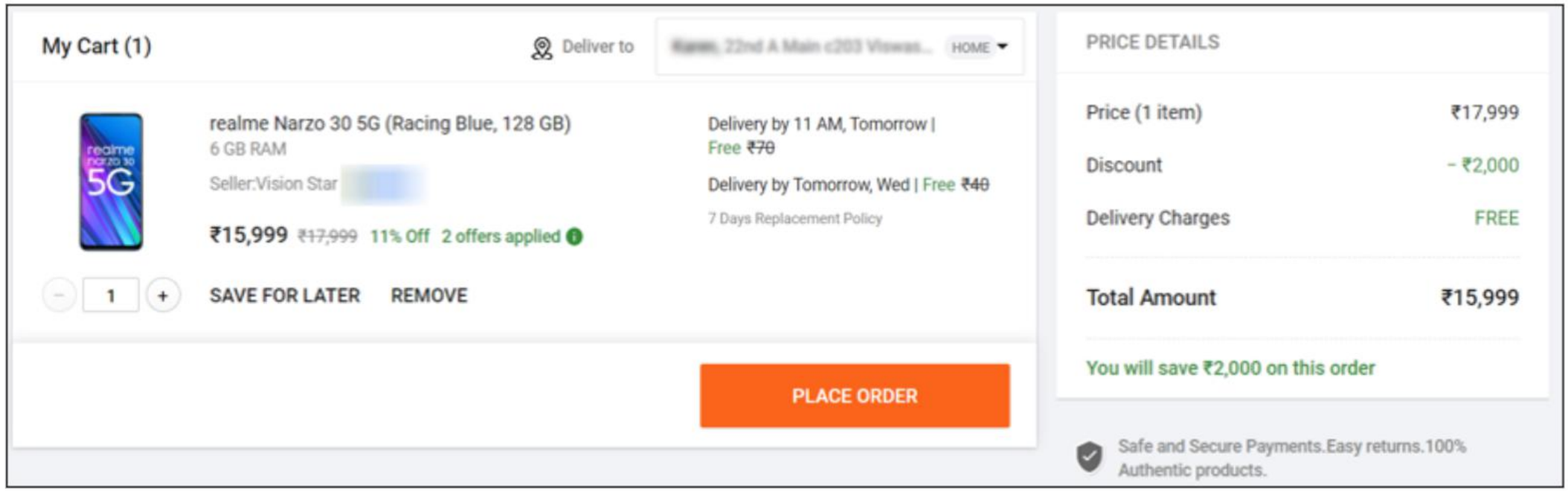
Slide Note

What is the structure of the data in the shopping cart?

This shopping cart contains data from multiple tables.

An Online Shopping Cart



What data does an online shopping cart hold?

Where does this data come from?

Slide Note

Data comes from multiple tables:

