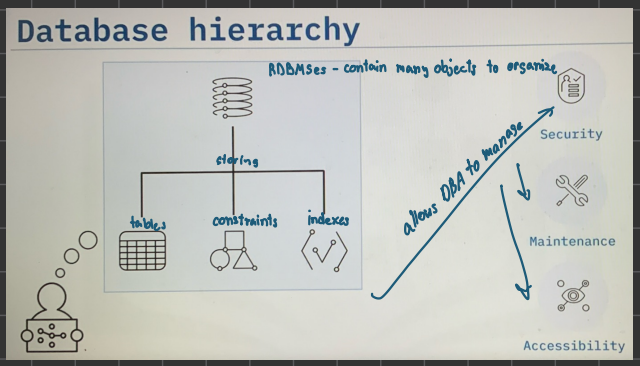
Database Objects

Objectives

After watching this video, you will be able to:

- Recall the hierarchy of database objects
- Describe an instance of a database
- Define the term schema
- List commonly used database objects

Database hierarchy

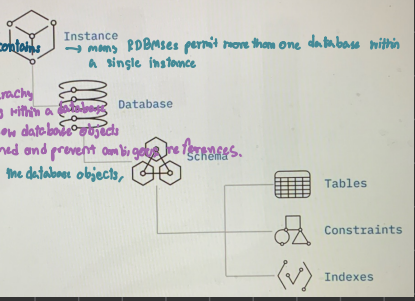


Database hierarchy

- Instance : a single way of organizing the database and everything it contains
- Database
- Schema : it is contained at some level in hierarchy; it is a logical grouping of objects within a database
- Database objects : define how database objects are named and prevent conflicts; are references.
  - Tables
  - Constraints
  - Indexes

within a schema are the database objects, including

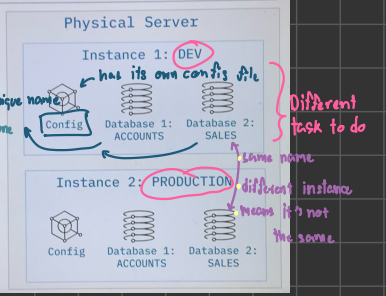
Instance → many RDBMSes permit more than one database within a single instance



Instance

- Is a logical boundary for databases, objects, and configuration
- Provides unique database server environment
- Allows isolation between databases

every database within an instance is assigned a unique name



can create more than one instance

Different task to do

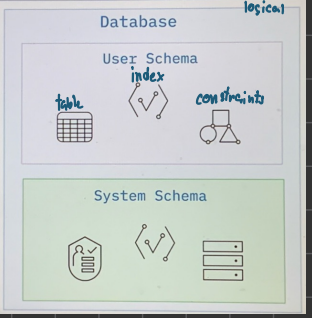
same name

different instance

means it's not the same

Schema → a specialize database object that provides a way to group other database objects logical

- Organize database objects
- Default schema is the user schema
- System schemas contain database configuration information

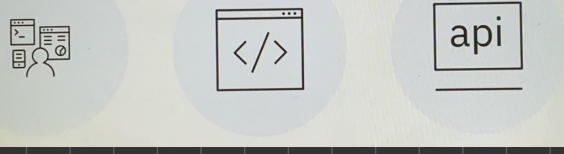


Database objects

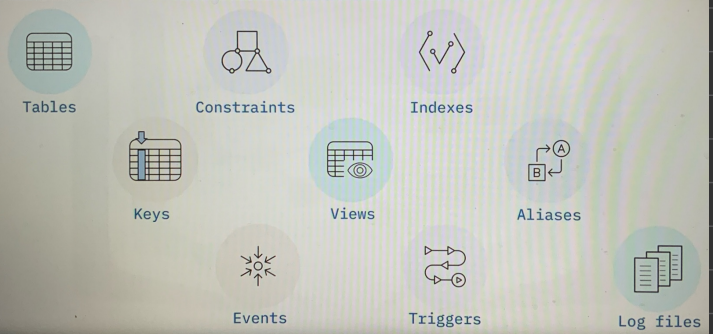
Database design includes defining database objects and their relationship to each other

Database objects

Create and manage with graphical tools, scripting, and APIs



Common database objects



System Objects and Database Configuration

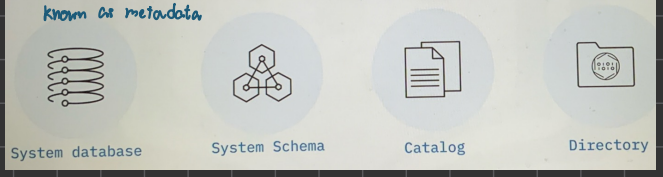
Objectives

After watching this video, you will be able to:

- Describe the purpose of system objects
- Recognize different types of system objects
- Describe how to use configuration files
- Explain common configuration settings
- Describe how to configure on-premises and cloud databases

