

**Introduction to Databases**

**Zeham Management Technologies BootCamp**

**by SDAIA**

**August 4th, 2024**

**Introduction to Databases**

**What is Database?** 

Database is s structured collection of data that can be

easily accessed, managed, and updated.

Types of Databases:

• **Relational Databases (RDBMS):** Organized into

tables with rows and columns.

• **Non-relational Databases (NoSQL):** Organized

differently, such as key-value pairs, documents, or

graphs.

• **Vector Databases:** Specialized for storing and

querying high-dimensional vectors.



**Types of Databases** • **Relational Databases (RDBMS):**

• **Examples:** MySQL, PostgreSQL, Oracle • **Features:** Structured schema, SQL query language, ACID properties.

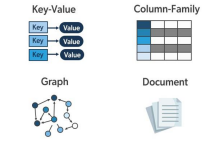
• **Non-relational Databases (NoSQL):**

• **Examples:** MongoDB, Cassandra, Redis • **Features:** Flexible schema, various data models (document, key-value, graph),

eventual consistency.

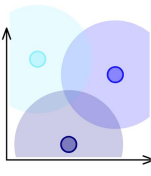
• **Vector Databases:**

• **Examples:** Pinecone, FAISS, Milvus • **Features:** Optimized for similarity search and nearest neighbor queries on high-dimensional vector data.

NoSQ L 

Source

RDBM S Source

Vector Database Source



**Database Components**

• **Relational Databases:**

• **Tables, Rows, Columns:** Basic structures for data storage.

• **Indexes, Keys, Constraints:** Improve performance and maintain data integrity. • **Non-relational Databases:**

• **Collections, Documents:** Organize data in flexible formats.

• **Indexes, Sharding:** Optimize performance and scalability.

• **Vector Databases:**

• **Vectors:** Represent data in high-dimensional space.

• **Indexes:** Specialized indexing techniques for efficient vector search (e.g., HNSW, IVF).

• **Query Languages:**

• **SQL:** Standard language for relational databases.

• **NoSQL Queries:** Vary by database type, e.g., MongoDB query language. • **Vector Queries:** Similarity search and nearest neighbor queries.



**Importance of Databases in Data Science**

• Store and analyze historical data for business insights. **Data Warehousing and Business**

**Intelligence**

• Process and analyze data as it is generated. **Real-time Data Processing and**

**Analytics**

• Store and access large datasets for model training. **Machine Learning Model Training**

**and Deployment**

**Big Data Analytics** • Handle and analyze massive datasets efficiently.

• Vector databases for applications like recommendation systems, image

recognition, and natural language processing. **High-Dimensional Data Search**

****

**Data Management Challenges**

**Handling Large Volumes of Data**

• **Efficient storage and retrieval of large datasets.**

**Ensuring Data Quality and Consistency**

• **Maintain accuracy and integrity of data.**

**Maintaining Data Security and Privacy**

• **Protect sensitive data from unauthorized access.**

**Managing Database Performance and Scalability**

• **Optimize performance and scale with growing data.**

**Vector Data Challenges**

• **Efficient indexing and querying of high-dimensional data.**

****

**Database Components**

• **Relational Databases:**

• **Tables, Rows, Columns:** Basic structures for data storage.

• **Indexes, Keys, Constraints:** Improve performance and maintain data integrity. • **Non-relational Databases:**

• **Collections, Documents:** Organize data in flexible formats.

• **Indexes, Sharding:** Optimize performance and scalability.

• **Vector Databases:**

• **Vectors:** Represent data in high-dimensional space.

• **Indexes:** Specialized indexing techniques for efficient vector search (e.g., HNSW, IVF).

• **Query Languages:**

• **SQL:** Standard language for relational databases.

• **NoSQL Queries:** Vary by database type, e.g., MongoDB query language. • **Vector Queries:** Similarity search and nearest neighbor queries.



**SQL Databases**

**Database Components** 

**Structured Query Language (SQL)** is a standardized

language used to manage and manipulate relational

databases.



**Database Components**

Data retrieval and reporting. 

Data manipulation and transformation.

Data integration and ETL (Extract, Transform,

Source Load) processes.



