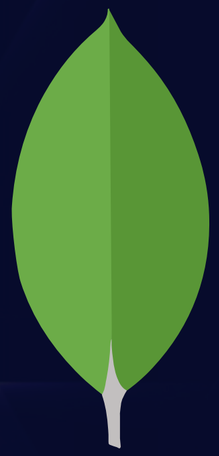
MASYNCTECH
SCHOOL

FULLSTACK WEB DEV



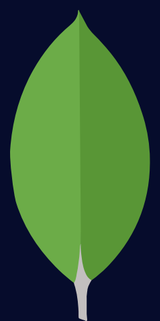
mongoDB

BACKEND DEVELOPMENT

MASYNCTECH MASYNCTECH www.masynctech.com

MASYNCTECH
SCHOOL

WHAT'S MongoDB?



mongoDB

CORE CONCEPTS

WHAT IS MONGODB?

MASYNCTECH SCHOOL

High-Level Explanation

- MongoDB: NoSQL database
- High performance, availability, scalability
- Stores data in JSON-like documents
- Supports complex, hierarchical data structures

Basic Explanation

- Analogy: Toy box as MongoDB
- Varied toys, shapes, sizes, no uniformity
- Add new toy types freely
- No need to organize beforehand, like MongoDB data flexibility

When to use?

- MongoDB use cases:
  - Large data volumes unsuitable for relational DBs
  - Real-time analytics, high-speed logging, caching, visualization
  - Mobile apps, IoT, real-time gaming, user info, real-time data
  - Agile projects with flexible schema for quick iterations

Deep Dive

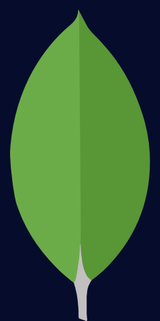
- MongoDB revolutionized databases
- Challenged relational DB dominance
- RDBMS uses rigid tables, predefined schema
- MongoDB schema-free, dynamic structure
- Insert documents without prior structure definition
- Documents have unique fields, flexible data structures

Summary

- MongoDB disrupted RDBMS with flexible schema-less docs
- Unlike rigid RDBMS tables, MongoDB allows doc insertion
- No need to define structure upfront
- Adaptability and dynamic structure modifications

MASYNCTECH
SCHOOL

WHY
MONGODB



mongoDB

CORE CONCEPTS

WHY MONGODB?

MASYNCTECH
SCHOOL

High-Level Explanation

- **MongoDB advantages:**
  - Scalability
  - Flexibility
  - Performance
- Rapid development
- Suited for large data, varying schemas, horizontal scaling

Deep Dive

- **MongoDB features:**
  - Schema flexibility: No fixed schema
  - Horizontal scalability: Easy server scaling
  - High availability: Replication, failover
  - JSON-like document storage: Aligns with coding
  - Complex data types: Arrays, nested docs supported

Analogue

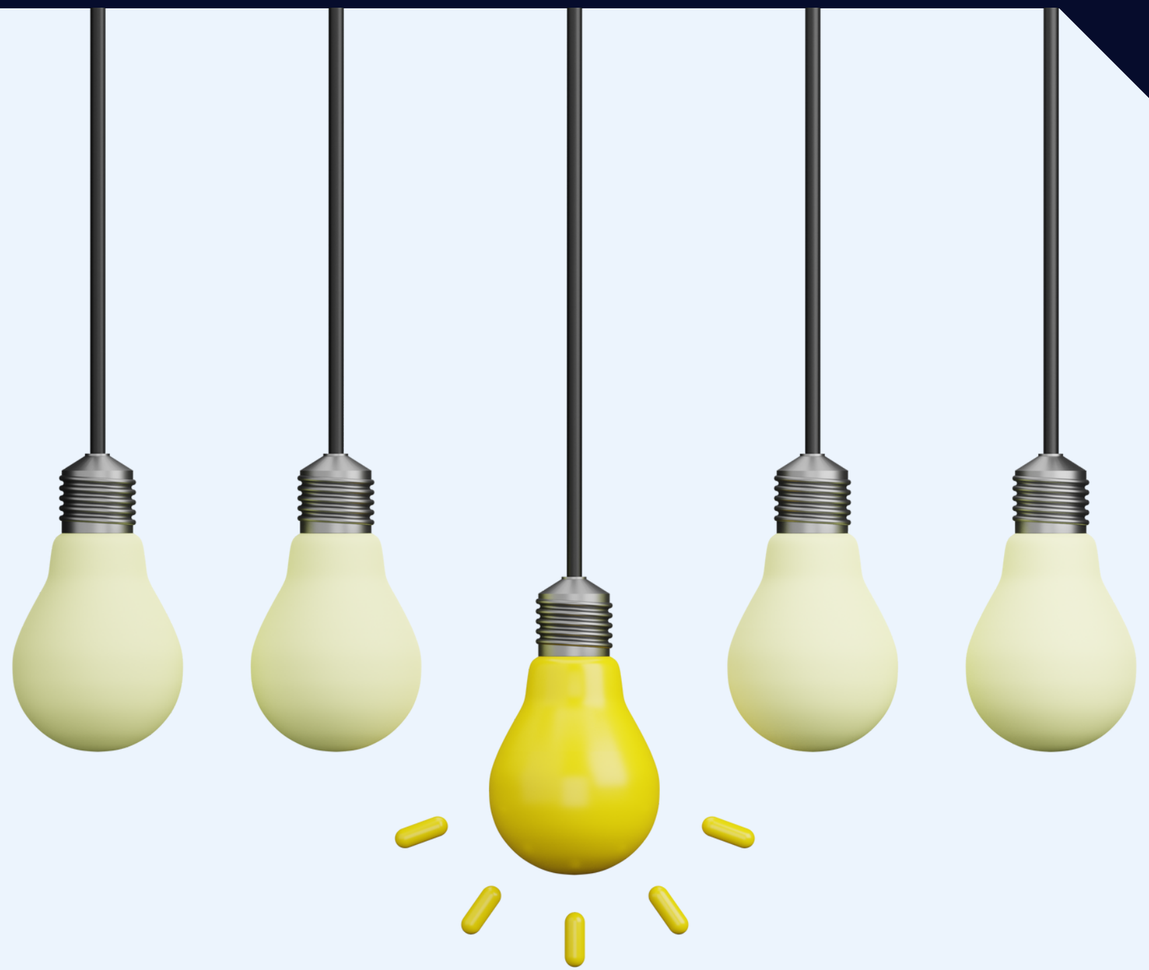
- Analogy: MongoDB as a magic LEGO board
- Doesn't restrict block color or size
- Add different blocks freely
- Board expands automatically if needed

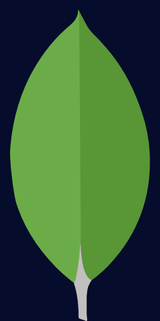
When to use?

