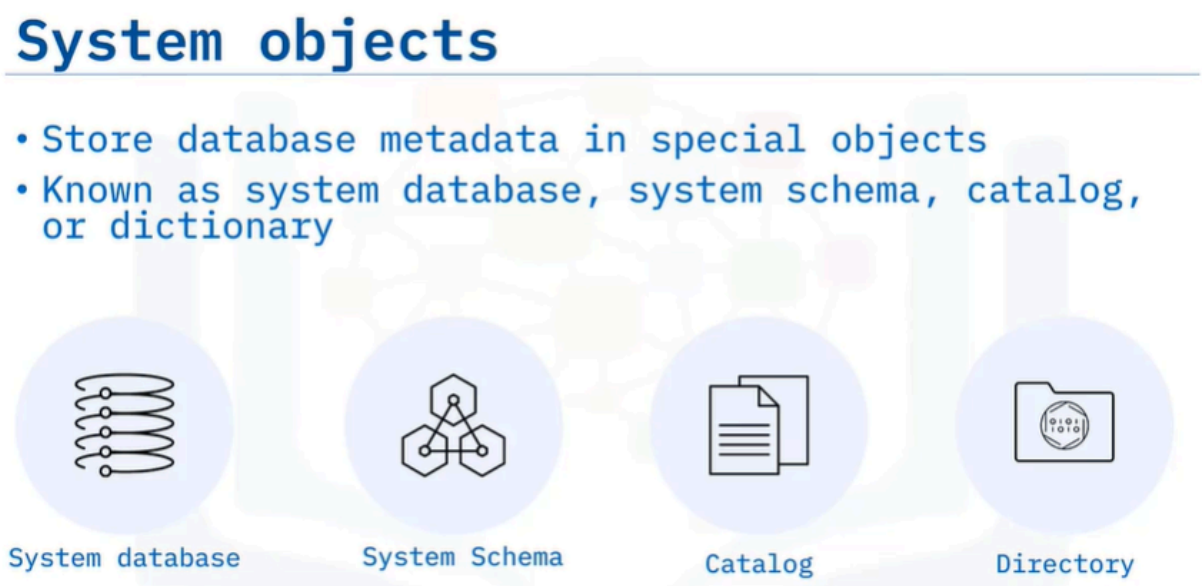
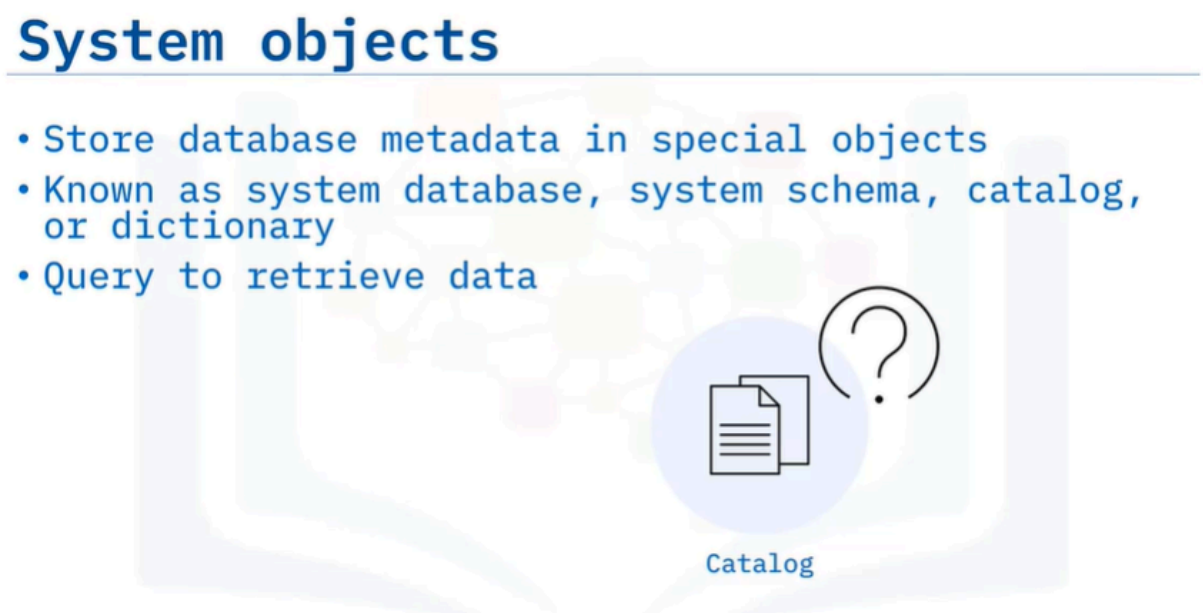
System Objects and Database Configuration

Sure! Let's dive into the concept of **system objects and database configuration** in a way that's easy to understand.

