Week 4 - Troubleshooting & Automation

Tipo	Anotaciones
Posición	
Etiquetas	
Bibliografía	
Fecha de creación	@June 14, 2022 7:01 PM
Property	

▼ Troubleshooting & Automation

▼ Troubleshooting Common Issues

Troubleshooting basics

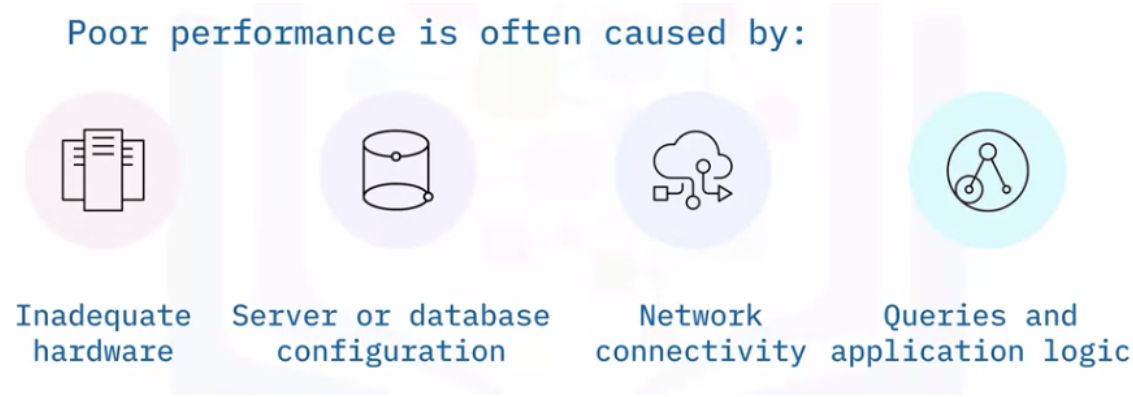
Troubleshooting is a process of identifying and then solving a problem. It begins by answering the following questions:

- What are the symptoms?
- Where is the problem happening?
- When does the problem happen?
- Under which conditions does the problem happen?
- Is the problem reproducible?

Common problems

- **Poor performance**: Poor performance can result in slow response to user queries or applications accessing the database.
- **Configuration**: Improperly configured clients, servers, or databases can cause a wide range of problems, including poor performance, crashes, errors, or even database corruption.
- **Connectivity**: Poor connectivity can cause poor performance, time outs, or a variety of errors when interacting with the database.

Common performance issues

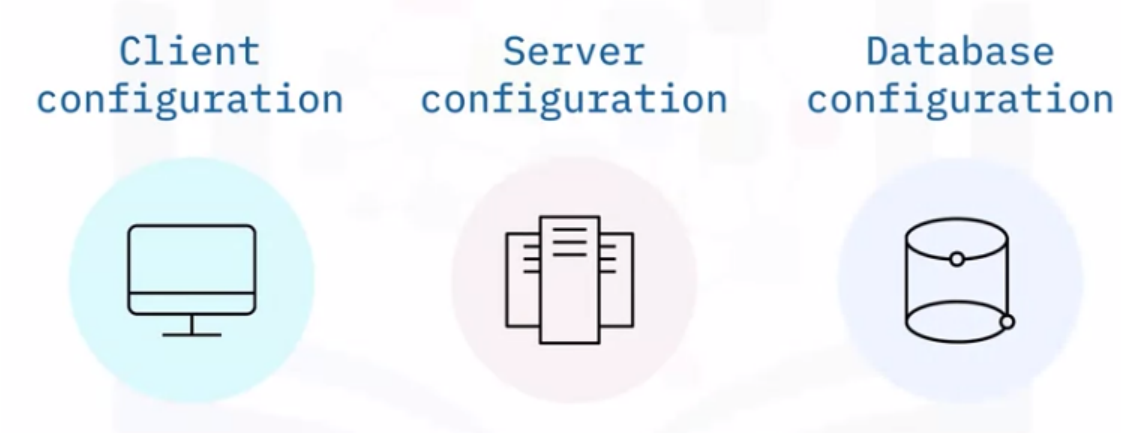


Poor performance is usually caused by high latency for disk reads and writes, slow processing time by the server, or a poor connection between the server and client. Any of these could be rooted in one or more of the following:

- Inadequate server hardware or configuration. For example, the server could be running out of disk space, memory, or processing power. Increasing these resources on the database server is called **vertical scaling**. Adding additional database partitions and shards, to improve performance is called **horizontal scaling**.
- Improper configuration. A poorly configured database may be operational but still unable to meet demand. For example, it may need to allow more connections, or it may need changes to its buffering and indexing settings to keep up with queries and return results quickly.
- Connectivity. A slow or poor network connection, or limited bandwidth between the client and database can cause high latency and processing times.

- Queries and Application Logic. A poorly written database query or improper application logic (such as unnecessary locking of database objects) can also result in performance issues.

Common configuration issues



An improperly or sub-optimally configured client, server, or database can cause any number of problems that can manifest in many ways. For example:

- A user with an incorrectly configured client or incorrect driver might be unable to connect to the database.
- A poorly or incorrectly configured server can reduce performance, cause time outs or any number of possible errors.
- The database may need to be configured to allow more connections or other settings, like caching or indexing, and may need to be adjusted to correct problems or improve overall performance.

