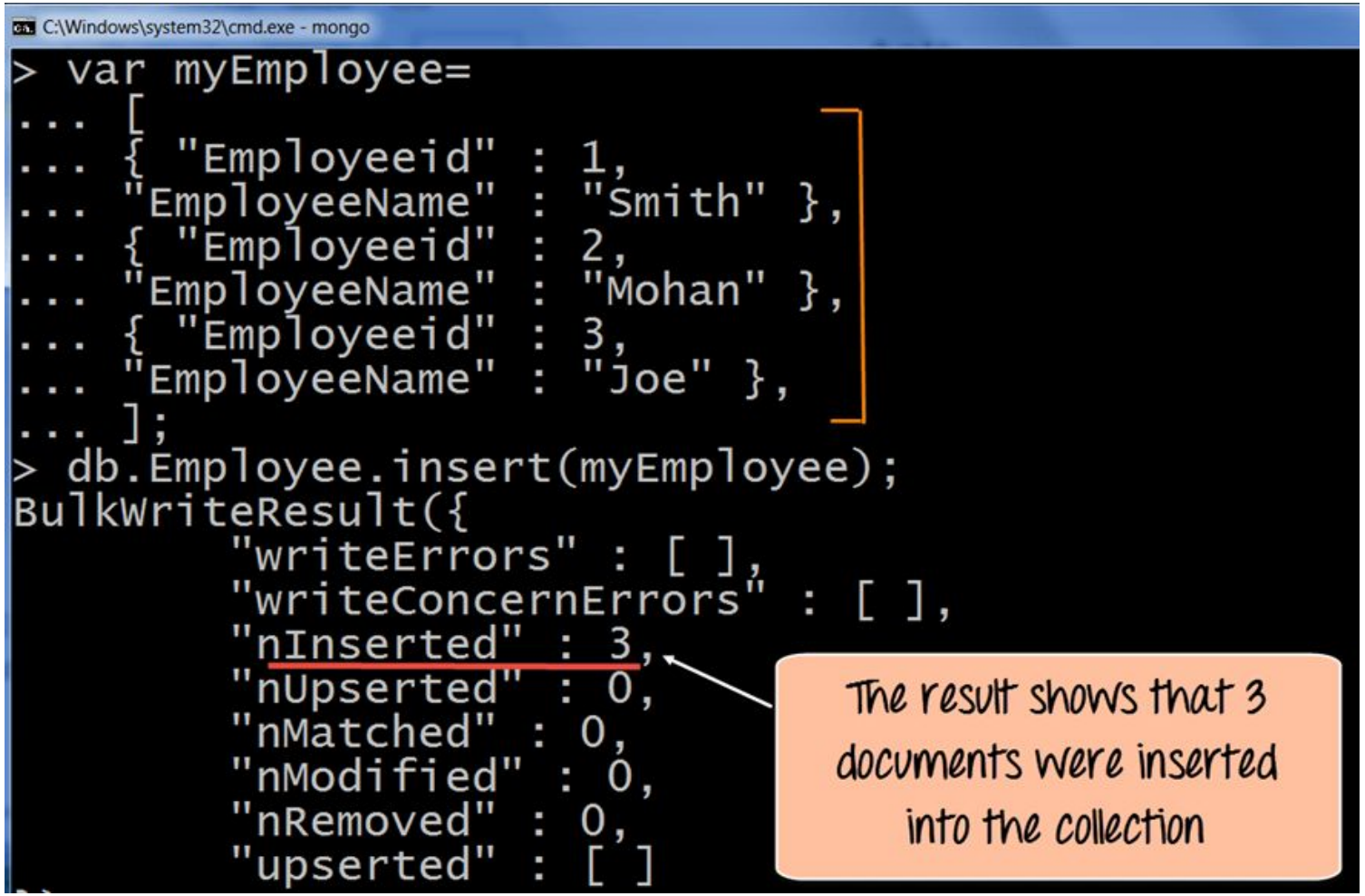
"insert" command to insert the document into the collection

2 key value pairs being added



The result shows that one document was added to the collection

Add MongoDB Array using insert()