- **MongoDB benefits in:**
  - Real-time analytics with large data
  - Content management systems with diverse content
  - Mobile app development with rapid iterations

MASYNCTECH
SCHOOL

SQL VS MONGODB





mongoDB

CORE CONCEPTS

HOW MONGODB DIFFERS FROM SQL DATABASES

MASYNCTECH SCHOOL

High-Level Explanation

- **MongoDB vs. SQL databases:**
  - MongoDB: NoSQL, document-based
  - SQL: Relational, table-based
  - Differences in architecture, data storage, querying, scalability

Deep Dive

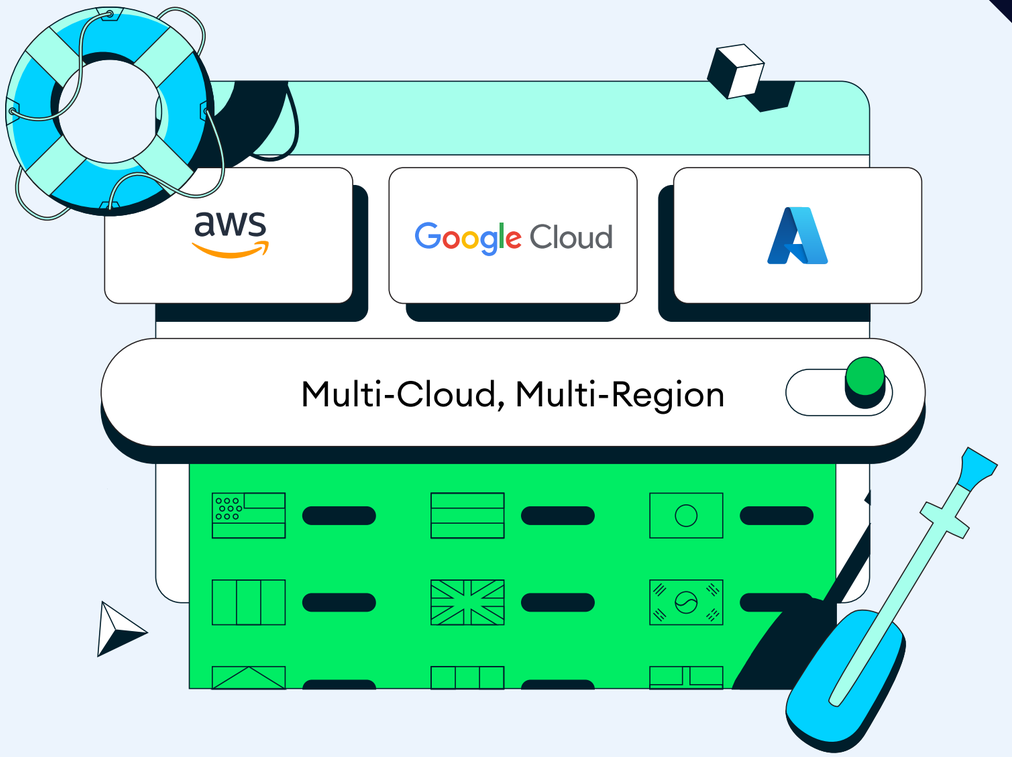
- **MongoDB vs. SQL databases**:
  - **Data Model:** MongoDB - JSON-like, flexible; SQL - tables, rigid
  - **Scalability**: MongoDB - horizontal; SQL - vertical
  - **Query Language**: MongoDB - method-based, simple; SQL - SQL, complex
  - **Transactions**: MongoDB - eventual consistency; SQL - ACID support
  - **Indexing**: MongoDB - varied types, e.g., geospatial, text

Analogue

- Analogy: SQL as a school locker
  -Specific organization, like books on one shelf
- MongoDB as a toy chest at home
  - Flexible, holds various items freely, e.g., toys, basketball

MASYNCTECH
SCHOOL

MONGODB ATLAS



mongoDB

CORE CONCEPTS

MONGODB ATLAS OVERVIEW

MASYNCTECH SCHOOL

High-Level Explanation

- MongoDB Atlas: Fully managed, cloud-based DB service
- Host and manage MongoDB DBs effortlessly
- Eliminates on-premise infrastructure setup and maintenance

Analogue

- Analogy: MongoDB Atlas as a magical toy store for data
- Effortless growth with increasing data needs
- Built-in security measures to safeguard data
- Accessibility for users from various locations

TIPS

- Considerations for MongoDB Atlas:
  - Costs can escalate beyond the free tier.
  - Limited custom configuration options.
  - Compliance with data localization laws when selecting data centers.

Deep Dive

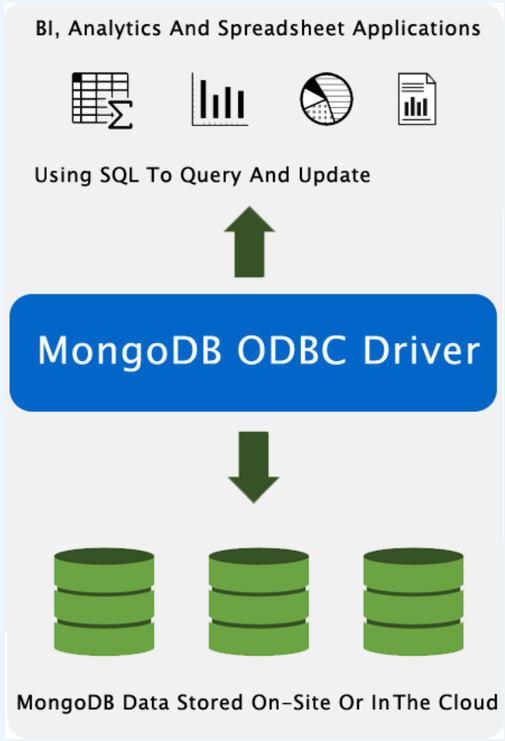
- **MongoDB Atlas features:**
  - Managed Service: Server, backup, update management
  - Scalability: Auto vertical and horizontal scaling
  - Security: Encryption, access control, VPC peering
  - Data Distribution: Global database distribution
  - Monitoring: Performance, query insights, custom alerts
  - Multi-Cloud Support: AWS, Google Cloud, Azure distribution

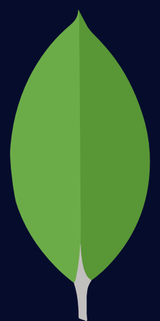
When to use?

- **MongoDB Atlas suits:**
  - Startups seeking quick, hassle-free DB solution
  - Enterprises needing scalable, secure infrastructure
  - Any case avoiding manual DB hardware/software management

MASYNCTECH
SCHOOL

MONGODB DRIVERS





mongoDB

CORE CONCEPTS

MONGODB DRIVERS OVERVIEW

MASYNCTECH SCHOOL

High-Level Explanation

- MongoDB drivers: Connect apps with MongoDB DBs
- Bridge for multiple languages (JavaScript, Python, C#, etc.)
- Enable querying and operations on MongoDB data

Deep Dive

- MongoDB drivers: Translate language commands to MongoDB
- Language-specific drivers maintained by MongoDB or community
- Optimization, security, and compatibility for smooth interaction

Analogue

- Analogy: MongoDB drivers as language translators
- Console speaks English, you speak French
- Drivers bridge communication between programming languages and MongoDB

When to use?

- MongoDB drivers are great for:
  - Apps needing data storage, retrieval, or management in MongoDB
  - Ensuring MongoDB feature compatibility across programming languages

Best Practices

- Keep MongoDB driver up to date for compatibility, security.
- Prefer officially supported, community-maintained drivers for updates, support.