Customer table

Shipment table

Seller table

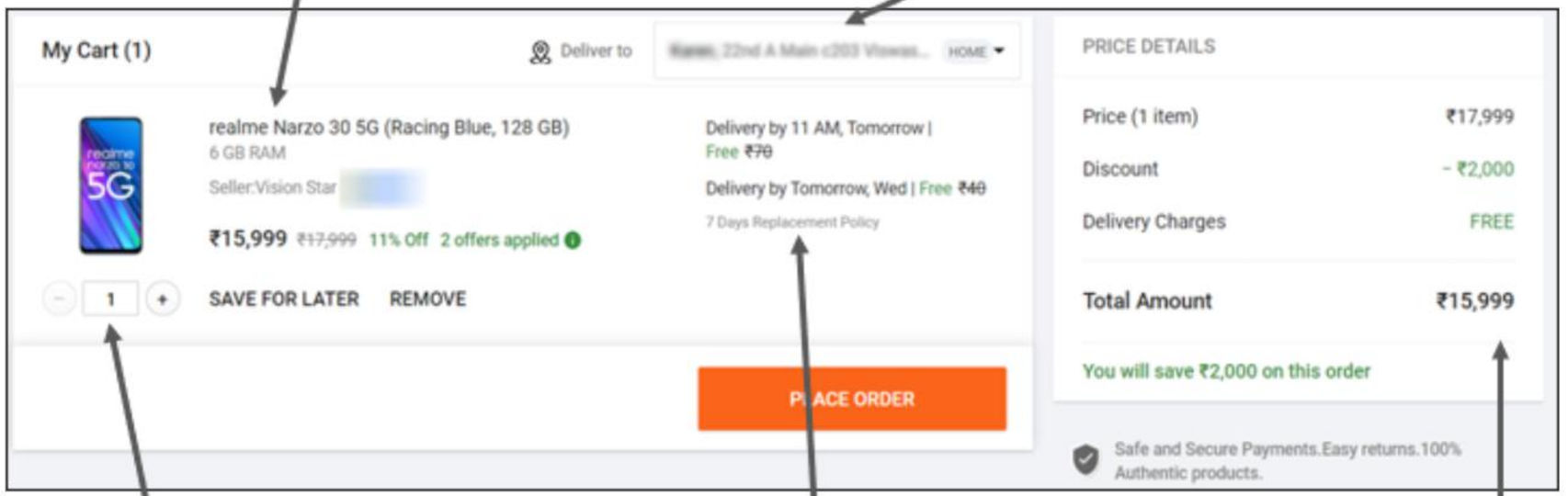
Product table

Product Price table

Online Shopping Cart - Explained

Seller details come from the seller table.

Shipping details come from the shipment table.



Specifications about the product (realMe 5G) come from the product table.

The data about delivery come from the delivery details table.

Price details come from the product price table.

Slide Note

Invoice

Tax Invoice

Sold By: [redacted] **Retail Private Limited ,**
Ship-from Address: [redacted]
GSTIN - 06AAICA4872D1ZS

Invoice Number # FAC7XD2100402820

Order ID: OD118819382028122000
Order Date: 05-06-2020
Invoice Date: 06-06-2020
PAN: AAICA[redacted]
CIN: U52100DL201[redacted]00

Bill To
[redacted]
22nd A Main e203 Viswas
Phone: xxxxxxxxxx

Ship To
[redacted]
22nd A Main e203 Viswas
Phone: xxxxxxxxxx

**Keep this invoice and manufacturer box for warranty purposes.*

Total items: 1

Product	Title	Qty	Gross Amount ₹	Discount ₹	Taxable Value ₹	IGST ₹	Total ₹
Without Call Function FSN: SMWFQZ8NXG6YBM8E HSN/SAC: 85176290	**Realme Watch** Warranty: 1 Year Manufacturer Warranty **IGST:** 18.000 %	1	3999.00	0.00	3388.98	610.02	3999.00
	Total	**1**	**3999.00**	**0.00**	**3388.98**	**610.02**	**3999.00**

Grand Total **₹ 3999.00**

[redacted]

What data is displayed in an online-generated Invoice?

Where does the data come from?

Slide Note

Data comes from multiple tables:

Customer table

Orders table

Seller table

Product Table

Invoice - Explained

Retailer details come from the seller table.

Order details come from the orders table.

Tax Invoice

Sold By: [redacted] **Retail Private Limited ,**

Ship-from Address: [redacted]

GSTIN - 06AAICA4872D1ZS

Invoice Number # FAC7XD2100402820

Order ID: OD118819382028122000

Order Date: 05-06-2020

Invoice Date: 06-06-2020

PAN: AAICA[redacted]

CIN: U52100DL201[redacted]00

Bill To

[redacted]

22nd A Main e202 Viswas

Phone: xxxxxxxxxx

Ship To

[redacted]

22nd A Main e202 Viswas

Phone: xxxxxxxxxx

**Keep this invoice and manufacturer box for warranty purposes.*

Total items: 1

Product	Title	Qty	Gross Amount ₹	Discount ₹	Taxable Value ₹	IGST ₹	Total ₹
Without Call Function FSN: SMWFQZ8NXG6YBM8E HSN/SAC: 85176290	**Realme Watch** Warranty: 1 Year Manufacturer Warranty **IGST:** 18.000 %	1	3999.00	0.00	3388.98	610.02	3999.00
	Total	**1**	**3999.00**	**0.00**	**3388.98**	**610.02**	**3999.00**

Specifications about the product come from the product table.