Printing in JSON format

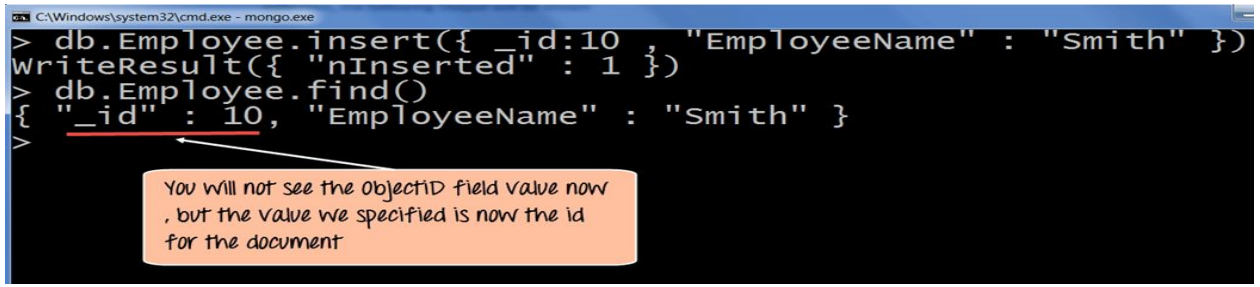
```
> db.Employee.find().forEach(printjson);
{
        "_id" : ObjectId("563479cc8a8a4246bd27d784"),
        "Employeeid" : 1,
        "EmployeeName" : "Smith"
}
{
        "_id" : ObjectId("563479d48a8a4246bd27d785"),
        "Employeeid" : 2,
        "EmployeeName" : "Mohan"
}
{
        "_id" : ObjectId("563479df8a8a4246bd27d786"),
        "Employeeid" : 3,
        "EmployeeName" : "Joe"
}
```

You will now see each document printed in json style

What is Primary Key in MongoDB?

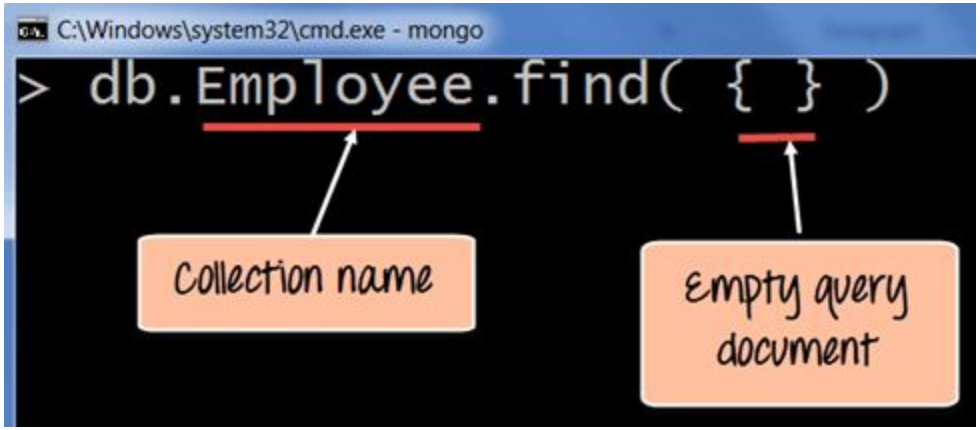
```
> db.Employee.find().forEach(printjson);
{
        "_id" : ObjectId("563479cc8a8a4246bd27d784"),
        "Employeeid" : 1,
        "EmployeeName" : "Smith"
}
{
        "_id" : ObjectId("563479d48a8a4246bd27d785"),
        "Employeeid" : 2,
        "EmployeeName" : "Mohan"
}
{
        "_id" : ObjectId("563479df8a8a4246bd27d786"),
        "Employeeid" : 3,
        "EmployeeName" : "Joe"
}
>
```