System objects are like the behind-the-scenes helpers of a database. They store important information about the database itself, such as the names of tables, the types of data in each column, and who has permission to access what. Think of it like a library catalog that tells you where to find each book, what each book is about, and who can borrow them. This information is crucial for understanding how the database is performing and for managing it effectively.

When it comes to database configuration, it's all about setting up the database to work properly. Just like you might adjust the settings on your phone or computer to make it run better for your needs, databases have configuration files that store settings like where to find data, how much memory to use, and how to handle connections. These settings can be adjusted based on whether the database is running on your own server or in the cloud, making it flexible and adaptable to different environments.

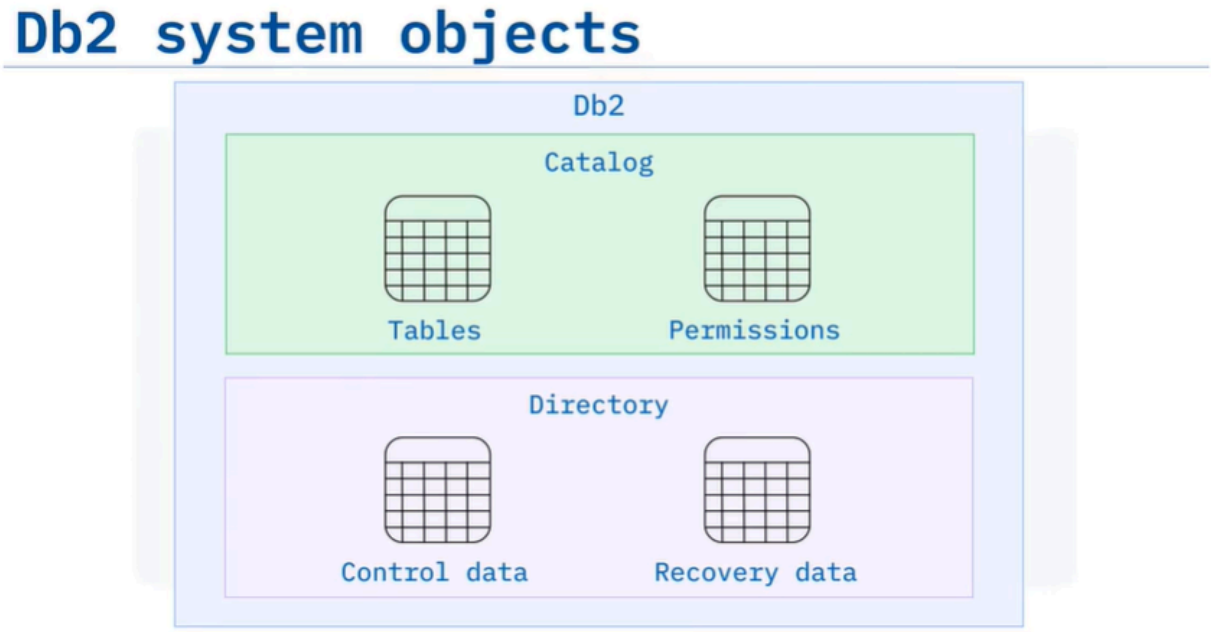




- Whether trying to find out if new indexes are helpful, the transaction count on a database at a specific time, or who's connected to the database at any given time, data that allows you to know how your databases are performing is crucial. RDBMSes store information about their databases, known as metadata, in special databases, schemas, or catalogs. They store specific types of information about the database, such as the name of a database or