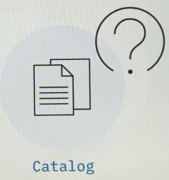
System objects

- Store database metadata in special objects
- Known as system database, system schema, catalog, or dictionary

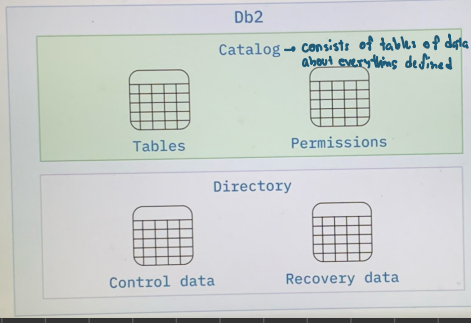


System objects

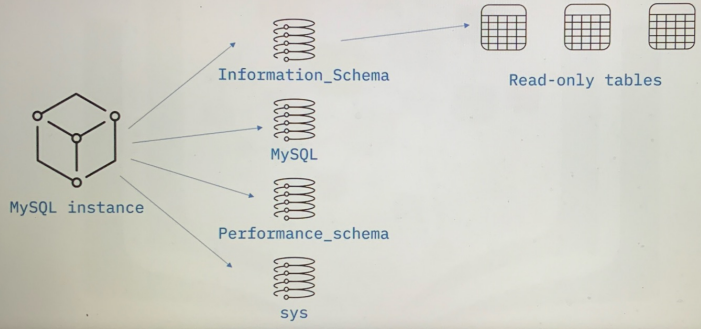
- Store database metadata in special objects
- Known as system database, system schema, catalog, or dictionary
- Query to retrieve data



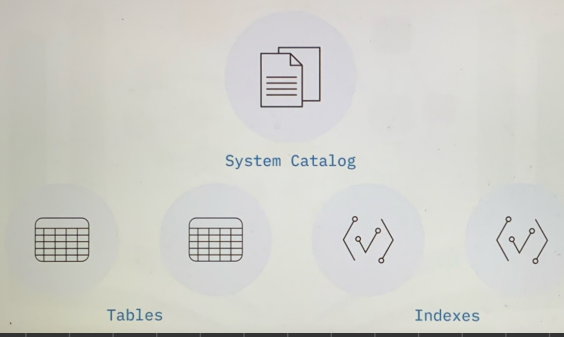
Db2 system objects



MySQL system objects

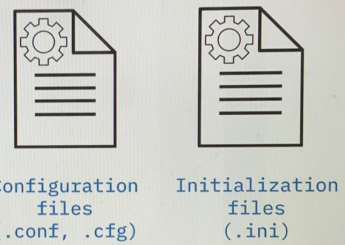


PostgreSQL system objects



Configuration files

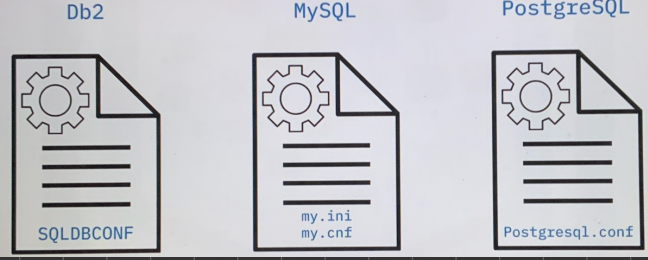
- Set configuration parameters during installation
- Save into files
  - Configuration
  - Initialization



Common configuration settings

- Initial configuration settings for a database can include:
- Location of data files and log files
- Port the server listens on
- Memory allocation
- Connection timeout
- Maximum packet size

Configuration file examples



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

Database Storage

Objectives

After watching this video, you will be able to:

- Differentiate between physical and logical storage
- Describe Tablespaces and explain their benefits
- Describe Storage Groups and their function
- Identify when to use partitions

• DBA use tablespace to logically organize database object where data is stored

→ structures that contain database object such as tables, indexes, large objects and long data

Plan database storage

Determine capacity to plan for growth

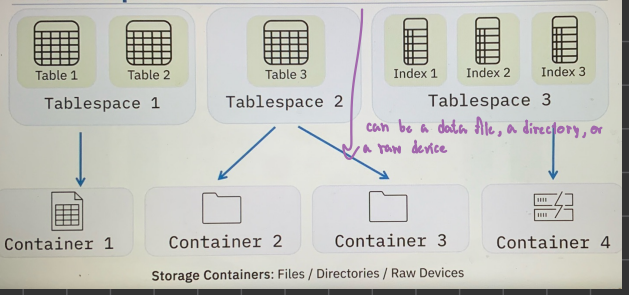
Scale cloud storage easily

Plan storage for performance

Physical storage separated from the logical database design

can manage data through a logical object without being concerned about the nature of the physical storage