Grand Total ₹ 3999.00

[redacted] Retail Private Limited

Customer data come from the customer table.

Slide Note

Think and Tell

- If the data comes from multiple tables, how is it retrieved?
- Are joins an efficient means of retrieving data?
- Does writing multiple joins decrease the performance of an application?

Slide Note

Use a NoSQL Database (MongoDB) to Manage Semi-Structured and Unstructured Data

Slide Note

Learning Objectives

- Explain Structured and Unstructured Data
- Understand Model Data in a NoSQL
- Explain Document Database
- Structuring Data in MongoDB
- Use MongoDB for Database Operations

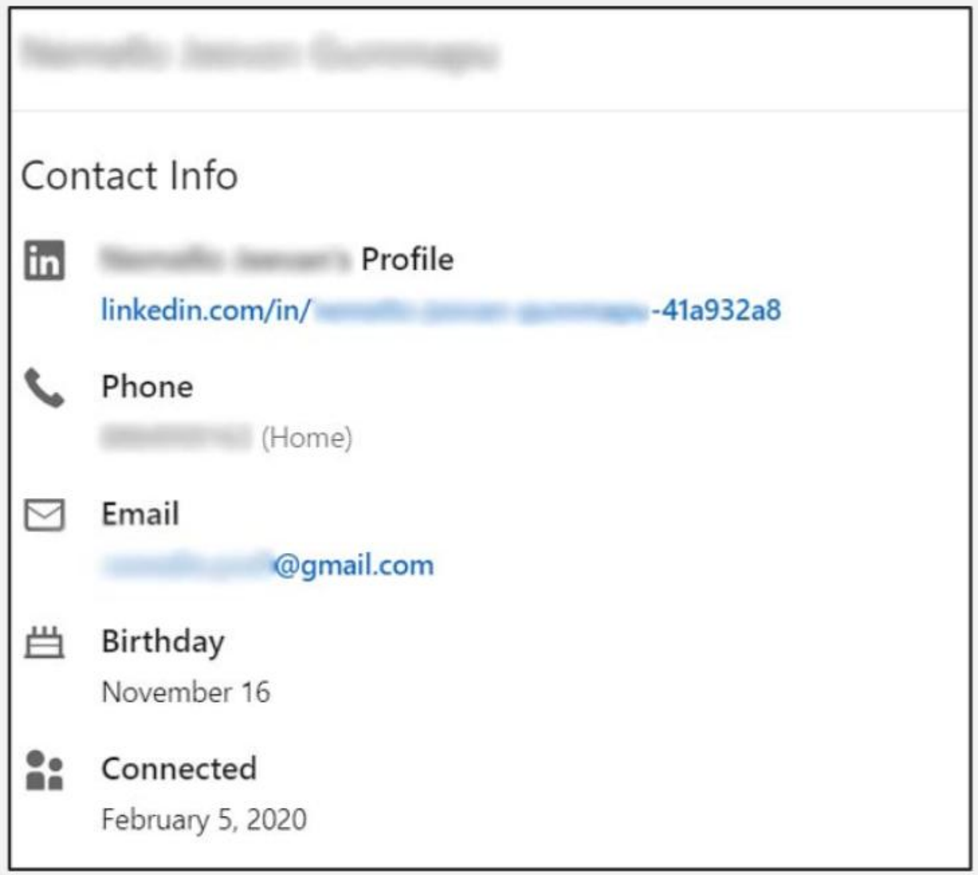
Slide Note

Explain Structured and Unstructured Data

Slide Note

Structured Data

- Structured data consists of clearly defined data types whose pattern makes them easily searchable.
- Structured data is highly specific and is stored in a predefined format.
- The contact information page of a website has a defined structure and can be stored in an RDBMS.
- The RDBMS provides a schema, or predefined format, to store the contact information.



Slide Note

Unstructured data – “everything else” – is comprised of data that is usually not as easily searchable, including formats like audio, video, and social media postings.

Organizations used to store only some key transactional data and a few basic things about their customers. Today, however, organizations can no longer cherry-pick a few key pieces of data. They need to store just about everything.

