Cheatography

MySQL Cheat Sheet

by Arshdeep via cheatography.com/201979/cs/42811/

Databases

Database is an organized collection of related information.

Databases support storage and manipulation of data

We need databases to store large amounts of information, to keep data organized, to protect data, to analyze data, and to scale with demand.

ACID Properties

ACID properties are fundamental for ensuring data reliability, consistency, and integrity in database systems.	They provide a set of guarantees that enable transactions to operate correctly in a multi-user environment.
Applications requiring strict data integrity and reliability, such as banking systems, e-commerce platforms, and inventory management systems, rely heavily on ACID-compliant databases.	Relational database management systems (RDBMS) like Oracle, MySQL, and PostgreSQL typically adhere to the ACID properties to ensure data consistency and reliability.
Atomicity: Refers to the all-or-nothing principle. Ensures that a transaction is either completed in its entirety or not at all. If any part of a transaction fails, the entire transaction is rolled back, preserving data integrity.	Consistency: Guarantees that the database remains in a valid state before and after the execution of transactions. Enforces integrity constraints and rules defined for the database. All changes made by a transaction must adhere to the predefined consistency constraints.
Isolation: Ensures that concurrent execution of transactions produces results equivalent to those achieved through serial execution. Transactions appear to execute in isolation from each other, even though they may be executed concurrently. Prevents interference between transactions, maintaining data integrity and consistency.	Durability: Ensures that once a transaction is committed, its effects persist even in the event of system failures. Changes made by committed transactions are permanent and are stored in non-volatile memory (such as disk) to withstand crashes or restarts. Guarantees that committed transactions survive system failures and are not lost or rolled back.

NoSQL

NoSQL databases are non-relational databases designed for handling large volumes of unstructured, semi-structured, or structured data.	They offer flexible schema design and horizontal scalability to manage diverse data types and high-velocity data ingestion.
Flexible Schema: NoSQL databases allow dynamic schema creation, enabling storage of varying data structures within the same database.	Horizontal Scalability: NoSQL databases scale horizontally by adding more servers or nodes to distribute data and load across the cluster.
High Performance: Designed for high-speed data processing and low-latency access, making them suitable for real-time applications.	Distributed Architecture: Data is distributed across multiple nodes, providing fault tolerance and redundancy for increased reliability.

Normalization

Definition: The process of organizing data in a database to reduce redundancy and dependency.	Objective: Enhance data integrity, minimize anomalies, and improve database efficiency.
First Normal Form (1NF): Ensures atomicity of data. No repeating groups or arrays.	Second Normal Form (2NF): Non-key attributes are fully functionally dependent on the primary key. Eliminates partial dependencies.
Third Normal Form (3NF): Eliminates transitive dependencies. Non-key attributes depend only on the primary key.	Boyce-Codd Normal Form (BCNF): A stronger version of 3NF. Every determinant is a candidate key. Avoids certain types of anomalies.

Database Management System (DBMS)

DBMS is a collection of programs that enables its users to access databases, manipulate data, reporting/representation of data.

DBMS manages the data, the database engine, and the database schema, allowing for data to be manipulated or extracted by users or other programs.

Types of DBMS

Relational DBMS (RDBMS) Organizes data into tables with rows and columns. Utilizes Structured Query Language (SQL) for data manipulation. Examples: MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server.

NoSQL DBMS Suited for unstructured or semi-structured data. Offers flexibility in schema design. Types: Document-oriented, Key-value, Column-oriented, Graph databases. Examples: MongoDB, Cassandra, Redis, Neo4j.

Object-Oriented DBMS (OODBMS) Stores data in the form of objects. Supports object-oriented features like inheritance and encapsulation. Examples: db4o, ObjectDB.

Graph DBMS Optimized for managing and querying graph data structures. Ideal for interconnected data applications like social networks. Examples: Neo4j, Amazon Neptune, JanusGraph.

Time-Series DBMS Specialized for managing data with timestamps. Commonly used in IoT and financial data analysis. Examples: InfluxDB, TimescaleDB, Prometheus.

In-Memory DBMS Stores data primarily in system memory for faster access. Suitable for applications requiring high-speed data processing. Examples: Redis, MemSQL, VoltDB.

RDBMS

RDBMS is a type of database management system that stores data in the form of tables with rows and columns.	Data is organized into related tables, and relationships between tables are established using keys. It employs Structured Query Language (SQL) for querying and managing the database.
Tables: Data is stored in tables consisting of rows and columns.	Rows: Each row represents a record or entity in the database.
Columns: Each column represents a specific attribute or field of the data.	Keys: Primary keys uniquely identify each row in a table, while foreign keys establish relationships between tables.

Types of NoSQL

Document Store: Stores semi-structured data in flexible JSON or BSON documents. Example: MongoDB, Couchbase, CouchDB.	Key-Value Store: Simplest NoSQL model, storing data as key-value pairs. Example: Redis, Amazon DynamoDB, Riak.
Column Family Store: Organizes data into columns instead of rows, suitable for wide-column databases. Example: Apache Cassandra, HBase.	Graph Database: Designed for managing highly interconnected data, using graph structures. Example: Neo4j, Amazon Neptune, JanusGraph.

SQL

Definition: Standardized language for managing relational databases.

Designed for querying, updating, and managing data.

Tips:

Use aliases to simplify column names in queries.

Utilize indexes for faster data retrieval.

Regularly backup and maintain databases to prevent data loss.

SQL Data Types

String Data Types: TEXT, CHAR, VARCHAR, ENUM, SET	Numeric Data Types: INT, FLOAT, DOUBLE, etc.
Date and Time Data Types: DATE, TIME, TIMESTAMP, DATETIME, YEAR	JSON Data Type: Store JSON documents in the JSON column