# Module 3
# Data Tier

## Intro to Relational DBs

Presented by
*Domenico Mossali*

IBM Client Innovation Center - Bari

Bari, 27/11/2023

IBM Client Innovation Center
**Italy**

# Who am I?

**Domenico Mossali**

Senior practitioner – Business Analyst

In IBM Client Innovation Center since 2019

E-mail: domenico.mossali-cic-it@ibm.com
Slack: @Domenico Mossali



IBM Client Innovation Center
**Italy**

# Agenda

1. **Basic Intro to Databases**
2. Relational Database Overview



# What a Database is?

A **database** is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex they are often developed using formal design and modeling techniques.

The **database management system** (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyse the data. The **DBMS software** additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database.

# Database Overview

Existing DBMSs provide various functions that allow management of a database and its data which can be classified into four main functional groups:

- **Data definition** – Creation, modification and removal of definitions that define the organization of the data.
- **Update** – Insertion, modification, and deletion of the actual data.
- **Retrieval** – Providing information in a form directly usable or for further processing by other applications. The retrieved data may be made available in a form basically the same as it is stored in the database or in a new form obtained by altering or combining existing data from the database.
- **Administration** – Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity.

# Database Overview #2

Physically, database servers are dedicated computers that hold the actual databases and run only the DBMS and related software.

Database servers are usually multiprocessor computers, with generous memory and RAID disk arrays used for stable storage.

DBMSs are found at the heart of most database applications.

Since DBMSs comprise a significant market, computer and storage vendors often take into account DBMS requirements in their own development plans.

# Database Overview #3

Application developers in the twenty-first century face a dizzying bevy of database decisions. There are hundreds of different databases available to choose from, and many are solid pieces of general-purpose technology. On the other hand, almost every commercially backed database can claim some important-sounding customers as references, regardless of how niche the database itself is.

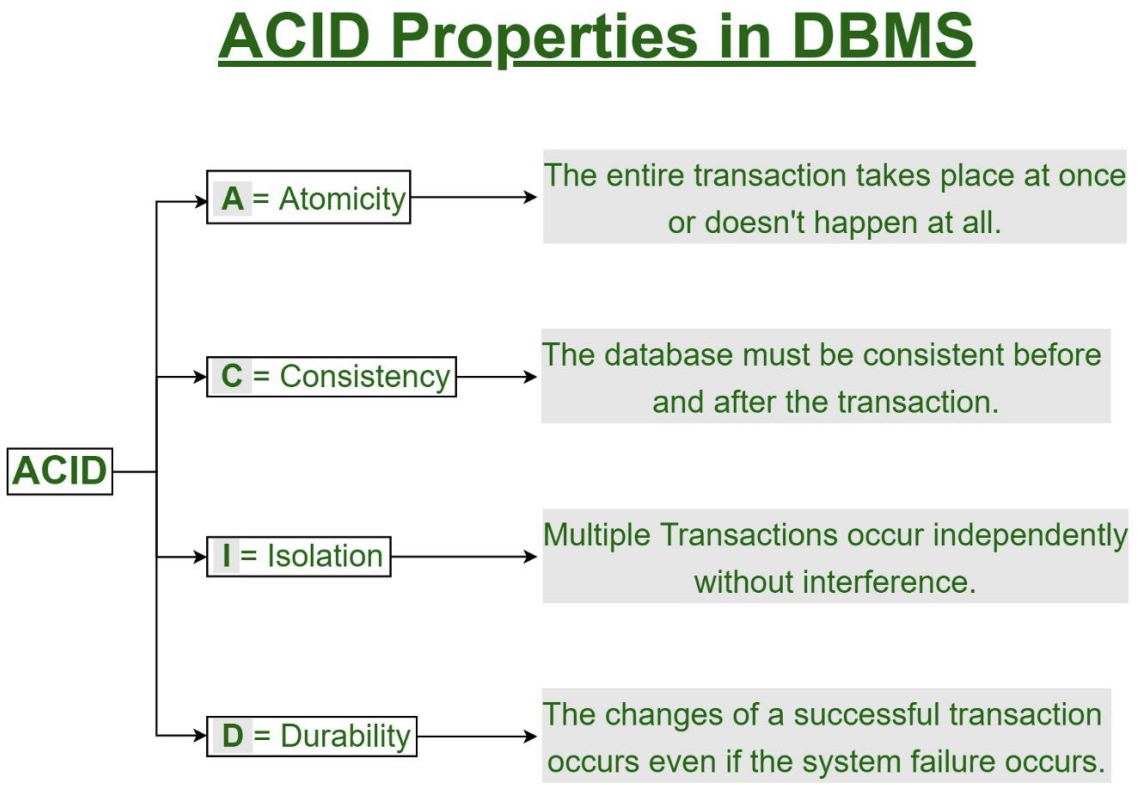
In order to make some sense of the landscape, it's helpful to have a taxonomy handy. For better or worse, the most popular taxonomy from the past 10 years divides the landscape into two classes: **SQL** (relational databases) and **NoSQL** (everything else).