Client configuration issues

Common issues with client configuration can be caused by incorrect login credentials, an incorrect host name or IP address, or even a corrupt or outdated connection driver. To fix these problems, check the client's driver configuration and verify the following:

- The username and password specified in the connection settings are correct. Be sure to verify that the client is also configured to use the correct type of authentication, such as Windows or SQL authentication, for example.

- The connection configuration settings, such as IP address, host name, and server name, are correct.
- The driver version for the database application is up to date and correct.

Server configuration issues

Server configuration also significantly impacts performance and operation. Some examples of things you might change or configure to improve performance or correct a problem include:

- Add more physical RAM or increase the memory assigned to the server.
- Add more physical disk space or assign additional disk space to the server.
- Consider upgrading the CPU or assigning more processing power to the server.
- Consider defragmenting the hard disk. Fragmented data degrades overall performance.
- Sometimes configuring the storage system appropriately can alleviate performance issues, for example, placing frequently accessed tables on a faster disk.
- Bugs in operating systems or in RDBMS software can result in errors and server crashes, so ensure you regularly apply software patches and security updates to guard against this.

Database configuration issues

The configuration of the database is something you need to monitor and continually evaluate to ensure it meets demand. Some examples of configuration settings you might need to change or correct are:

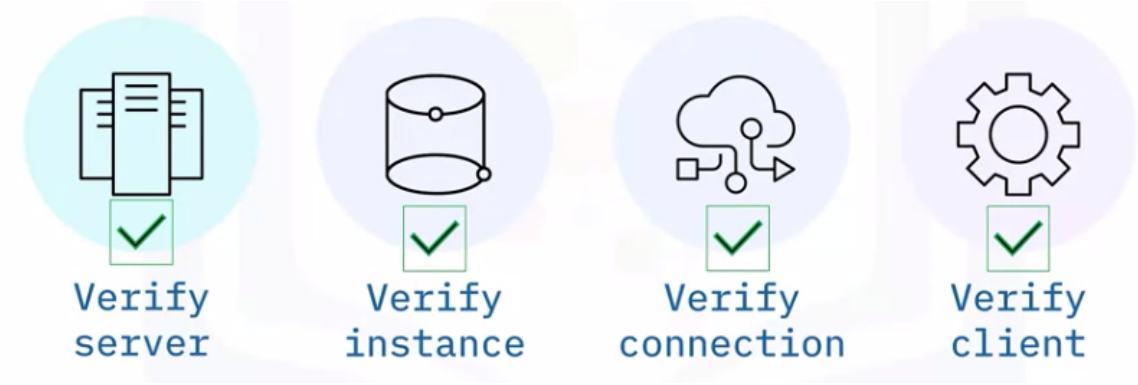
- Increase the number of allowed connections to the database to meet increasing demand.
- Change database buffering to improve performance.
- Change database indexing to improve performance.

Connectivity issues

Poor connectivity between the client and the database server can cause a wide range of problems, including poor performance, error messages, or loss of function. Some of the most common connectivity problems are typically caused by one of the following:

- The database server cannot be reached or is not running properly.
- The database instance on the server is not running properly.
- The client login credentials are incorrect, or missing security settings such as for SSL connections.
- The client configuration is incorrect.

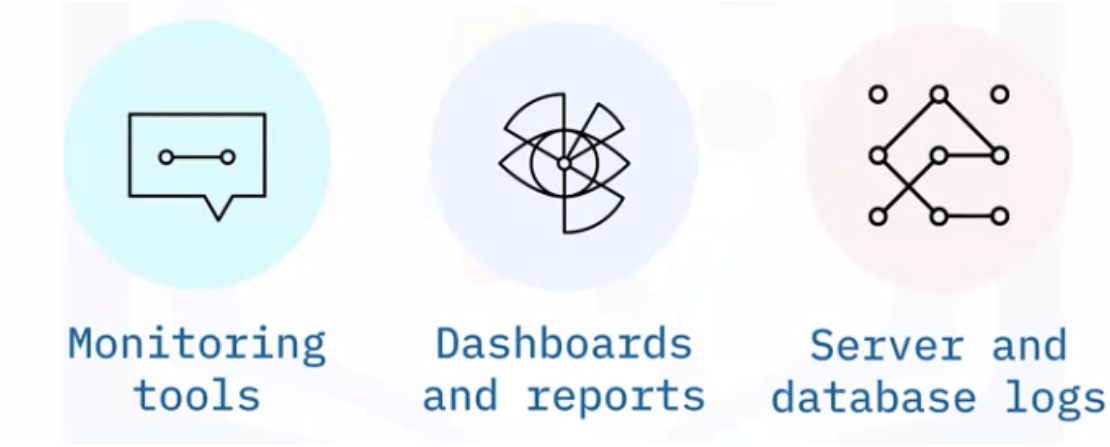
Common connectivity solutions



- Verify that the database server is running properly. The exact procedure depends on your configuration and environment. For example, you may need to physically check an on-premises server. Or you may need to verify that a virtual machine in a cloud service is running.
- Verify that the database instance on the server is running. This process varies depending on operating system and database. For example, on a Windows-based system you could use the Task Manager to verify that the instance is running. On a Db2 configuration, you could run db2cmd.exe and then issue commands in the command line.
- Verify that the database can be reached from the client. A common method is to use the PING command from the client to communicate with the server’s IP address or host name.