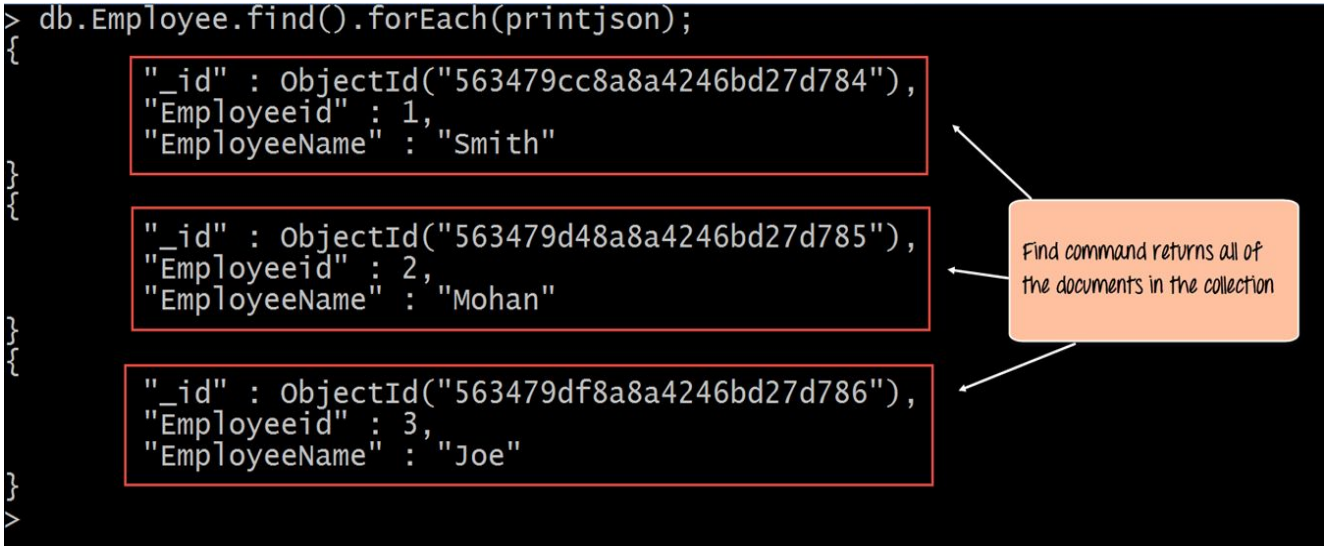
Every row has a unique object id



You will not see the ObjectID field value now , but the value we specified is now the id for the document

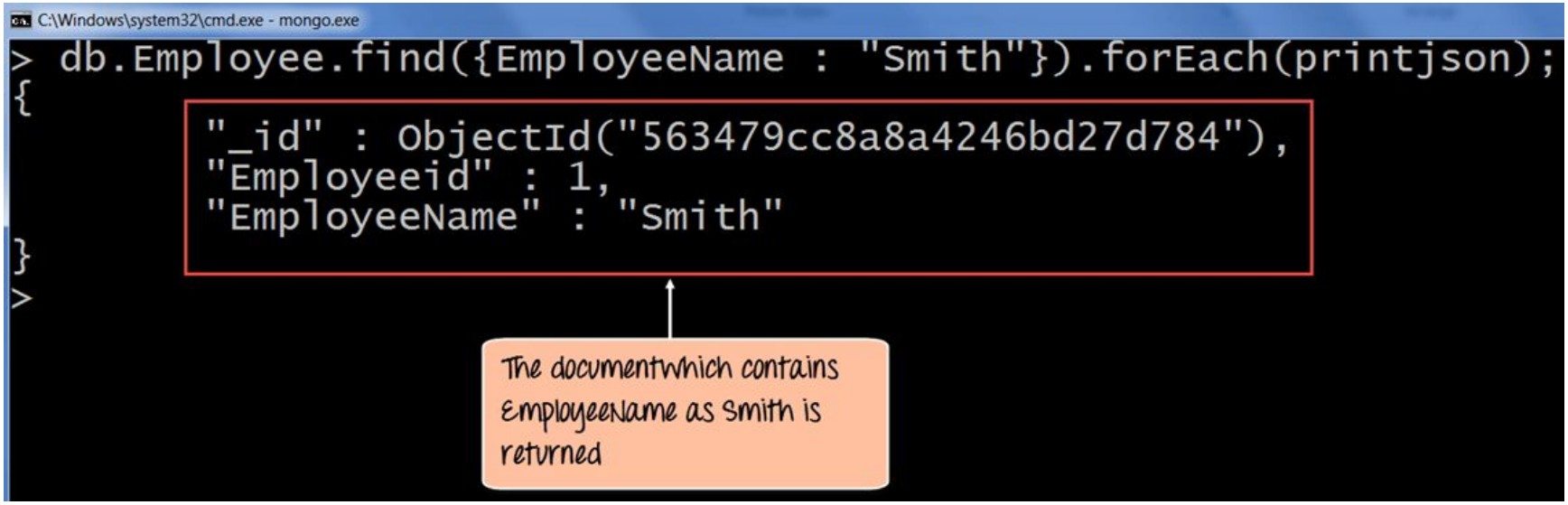
MongoDB Query Document using find()

