



GUIA BÁSICO PARA ADMINISTRAÇÃO DE BANCOS DE DADOS

**POSTGRESQL
E
MYSQL (MARIADB)**

PROF. EDUARDO MAROÑAS MONKS

Changelog

16/02/2025	Versão inicial
09/03/2025	Ajustes nas Dicas para Gerenciamento de Banco de Dados Revisão do texto



Sumário

Sobre o Autor.....	4
Sobre esta Obra.....	5
Introdução.....	6
1) PostgreSQL.....	8
Criar banco de dados.....	8
Remover banco de dados.....	9
Criar tabela.....	10
Remover tabela.....	10
Criar usuários.....	11
Operações com usuários.....	11
Permissões de usuários.....	13
Trocar senha.....	14
Tamanho do banco.....	15
Tamanho de tabela.....	16
Modo somente leitura (manutenção).....	16
Rotinas de checagem.....	17
Modo Monousuário.....	17
Backup.....	18
Restore.....	18
Logs.....	19
Tuning.....	20
Monitoramento.....	21
psql - Atalhos.....	22
Ferramentas complementares.....	23
2) MySQL (MariaDB).....	26
Criar banco de dados.....	27
Remover banco de dados.....	27
Criar tabela.....	28
Remover tabela.....	28
Criar usuários.....	29

Operações com usuários.....	29
Permissões de usuários.....	30
Trocar senha.....	31
Tamanho do banco.....	32
Tamanho de tabela.....	33
Modo somente leitura (manutenção).....	33
Rotinas de checagem.....	33
Modo Monousuário.....	34
Backup.....	34
Restore.....	35
Logs.....	35
Tuning.....	37
Monitoramento.....	38
mysql - Atalhos.....	39
Ferramentas complementares.....	41
3) Exemplos de queries SQL.....	44
4) Dicas de Gerenciamento de Bancos de Dados.....	46
5) Considerações Finais.....	47
Referências bibliográficas.....	48

Sobre o Autor

Possui graduação em Bacharel em Análise de Sistemas pela Universidade Católica de Pelotas (1998), Mestrado em Computação pela Universidade Federal do Rio Grande do Sul (2006) e Doutorado pela Universidade Federal de Pelotas (2023). Começou a atuar na área de redes em 1995 como estagiário na UCPEL, onde trabalhou por 14 anos. No período de 1999 a 2004 e 2014 a 2017, atuou como docente em cursos de Ciência da Computação e Tecnologia em Análise e Desenvolvimento de Sistemas na UCPEL. Em 2017, passou a exercer a função de docente da UniSenac Pelotas, em que esteve na coordenação da criação do curso superior em Redes de Computadores, curso que atingiu a nota máxima no ENADE de 2017. Em 2014 tornou-se Analista de TI na UFPEL, atuando na administração de redes. Possui certificados Cisco CCAI, Cisco CCNA e LPIC-1.



Para saber mais: Canal do Youtube criado em 2009 com mais de 600 vídeos sobre a área de redes, disponível em <http://www.youtube.com/emmonks>



Aviso Importante

Sobre esta Obra

Este e-book pode ser usado de forma livre, desde que os créditos e as referências sejam publicados. As informações contidas no documento são de minha autoria, fruto de experiências de mais de 25 anos atuando na área. Nos casos em que são apontadas ferramentas ou sites, poderá haver inconsistências no acesso futuro e poderão causar indisponibilidades.

Se este e-book tiver ajudado de alguma forma, por favor, contribua com o valor que achar justo como forma de agradecimento, um cafezinho está ótimo. Obrigado!

Chave do PIX: **7f7fa84a-6ce9-45da-a42f-a77d18eb232c**



Introdução


Este livro tem como objetivo disponibilizar um guia prático para administração de bancos de dados PostgreSQL e MySQL (MariaDB). Estes bancos de dados são duas opções das mais populares em código-fonte aberto e são amplamente utilizados como repositório de dados em milhares de sistemas. Os sistemas gerenciadores de bancos de dados possuem uma linguagem padrão que é a SQL (*Structure Query Language*) usada para manipulação da estrutura e dos dados. Entretanto, diversas rotinas na administração de bancos de dados também utilizam o formato de queries SQL para interação.

A administração de banco de dados é composta por procedimentos que estão relacionados à infraestrutura e são exemplos a instalação, a configuração do serviço, a criação de usuários e de permissões, o backup e a restauração, replicação, tuning e monitoramento. A modelagem do banco de dados e as requisições para acesso aos registros por meio da linguagem SQL não são procedimentos de rotina para os administradores de redes. Entretanto, saber o básico ajudará bastante no diagnóstico de problemas!