**SQL Concepts**

**Tables, Rows, and Columns:**

• **Tables:** Collections of related data organized in rows and

columns.

• **Rows:** Individual records in a table.

• **Columns:** Attributes or fields of the data.





**SQL Concepts Cont.**

**Primary Keys and Foreign Keys:**

• **Primary Key:** Unique identifier for each row in a table.

• **Foreign Key:** A field in one table that uniquely identifies a row of another table.

Source



**SQL Concepts Cont.**

| **Primary key** | **Foreign key** |
| --- | --- |
| It must contain unique values. | It can contain duplicate values. |
| It cannot contain null values. | It can contain null values. |
| A database can have only one primary key. | A database can have more than one foreign key. |
| It is used to identify the records in a table uniquely. | It is used to make a relation between two tables. |



**SQL Concepts Cont.**

**Data Types and Constraints:**

• **Common data types:** INTEGER, VARCHAR, DATE, etc. • **Constraints:** Rules applied to data (e.g., NOT NULL, UNIQUE).

Source

Source



**SQL Concepts Cont.** 

**Basic SQL Operations:**

• **SELECT:** Retrieve data from a table.

• **INSERT:** Add new data to a table.

• **UPDATE:** Modify existing data.

• **DELETE:** Remove data from a table.

Source



**SQL Queries**

**Basic Query Structure:**

SELECT [columns] FROM [table] WHERE [conditions]





**SQL Queries (Example 1)**

****

****

****

**SQL Queries (Example 2)**

****

****

**SQL Queries (Example 3)**

****

****

****

****

**SQL Joins**

**Joins:**

• **INNER JOIN:** Select records with matching values in both tables.

• **LEFT JOIN:** Select all records from the left table and matched records from the right table. • **RIGHT JOIN:** Select all records from the right table and matched records from the left table. • **FULL JOIN:** Select all records when there is a match in either left or right table.

**Example Database:**

****

**SQL Joins (INNER JOIN)**

****

****

****

**SQL Joins (LEFT JOIN)**

****

****

****

**SQL Joins (RIGHT JOIN)**

****

****

**SQL Joins (FULL OUTER JOIN)**

****

****

**SQL Aggregation**

**Aggregation Functions:**

**COUNT:** Returns the number of rows.

**SUM:** Returns the total sum of a numeric column. **AVG:** Returns the average value of a numeric column. **MIN:** Returns the smallest value.

**MAX:** Returns the largest value.

**Example Database:**

****

**SQL Aggregation (COUNT)**

****

****

****

**SQL Aggregation (SUM)**

****

****

****

**SQL Aggregation (AVG)**

****

****

****

**SQL Aggregation (MAX)**

****

****

****

**SQL Aggregation (MIN)**

****

****

****

**SQL Aggregation (MIN)**

****

****

****

**SQL Grouping Data**

**Grouping Data:**

• **GROUP BY:** Group rows that have the same values in specified columns.

• **HAVING:** Filter groups based on a specified condition.

**Example Database:**

****

**SQL Grouping Data (GROUP BY)**

****

****

****

**SQL Grouping Data (HAVING)**

****

****

****

**SQL Sorting**

**Example Database:**

****

**Sorting Data:**

• **ORDER BY:** Sort the result set by one or more

columns.

**Tutorials and Exercises**

**Exercises:**

3- Advanced Machine Learning/1-Introduction to Database /LAB/SQL\_ Exercises.pdf



**NoSQL Database**

**Database Components** 

NoSQL databases provide flexible schemas and horizontal scalability.

They are designed to handle large volumes of unstructured or semi

structured data. Four main types of NoSQL databases: **Document**, **Key**

**Value**, **Column-Family**, and **Graph**.



**NoSQL databases rule** 

NoSQL databases has important rule in data science such as:

• Handle large-scale data processing and real-time analytics.

• Efficiently store and query hierarchical data structures.

• Support for complex queries and aggregation.



**JSON and MongoDB**

**JSON (JavaScript Object Notation):**

• Lightweight data interchange format.

• Easy to read and write for humans and machines.

• Commonly used for transmitting data in web applications (e.g., APIs).

**MongoDB:**

• Document-oriented NoSQL database.

• Stores data in BSON format (Binary JSON).