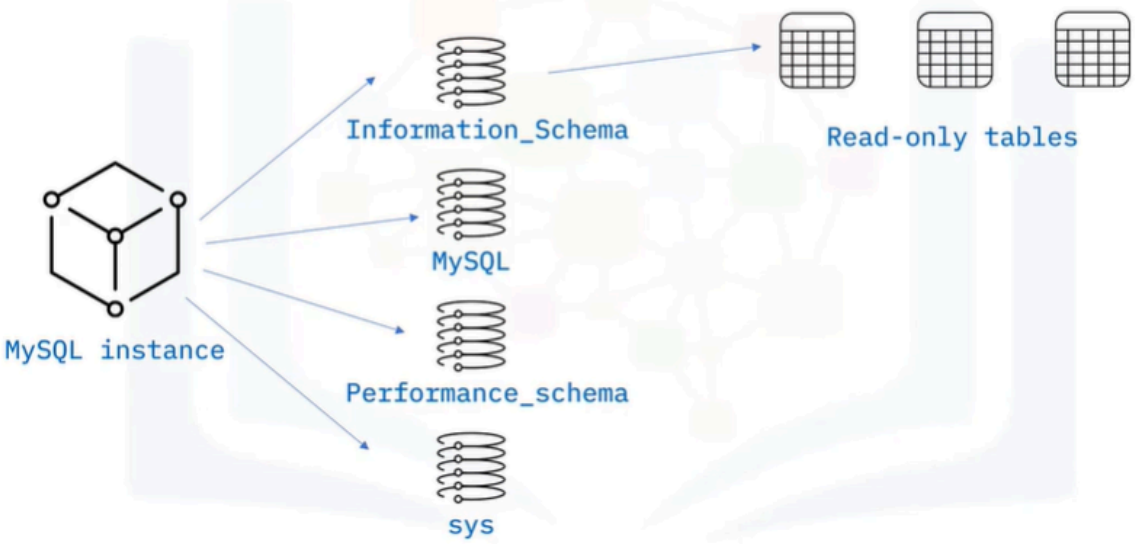
table, the data type of a column, or access privileges, known as metadata. Other terms used for this information store are data dictionary and system catalog.

- The RDBMS controls and updates the system objects, but you can query the metadata tables to discover information about the objects in the database.



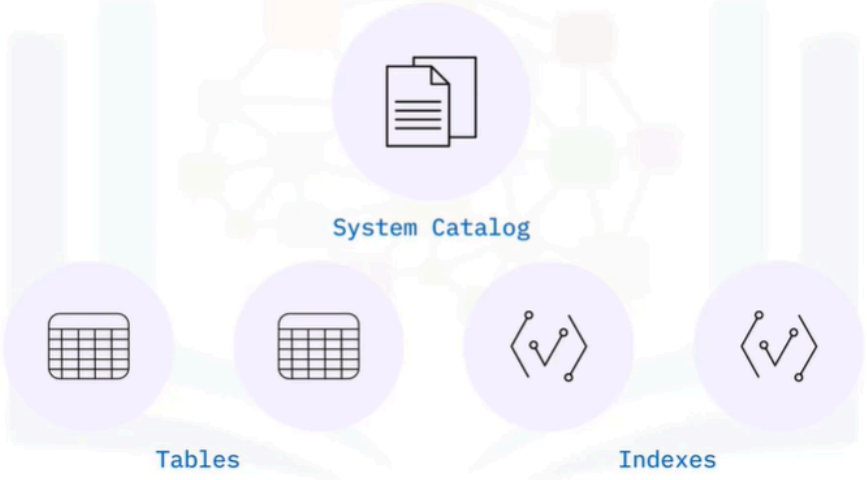
- Different RDBMSes use different names for their metadata stores: Db2 uses the catalog and the directory. The catalog consists of tables of data about everything defined to the Db2 system. When a user creates, alters, or drops any object, Db2 inserts, updates, or deletes rows of the catalog that describe that object and how it relates to other objects. The directory contains the Db2 internal control data used by Db2 during its normal operation. Among other things, the directory contains database descriptors, access paths for applications, and recovery and utility status information.