- Verify that the client connection driver is configured correctly.

Troubleshooting tools



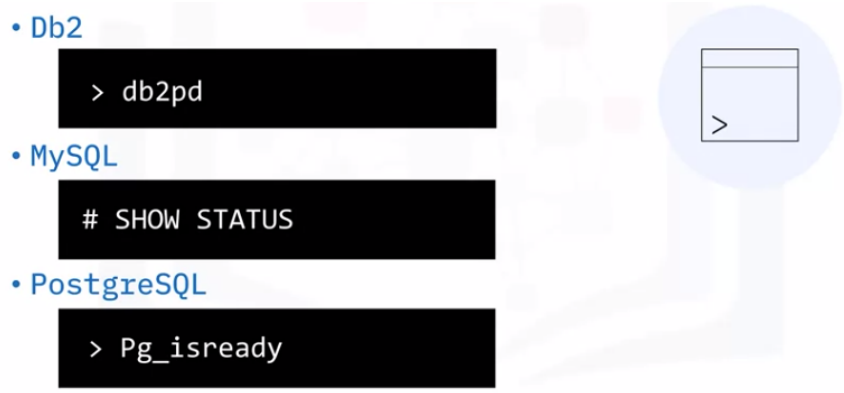
▼ Using Status Variables, Error Codes, and Documentation

Getting server status

- Databases have utilities to check health and operational status
  - Command line
  - GUI

```
#SERVICE MYSQL STATUS

[dbadm@example etc]$ su - root
Password:
[root@example ~]#
[root@example ~]#
[root@example ~]# service mysql
status
MySQL running (5089) [ OK ]
[root@example ~]#
```



Using status variables

`SHOW GLOBAL STATUS`

- A GLOBAL status variable may represent status for some aspect of the server itself (for example, Aborted_connects), or the aggregated status over all connections to MySQL (for example, Bytes_received and Bytes_sent). If a variable has no global value, the session value is displayed.

`SHOW SESSION STATUS`

- A SESSION status variable, sometimes called a LOCAL status variable, represents values for the current connection.

You can also use a LIKE clause and pattern with a SHOW SESSION statement to show status variable information that matches a specific pattern.

```
SHOW STATUS LIKE 'pattern';
SHOW STATUS LIKE 'Key%';
```

```
mysql> SHOW STATUS LIKE 'Key%';
+--------------------+----------+
| Variable_name      | Value    |
+--------------------+----------+
| Key_blocks_used    | 14955    |
| Key_read_requests  | 96854827 |
| Key_reads          | 162040   |
| Key_write_requests | 7589728  |
| Key_writes         | 3813196  |
+--------------------+----------+
```

Using GUIs to get server status

Windows SQL Server GUI Tools

- Activity Monitor

- System Monitor

Reviewing logs

There are many possible log files you can use to help identify when and where an error occurred. The most used ones are:

- **Server and OS logs**: log general server activity, connectivity, and other aspects of the server or servers running the database, and
- **Database error logs**: log information and errors specific to the database being operated, such as a MYSQL or PostgreSQL system.

Log file examples (SQL)

Log files are important tools for discovering when an error occurred and the description of the error.

- **Error Log file**: is created every time SQL server is started
- **Event Log file:** shows informational and error events
- **Default Trace Log:** is an optional log file that tracks all database configuration changes

Decoding errors

Where to find documentation and troubleshooting help:

- **Db2:** ibm.com/docs/db2
- **PostgreSQL:** postgresql.org/docs
- **MSSQL:** docs.microsoft.com/sql
- **MySQL:** dev.mysql.com/doc/

▼ Using Logs for Troubleshooting

Troubleshooting with logs

- Troubleshooting is a systematic process used to locate the cause of a fault in a computer system to provide details on correcting the relevant hardware and software issues.

- Approaching troubleshooting using a logical and systematic approach is essential to a successful resolution.
- Troubleshooting is one of the main reasons people create logs. When a problem occurs, you'll want to diagnose it to understand why it happened and what the cause was.