# The ACID properties

In computer science, **ACID** (*atomicity*, *consistency*, *isolation*, *durability*) is a set of properties of database transactions intended to guarantee data validity despite errors, power failures, and other mishaps.

In the context of databases, a sequence of database operations that satisfies the ACID properties (which can be perceived as a single logical operation on the data) is called a transaction.

For example, a transfer of funds from one bank account to another, even involving multiple changes such as debiting one account and crediting another, *is a single transaction*.



# The ACID properties #2

## Atomicity

Transactions are often composed of multiple statements. Atomicity guarantees that each transaction is treated as a single "unit", which either succeeds completely or fails completely: if any of the statements constituting a transaction fails to complete, the entire transaction fails and the database is left unchanged.

## Consistency

Consistency ensures that a transaction can only bring the database from one valid state to another, maintaining database invariants: any data written to the database must be valid according to all defined rules.

## Isolation

Transactions are often executed concurrently (e.g., multiple transactions reading and writing to a table at the same time). Isolation ensures that concurrent execution of transactions leaves the database in the same state that would have been obtained if the transactions were executed sequentially.

## Durability

Durability guarantees that once a transaction has been committed, it will remain committed even in the case of a system failure (e.g., power outage or crash).

# Agenda

1. Basic Intro to Database
2. **Relational Database Overview**

IBM Client Innovation Center
**Italy**

# A Relational Database Overview

In short, SQL databases support SQL — a domain-specific language for querying and manipulating data in a relational database. The "relational" in a relational database refers to the "relational model" of data management devised by IBM researcher E.F. Codd in the early 1970s.

A database is a means of storing information in such a way that information can be retrieved from it.
In simplest terms, a **relational database** is one that presents information in tables with rows and columns.

A table is referred to as a relation in the sense that it is a collection of objects of the same type (rows). Data in a table can be related according to common keys or concepts, and the ability to retrieve related data from a table is the basis for the term relational database.
A Database Management System (DBMS) handles the way data is stored, maintained, and retrieved. In the case of a relational database, a Relational Database Management System (RDBMS) performs these tasks.

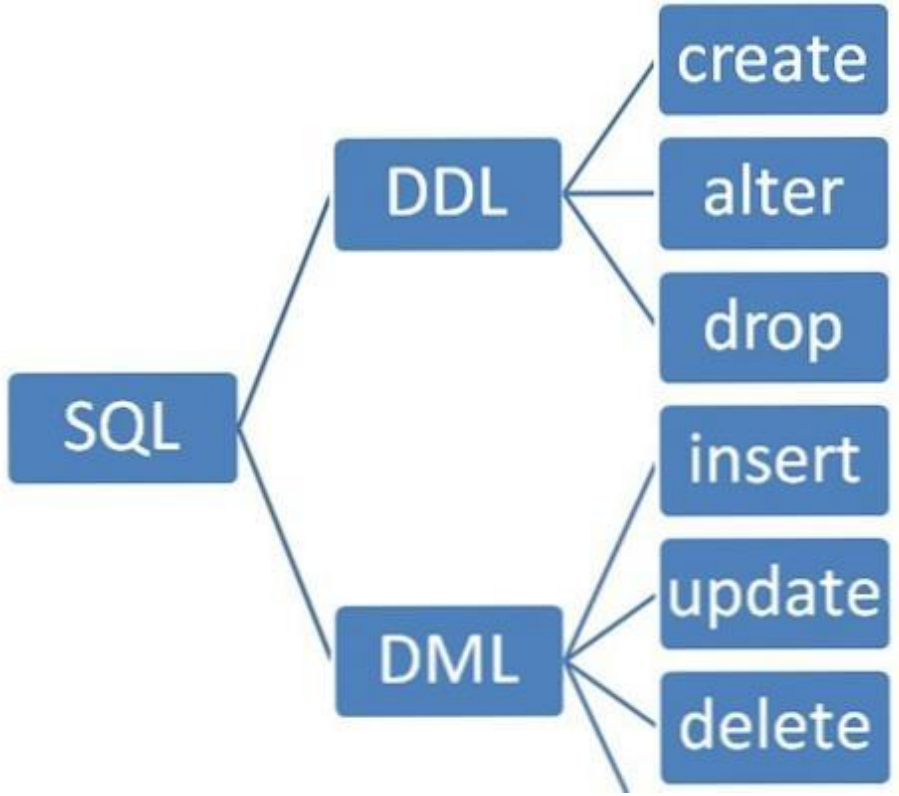
From Oracle Java Documentation: https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html

# Common SQL Commands #1

**Common SQL Commands**

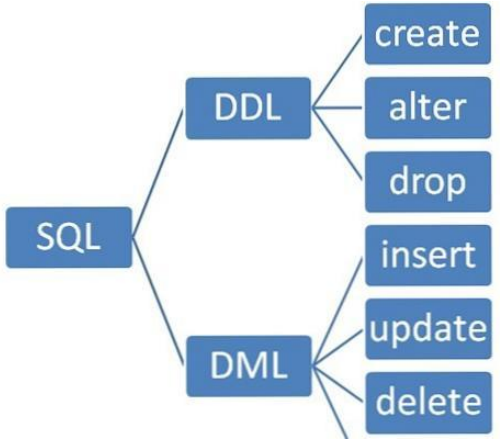
SQL commands are divided into categories, the two main ones being **Data Manipulation Language** (DML) commands and **Data Definition Language** (DDL) commands.