As the cost of storage drops (even for SSDs), organizations are doing just that. There is also an expectation from customers, partners, and regulators that the organization should store everything in a usable format that will benefit them as well.

The growing amount of unstructured data presents a problem for relational databases. The rows and columns of a relational database are ideal for storing sets of values. Still, most information is composed of much more than that.

Consider something like a person’s medical record. It is incredibly heterogeneous. It includes values (name, date of birth), relationships (to family members or care providers, to symptoms and medications), geospatial data (addresses), metadata (provenance, security attributes), images (CAT scan), and free text (doctors’ notes, transcripts).

Unstructured Data

- A webpage can have different types of data displayed on a single page, like:
  - Geospatial data
  - Aggregate data
  - Audio data
  - Video data
  - Images

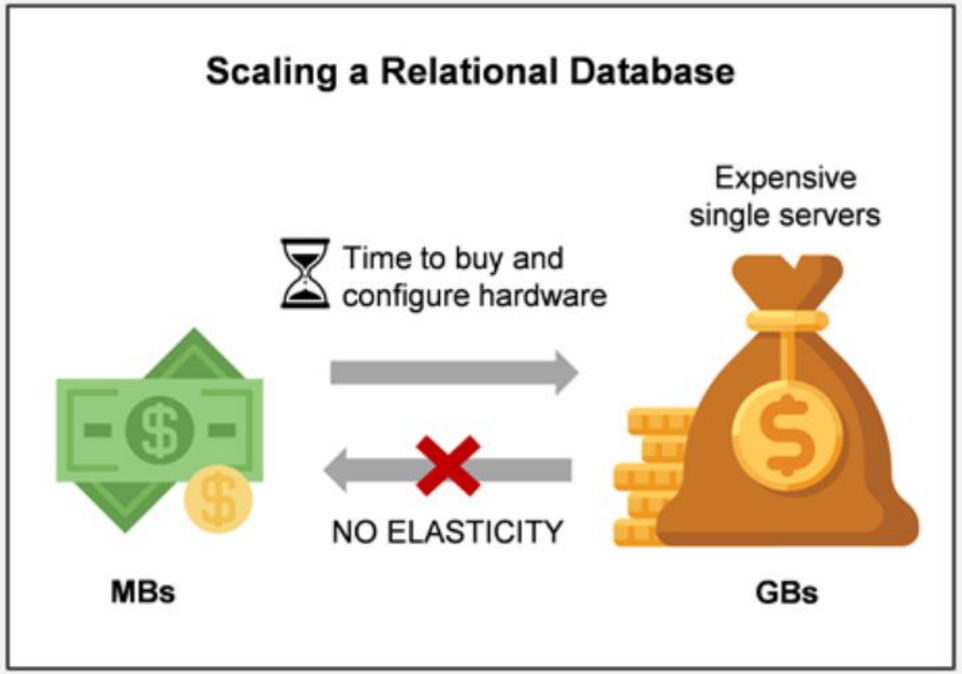
Slide Note

Relational databases are designed to run on a single server in order to maintain the integrity of the table mappings and avoid the problems of distributed computing.

With this design, if a system needs to scale, customers must buy bigger, more complex, and more expensive proprietary hardware with more processing power, memory, and storage. Upgrades are also a challenge, as the organization must go through a lengthy acquisition process, and then often take the system offline to actually make the change.

It's expensive to scale out. In some cases, can even be impossible. At certain levels of data volume, relational database systems simply do not feasibly scale out. This is usually due to the high terabytes to petabyte size.

In addition to this, with a high volume of data coming in at real time, relational databases are typically not designed to handle such a problem.



Scaling an RDBMS

- RDBMS can be used to store collosal amounts of data on the web.
- They are designed to run on a single server to maintain integrity.
- To store all the data, we need to scale the RDBMS system.

Is this an effective solution?

Slide Note

When people use the term “NoSQL database”, they typically use it to refer to any non-relational database. Some say the term “NoSQL” stands for “non-SQL” while others say it stands for “not only SQL.” Either way, most agree that NoSQL databases are databases that store data in a format other than relational tables.