• Dynamic schemas and rich query capabilities.



**JSON vs Python Dictionary** JSON is a format for storing and sharing data. JSON is text formatted using JavaScript object notation. JSON and Python Dictionary are similar but there are small differences like:

• **Syntax:**

• **JSON:** Text-based, double quotes

• **Dictionary:** Native Python, flexible

quotes

• **Data Types:**

• **JSON:** Limited (strings, numbers,

booleans, etc.)

• **Dictionary:** Supports complex types

• **Usage:**

• **JSON:** Data interchange

• **Dictionary:** In-program data

manipulation

Source



**JSON and Python Objects**

You can convert Python objects of the

following types, into JSON strings: 

• dict

• list

• tuple

• string

• int

• float

• True

• False

• None Source

**Python equivalents in JSON**

****Source



**Tutorials and Exercises**

**Tutorial (Notebook):**

3- Advanced Machine Learning/1-Introduction to Database /LAB/JSON.ipynb

**Exercises:**

3- Advanced Machine Learning/1-Introduction to Database /LAB/JSON\_exercise.ipynb



**Starting with MongoDB**

**To start with MongoDB, do the following steps:**

1. Create an account at the official website of MongoDB.

2. Build a free cluster.

3. Chose the cloud provider and the region.

4. Select a cluster tier (the free tier, M0, is sufficient for most small projects).

5. Create a database user.

6. Set up network access.

7. Connect to your cluster and get your connection string ‘add your database username and password to the connection string as specified in the documentation’.

8. Add it to your environment variable ‘.env’

9. Install PyMongo using `pip install pymongo`.



**Starting with MongoDB Cont.**

Connect to MongoDB using the connection string, initiate the client to connect to MongoDB. Then, create a database with the name you desire and create a collection ‘which is a table in SQL’.





**Starting with MongoDB (Insert Documents)** You can insert single single document or multiple documents using python dictionary or JSON as the document, use the insert\_one method or a list of dictionaries using insert\_many method.



**Starting with MongoDB (Query Documents)** You can query single single document or multiple documents using python dictionary or JSON as conditions to filter your search, use the find\_one method to get the first one that matches your query or a list of dictionaries using find method.



**Starting with MongoDB (Update Documents)** You can update a single document or multiple documents using a Python dictionary or JSON as conditions to filter the documents to update. Use the update\_one method to update the first document that matches your condition or use the update\_many method to update multiple documents.





**Starting with MongoDB (Delete Documents)** You can delete a single document or multiple documents using a Python dictionary or JSON as conditions to filter the documents to delete. Use the delete\_one method to delete the first one that matches your condition or use the delete\_many method to delete multiple documents.





**Tutorials and Exercises**

**Tutorial (Notebook):**

3- Advanced Machine Learning/1-Introduction to Database /LAB/No\_SQL.ipynb

**Exercises:**

3- Advanced Machine Learning/1-Introduction to Database /LAB/NoSQL\_ Exercises.pdf



**Vector Database**

**Introduction to Vector Databases**

Vector databases are specialized databases designed to handle high-dimensional vector data. Essential for applications in AI, machine learning, and data science that involve similarity searches, clustering, and nearest neighbor search.

Source Source

**What does Vector Database**

**Vector databases** store embeddings, which are high-dimensional vector representations of various inputs such as words, sentences, images, and audio. These embeddings capture the semantic information of the input data. We will explore this topic in more detail later.

Source Source

**Why not RDMBS/NoSQL databases?** While these databases are excellent at managing structured data with fixed schemas, they often face challenges with unstructured or high-dimensional data like images, audio, and text. Traditional databases are not optimized for efficiently searching for similar items within large datasets, particularly when dealing with high-dimensional vector data, resulting in poorer performance at scale.

Source Source

**Why Should We Use Vector Databases**

Long-term

Memory

Overcome

Cost

Efficiency

Context Size

Limitation



**Long-term Memory**

**Storing Information for Later Use:**

• Encoding data like text, images, or audio into embeddings helps retain their meanings. • Similar data is stored closer together, facilitating better information retrieval. • Enables your product to persistently retain and manage large amounts of knowledge over time.