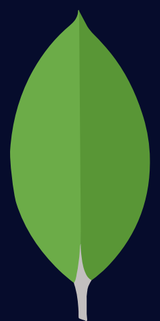
Summary

- MongoDB drivers as translators for apps
- Convert commands to MongoDB language
- Ensure compatibility, optimization, security

MASYNCTECH
SCHOOL

MONGODB

INSTALLATION & OFFICIAL DOCS

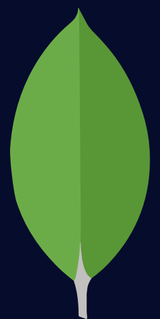


mongoDB

GETTING STARTED

MASYNCTECH
SCHOOL

MONGODB CONNECT FUNCTION

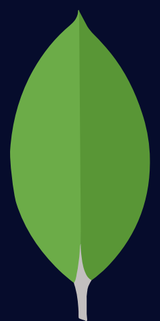


mongoDB

GETTING STARTED

MASYNCTECH
SCHOOL

MONGODB CONNECTION STRING

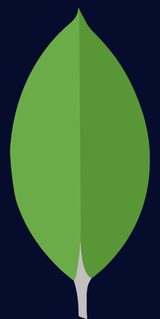


mongoDB

GETTING STARTED

MASYNCTECH
SCHOOL

MONGODB COMPASS



mongoDB

GETTING STARTED

MASYNCTECH
SCHOOL

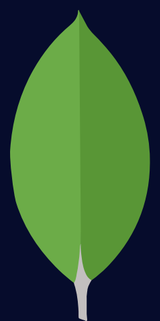
MONGODB VSCODE EXTENSION

mongoDB

GETTING STARTED

MASYNCTECH
SCHOOL

MONGODB COMMON TERMINOLOGIES



mongoDB

GETTING STARTED

MONGODB IMPORTANT TERMS

MASYNCTECH SCHOOL

High-Level Explanation

- MongoDB key terms:
  - Database
  - Collection
  - Document
  - Index
  - Sharding
  - Replication
  - Aggregation
  - And more

Analogue

- Analogy: MongoDB terms in a library
- Database as the library
- Shelves as collections
- Books as documents
- Bookmarks as indexes
- Friend's library for sharding
- Copying books for replication
- Superhero list creation as aggregation

Deep Dive

- **MongoDB key terms:**
  - **Database**: Top-level data container
  - **Collection**: Like a table, flexible schema
  - **Document**: JSON-like data unit
  - **Index**: Enhances query performance
  - **Sharding**: Horizontal scaling
  - **Replication**: Data sync for high availability
  - **Aggregation**: Data transformation and computation
  - **Cursor**: Records traversal
  - **Embedded Documents**: Nested data
  - **ACID Transactions**: Data integrity support (v4.0+)

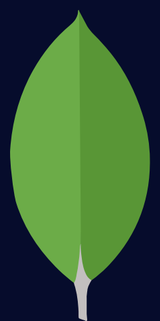
Summary

- **Key MongoDB terms:**
  - **Database**: Data container
  - **Collection**: Holds documents
  - **Document**: Basic data unit
  - **Index**: Query speed-up
  - **Sharding**: Data distribution
  - **Replication**: Data synchronization
  - **Aggregation**: Complex queries

MONGODB IMPORTANT TERMS

MASYNCTECH
SCHOOL

MONGODB WITH NODEJS



mongoDB

CRUD OPERATIONS

MASYNCTECH
SCHOOL

MONGODB

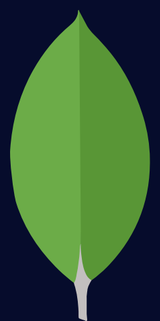
CRUD OPERATIONS

mongoDB

CRUD OPERATIONS

MASYNCTECH
SCHOOL

MONGODB
INSERTING DOCUMENTS



mongoDB

CRUD OPERATIONS

INSERTING DOCUMENTS

MASYNCTECH SCHOOL

High-Level Explanation

- MongoDB document insertion methods:
  - `**insertOne()**`: Insert single document
  - `**insertMany()**`: Insert multiple documents
  - `**insert()**`: Legacy insert method- Bridge for multiple languages (JavaScript, Python, C#, etc.)
- Enable querying and operations on MongoDB data

Analogue

- **Analogy**: MongoDB insertion methods as adding pictures to a folder
- `**insertOne()**`: Adding one picture
- `**insertMany()**`: Adding many pictures at once
- `**insert()**`: Old method, less commonly used

Best Practices

- MongoDB insertion best practices:
  - Check for errors and handle them
  - For large datasets, use `insertMany()` with `ordered: false`
  - Avoid `insert()` in new apps for modern practices

Deep Dive

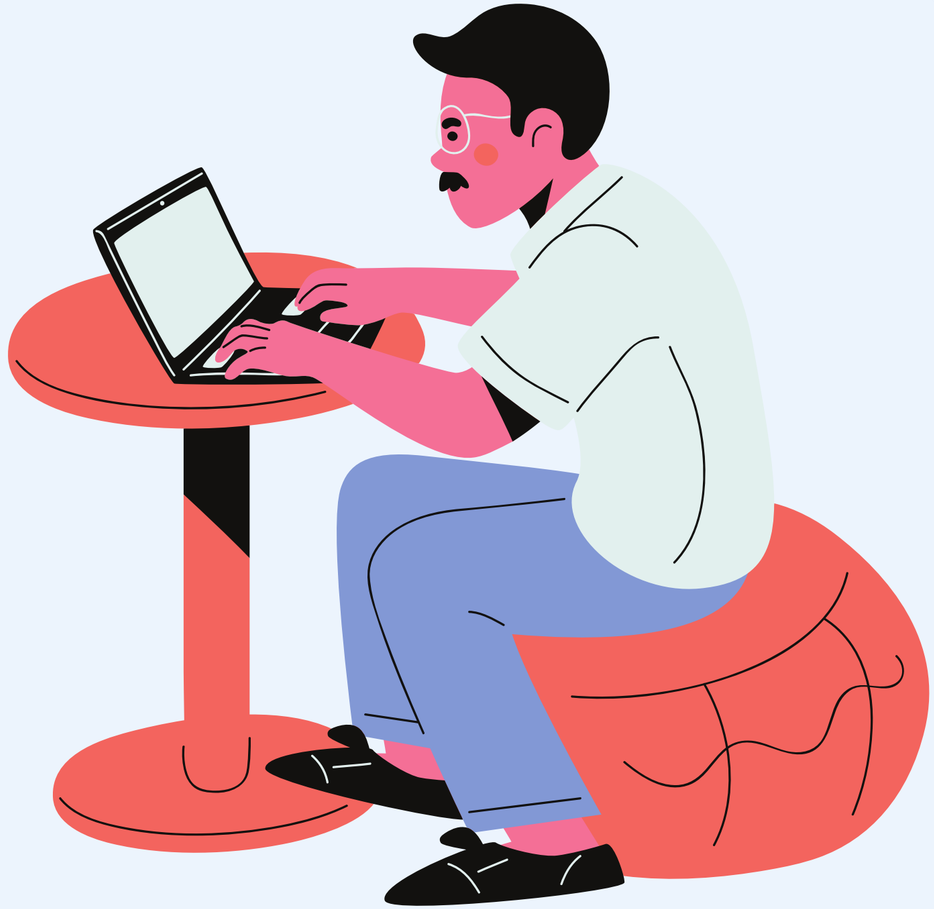
- MongoDB document insertion methods:
  - `**insertOne()**`: Single document, optimized
  - `**insertMany()**`: Multiple documents, bulk insert
  - `**insert()**`: Legacy, single/array, not recommended for new apps
- Various options for write concerns, validation, duplicate keys

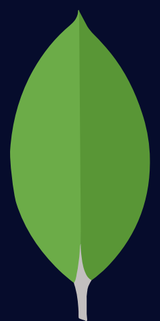
When to use?