A common misconception is that NoSQL databases or non-relational databases don’t store relationship data well. NoSQL databases can store relationship data—they just store it differently than relational databases do. In fact, when compared with SQL databases, many find modeling relationship data in NoSQL databases to be easier than in SQL databases, because related data doesn’t have to be split between tables.

NoSQL data models allow related data to be nested within a single data structure.

NoSQL databases emerged in the late 2000s as the cost of storage dramatically decreased. Gone were the days of needing to create a complex, difficult-to-manage data model simply for the purposes of reducing data duplication. Developers (rather than storage) were becoming the primary cost of software development, so NoSQL databases optimized for developer productivity.

Model Data in an NoSQL (Not Only SQL)

Slide Note

Document databases store data in documents similar to JSON (JavaScript Object Notation) objects. Each document contains pairs of fields and values. The values can typically be of various types, including strings, numbers, Booleans, arrays, or objects. Their structures align with the objects developers work on within the code. Because of their various field value types and powerful query languages, document databases are great for various use cases. They can be used as a general-purpose database. They can horizontally scale out to accommodate large data volumes. According to DB engines, MongoDB is consistently ranked as the world's most popular NoSQL database and is an example of a document database.

Key-value databases are simpler databases where each item contains keys and values. A value can typically only be retrieved by referencing its key, so it is typically simple to learn how to query for a specific key-value pair. Key-value databases are great for use cases where you need to store large amounts of data but don't need to perform complex queries to retrieve it. Common use cases include storing user preferences or caching. Redis and DynamoDB are popular key-value databases.

Wide-column stores store data in tables, rows and dynamic columns. Wide-column stores provide much flexibility over relational databases because each row is not required to have the same columns. Many consider wide-column stores to be two-dimensional key-value databases. Wide-column stores are great for when you need to store large amounts of data, and you can predict what your query patterns will be. Wide-column stores commonly store the Internet of Things and user profile data. Cassandra and HBase are two of the most popular wide-column stores.

Graph databases store data in nodes and edges. Nodes typically store information about people, places and things, while edges store information about the relationships between the nodes. Graph databases excel in use cases where you must traverse relationships to look for patterns such as social networks, fraud detection, and recommendation engines. Neo4j and JanusGraph are examples of graph databases.

NoSQL Database

- main types are:
- docNoSQL databases are non-tabular; they store data differently than relational tables.
- NoSQL databases come in a variety of types based on their data model.
- The four ument
  - key-value
  - wide-column
  - graph
- They provide flexible schemas and scale easily with large amounts of data and high user loads.
  Check document databases and key-value databases for more information.

Taxonomy of NoSQL

- Key-value
- Graph database
- Document-oriented
- Column family

Slide Note

SQL databases are most often implemented in a scale-up architecture, which is based on using ever-larger computers with more CPUs and more memory to improve performance.

NoSQL databases were created in Internet and cloud computing eras which made it possible to easily implement a scale-out architecture. In a scale-out architecture, scalability is achieved by spreading the storage of data and the work to process the data over a large cluster of computers. To increase capacity, more computers are added to the cluster.

This scale-out architecture is particularly painless to implement in cloud computing environments where new computers and storage can be easily added to a cluster.

NoSQL databases support widely used data formats.

NoSQL databases have become popular because they store data in simple straightforward forms that can be easier to understand than the type of data models used in SQL databases.

Adoption of NoSQL databases has primarily been driven by uptake from developers who find it easier to create various types of applications compared to using relational databases.

The scale-out architecture that most NoSQL databases use not only provides a clear path to scaling to accommodate huge data sets and high volumes of traffic but delivers a database using a cluster of computers that also allows the database to expand and contract capacity automatically.

Advantages of No SQL Database

- Handles large volumes of data at high speed with a scale-out architecture
- Stores unstructured, semi-structured, and structured data
- Enables easy updates to schemas and fields
- They are developer-friendly
- Takes full advantage of the cloud to deliver service at zero downtime

Slide Note

• NoSQL databases are designed to break away from the rows and columns of the relational database model. But it's a common mistake to think that NoSQL databases don't have any sort of data model. A useful description of how the data will be organized is the beginning of a schema.