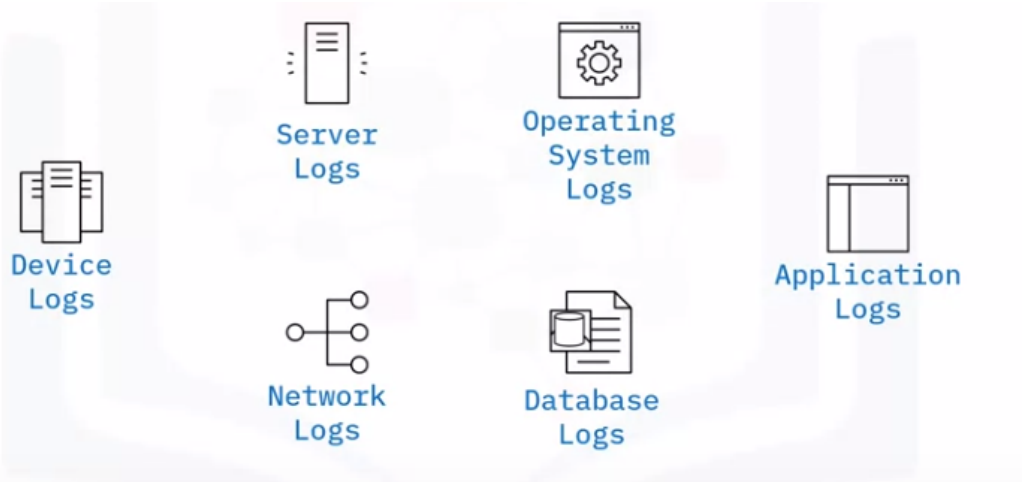
What are diagnostic logs?

A diagnostic log tracks what is happening in a system component or an application such as a web server or a database. It contains information about an event, or a problem encountered when processing requests. The diagnostic log is a record of significant events and errors in chronological order and is very useful for diagnosing or troubleshooting problems.

The diagnostic logs are also sometimes referred to as troubleshooting logs, error logs, or event logs. It is essential to recognize that they may contain events, informational messages, warnings, or errors and their details.

Some logs may have fewer or more categories of diagnostic messages such as FATAL and PANIC as additional classes of errors or may label them differently, for example, NOTE or NOTICE instead of Information.

Types of diagnostic logs

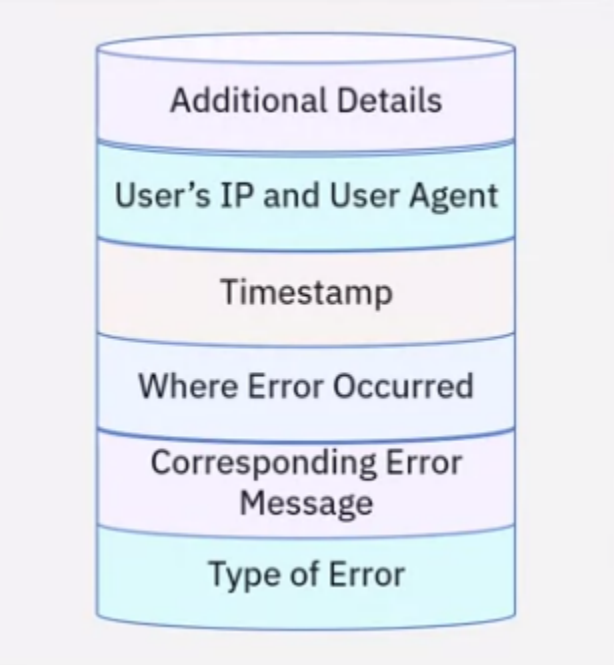


While the systems administrator may monitor the other logs, the DBA needs to be intimately familiar with the database diagnostic logs, and review them frequently. These database diagnostic logs are generally separate from the database transaction logs or query logs.

Working with log files

- May be possible to configure location of log
- Many log files are in plain text format
- Some logs may require special tools to view or filter contents