A linguagem SQL (*Structure Query Language*) é o padrão para interação com os bancos de dados atuais. Entretanto, existem bancos de dados que não são estruturados e utilizam linguagens próprias, por exemplo o MongoDB. Os comandos SQL são interpretados pelo banco e são aplicados por meio de um cliente ou de uma console. São exemplos de ações aplicados por meio de comandos SQL:

- Criar: Tabelas, índices e outros objetos
- Ler: Extrair dados de tabelas através de consultas
- Atualizar: Modificar dados existentes nas tabelas
- Deletar: Remover dados de tabelas

A linguagem SQL é um padrão que deve ser seguido por todos os bancos de dados, porém existem particularidades em cada banco de dados. São usados comandos no formato do SQL, mas que não seguem o padrão. Estes comandos são específicos e definidos pelos desenvolvedores dos bancos de dados, basicamente é uma linguagem própria. A aplicação dos comandos e configurações de administração podem ser realizados por meio de arquivos ou com



o uso de comandos interativos direto na console do banco. Para causar efeito alguns comandos necessitam reiniciar o serviço do banco de dados. Este livro trata destes comandos específicos que são utilizados para a administração dos bancos de dados PostgreSQL e MySQL (MariaDB).

Espero que seja uma leitura proveitosa e que ajude a melhorar as rotinas de administração de bancos de dados PostgreSQL e MySQL (MariaDB).

Boa leitura!



1) PostgreSQL

O banco de dados PostgreSQL é um projeto em código-fonte aberto bastante popular. O começo foi com o Projeto POSTGRES iniciado nos anos 80 na Universidade de Berkeley. Nos anos 90, com o nome de Postgres95 foi disponibilizado como software livre. Os desenvolvedores tinham como foco o padrão SQL (*Strucuture Query Language*). Posteriormente, o nome foi simplificado para PostgreSQL. O PostgreSQL é um banco robusto, confiável e possui uma comunidade bastante ativa e está disponível para Linux, Windows, MacOS, FreeBSD entre outros. A versão do banco de dados PostgreSQL utilizada para os exemplos de comandos foi a **13.18** sendo executada em sistema operacional **Linux** na distribuição **CentOS Stream release 9**.

As configurações do PostgreSQL ficam em dois arquivos principais:

- **postgresql.conf** : É o arquivo de configuração principal, contendo parâmetros que afetam o comportamento geral do servidor
- **pg_hba.conf** : Onde estão definidas as regras de autenticação para conexões de clientes ao servidor

Os arquivos de configuração e os dados onde ficam os registros, por padrão em servidores Linux, estão localizados em /var/lib/pgsql/data (este caminho está definido no arquivo de configuração postgresql.conf). O cliente nativo é o aplicativo psql, o usuário padrão é o postgres, que tem o papel de usuário similar ao “root” (SUPERUSER). A porta de comunicação padrão é a TCP 5432.



*Os comandos precedidos de “#” devem ser executados no shell do sistema operacional. Os outros comandos são executados dentro da console do **psql***

Criar banco de dados

Cria o banco "empresa" com as configurações padrão

```
CREATE DATABASE empresa;
```

Cria o banco "empresa", sendo o dono o usuário aluno, com a codificação UTF8 e o locale en_US (Inglês americano)

```
CREATE DATABASE "empresa"  
  
WITH OWNER "aluno"  
  
ENCODING 'UTF8'  
  
LC_COLLATE = 'en_US.UTF-8'  
  
LC_CTYPE = 'en_US.UTF-8';
```

Clonar o banco empresa para empresaX e o usuário postgres será o proprietário do novo banco criado.

```
CREATE DATABASE empresaX WITH TEMPLATE empresa OWNER postgres;
```



O banco empresa deve estar sem atividade ou em somente leitura no momento da cópia para manter a integridade da cópia.

Para encerrar as atividades no banco empresa:

```
SELECT pg_terminate_backend(pg_stat_activity.pid)  
  
FROM pg_stat_activity  
  
WHERE pg_stat_activity.datname = 'empresa'  
  
AND pid != pg_backend_pid();
```

Remover banco de dados

Remove o banco de dados "empresaX"

```
drop database empresaX
```

Forçar a remoção do banco empresa, caso tenha atividades em andamento

```
drop database empresaX FORCE
```

Criar tabela

Cria a tabela fornecedores

```
CREATE TABLE fornecedores (  
    cod_forn      integer,  
    nome  varchar(80)  
);
```