Generally speaking, because NoSQL databases are designed to store data that does not have a fixed structure that is specified prior to developing the physical model, developers focus on the physical data model. They are typically developing applications for massive, horizontally distributed environments. This puts an emphasis on figuring out how the scalability and performance of the system will work. But they still need to think about the data model they will use to organize the data.

NoSQL Database Design and Data Modelling

- Schema Design for NoSQL Databases:
  - NoSQL databases do not have a schema in the same rigid way as relational databases.
  - Each of the main types of the NoSQL database has an underlying structure that is used to store the data.
  - The data organization is very flexible.
- NoSQL Data Modelling:
  - NoSQL databases fall into four main categories or types.
  - Each database has a data model that reflects its category.

Slide Note

Explain Document Database

Slide Note

The Document Database

- A document database is a non-relational database designed to store and query data as JSON-like documents.
- Document databases store data in a document data model using JSON (JavaScript Object Notation) or XML objects.
- Each document contains a markup that identifies fields and values.
- The values can vary over the usual types, including strings, numbers, Boolean, arrays, and nested data.

Slide Note

MongoDB – The Document Database

- MongoDB is a general-purpose, document-based, and distributed database built for modern application developers.
- The advantages of using MongoDB are:
  - Documents (i.e., objects) correspond to native data types in many programming languages.
  - Embedded documents and arrays reduce the need for expensive joins.
  - MangoDB provides dynamic schema support and fast read operations.
  - MongoDB represents data as JSON documents in a binary-encoded format called BSON.

Slide Note

Structuring Data in MongoDB

Slide Note

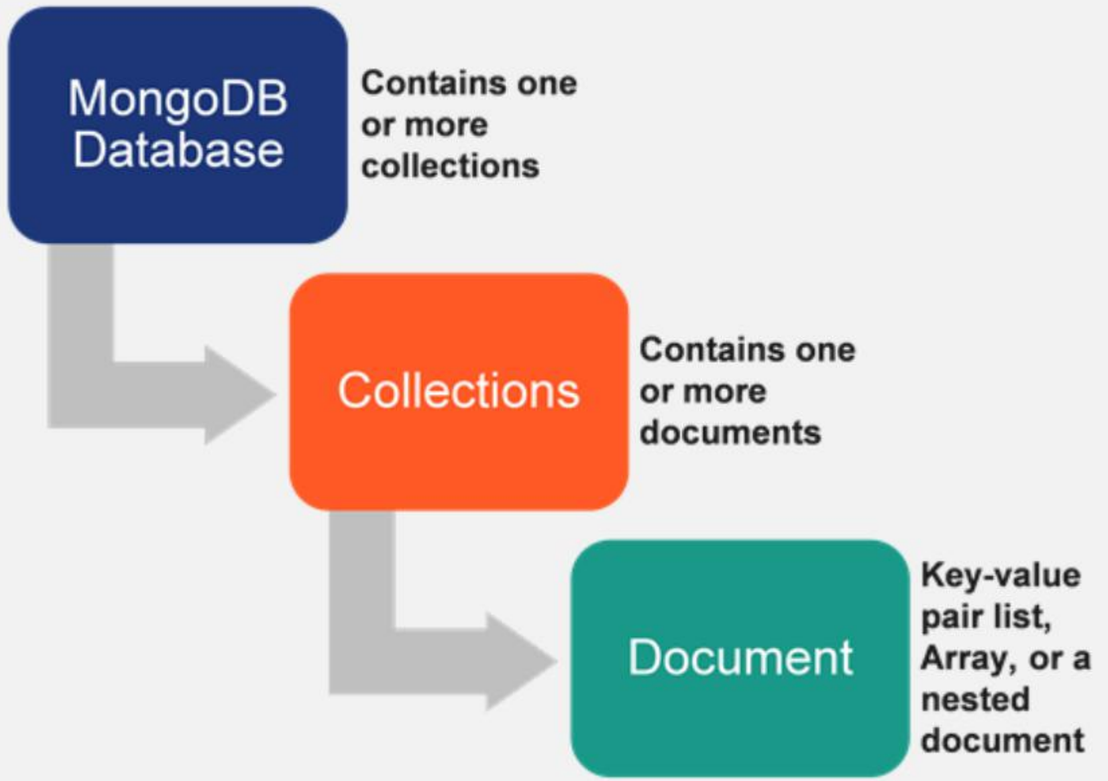
Mongo Document Explained

A record in MongoDB is a document, which is a data structure composed of field and value pairs.