- Best practices for MongoDB insertion methods:
  - `**insertOne()**`: For single document insertion
  - `**insertMany()**`: For bulk insertions
  - **Avoid `insert()`** in modern applications

MASYNCTECH
SCHOOL

DEMO TIME





mongoDB

CRUD OPERATIONS

MONGODB READ OPERATIONS

MASYNCTECH SCHOOL

mongoDB

CRUD OPERATIONS

READ OPERATIONS

MASYNCTECH
SCHOOL

High-Level Explanation

- MongoDB read operations:
  - `**find()**`: Retrieve multiple documents.
  - `**findOne()**`: Retrieve a single document.
  - `**aggregate()**`: Complex data transformation and computation.
  - Options for field selection, sorting, skipping, and joins between collections.

Analogue

- Analogy: MongoDB read operations as actions with a digital locker
- `**findOne()**`: Retrieving one item
- `**find()**`: Retrieving specific items
- `**countDocuments()**`: Counting the items

Deep Dive

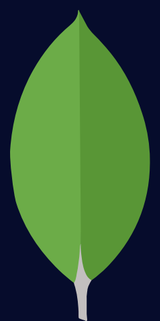
- MongoDB read operations:
  - `**find()**`: Get multiple docs, query, projection, sorting, limits.
  - `**findOne()**`: Get single doc based on query.
  - **Aggregation Pipeline:** Data operations sequence.
  - `**countDocuments()**`, `**estimatedDocumentCount()**`: Counting methods.

When to use?

- MongoDB read operation recommendations:
  - `**find()**`: For multiple docs with specific conditions
  - `**findOne()**`: For a single matching doc
  - **Aggregation:** Complex multi-step queries

MASYNCTECH
SCHOOL

DEMO TIME



mongoDB

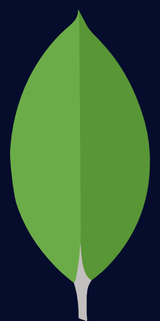
CRUD OPERATIONS

MASYNCTECH
SCHOOL

MONGODB

UPDATE

DOCUMENTS



mongoDB

CRUD OPERATIONS

UPDATE DOCUMENTS OPERATIONS

MASYNCTECH
SCHOOL

High-Level Explanation

- MongoDB document update methods:
  - `**updateOne()**`: Update a single document.
  - `**updateMany()**`: Update multiple documents.
  - `**findOneAndUpdate()**`: Find and update a document.
  - All methods work based on specified conditions.

Deep Dive

- MongoDB document update methods:
  - `**updateOne()**`: Update one matching doc.
  - `**updateMany()**`: Update multiple matching docs.
  - `**findOneAndUpdate()**`: Find, update one doc, and return it.

Analogue

- Analogy: MongoDB update methods as renaming toys in a labeled toy box
- `updateOne()`: Renaming a single toy
- `updateMany()`: Renaming multiple toys with the same label
- `findOneAndUpdate()`: Finding, renaming, and inspecting a toy

When to use?

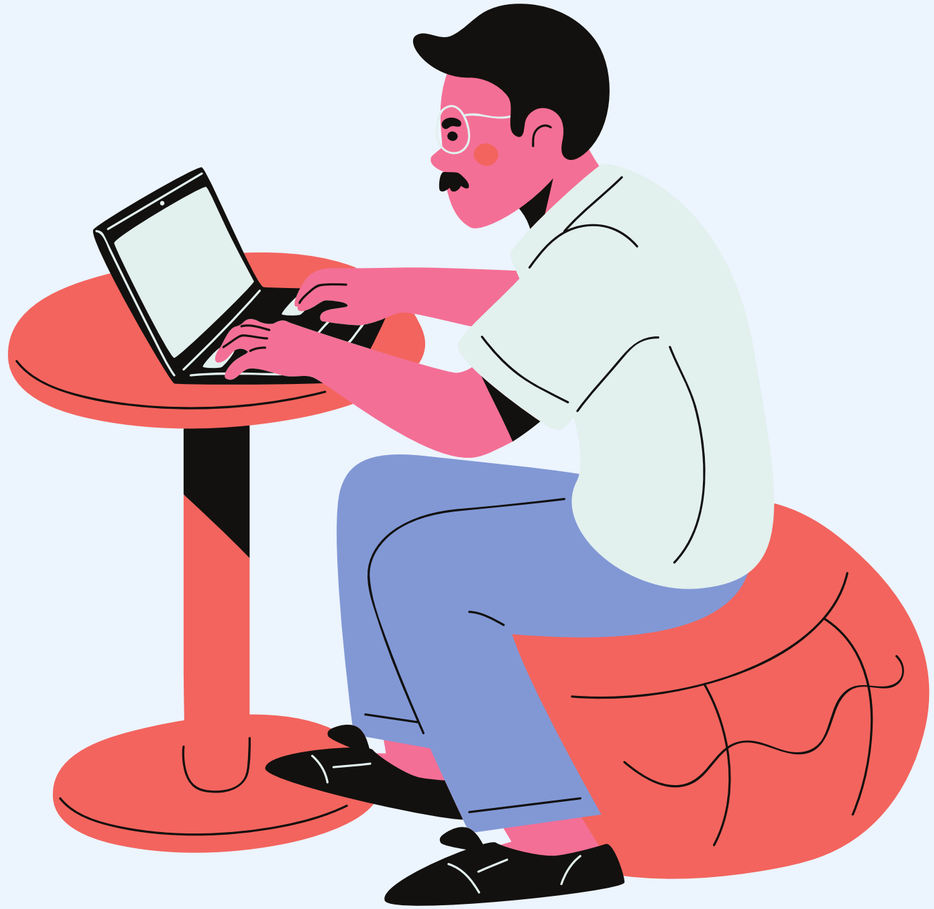
- MongoDB update method recommendations:
  - `**updateOne()**`: For updating a specific document.
  - `**updateMany()**`: When updating multiple documents.
  - `**findOneAndUpdate()**`: For fetch-before/after updates, like transactions or conditional updates.

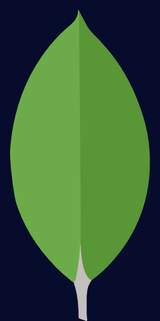
Best Practices

- MongoDB update best practices:
  - Upsert for creating if not exist.
  - Use atomic operators like `$set`, `$inc`.
  - Index fields frequently queried for updates.

MASYNCTECH
SCHOOL

DEMO TIME





mongoDB

CRUD OPERATIONS

MASYNCTECH
SCHOOL

MONGODB

DELETE

DOCUMENT

mongoDB

CRUD OPERATIONS

DELETING DOCUMENTS OPERATIONS

MASYNCTECH SCHOOL

High-Level Explanation

- MongoDB document deletion methods:
  - `**deleteOne**()`: Delete one matching doc.
  - `**deleteMany**()`: Delete multiple matching docs.
  - `**findOneAndDelete**()`: Find, delete, and return one doc.
  - All methods operate based on specified conditions.

Analogue

- MongoDB delete methods illustrated with LEGO:
  - `**deleteOne**()`: Dismantling one specific LEGO creation.
  - `**deleteMany**()`: Dismantling multiple LEGO creations of the same type.
  - `**findOneAndDelete**()`: Finding, dismantling, and retaining pieces from a specific LEGO creation.