Cria a tabela adesivoB a partir do conteúdo da tabela adesivo

```
INSERT INTO adesivoB (conteudo, dia, fim)  
SELECT conteudo, dia, fim  
FROM adesivo;
```

Remover tabela

Remove as tabelas adesivoB e adesivoC

```
DROP TABLE adesivoB, adesivoC;
```

Limpa o conteúdo da tabela adesivoB

```
TRUNCATE adesivoB;
```

Limpa o conteúdo da tabela adesivoB, zerando os contadores de sequência

```
TRUNCATE adesivoB RESTART IDENTITY;
```

Criar usuários

Criar usuário (role)

```
CREATE USER novo_usuario WITH PASSWORD 'senha_segura';
```

```
CREATE ROLE novo_usuario LOGIN PASSWORD 'senha_segura';
```

Exemplos:

```
CREATE USER aluno WITH PASSWORD '@T3tesT32o24@';
```

```
CREATE ROLE aluno5 LOGIN PASSWORD '@P4ssW0rd$';
```

Cria o aluno3 com privilégios de superusuário e acesso remoto

```
CREATE ROLE aluno3 SUPERUSER LOGIN PASSWORD '@P4ssW0rd$';
```



O `CREATE USER` é equivalente ao `CREATE ROLE` exceto que o `CREATE USER` inclui o `LOGIN` (permissão de acesso remoto) por default e o `CREATE ROLE` não.

Operações com usuários

Para trocar a senha do usuário aluno3

```
ALTER ROLE aluno3 PASSWORD '@P4ssW0rd$45';
```

Para remover os privilégios de superusuário do aluno5

```
ALTER ROLE aluno5 NOSUPERUSER;
```

Para retirar a permissão de conexão remota do aluno3

```
ALTER ROLE aluno3 NOLOGIN;
```

Para alterar o nome do usuário aluno2 para aluno4 (o password será resetado)

```
ALTER ROLE aluno2 RENAME TO aluno4;
```

Para configurar o tempo de validade (1º/05/2026) do usuário aluno1

```
ALTER ROLE aluno1 valid until '2026-05-01';
```

Remover um usuário

```
DROP ROLE aluno3;
```



Caso o usuário seja dono de algum objeto não será possível removê-lo antes de trocar o dono ou remover os objetos

Troca a propriedade dos objetos do aluno3 para postgres

```
reassign owned by aluno3 to postgres;
```

Remove todos os objetos de propriedade de aluno3

```
drop owned by aluno3;
```

Listar o usuário atual

```
SELECT current_role;
```

Faz um “sudo”, assumindo a identidade de outro usuário

```
SET ROLE aluno;
```

Para retornar ao usuário original

```
RESET ROLE;
```

Listar os usuários existentes

```
\du  
  
select * from pg_catalog.pg_user;
```

Permissões de usuários

Concede todos os privilégios para o usuário "aluno6" no banco de dados "empresa"

```
GRANT ALL PRIVILEGES ON DATABASE empresa TO aluno6;
```

Conecta no banco "empresa" e concede todos os privilégios nas tabelas do banco para o usuário "aluno6"

```
\c empresa  
  
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO "aluno6";
```



Para verificar os privilégios usar o comando "\z"

Conecta no banco "empresa" e concede o privilégio de SELECT nas tabelas do banco para o usuário "aluno6"

```
\c empresa  
  
GRANT SELECT ON ALL TABLES IN SCHEMA public TO "aluno6";
```

Concede todos os privilégios na tabela

```
GRANT ALL PRIVILEGES ON adesivo TO "aluno6";
```


Altera os privilégios padrão para ALL no esquema public para o usuário aluno

```
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL ON TABLES TO aluno;
```

Remove o privilégio padrão de INSERT no esquema public para o usuário aluno

```
ALTER DEFAULT PRIVILEGES IN SCHEMA public REVOKE INSERT ON TABLES FROM aluno;
```

Para listar os privilégios do usuário aluno

```
SELECT * FROM information_schema.role_table_grants where grantee = 'aluno';
```



A mudança dos privilégios padrão ALTER DEFAULT PRIVILEGES só afetam as tabelas futuras!