Slide Note

Mongo Document Explained (contd.)

- MongoDB stores documents in collections.
- Collections are analogous to tables in relational databases.



Slide Note

Use MongoDB for Database Operations

Slide Note

```
> use products_db
switched to db products_db
```

Creating Database Operation

- In MongoDB, databases hold one or more collections of documents.
- Create a database in MongoDB using the `use` keyword.
- If the database with the same name is available, mongo will switch to the database
- If not a database with the specified name is created.

```
use <databasename>
```

Slide Note

By default, the _id field is String. It can be changed when creating the document also.

In MongoDB, documents do not use sequential numeric values for identifiers. Instead, they use a 12-byte hexadecimal value composed from the following pieces of data:

• A 4-byte value representing the seconds since the Unix epoch,

• a 3-byte machine identifier,

• a 2-byte process id, and

• a 3-byte counter, starting with a random value.

This results in our objects having identifiers that resemble the following:

ObjectId(5b5615914434ad438bf3ea43)

Insert Values

- Create a collection and insert values:

`db.<collection name>.insertOne (<values>)`

- The `_id` field is automatically generated by MongoDB.
- This is like the primary key field that is used to uniquely identify a record or a document.
- The `_id` or `Object ID` generated is a 12-byte hexadecimal value.

```
> db.inventory.insertOne(
...     { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("60dc389d3d6a77f598d222f0")
}
```

Slide Note

Inserting a Document Into a Movie Collection

Create a movie_db database in MongoDB and insert the specified records into the movies collection.

Use insertOne() to insert the record below:
Title : The Hobbit - An Unexpected Journey
Writer : J.R.R. Tolkein
Year : 2012
Franchise : The Hobbit

Install the MongoDB shell from here.
Click here for the solution.

DEMO

Slide Note

Quick Check

The command to get help for all the methods in the collection is _____.

1. `db.help()`
2. `db.mycoll.help()`
3. `rs.help()`
4. `help mr`

Slide Note

Quick Check: Solution

The command to get help for all the methods in the collection is _____.

1. `db.help()`
2. **`db.mycoll.help()`**
3. `rs.help()`
4. `help mr`

Menu

00:27:00:00:40 27/ 40

Slide Note

```
> db.inventory.insertMany([
...     { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
...     { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
...     { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
...     { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
...     { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
... ]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("60dc39763d6a77f598d222f1"),
                ObjectId("60dc39763d6a77f598d222f2"),
                ObjectId("60dc39763d6a77f598d222f3"),
                ObjectId("60dc39763d6a77f598d222f4"),
                ObjectId("60dc39763d6a77f598d222f5")
        ]
}
```

Inserting Multiple Records

- Insert multiple records into a collection using the command below:

  `db.<collection name>.insertMany()`

- An array of documents is passed to the method.

Slide Note

Mongo shell installation steps

Install MongoDB shell

Download the zip file and extract it and save in a desired location.

Navigate to the bin folder where the mongosh file is present, copy the path.

In the system environment variables add the path and click ok.

For more details on how to set the path for mongo refer to the documentation.

Open a terminal and enter mongosh to enter the mongo terminal.

Inserting Multiple Documents Into a Movie Collection

Use insertMany() to insert the records below into the movies collection of the movie_db database.

title:"The Desolation of Smaug", writer:"J.R.R. Tolkein", year: 2013, franchise:"The Hobbit“

title:"The Battle of the Five Armies", writer:"J.R.R. Tolkein", year: 2015, franchise:"The Hobbit“

title:"The Lord of the Rings: The Two Towers", writer:"J.R.R. Tolkein", year: 2002, franchise:"The Lord of the Rings“

DEMO

Slide Note

Inserting Multiple Documents Into a Movie Collection (contd.)