Diagnostic logs

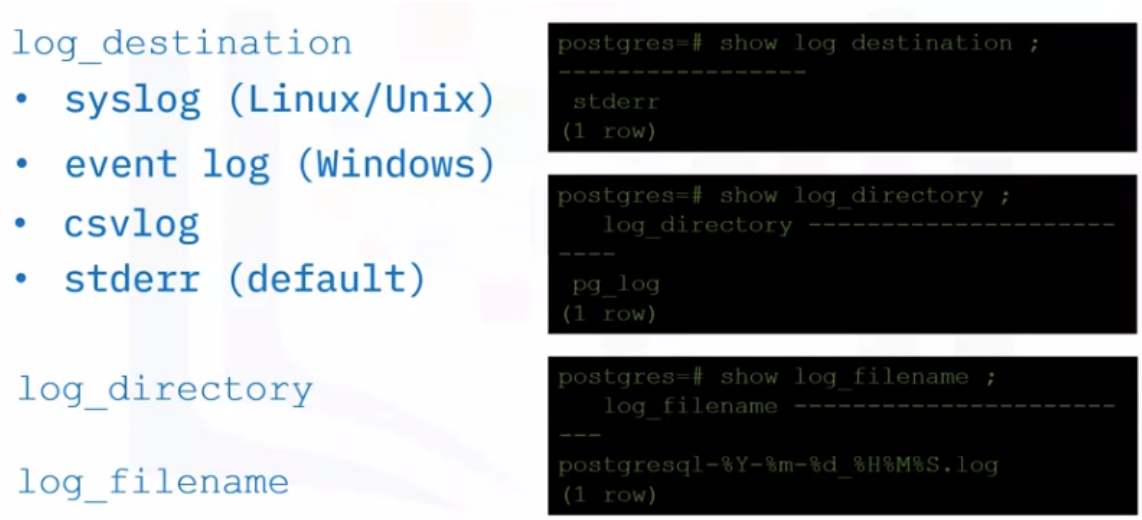


Db2 diagnostic log

```
2006-02-09-18.07.31.059000-300 I1H917            LEVEL: Event
PID     : 3140                 TID  : 2864       PROC : db2start.exe
INSTANCE: DB2                  NODE : 000
FUNCTION: Db2, RAS/PD component, _pdlogInt, probe:120
START   : New Diagnostic Log file
DATA #1 : Build Level, 124 bytes
Instance "DB2" uses "32" bits and Db2 code release "SQL09010"
with level identifier "01010107".
Informational tokens are "Db2 v9.1.0.190", "s060121", "", Fix Pack "0".
DATA #2 : System Info, 1564 bytes
System: WIN32_NT MYSRVR Service Pack 2 5.1 x86 Family 15, model 2, stepping 4
CPU: total:1 online:1 Cores per socket:1 Threading degree per core:1
Physical Memory(MB): total:1024 free:617 available:617
Virtual  Memory(MB): total:2462 free:2830
Swap     Memory(MB): total:1438 free:2213
Information in this record is only valid at the time when this file was created
(see this record's time stamp)
```

```
$> db2 get DBM CFG show detail | grep DIAG
Diagnostic data directory path (DIAGPATH) = /home/db2inst1/sqllib/db2dump
Diagnostic error capture level (DIAGLEVEL) = 4
```

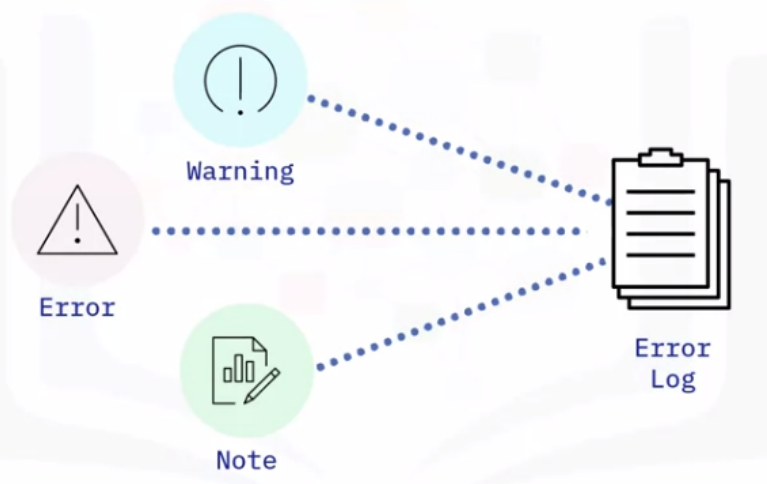
PostgreSQL server logs



MySQL server logs

MySQL has several log for troubleshooting:

- General Query log
  - Connections and queries received from clients
- Slow Query log
  - Queries that take longer than log_query_time
- Error log
  - Diagnostic and error messages from MySQL D – the MySQL daemon or server process
  - In addition to the error messages, the MySQL error log also includes warnings and notes during database server operation, startup, and shutdown



MySQL error log

- Can be configured to write to:
  - event log - enabled by default in Windows
  - syslog - used on Linux or Unix
  - stderr - enabled by default on Linux or Unix
  - specific file
- Specify using **log_error**

```
[mysqld]
log_error = /var/log/mysql/error.log
```

- Configure how much is stored in the log:

Level	Includes	Default
1	Errors	
2	Errors and warnings	
3	Errors, warnings and notes	Default

- Use **log_err_verbosity:**

```
[mysqld]
log_error_verbosity=2 # error and warning messages only
```