Best Practices

- Perform data backups before large-scale deletions.
- Set the appropriate `write concern` for acknowledgment.
- Double-check your filters to avoid unintended deletions.

Deep Dive

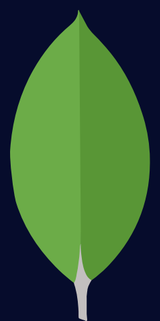
- MongoDB document deletion methods:
  - `**deleteOne**()`: Deletes one matching doc.
  - `**deleteMany**()`: Deletes multiple matching docs.
  - `**findOneAndDelete**()`: Finds, deletes, and returns one doc.

When to use?

- Use `**deleteOne**()` for specific, single document removal.
- Use `**deleteMany**()` for bulk document removal based on criteria.
- Use `**findOneAndDelete**()` when you need to inspect a document before deletion.

MASYNCTECH
SCHOOL

DEMO TIME

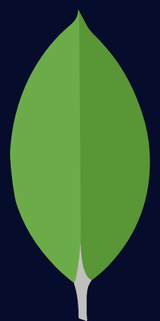


mongoDB

CRUD OPERATIONS

MASYNCTECH
SCHOOL

MONGODB BULKWRITE OPERATION



mongoDB

CRUD OPERATIONS

USING BULKWRITE

MASYNCTECH
SCHOOL

High-Level Explanation

- **bulkWrite():** This method allows you to perform multiple write operations in a single command by providing an array of write operation objects as its argument.

Deep Dive

- `**bulkWrite**`: Mix of insert, update, delete in one command
- Boosts performance, especially for large-scale data
- Improves efficiency over individual operations

Analogue

- Analogy: `bulkWrite` as managing tasks in a to-do list
- Handle multiple tasks at once
- Speed up and improve efficiency

When to use?

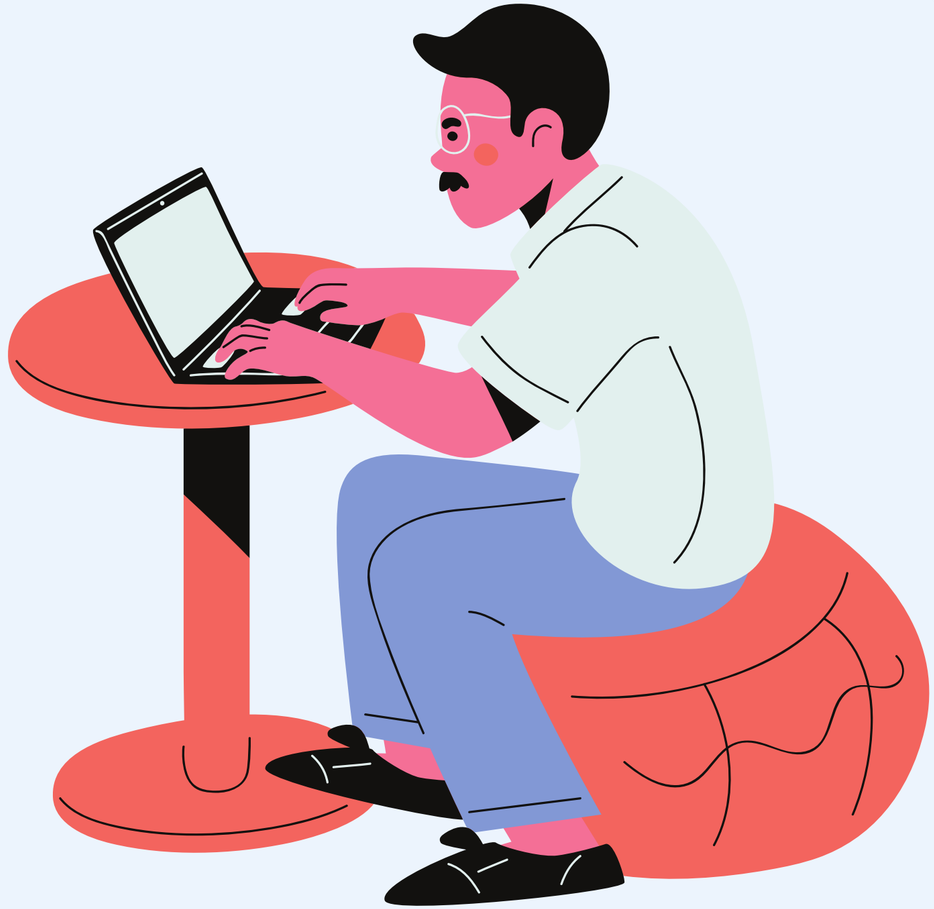
- Use `**bulkWrite**` for various write operations together.
- Enhance performance for bulk tasks.

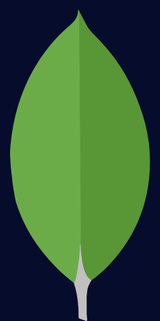
Best Practices

- Ordered operations for sequential transactions.
- Unordered operations for improved performance without sequence constraints.

MASYNCTECH
SCHOOL

DEMO TIME





mongoDB

CRUD OPERATIONS

MASYNCTECH
SCHOOL

UNDERSTANDING BSON DATA TYPES

mongoDB

CRUD OPERATIONS

WORKING WITH DIFFERENT DATA TYPES

MASYNCTECH
SCHOOL

High-Level Explanation

- MongoDB data types cover basic (strings, integers) to complex (embedded docs, arrays)
- Understanding how to define, query, manipulate is essential
- Span a wide range of use cases

Deep Dive

- Data types have distinct purposes and functionalities.
- `ObjectId` for unique identifiers, `Decimal128` for precision.
- Type casting may be necessary for comparisons.
- Choose the appropriate type for each task.

Analogue

- Analogy: Toy box with different toy types
- Action figures for storytelling
- Building blocks for construction
- Toy cars for races
- Each data type has a distinct purpose

When to use?

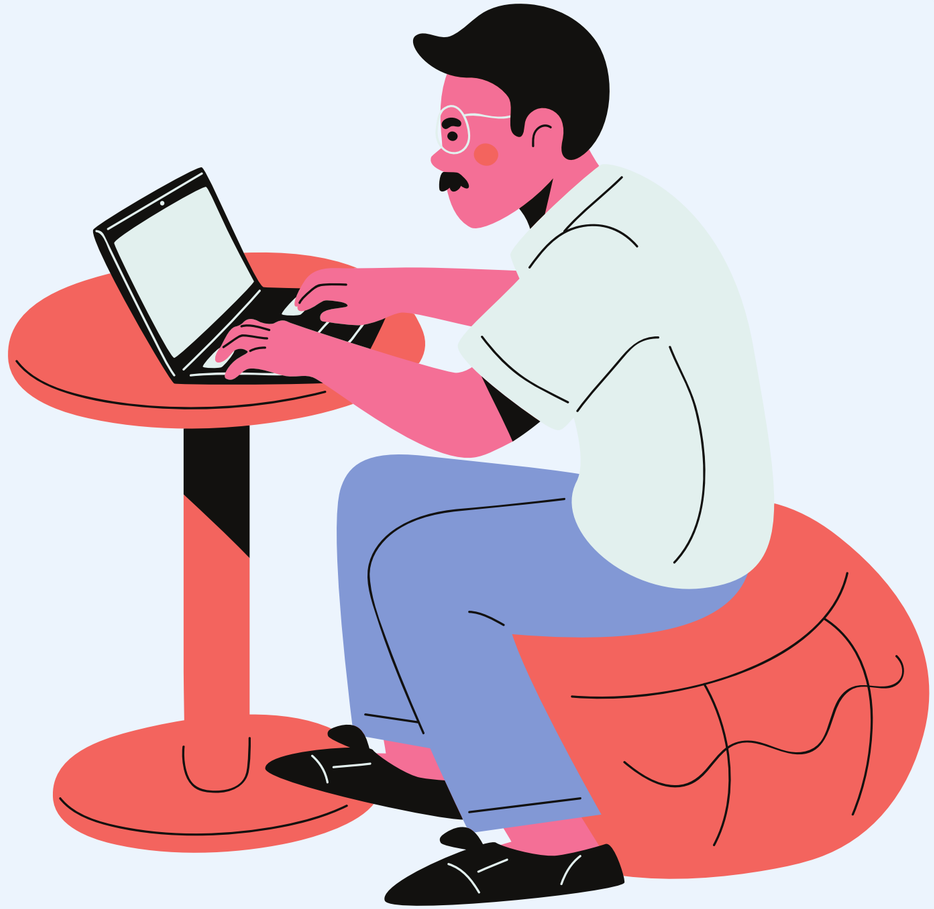
- Knowledge of data types important when:
  - Designing database schema
  - Running type-specific queries
  - Optimizing storage and performance