title:"The Lord of the Rings: The Return of the King", writer:"J.R.R. Tolkein", year: 2003, franchise:"The Lord of the Rings“

title:"The Chronicles of Narnia : The Lion, the Witch and the Wardrobe", writer:"C S Lewis", year: 2005, franchise:"The Chronicles of Narnia“

title:"The Chronicles of Narnia : Prince Caspian", writer:"C S Lewis", year: 2008, franchise:"The Chronicles of Narnia“

DEMO

Slide Note

Inserting Multiple Documents Into a Movie Collection (contd.)

title:"The Chronicles of Narnia : The Voyage of the Dawn Treader ", writer:"C S Lewis", year: 2010, franchise:"The Chronicles of Narnia"

Use the Mongo shell to execute the queries.
Click here for the solution,

DEMO

Slide Note

Querying a Collection

- **Displaying the data from a collection:**

```
db.<collection name>.find()
```

- **Displaying data in a formatted manner:**

```
db.<collection name>.find().pretty()
```

```
> db.inventory.find()
{ "_id" : ObjectId("60dc389d3d6a77f598d222f0"), "item" : "canvas", "qty" : 100, "tags" : [ "cotton" ], "size" : { "h" : 28, "w" : 35.5, "uom" : "cm" } }
> db.inventory.find().pretty()
{
        "_id" : ObjectId("60dc389d3d6a77f598d222f0"),
        "item" : "canvas",
        "qty" : 100,
        "tags" : [
                "cotton"
        ],
        "size" : {
                "h" : 28,
                "w" : 35.5,
                "uom" : "cm"
        }
}
```

Slide Note

Conditional Querying

- **Specify Equality – Find all products that have a status as ‘D’.**
  - `db.inventory.find( { status: "D" } )`
- **Specify Conditions Using Query Operators – Find all products that have the status ‘A’ or ‘D’.**
  - `db.inventory.find( { status: { $in: [ "A", "D" ] } } )`
- **Specify AND Conditions – Find all products that have status ‘A’ and quantity less than 30.**
  - `db.inventory.find( { status: "A", qty: { $lt: 30 } } )`
- **Specify OR Conditions – Find all products that have status ‘A’ or quantity less than 30.**
  - `db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ] } )`

For query operators refer to the Mongo official document.

Slide Note

Conditional Querying (contd.)

- **Specify AND as well as OR Conditions – Find all products that have the status 'A', or the quantity less than 30, or the item must not begin with the letter 'p'.**
  - ```
    db.inventory.find( {
                    status: "A",
                            $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]
    } )
    ```
- **Match an Embedded/Nested Document**
  - ```
    db.inventory.find( { size: { h: 14, w: 21, uom: "cm" } } )
    ```

Slide Note

Query the Movies Collection

1. Fetch all documents in a formatted manner.
2. Find all documents with the author set to " J.R.R. Tolkein ".
3. Fetch all documents where the franchise is “The Chronicles of Narnia“ or “The Hobbit”.
4. Fetch all documents with franchise set to "The Hobbit“.
5. Fetch all movies released between 2003 and 2009.

Use the Mongo shell to execute the queries.
Click here for the solution,

DEMO

Slide Note

Quick Check

The `$gte` query operator _____.

1. matches values that are greater than a specified value.
2. matches values that are equal to a specified value.
3. is an invalid query operator.
4. matches values that are greater than or equal to a specified value.

Slide Note

Quick Check: Solution

The `$gte` query operator _____.

1. **matches values that are greater than a specified value.**
2. **matches values that are equal to a specified value.**
3. **is an invalid query operator.**
4. **matches values that are greater than or equal to a specified value.**

Slide Note

Delete Documents

- To delete the documents in a collectio n, the commands below can be used:
  - `db.collection.deleteOne(<filter>)`
    - Removes the first document that matches the filter.
  - `db.collection.deleteMany(<filter>)`
    - Removes all documents that match the filter from a collection.
- `<filter>` - Specifies deletion criteria using the query operators.

Slide Note

Delete Documents - Movies

1. Delete the movie, “The Hobbit - An Unexpected Journey“.
2. Delete the movies with the franchise “The Lord of the Rings”

Use the Mongo shell to execute the queries.
Click here for the solution.

DEMO