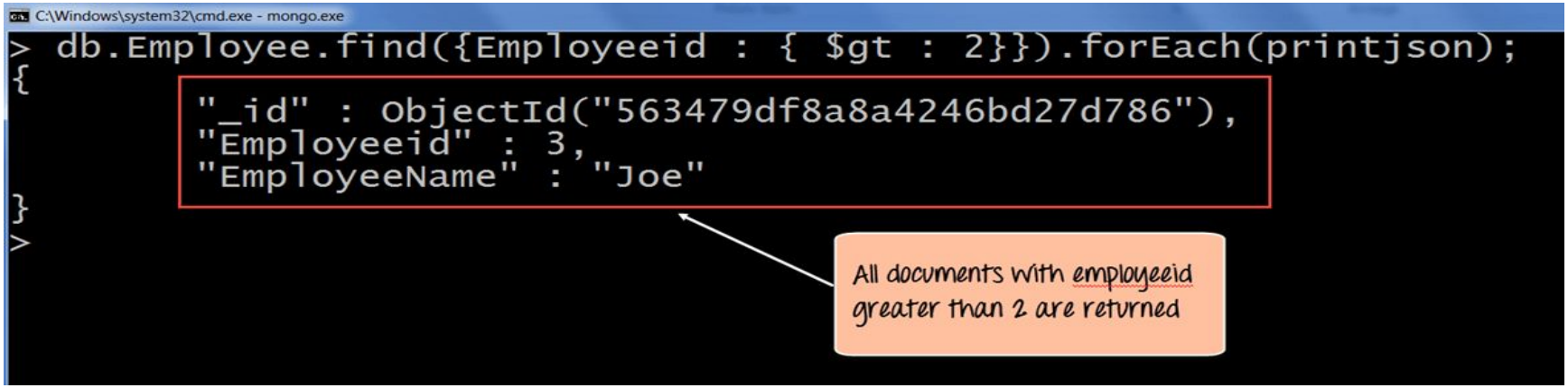




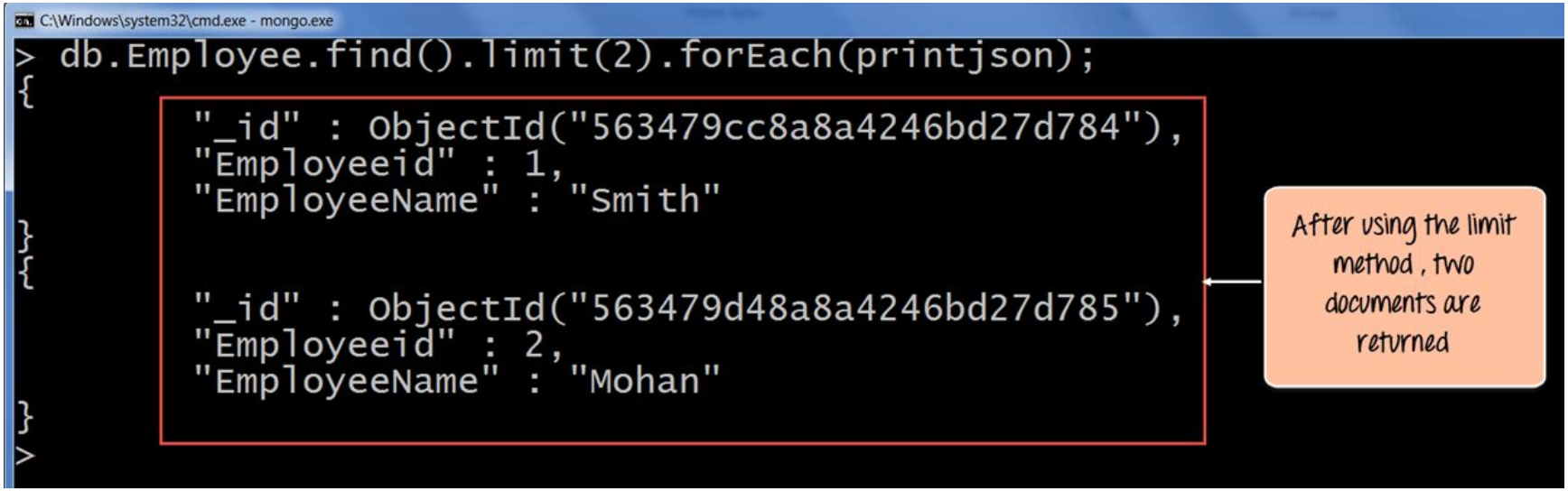
Unit 5

ADVANCE DATABASE AND NOSQL DATABASES