Source



**Overcome Context Size Limitation**

**Handling Large Data Efficiently:**

• Vector databases store shorter sections of large data within the prompt context. • Fetch only the semantically relevant sections from the vector database.

• Overcome token limitations by using relevant shorter sections.



Source



**Cost Efficiency**

**Cost-effective Solution:**

• Generating embeddings is cheaper than invoking a full LLM API call.

• Querying embeddings against the database is much more affordable.

• Embeddings retain semantic information, offering a balance between cost and performance. 

Source



**Key Features of Vector Databases Vector databases offers various features such as:**

• **Indexing:** Advanced indexing techniques (e.g., HNSW, IVF, PQ) for fast similarity searches. • **Distance Metrics:** Support for various distance metrics (e.g., Euclidean, cosine similarity). • **Scalability:** Distributed architectures for horizontal scaling.

• **Integration:** Seamless integration with machine learning frameworks and tools.

hierarchical navigable small world (HNSW) inverted file index (IVF)

Product Quantization (PQ)



Source

Source Source

**Tutorials and Exercises**

**Tutorial (Notebook):**

3- Advanced Machine Learning/1-Introduction to Database /LAB/vectorDB.ipynb



**Collecting Data from Various Sources**

**Introduction to Data Collection Methods**

**Data Collection** is a foundational step in the field of data science 

and decision-making.

It is gathering information from various sources to create datasets

that accurately represent the phenomenon or subject of interest.

This process can be performed using *surveys*, *experiments*, *web*

*scraping*, and *accessing public datasets* or *APIs*.

**Data Collection Considerations:**

The goal of data collection in data science is to amass data that is

**relevant**, **accurate**, and of **high quality**.



**Primary versus Secondary Approaches**

Direct

Approach

In-Direct Approach

Surveys & Experiments

Primary Approach

Use-Case

Tailored

Newly

collected Data

Pre-existing datasets &

Web

scraping

Secondary Approach

Use-Case

Tailored

Using

already

gathered

Data



**Direct Data Collection Approach** 

**Primary Approach**

**The Direct Data Collection Approach** refers to a method of gathering information straightforwardly from the source for the first time.

Surveys and experiments are two fundamental methods for data collection in various fields, including *social sciences*, *marketing*, *health research*, and *many areas of data science*.



**Surveys Experiments**

****

**Survey** 

**Direct Data Collection Approach**

Surveys gather self-reported data through formats like online questionnaires and interviews. They're useful for collecting opinions and behaviors.

**Their advantages include:**

**Scalability:** Efficiently reach many respondents.

**Versatility:** Collect diverse data, from demographics to opinions.

**Comparability:** Standardized questions facilitate cross-group analysis.

Limitations include biases from question phrasing, respondent interpretation, and accuracy of responses



**Experiments** 

Direct Data Collection Approach

Experiments manipulate variables to study effects and infer causality, often under controlled settings like labs, though they can also be in the field or online.

**Key features include:**

**Control**: Ability to manage conditions and isolate variables.

**Randomization**: Random assignment to groups to limit bias and support causal conclusions.

**Repeatability**: Can be replicated to confirm findings.

Valuable for exploring cause-and-effect

Limitations include high costs, time demands, and practical or ethical limitations.



**In Direct Data Collection Approach** 

**Secondary Approach**

**Indirect(Secondary) Approach:** Involves using data that has already been collected by someone else for a different purpose and leveraging existing resources to gather information that can be applied to the current research.

**Web scraping and API Calling are two fundamental secondary methods for data collection:**

****

**API Web Scraping**

****

**Web Scraping for Data Collection** 

**In-Direct Approach**

**Web scraping** is the process of extracting data from websites, automating the collection of information available online. It serves as a powerful tool in data collection, enabling analysts and scientists to gather vast amounts of data quickly, which is essential for analysis, research, and decision-making processes.

**Key tools and technologies for web scraping include:**

**Beautiful Soup:** A Python library for parsing HTML and XML documents. It's widely used for simple projects and tasks that require quick data extraction from websites. 

**Selenium:** Originally a tool for testing web applications, Selenium can automate web browser interaction, making it suitable for scraping dynamic content that requires interaction with the webpage. 

