Week1_DatabaseManagement

The database life cycle

`Requirements Analysis -> Design and Plan -> Implementation -> Monitoring and Maintenance`

Requirements and analysis

- Understand purpose and scope of the database
    - what data is envolved
    - talk to the users and producers of data
    - develop samples on how users will use the data
- Work with stakeholders
    - Developers
    - Data engineers
    - End-users
    - Other DBAs
    - Technology managers
    - Administrators
- Analyze need for database
- Clarify goals for database
- Identify users

Design and Plan

- Work with database objects
    - Instances
    - Databases
    - Tables
    - Indexes
- Database Modelling
- Capacity and Growth Planning (determining server resources)

Implementation

- Create and configure database objects
- Grant access for database users, groups and roles

- Automate repeating tasks (backups, restores, deployments)
- Deploy data movement

Monitor and maintain

- Monitor system for performance issues
- Review reports
- Apply upgrades and patches to RDBMSes
- Automate deployments and routine tasks
- Troubleshoot issues
- Security and Compliance
    - Ensure data is secure and only authorized users can access it
    - Review logs, monitor failed logins and data access activity
    - Maintain databse permissions - grand/revoke access

Database Objects

Database hierarchy

Slight variations may occur:

`Instance -> Database -> Schema -> Database Objects`

- Instance is a logical bounday for databases, objects and configuration
- Provides unique database server environment
- Allows isolation between databases
- Schema organize database objects (tables, index, constraints, and other objects)
- Default schema is the user schema
- System schemas contain database configuration information
- Database idesign includes defining databse objects
- Create and managed with graphical tools, scripting and APIs
- Tables, constraints, Indexes, Keys, Views, aliases, events, triggers, log files

---

System Objects and Database Configuration

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

Configuration files

- Set configuratin parameters during installation (location of the data directory, port number, memory allocation, etc)
- Information saved into configuratio nfiles, and the database use this files to set parmeters as it starts up
- Initial configuration settings for a db can include: location of data files and log files, port the server listens on, memory allocation, connection timeout, maximum packet size

How to configure databases

On-premises:

1. Stop the database service
2. Modify the configuration file
3. Restart the database service

Cloud-based:

- Use graphical tools or apis to modify the setting
- Database scales dynamically

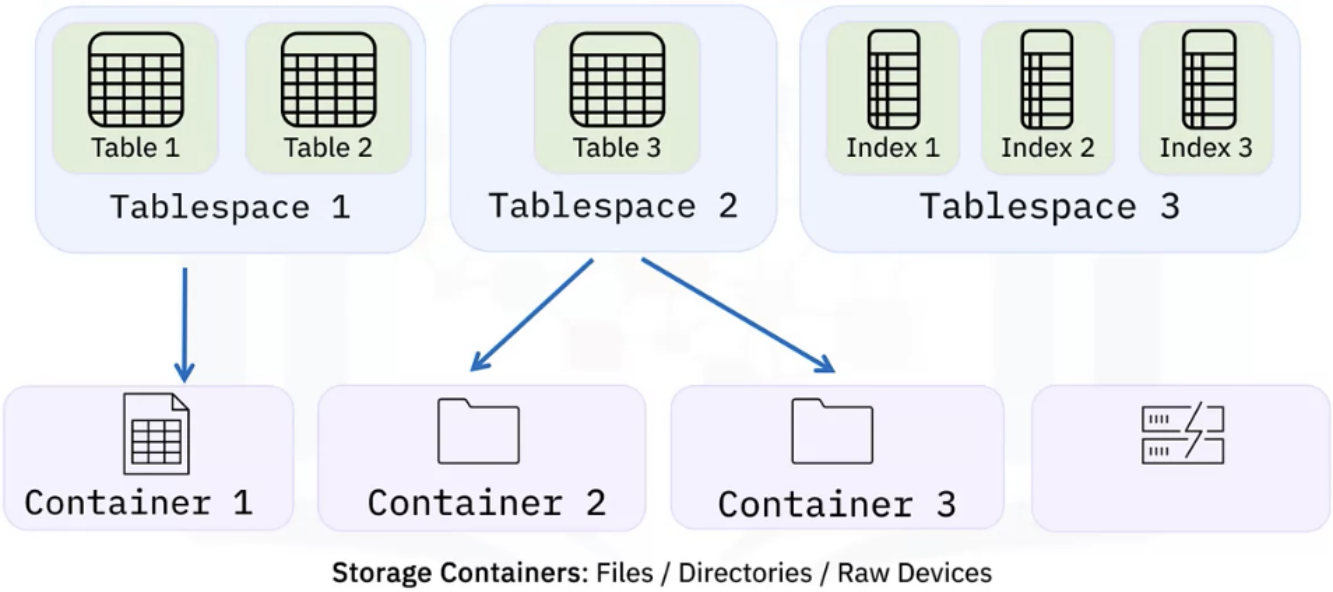
Database Storage

Plan database storage

- Determine capacity to plan for growth
- In cloud-based it scales storage easily
- On-premise you can plan storage for performance
- RDMS separates physical storage from the logical database design

Tablespaces and Containers

Tablespaces define the mapping between logical database objects and the physical storage containers that holds the data for this objects.
Storage Containers can be Files, Directories, Raw Devicies



Storage Containers: Files / Directories / Raw Devices

It optmizes performance: place a heavily used files on fast media
Recoverability: make backup and restore operations more convenient
Storage Management: extend the datafiles or containers as required

Storage Groups

It is a group of storage containers based on similar performance characteristics
You can organize your data in storage based on temperature (temperature refers to how often data is accessed (hot, warm or cold groups))

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 datawarehousing

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:

- **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;`

- **CREATE TABLE**

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

```
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.

Summary

- An instance is a logical boundary for a database or set of databases where you organize database objects and set configuration parameters.
- Common database objects are items that exist within the database such as tables, constraints, indexes, keys, views, aliases, triggers, events, and log files.
- Different RDBMSs use different names for their system objects. Most use the terms system schema, system tables, catalog, or directory.
- Database storage is managed through logical database objects and physical storage.