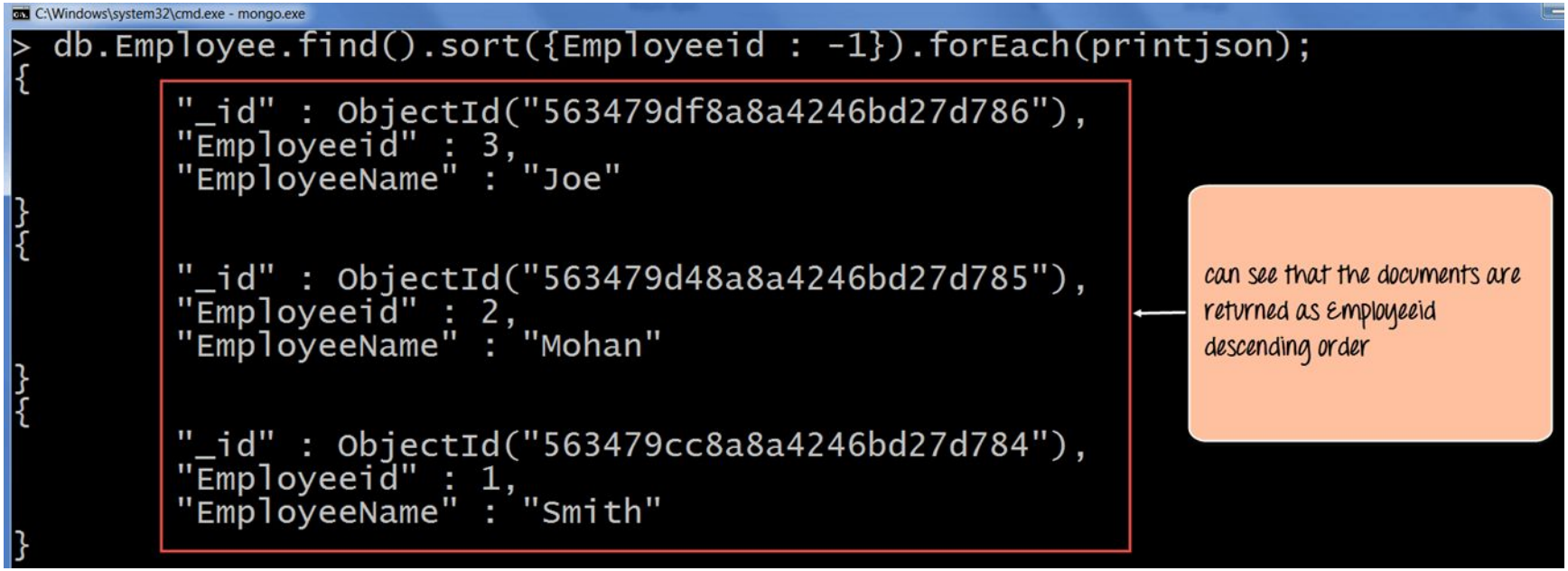




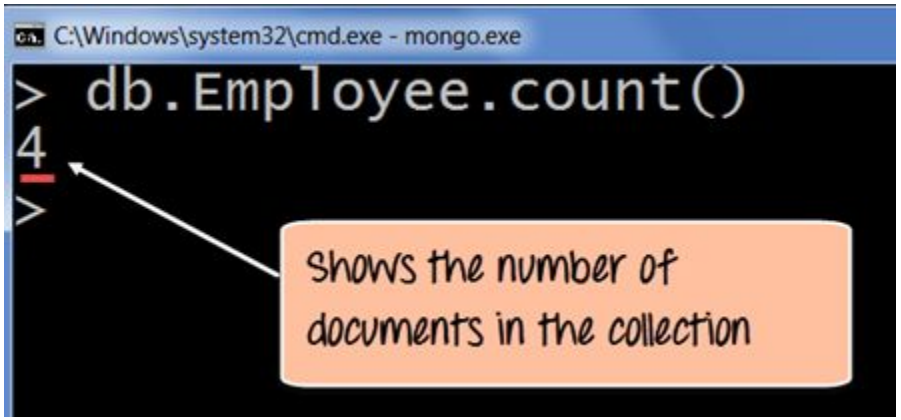
MongoDB Limit Query Results

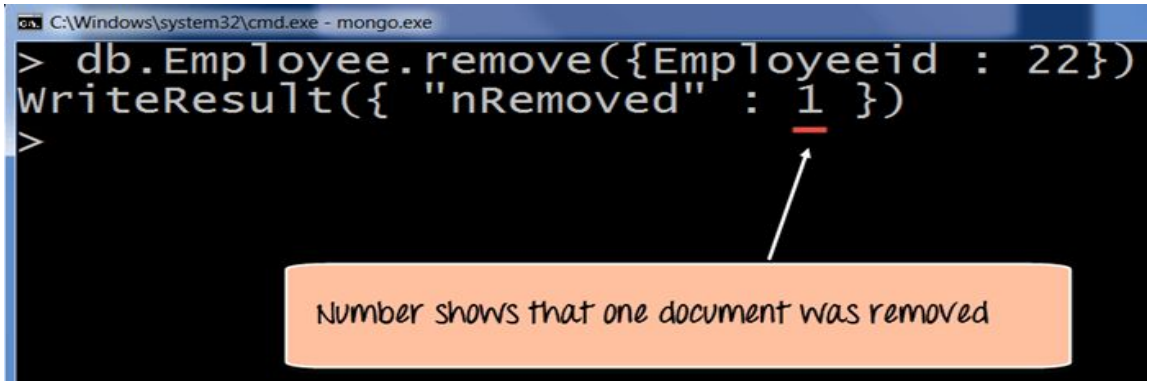