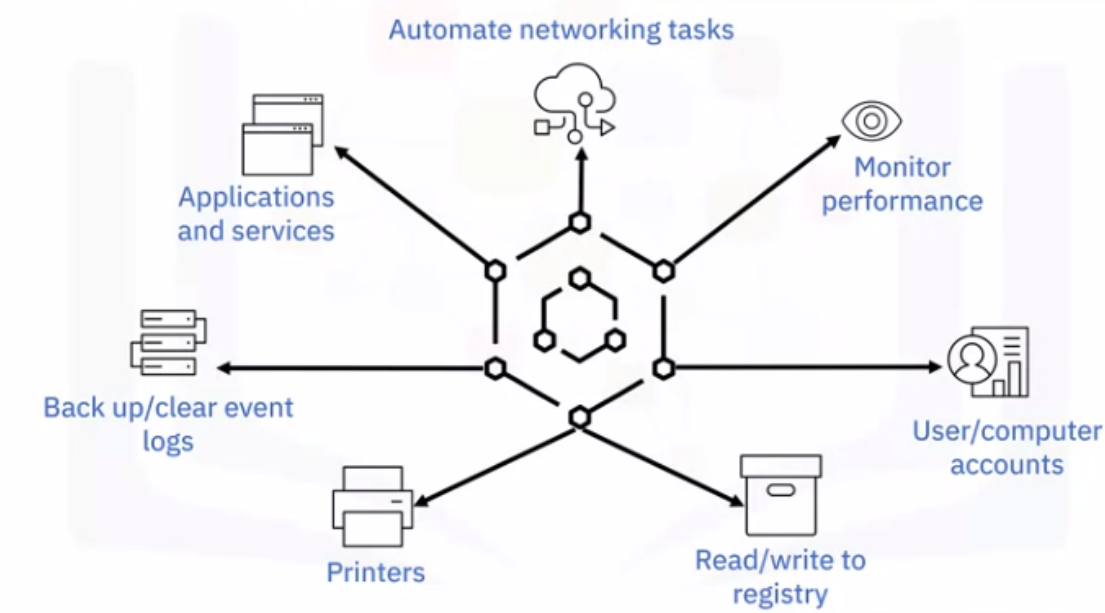
▼ Overview of Automating Database Tasks

Identifying automated tasks

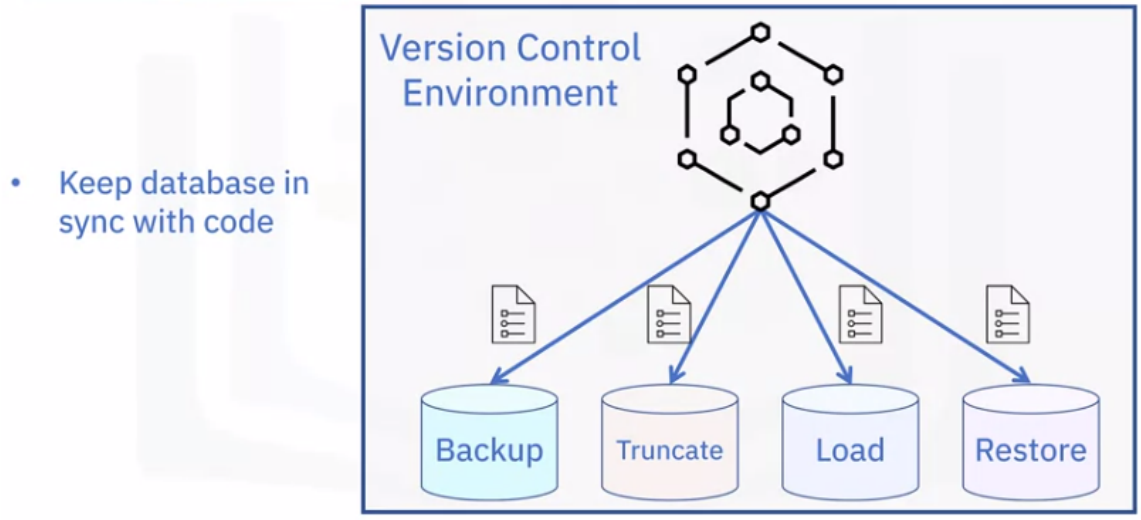
- Leverages unattended processes and self-updating procedures.
- Results in fewer deployment errors and higher reliability and speed on change implementations.
- Enables staff to focus on more important tasks and coding.
- Ideal tasks to automate are time-consuming and repetitive ones.

Using script to automate tasks

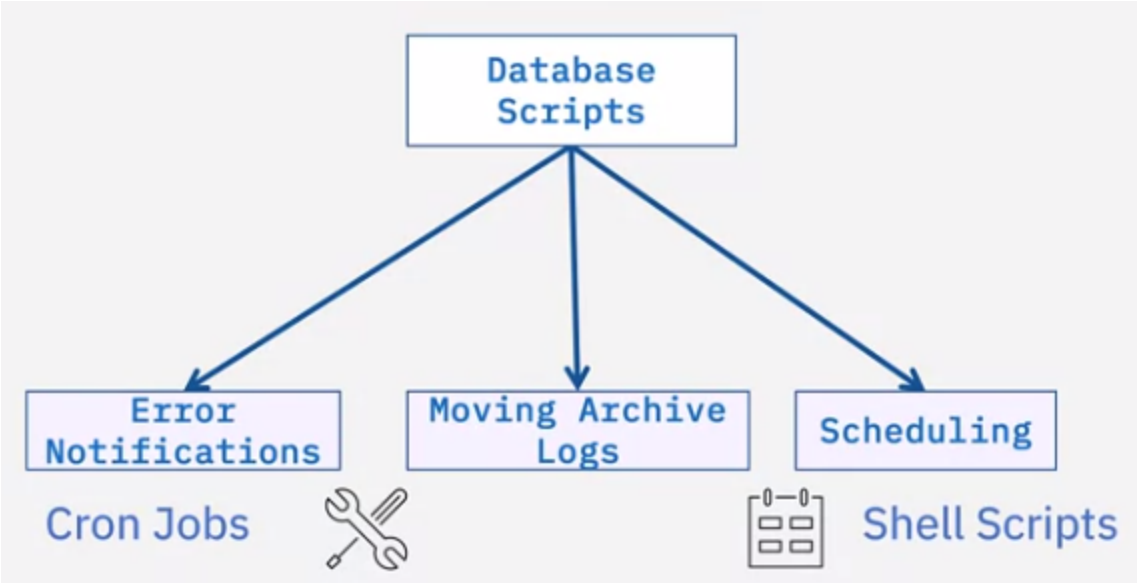
Script automation is the process of using software to leverage and re-use existing scripts to deliver automation in a managed framework without requiring custom scripts to be developed and maintained each time. Scripts are written to carry out routine, yet important, jobs which include:



There are several methods and tools for automating most database administration tasks. Some of these come built into a database, and others are performed by DBAs and developers either via scripts or database code. Common scrips that DBAs can write include processes to:



You can also write scripts to perform certain database administrator (or DBA) tasks, such as:



Advantages of automating tasks

- Increase throughput and productivity
- Improve quality or increase predictability of quality
- Improve consistency of process or product
- Increase consistency of outputs or results
- Free up staff to perform other activities

- Provide higher-level jobs in automated processes

Automated database tasks examples

- **Database Health Check** – the process of inspecting a database to see how healthy and efficient the system is.
- **Alert Log File Cleanup** – the process of deleting the chronological log of messages and errors written out by the database. Typical messages found in this file include database startup, shutdown, log switches, space errors, and so on.
- **Trace File Cleanup** – the process that deletes the trace file, or backup file, which is a snapshot that shows the process that was executing and modules that were loaded for an app at a point in time.
- **And Data Dictionary Statistics** – the process where the system gathers a collection of names, definitions, and attributes about data elements that are being used or captured in a database, information system, or part of a research project. A Data Dictionary provides metadata about data elements.
- **Database Configuration Check** – the process to check to see whether your database configuration still complies with the current recommendations for your system.
- **Schema Object check** – the process of monitoring your database changes to quickly identify the weak links and problematic queries.
- **Routine Daily Tasks using GUI Tools** – the process of performing everyday functions in the database with a GUI, for example running reports and backing up files.

Automating database testing

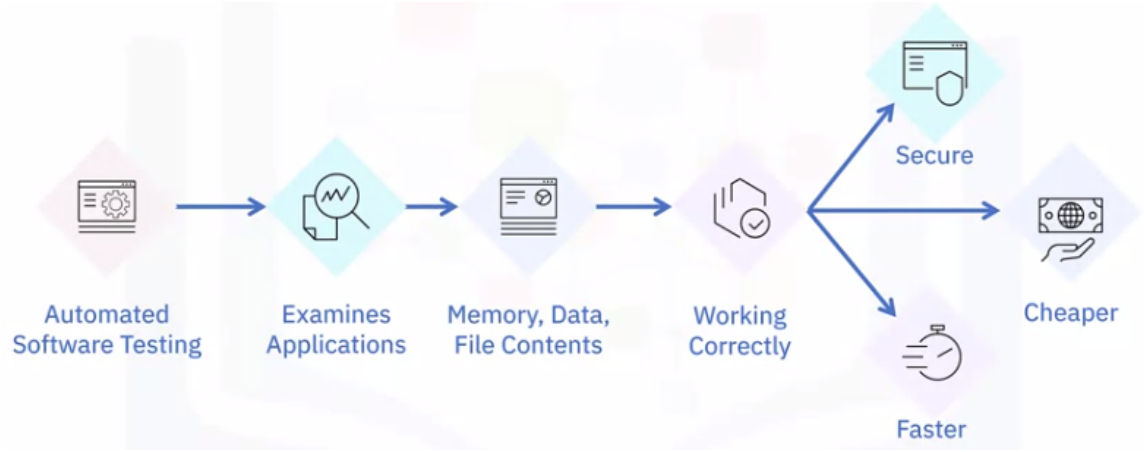
Database testing involves checking the database to ensure that everything is correct and running properly using a controlled testing environment. Database testing involves checking:

- Schema
- Tables
- Triggers, and so on

Database testing:

- Helps to prevent data loss
- Saves aborted transaction data
- Prohibits unauthorized access to the information
- Checks the integrity and consistency of data

Automating database testing benefits



▼ Automating Reports and Alerts

Reports

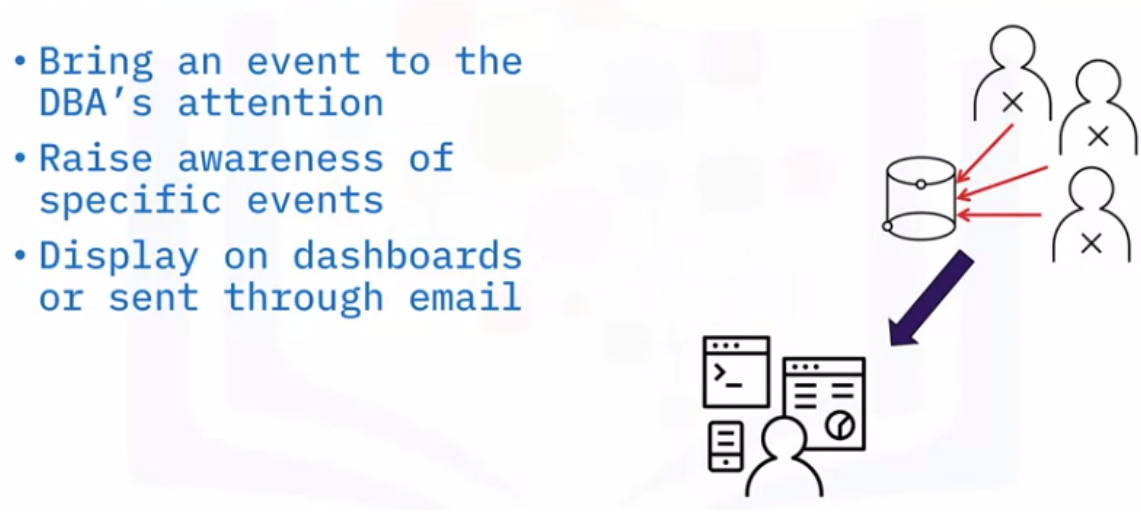
RDBMSs include reporting functionality that gives you insights into the health of your databases, like the number of users connecting successfully or failing to connect, the amount of space used and the rate of increase, and the number of queries executed against the database. You can create and configure reports with specific metrics to give you the information you need. Running reports on database health allows you to address issues before they become serious problems while allowing you to keep track of trends over time and help you to predict future needs and prepare for them. You can automate reports to run daily, weekly, or monthly, depending on your needs.