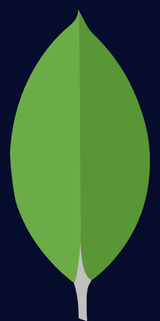
Best Practices

- Select data type aligned with application logic.
- Employ suitable query operators for efficient queries.
- Beware of type-casting and type-coercion in comparisons.

MASYNCTECH
SCHOOL

DEMO TIME



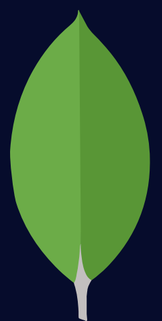


mongoDB

CRUD OPERATIONS

MASYNCTECH
SCHOOL

QUERYING DATA



mongoDB

QUERYING

THE CURSOR OBJECTS

MASYNCTECH SCHOOL

High-Level Explanation

- MongoDB cursor: Iterates query result set
- Generated by "find" operation
- Used to iterate matching documents

Deep Dive

- MongoDB cursor: Versatile tool for data manipulation
- Methods include sorting, limiting, mapping, etc.
- Doesn't immediately execute query
- Lazily retrieves documents during iteration
- Efficient for large data sets

Analogue

- Analogy: Magic wand as MongoDB cursor
- Points to toys (data) matching criteria
- Avoids dumping entire toy box (data set)
- Efficient data retrieval and selection

When to use?

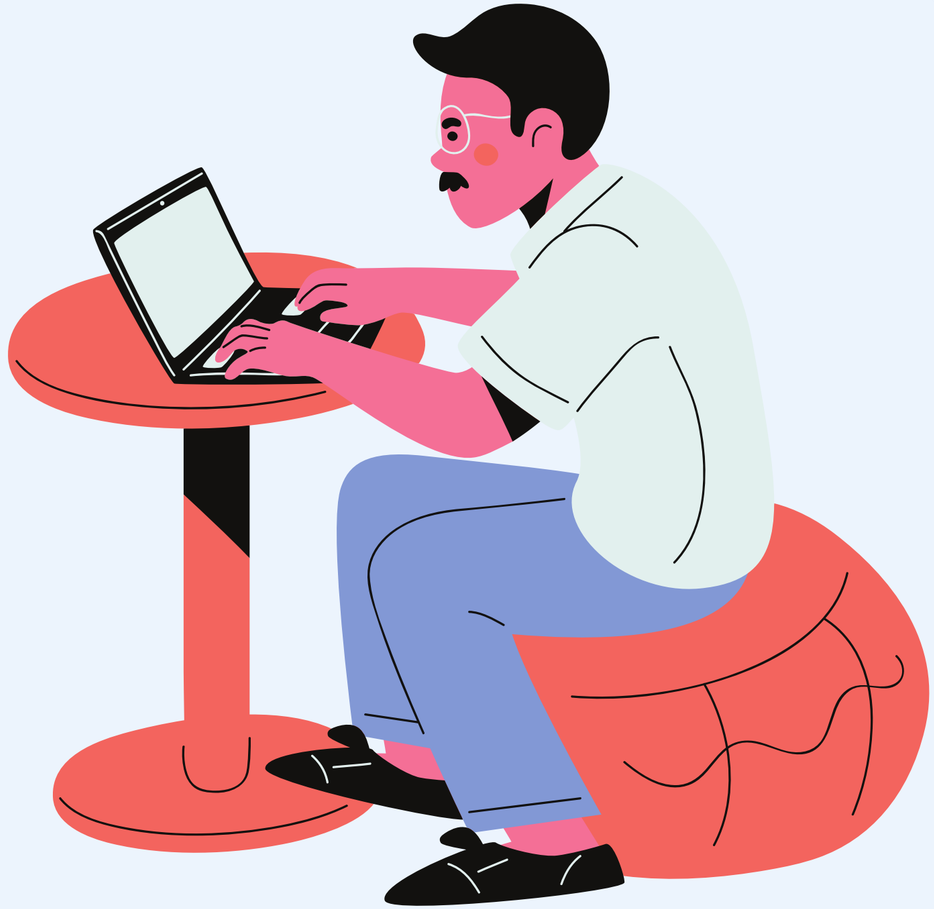
- Cursor in MongoDB: Useful for large document collections
- Apply operations like sorting, limiting
- Stream large data not fitting in memory

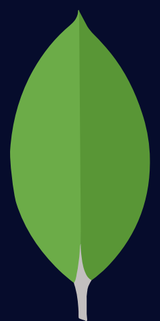
Best Practices

- Use pagination for large result sets
- Employ projection for optimized field retrieval
- Explicitly close cursor when done, especially outside cursor-based iterations

MASYNCTECH
SCHOOL

DEMO TIME



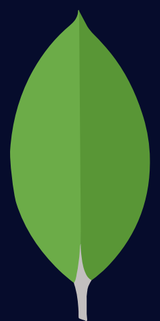


mongoDB

QUERYING

MASYNCTECH
SCHOOL

QUERY OPERATORS



mongoDB

QUERYING

QUERY OPERATORS (`$GT`, `$LT`, ETC.)

High-Level Explanation

- MongoDB query operators: Filter and manipulate data
- Common ones: `$eq`, `$gt`, `$gte`, `$in`, `$lt`, `$lte`, `$ne`, `$nin`

When to use?