MongoDB Sort by Descending Order

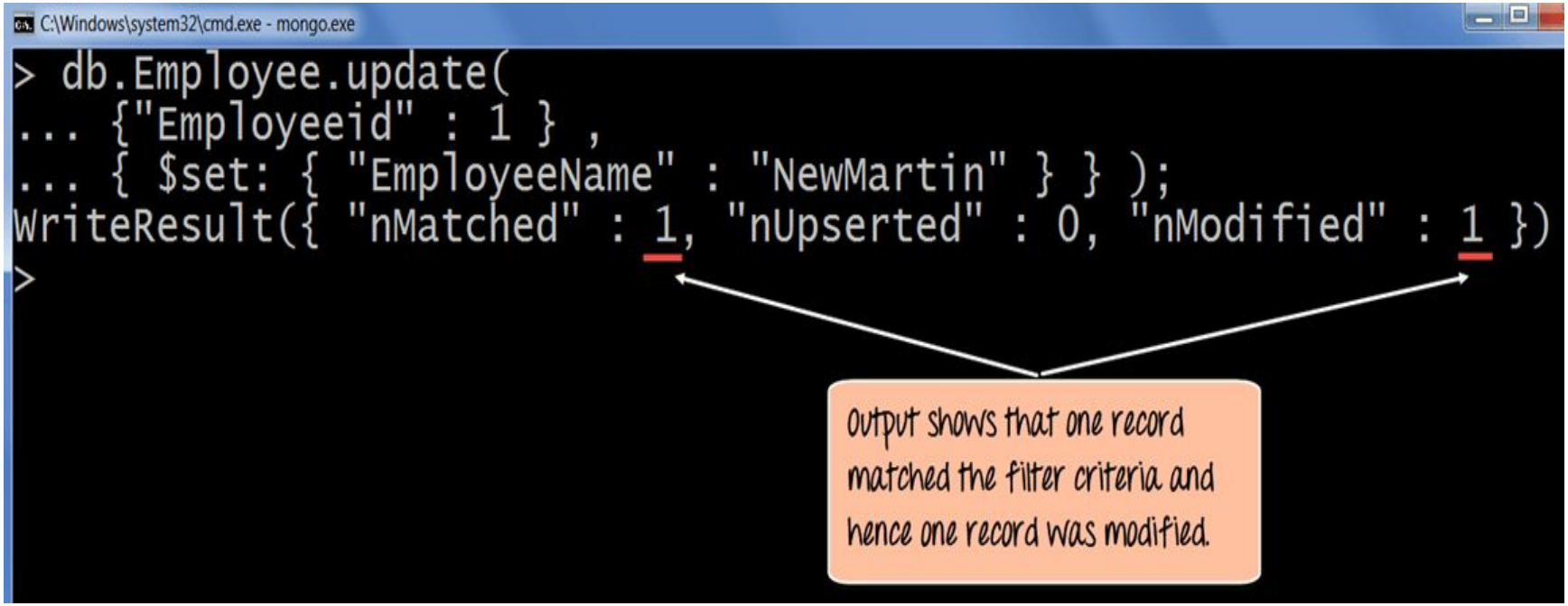


MongoDB Count() & Remove() Functions

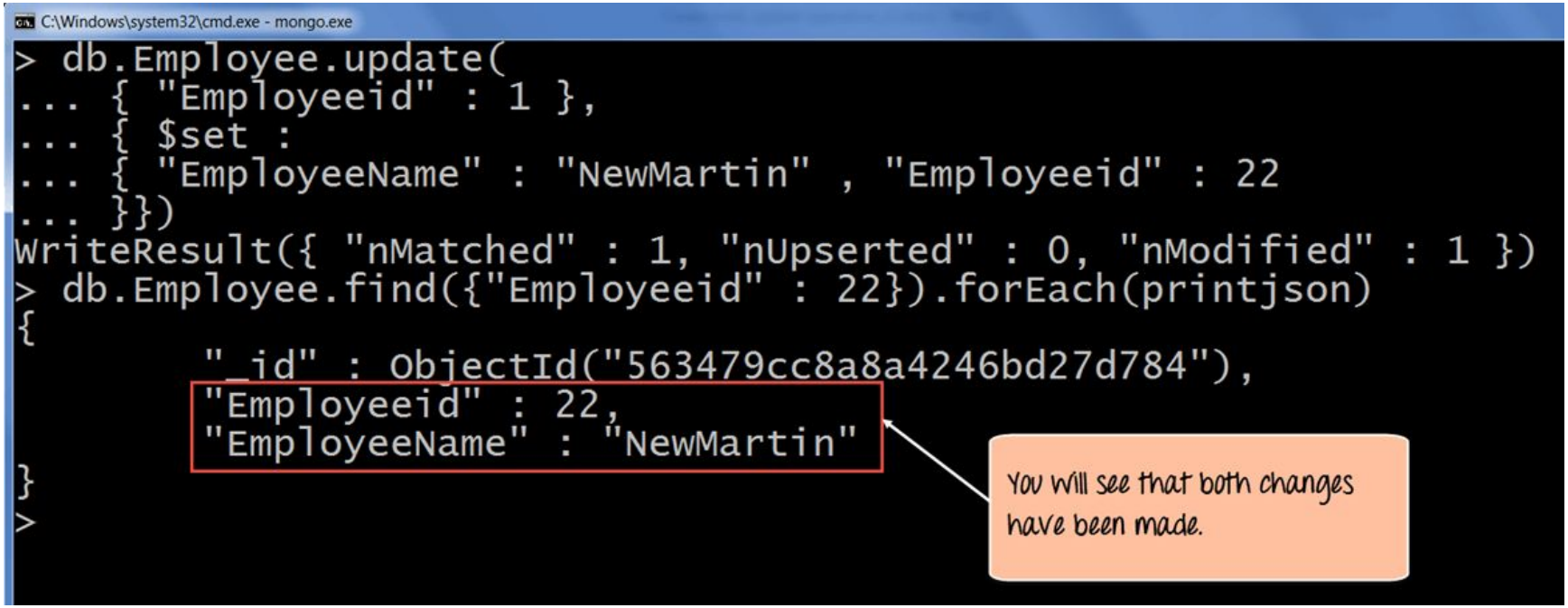


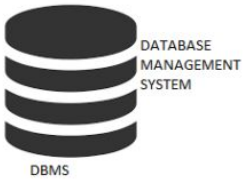


MongoDB Update() Document



Updating Multiple Values





Unit 5

ADVANCE DATABASE AND NOSQL DATABASES