Automated reports

Some companies use third-party reporting tools that provide extra options and features.

Notifications

Notifications give you, the DBA, an opportunity to track specific database events. You might choose to be notified when a user attempts to log in but fails. A few events like this are part of daily life, as users forget or mistype their passwords, but a cluster of login failure notifications may indicate a malicious attempt to gain access to data. You can receive automated notifications through SMS messages, email, or via a dashboard. You need to configure your preferred option.



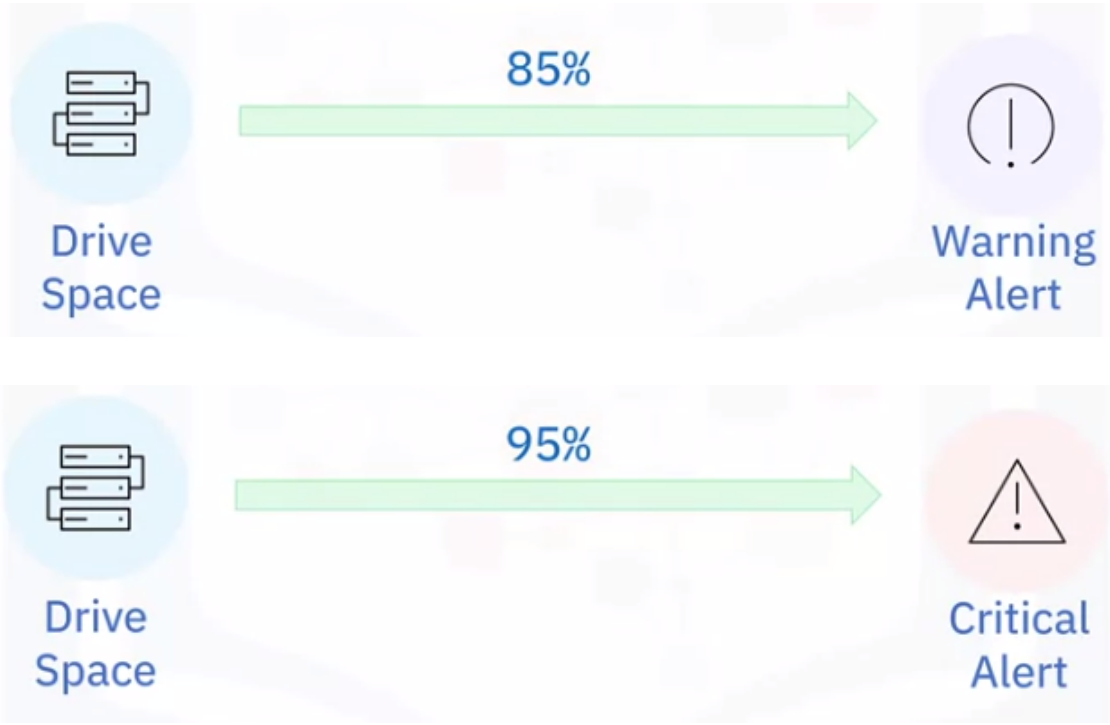
Alerts

Alerts quickly make you aware of issues that require your urgent attention. They are triggered by events like

- Catastrophically low drive space or memory
- Scheduled jobs that have failed to complete
- Error-level events in the error log

You need to determine which alerts are appropriate for your environment and configure them to immediately reach the DBA on duty. When configuring alerts and notifications, be careful not to set so many that you cannot respond to them all.

Alerts use two or more thresholds to help communicate the severity of an event. A warning threshold, when the system sends a warning message, and a critical threshold, when the system sends a critical alert message. A best practice is to send out warning alerts when the threshold for the specified event reaches 85% and a critical alert when the threshold reaches 95%. You can customize these values to suit your environment.



Automate reports, notifications, and alerts

Most RDBMSs enable you to configure the content and frequency of reporting through a graphical interface. Many provide sample reports and enable you to configure your own if necessary.

Notifications and alerts function similarly, and you can configure them through

- A graphical interface
- A command-line tool
- A script

The process of automating reports, notifications, and alerts varies depending on which RDBMS you are using.

🛠️ **Ungraded External Tool:** Hands-on Lab: Troubleshooting with PostgreSQL
Ungraded External Tool: Hands-on Lab: Automating Tasks in MySQL using Shell Scripts