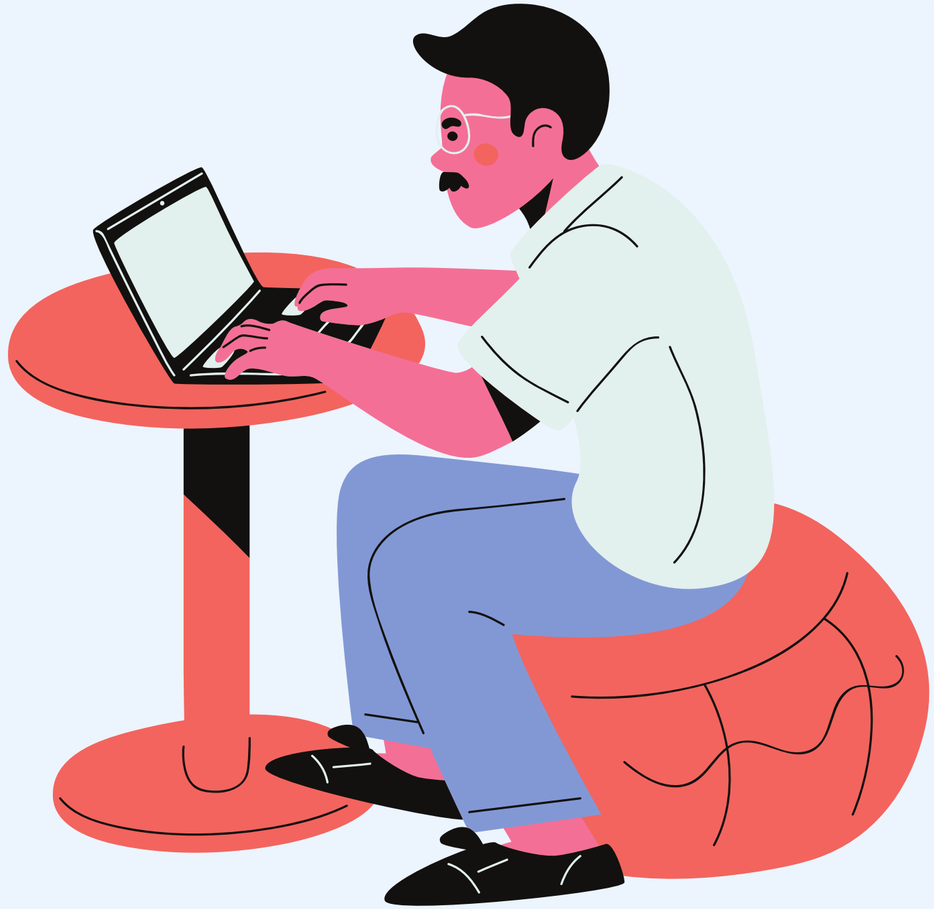
- Query operators for specific data filtering
- Example: E-commerce platform - filter items by price, reviews
- Simplify complex queries with operators

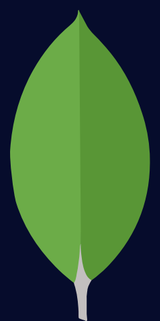
Deep Dive

- Query operators enhance query versatility
- Perform operations like equal, greater-than, etc.
- Common operators:
  - `**$eq**`: Equal to specified value
  - `**$gt**`: Greater than specified value
  - `**$gte**`: Greater than or equal to specified value
  - `**$in**`: In specified array values
  - `**$lt**`: Less than specified value
  - `**$lte**`: Less than or equal to specified value
  - `**$ne**`: Not equal to specified value
  - `**$nin**`: Not in specified array values
- Used in `.find()` or `.update()` for document filtering or modification.

MASYNCTECH
SCHOOL

DEMO TIME



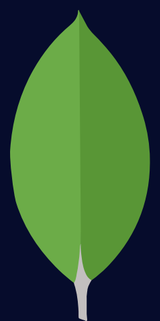


mongoDB

QUERYING

MASYNCTECH
SCHOOL

LOGICAL OPERATORS



mongoDB

QUERYING

LOGICAL OPERATORS (`$OR`, `$AND`, ETC.)

MASYNCTECH SCHOOL

High-Level Explanation

- Logical operators combine multiple query conditions
- Useful for complex queries with multiple criteria

Analogue

- Analogy: Logical operators like magic chest commands
- `**$or**`: Show golden keys OR diamond keys
- `**$and**`: Show keys that are both golden AND diamond

Deep Dive

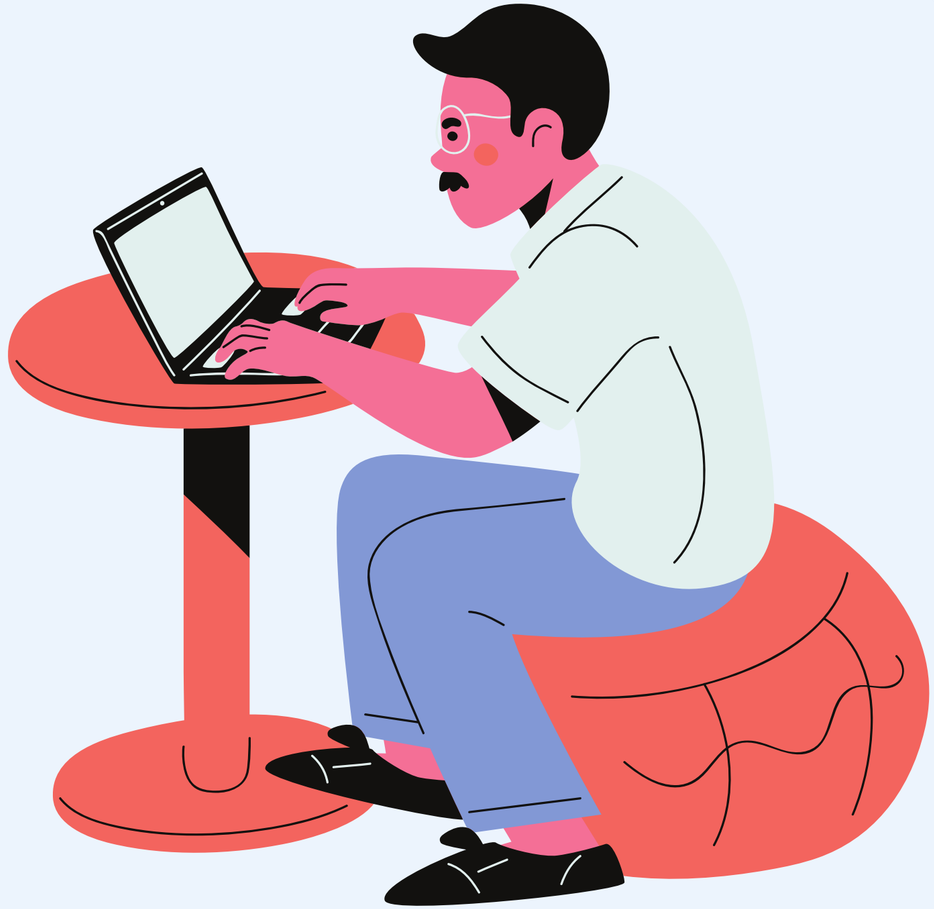
- Logical operators for complex query conditions
- `**$or**`: Logical OR on multiple queries
- `**$and**`: Logical AND between queries
- `**$nor**`: Negates `$or` operation
- `**$not**`: Negates wrapped query expression
- Useful for combining multiple conditions, often at query document root or nested.

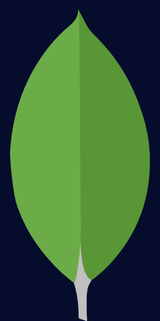
When to use?