**Scrapy:** An open-source and collaborative framework for extracting the data you need from websites. It's designed for web scraping. Scrapy is highly efficient, scalable, and versatile, making it suitable for large-scale web scraping projects.



**Beautiful Soup for Data Collection** 

**In-Direct Approach** 

**Typical Steps to handle a website in Beautiful Soup**

▪ Fetching the web page content using requests.

▪ Parsing the content with Beautiful Soup to create a parse tree.

▪ Using Beautiful Soup's searching and navigation methods to find

relevant data.

▪ Extracting and processing the data you need from the elements

found.

▪ Iteratively refining your approach based on the specific

requirements of your web scraping project and the structure of

the web pages you're working with.



**Selenium for Data Collection** 

**In-Direct Approach**

**What is Selenium?**

▪ An open-source automation tool primarily used for automating testing 

web applications.

▪ Allows for browser automation, enabling tasks to be performed as if a

real user is navigating the site so it can also render websites Dynamically.

**Why Use Selenium for Web Scraping?**

▪ **Dynamic Content:** Selenium can interact with webpages that load

content dynamically, making it ideal for scraping modern sites.

• **Real Browser Interaction:** Performs operations in a real browser

environment, allowing for actions like clicking buttons, filling forms, and

scrolling.



**Tutorials and Exercises**

**Tutorial (Notebook):**

3- Advanced Machine Learning/1-Introduction to Database /LAB/web\_scrape.ipynb

**Exercises:**

3- Advanced Machine Learning/1-Introduction to Database /LAB/web\_scrape\_exercise.ipynb



**API Access for Data Collection** 

**In-Direct Approach**

**APIs (Application Programming Interfaces) are software** are tools that allow different software applications to communicate with each other. They acts as intermediaries allowing different software applications to communicate, simplifying the process of data collection by providing structured ways to request and receive data.

**Advantages of Using APIs:**

**Efficiency:** Streamlines data access and functionality.

**Real-Time Data:** Offers access to live data, crucial for up-to-date application needs.

**Scalability:** Eases handling of growing data or demand with minimal infrastructure adjustments.

**Cost-Effectiveness:** More affordable than developing custom data collection systems.



**RapidAPI** 

**In-Direct Approach**

**RapidAPI** [link] is a comprehensive platform that 

aggregates thousands of APIs across various domains

It presents a unified platform for developers to

discover, connect, and manage APIs through a single,

standardized interface.

It offers access to diverse data sources across

various categories, including finance, sports,

entertainment, weather, and more.



**RapidAPI** 

**Considerations** In-Direct Approach

**API Limits:** Be aware of rate limits and quotas to avoid service interruptions. **Costs:** Understand the pricing model of the API and usage charges. **Security:** Keep your API key confidential to prevent unauthorized usage. **Performance:** Test response times and reliability.

**Documentation:** Read the API documentation thoroughly.

**Updates**: Stay informed about any changes or updates to the API. **Support:** Check the support options and community forums for help for Q&A.

**Getting Data from** 

Accessing high-quality data from various platforms is crucial for

data science projects. In this section, we explore three key 

platforms: **Kaggle**, **GitHub**, and **Hugging Face** .

Source



**Kaggle** 

Kaggle is a popular platform for data science

competitions, providing a variety of datasets for machine

learning and data analysis. Kaggle provides many

features like:

• **Datasets:** Access thousands of datasets across

numerous domains.

• **Competitions:** Participate in or view ongoing data

science challenges.

• **Kernels (Notebooks):** Explore code and analysis

shared by others.



**GitHub**

GitHub is a platform for version control and 

collaboration, hosting a vast number of open-source

projects that often include datasets and code. GitHub

provides many features like:

• **Repositories**: Explore and fork repositories that

contain datasets and code.

• **Issues and Discussions:** Engage with the community

for support and data-related discussions.



**Tutorials and Exercises**

**Now let's try to save the dataset in MongoDB and how to upload the model’s weights.**

**Tutorial (Notebook):**

3- Advanced Machine Learning/1-Introduction to Database/LAB/Storing&Retrieving\_ML\_Models.ipynb **Exercises:**

3- Advanced Machine Learning/1-Introduction to Database/LAB/Storing&Retrieving\_ML\_Models\_Exercise.ipynb