- DML commands deal with data, either retrieving it or modifying it to keep it up-to-date.
- DDL commands create or change tables and other database objects such as views and indexes.




From Oracle Java Documentation: https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html

# Common SQL Commands #2




**Common SQL Commands. Cont...**

A list of the more common DML commands follows:

- **SELECT** — used to query and display data from a database. The SELECT statement specifies which columns to include in the result set. The vast majority of the SQL commands used in applications are SELECT statements.

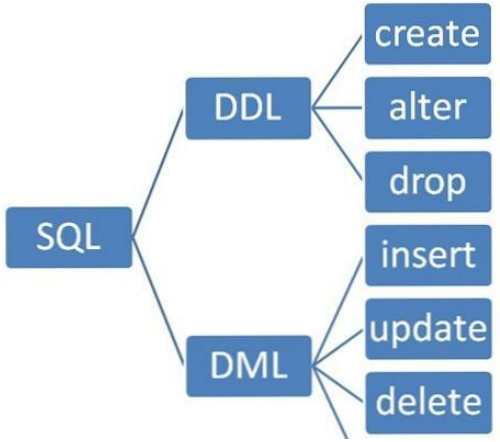
- **INSERT** — adds new rows to a table. INSERT is used to populate a newly created table or to add a new row (or rows) to an already-existing table.

- **DELETE** — removes a specified row or set of rows from a table

- **UPDATE** — changes an existing value in a column or group of columns in a table

From Oracle Java Documentation: https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html

# Common SQL Commands #3




**Common SQL Commands. Cont...**

The more common DDL commands follow:

- CREATE TABLE — creates a table with the column names the user provides. The user also needs to specify a type for the data in each column. Data types vary from one RDBMS to another, so a user might need to use metadata to establish the data types used by a particular database. CREATE TABLE is normally used less often than the data manipulation commands because a table is created only once, whereas adding or deleting rows or changing individual values generally occurs more frequently.

- DROP TABLE — deletes all rows and removes the table definition from the database.

- ALTER TABLE — adds or removes a column from a table. It also adds or drops table constraints and alters column attributes

From Oracle Java Documentation: https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html

# Relational Database Pros & Cons

**Pros**

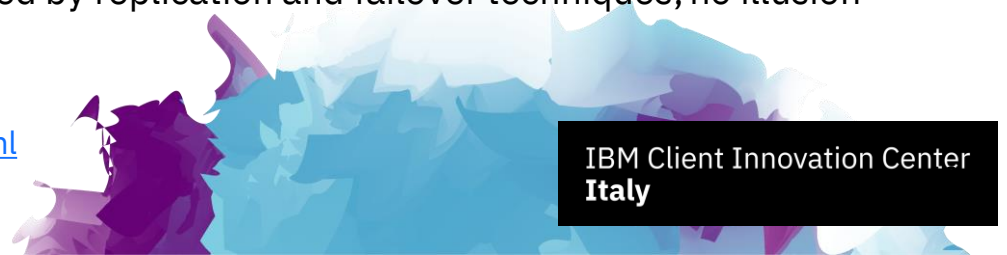
- Reduced data storage footprint due to normalization and other optimization opportunities. Often results in better performance and more efficient use of resources.
- Strong and well-understood data integrity semantics through ACID (Atomicity, Consistency, Isolation, Durability).
- Standard access to data via SQL.
- Generally more flexible query support capable of handling a broader range of workloads. SQL abstracts over the underlying implementation and allows the engine to optimize queries to fit their on-disk representation.

**Cons**

- Rigid data models that require careful up-front design to ensure adequate performance and resist evolution—changing a schema will often include downtime
- Scaling horizontally is challenging—either completely unsupported, supported in an ad-hoc way, or only supported on relatively immature technologies
- Non-distributed engines are generally a "single point of failure" that must be mitigated by replication and failover techniques; no illusion of infinite scalability

From Oracle Java Documentation: https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html





# Relational Database examples

In the following a short list, as an example, of more known relational DBs:

- PostgreSQL
- Db2
- MySQL
- Oracle
- SQL Server
- MariaDB

From Oracle Java Documentation: https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html

# Creation of a database

To create a new database in MySQL, you use the **CREATE DATABASE** statement with the following syntax:

```
CREATE DATABASE [IF NOT EXISTS] database_name
[CHARACTER SET charset_name]
[COLLATE collation_name]
```

In this syntax:

- First, specify name of the database after the the **CREATE DATABASE** keywords. The database name must be unique within a MySQL server instance. If you attempt to create a database with a name that already exists, MySQL will issue an error.
- Second, use the **IF NOT EXISTS** option to conditionally create a database if it doesn't exist.
- Third, specify the character set and collation for the new database. If you skip the CHARACTER SET and COLLATE clauses, MySQL will the default character set and collation for the new database.