- Logical operators for filtering with multiple conditions
- E.g., Online store: Find products in "Electronics" or "Toys" under $50

MASYNCTECH
SCHOOL

DEMO TIME





mongoDB

QUERYING

MASYNCTECH
SCHOOL

ARRAY QUERIES

mongoDB

QUERYING

ARRAY QUERIES

MASYNCTECH SCHOOL

High-Level Explanation

- MongoDB array queries: Target data within arrays
- Operators like `$in`, `$all`, `$size`, `$elemMatch`

Analogue

- Toy box analogy for array operators
- `$**all**`: Find boxes with car and ball
- `$**size**`: Locate boxes with exactly 5 toys

Deep Dive

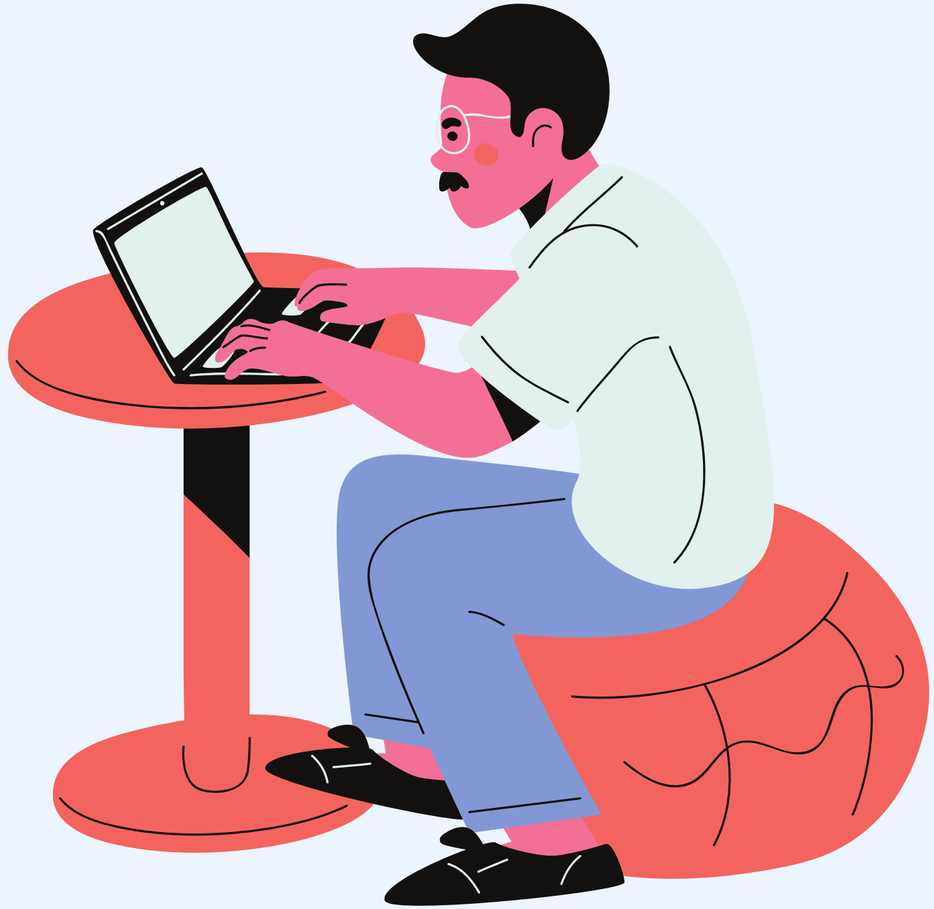
- `**$in**`: Match any element with any in specified array
- `**$all**`: Match arrays with all specified elements
- `$**size**`: Match arrays with specified element count
- `$**elemMatch**`: Match documents with array fields meeting multiple conditions

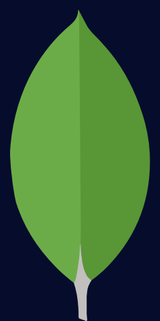
When to use?

- Array queries for filtering documents with arrays
- E.g., Shopping website: Find products with specific tags

MASYNCTECH
SCHOOL

DEMO TIME



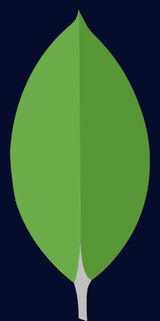


mongoDB

QUERYING

MASYNCTECH
SCHOOL

STRING SEARCH



mongoDB

QUERYING

STRING SEARCH

MASYNCTECH SCHOOL

High-Level Explanation

- MongoDB string search options:
  - `$**regex**` for regular expressions
  - `$**text**` search with text indexes

Analogue

- Analogy: `$regex` vs. `$text` search in a book
- `$regex`: Flipping through pages to find word
- `$text`: Special index guides to word location

Deep Dive

- `$**regex**`: Search with partial string patterns
  - Flexible but potentially less efficient

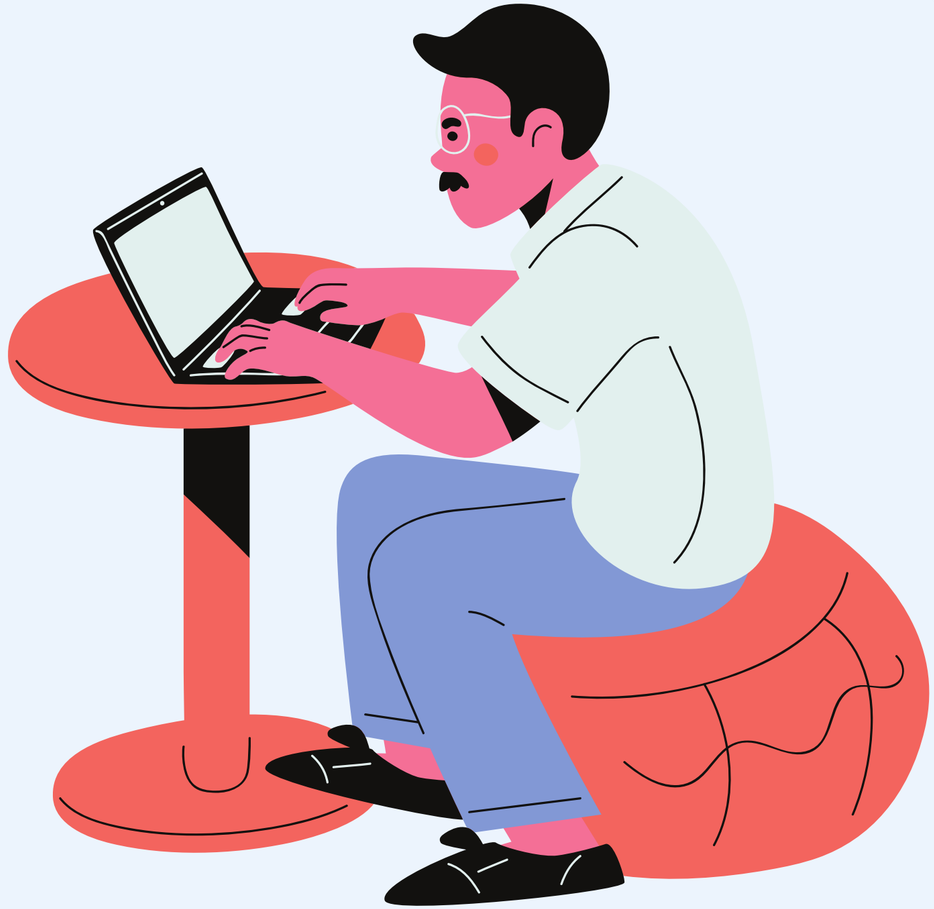
- `$**text**`: Full-text search
  - Requires text index on search fields

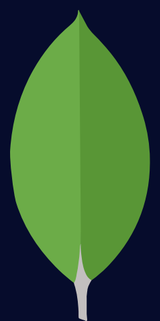
When to use?

- `$regex`: Simple partial matching
- `$text`: Advanced features like phrase search, exclusions, weightings

MASYNCTECH
SCHOOL

DEMO TIME





mongoDB

QUERYING