MySQL system objects

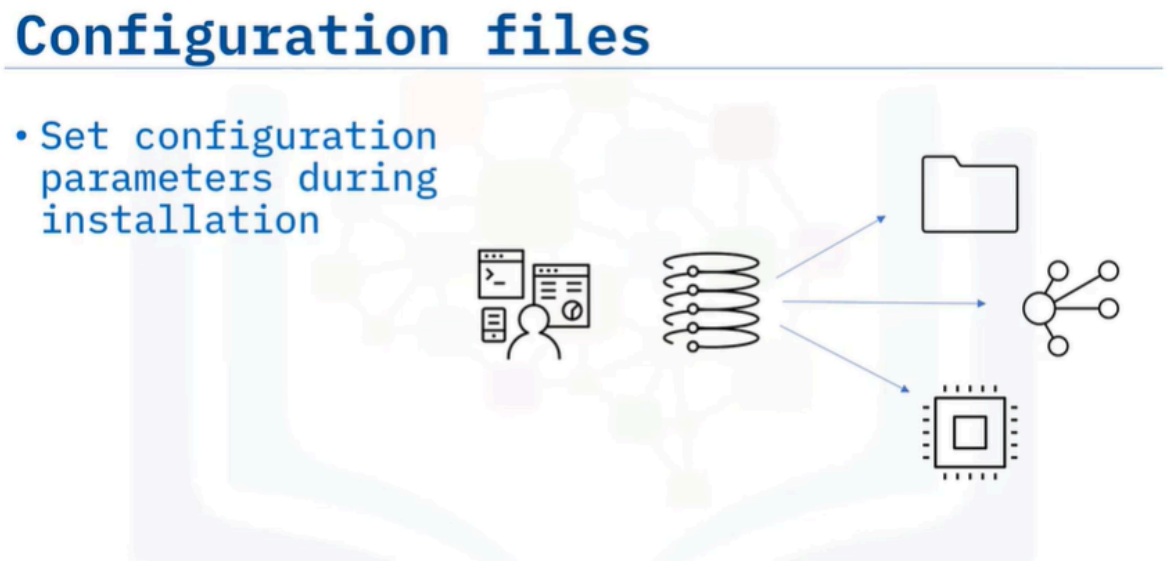


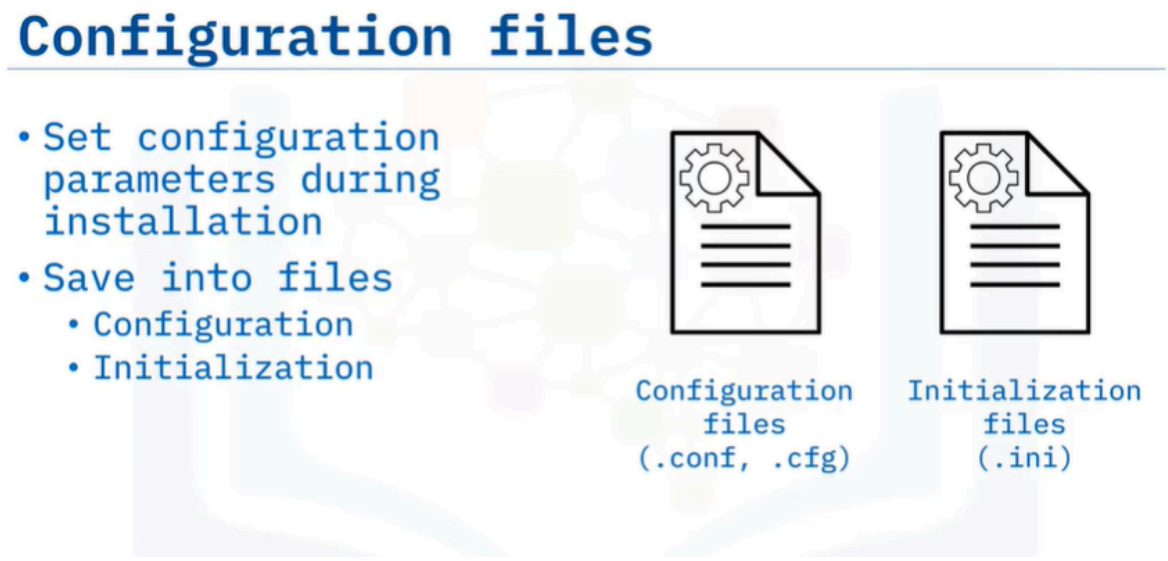
- MySQL uses a system schema to store database metadata. For example, each new MySQL instance has four system databases: information_schema, mysql, performance_schema, and sys. Each system database contains several read-only tables.