Tablespaces and Containers



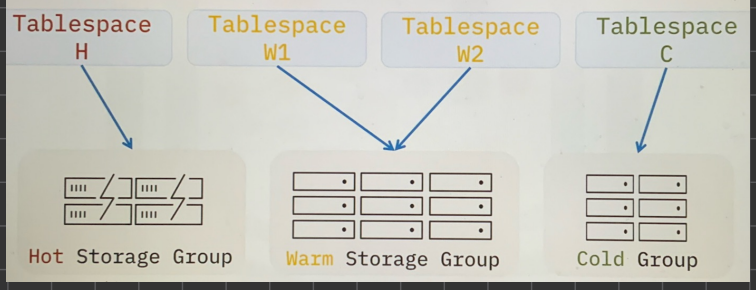
Tablespace benefits

Tablespaces separate logical database storage separate from physical storage

Performance	Recoverability	Storage management
Optimize performance - place a heavily used files on fast media	Make backup and restore operations more convenient	Extend the datafiles or containers as required

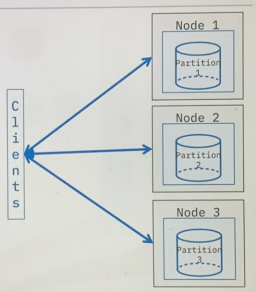
Storage groups

→ is a grouping of storage paths or container based on similar performance characteristics



Database partitions

- Data is managed across multiple partitions
- Split tables that contain very large quantities of data
- Partitions hold a subset of the data
- Common in data warehousing



Storage engines in MySQL

A storage engine is a software component that handles the operations that store and manage information in a database. MySQL is unusual among relational databases because it supports multiple storage engines.

Each storage engine has a particular set of operational characteristics, including the types of locks to manage query contention and whether the storage engine supports transactions. These properties have implications for database performance, so choose your storage engine based on the type of data you want to store and the operations you want to perform.

Most applications require only one storage engine for the whole database, but you can specify the storage engine on a table-by-table basis if your data has different requirements.

Storage engines available in MySQL

Engines	Description
InnoDB	• Default storage engine for MySQL 5.5 and later. • Suitable for most data storage scenarios. • Provides ACID-compliant tables and FOREIGN KEY referential-integrity constraints. • Supports commit, rollback, and crash recovery capabilities to protect data. • Supports row-level locking. • Stores data in clustered indexes which reduces I/O for queries based on primary keys.
MyISAM	• Manages non-transactional tables. • Provides high-speed storage and retrieval. • Supports full-text searching.
MEMORY	• Provides in-memory tables, formerly known as HEAP. • Stores all data in RAM for faster access than storing data on disks. • Useful for quick lookups of reference and other identical data.
MERGE	• Treats groups of similar MyISAM tables as a single table. • Handles non-transactional tables.
EXAMPLE	• Allows developers to practice creating a new storage engine. • Allows developers to create tables. • Does not store or fetch data.
ARCHIVE	• Stores a large amount of data. • Does not support indexes.
CSV	• Stores data in Comma Separated Value format in a text file.
BLACKHOLE	• Accepts data to store but always returns empty.
FEDERATED	• Stores data in a remote database.

Commands for working with Storage Engines

If you're a DBA working with storage engines in MySQL, you should be familiar with the following common commands.

SHOW ENGINES

Displays status information about the server's storage engines. Useful for checking whether a storage engine is supported, or what the default engine is.

```
mysql> SHOW ENGINES;
```

```
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
9 rows in set (0.28 sec)
```

CREATE TABLE

Creates a table using the storage engine specified in the ENGINE clause, as shown in the following examples:

```
CREATE TABLE Products (i INT) ENGINE = INNODB;

CREATE TABLE Product_Codes (i INT) ENGINE = CSV;

CREATE TABLE History (i INT) ENGINE = MEMORY;
```

If you do not specify the `ENGINE` clause, the `CREATE TABLE` statement creates the table with the default storage engine, usually **InnoDB**.

SET

For databases with non-standard storage needs, you can specify a different default storage engine using the set command.

Set the default storage engine for the current session by setting the default_storage_engine variable using the SET command. For example, if you are creating a database to store archived data, you can use the following command:

```
SET default_storage_engine=ARCHIVE;
```

To set the default storage engine for all sessions, set the default-storage-engine option in the my.cnf configuration file.

ALTER TABLE

You can convert a table from one storage engine to another using an `ALTER TABLE` statement. For example, the following statement set the storage enigne for the Products table to Memory:

```
ALTER TABLE Products ENGINE = MEMORY;
```

InnoDB is suitable for most data storage needs but setting and working with different storage engines in **MYSQL** gives you more control over how your data is stored and accessed. Using the most appropriate storage engine for your data brings operational benefits like faster response times or efficient use of available storage.