Database for Data Science

Data scientists often encounter daunting challenges when working with massive datasets that exceed the limitations of device RAM, resulting in sluggish code performance. Storing data on hard drives can further compound the inefficiencies in data processing. To tackle these issues, databases play a crucial role in organizing and managing data in a structured manner, be it stored in computer memory or the cloud.

As a data scientist, you will frequently need to design, create, and interact with databases in your projects. This may involve building databases from scratch or working with existing ones to effectively analyze and process data for meaningful insights.

1. Different Types of Databases Used in Data Science

Database Type	Description	Examples	Advantages	Disadvantages	Common Use Cases
Relational Database	Organizes data into tables with defined relationships between them using a schema	MySQL, PostgreSQL, Oracle	• Supports complex queries and transactions. • Ensures data integrity with constraints. • Provides scalability and performance optimizations	• May have limited flexibility in handling unstructured or semi-structured data. • May require significant upfront design and schema changes for data updates. • May have higher storage overhead	• Financial data • E-commerce • Human resources management
NoSQL Database	Does not rely on a fixed schema and can handle unstructured or semi-structured data	MongoDB, Cassandra, Amazon DynamoDB	• Provides flexibility in handling diverse data types. • Supports horizontal scaling and distributed computing. • Can be faster for certain use cases	• May lack support for complex queries and transactions. • May have limited data consistency and integrity. • May require additional effort for data modeling and indexing	• Social media • Internet of Things (IoT • Real-time analytics
Graph Database	Stores data as nodes and edges to represent relationships and is optimized for graph-based queries	Neo4j, OrientDB, Amazon Neptune	• Enables efficient representation of complex relationships. • Supports traversals and pattern matching. • Provides scalability for handling large, interconnected data	• - May not be suitable for data that does not have many relationships • May have performance limitations for non-graph-based queries. • May have a steeper learning curve for data modeling and querying	• Social networks • Fraud detection • Recommendation systems
Time-Series Database	Designed for storing and analyzing time-series data, such as sensor data, logs, or financial data	InfluxDB, TimescaleDB, OpenTSDB	• Optimized for handling time-based data with high write and query rates. • Provides time-based indexing and aggregation functions. • Supports retention policies for data aging	• May have limited support for complex data relationships. • May not be suitable for general-purpose data storage. • May require additional processing for data interpolation or filling gaps	• Sensor data analysis • Log data analysis • Financial market data

2. Different Types of Data Models Used in Database Design

Data Model	Description	Applications	Advantages	Disadvantages
Entity-Relationship (ER) Model	Represents entities (e.g., objects, concepts) as entities, attributes, and relationships between them	Most commonly used data model for conceptual database design	• Provides a graphical representation of entities and relationships. • Facilitates clear understanding of the data structure. • Supports data normalization and integrity constraints	• May not be suitable for complex real-world scenarios. • May require additional modeling techniques for handling dynamic or temporal data. • Limited support for object-oriented concepts
Relational Model	Represents data as tables (i.e., relations) with rows (i.e., tuples) and columns (i.e., attributes)	Most widely used data model for relational databases	• Supports a consistent and standardized way of organizing data. • Provides data integrity through normalization. • Enables powerful query capabilities and joins	• May have limitations in handling complex data relationships. • Requires defining a fixed schema. • May not be suitable for handling unstructured or semi-structured data
Object-Oriented Model	Represents data as objects with attributes and behaviors, and supports inheritance and encapsulation	Suitable for applications that require complex data modeling and support for object-oriented programming concepts	• Supports complex data relationships and encapsulation. • Provides inheritance and polymorphism. • Can represent real-world objects more accurately	• May have a steeper learning curve. • May have limited support in commercial databases. • May require additional efforts for query optimization
Hierarchical Model	Represents data as a tree-like structure with parent-child relationships	Suitable for hierarchical data representations	• Provides a simple and intuitive way of organizing data. • Supports efficient retrieval of hierarchical data. • Suitable for representing hierarchical relationships	• May have limitations in handling complex data relationships. • May not be suitable for handling dynamic or changing data. • May require additional efforts for maintaining data integrity

3. Different Methods for Querying Databases

Method	Description	Advantages	Disadvantages
SQL (Structured Query Language)	A domain-specific language used to query and manage relational databases	• Supports complex queries with powerful and standardized syntax. • Provides transaction management and data integrity. • Well-suited for handling large datasets	• May require a rigid schema design upfront. • May not be suitable for handling unstructured or semi-structured data. • May have limitations in handling horizontal scalability
NoSQL (Not Only SQL) Querying	Various query languages and approaches used in NoSQL databases, such as key-value, document, column-family, and graph databases	• Provides flexibility in handling diverse data types and structures. • Supports horizontal scalability and distributed computing. • Can be optimized for specific use cases	• May have limited support for complex queries and transactions. • May have limited data consistency and integrity. • May require additional effort for data modeling and indexing. • Querying syntax and capabilities can vary greatly depending on the NoSQL database type

4. Database Normalization

Normalization is a crucial process in database management that helps eliminate redundant data and ensures that only relevant data is stored in each table. By organizing data into different normal forms, it minimizes the chances of database inconsistencies and errors resulting from modifications like insertions, deletions, and updates.

We will be looking at an example with the normal forms. Assume, a video library maintains a database of movies rented out. Without any normalization in database, all information is stored in one table as shown below.

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

The three normal forms are:

- ### First Normal Form (1NF)

  - Data is stored in tables with rows uniquely identified by a primary key.
  - Data within each table is stored in individual columns in its most reduced form.
  - There are no repeating groups.

  Looking at the table above we can see that the Movies Rented column has multiple values for each row. To apply the first normal form we have to fix that by splitting it into multiple rows.

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

- ### Second Normal Form (2NF)

  - Everything from 1NF
  - Only data that relates to a table's primary key is stored at each table.

  Here we need to fix the keys situation. To do that we can add a primary key to our main table and move the movies rented to a new table.

Membership ID	Full Names	Physical Address	Salutation
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

Membership ID	Movies rented
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

- Third Normal Form (3NF)

- Everything from 2NF
- There are no in-table dependencies between the columns in each table.

To fix the transitive functional dependency in our 1st table, we need to make a new table for the salutations.

Membership ID	Full Names	Physical Address	Salutation ID
1	Janet Jones	First Street Plot No 4	2
2	Robert Phil	3rd Street 34	1
3	Robert Phil	5th Avenue	1

Membership ID	Movies rented
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Salutation ID	Salutation
1	Mr.
2	Ms.
3	Mrs.
4	Dr.

Now our little example is at a level that cannot further be decomposed to attain higher normal form types of normalization in DBMS. In fact, it is already in higher normalization forms.