PostgreSQL system objects



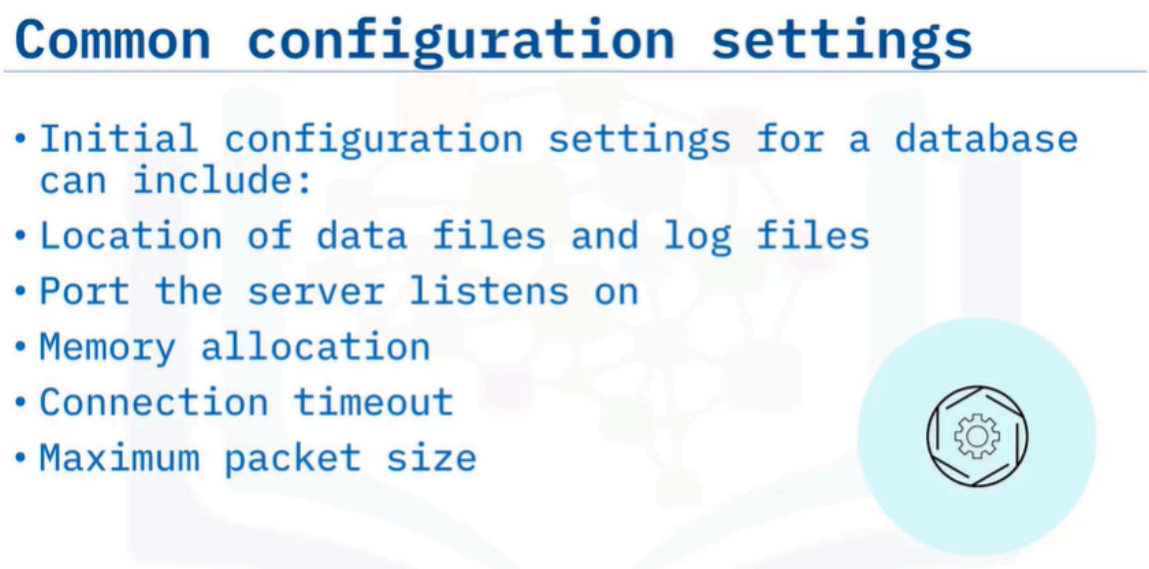
- PostgreSQL uses the system catalog, which is a schema with tables and views that contain metadata about all the other objects inside the database, and more. With it, you can discover when various operations happen, how tables or indexes are accessed, and whether the database system is reading information from memory or needing to fetch data from disk.





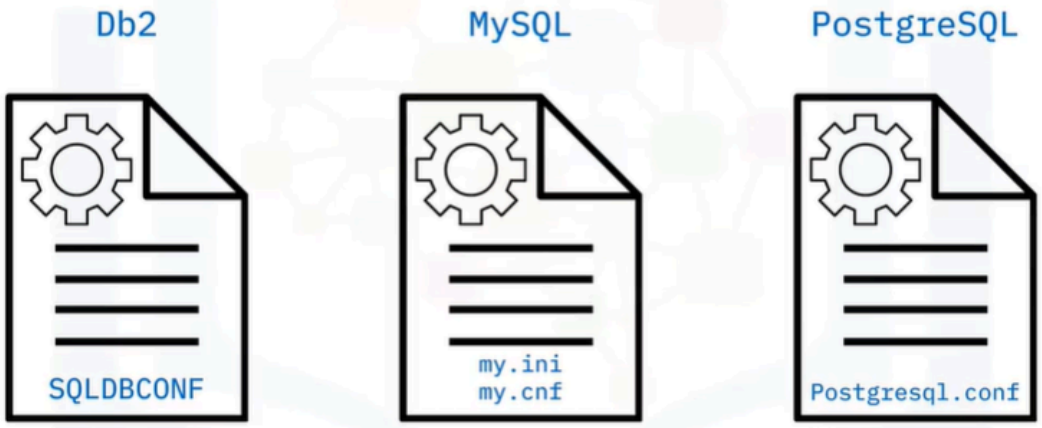
- During any database installation, you supply parameters like the location of the data directory, the port number that the service listens on for connections,

memory allocation, and much more. You can accept the default options or supply custom values to suit the environment. The database installation process saves this information into files, known as configuration or initialization files.



- The database uses the information in these files to set parameters as it starts up. Again, different RDBMSes use different files, file locations, and settings. Still, they all serve the same purpose: to provide the initial configuration information for the database as it starts up and operates. Configuration information can include general settings like the location of data and log files, and the port the server listens on for requests. Or configuration information can be more focused, like settings that affect performance, including memory allocation, a connection timeout, and the maximum packet size.

Configuration file examples



- To store configuration information: Db2 uses the SQLDBCONF file. MySQL uses my.ini files on Windows-based systems and my.cnf files on Linux-based systems. PostgreSQL uses the postgresql.conf file.

How to configure databases

On-premises	Cloud-based
1. Stop the database service 2. Modify the configuration file 3. Restart the database service	1. Use graphical tools or APIs to modify the setting 2. Database scales dynamically

- Suppose you want to modify these parameters after the database has started. In an on-premises RDBMS, you stop the database service, modify the file, and then restart the service, causing the database to read the modified settings. In

a Cloud-based RDBMS, you select configuration options as you create the database. One advantage of Cloud-based systems is that you can scale many configuration options, such as storage size and compute power, through a graphical interface as the service is running. You don't have to edit configuration files.

Summary

In this video, you learned that:

- System objects store database metadata
- Different RDBMSs use different names for their systems objects
- Configuration files store the information that the database needs as it initializes
- In an on-premise RDBMS, you edit the configuration file to change a setting
- In a Cloud-based RDBMS, you can scale settings dynamically