Trocar senha

Para trocar a senha do usuário aluno3

```
ALTER ROLE aluno3 PASSWORD '@P4ssW0rd$45';
```

Este comando altera a senha do usuário aluno6 para '@P4ssW0rd\$45'. Se a sessão estiver ativa, deve encerrar a conexão para forçar a nova senha

```
ALTER USER aluno6 WITH PASSWORD '@P4ssW0rd$45';
```

Para alterar a senha do usuário postgres (superusuário padrão)

```
ALTER USER postgres WITH PASSWORD '@P4ssW0rd$45';
```

Em caso de ser perdida a senha do usuário postgres

Alterar a forma de autenticação do localhost (127.0.0.1) para trust

Fazer um reload no serviço do Postgresql

```
# systemctl reload postgresql.service
```

Acessar a console *psql* com o usuário do Linux *postgres*

```
# su - postgres
```

```
# psql
```

Realizar a troca da senha do usuário *postgres*

```
ALTER USER postgres PASSWORD '@P4ssW0rd$45';
```



ALTER ROLE e *ALTER USER* são equivalentes

Tamanho do banco

Para listar todos os bancos de dados com o tamanho ocupado em disco

```
\/+
```

Para listar o banco “empresa” com o tamanho ocupado em disco

```
\/+ empresa
```

Para listar todos os bancos de dados com o tamanho ocupado em disco usando SQL

```
SELECT pg_database.datname AS "Bancos",  
pg_size_pretty(pg_database_size(pg_database.datname)) AS "Tamanho" FROM pg_database;
```

Para listar o tamanho em disco ocupado pelo banco de dados “empresa” usando SQL



```
SELECT pg_database.datname AS "Bancos",
pg_size_pretty(pg_database_size(pg_database.datname)) AS "Tamanho" FROM pg_database
where pg_database.datname = 'empresa';
```

Tamanho de tabela

Para listar todas as tabelas de um banco de dados com o tamanho ocupado em disco

```
\d+
```

Para listar o tamanho em disco da tabela adesivo4 do banco de dados conectado

```
SELECT
table_name
AS
"Tabela",pg_size_pretty(pg_total_relation_size(table_name::regclass))
AS
"Tamanho"FROM
information_schema.tables WHERE table_schema = 'public' and table_name='adesivo4' ORDER
BY pg_total_relation_size(table_name::regclass) DESC;
```

Modo somente leitura (manutenção)

Para bloquear todas as transações no banco (somente leitura)

```
ALTER SYSTEM SET default_transaction_read_only = on;

SELECT pg_reload_conf();
```

Para reativar as transações no banco (leitura/escrita)

```
ALTER SYSTEM SET default_transaction_read_only = off;

SELECT pg_reload_conf();
```

Rotinas de checagem

Este comando limpa e atualiza as estatísticas de uma tabela ou banco de dados, o que poder ser útil para manter o desempenho

```
VACUUM ANALYZE;
```

O comando vacuum realiza a desfragmentação de tabelas que podem reduzir bastante o espaço em disco e melhorar o tempo de buscas

```
VACUUM FULL;
```

Executa um vacuum na tabela adesivo

```
VACUUM adesivo;
```



Estes procedimentos podem causar indisponibilidade de acesso ao banco durante a execução (poderá levar horas, depende da complexidade do banco)

Modo Monousuário

Parâmetros para ativar o modo monousuário (--single).

```
# postgres --single -O -P -D $PGDATA $DB_NAME
```

-D local dos arquivos do banco

-O permite alterações nas tabelas do sistema

-P - ignora índices do sistema e atualiza os índices se as tabelas forem modificadas, importante para índices corrompidos

Exemplo de uso do modo monousuário para o banco video1. Após acessar em modo monousuário, foi executado o comando para reindexar os índices e para sair do ambiente a combinação de teclas CTRL+D.

```
# postgres --single -P -D /var/lib/pgsql/data/ video1
```

```
REINDEX SYSTEM video1;
```

CTRL+D para sair



O modo Monousuário é indicado para casos onde há corrompimento de índices ou tabelas.

Backup

Para realizar um backup (dump) do banco de dados “empresa”

```
pg_dump -U postgres -W -Fp -v -f backup_empresa.dump empresa
```

-W – solicita a senha do usuário

-Fp – formato em texto plano (padrão)

-v – modo verbose

-f – arquivo com o dump do banco



Script com rotinas completas de backup para Postgresql e MySQL disponível em https://github.com/emmonks/bkp_bancos

Restore

Para realizar a restauração do backup (dump) do banco de dados “empresa” para um novo banco de nome “empresa2”.

Como usuário postgres, criar o banco de dados empresa2

```
# createdb empresa2
```



Executar o com comando pg_restore

```
# pg_restore -d empresa2 backup_empresa.dump
```



Script com rotinas completas de backup para Postgresql e MySQL disponível em https://github.com/emmonks/bkp_bancos

Logs

Para ativar os logs, ajustar as configurações no postgresql.conf

```
log_destination = 'stderr'

logging_collector = on

log_directory = '/var/log/postgresql'

log_filename = 'postgresql-%Y-%m-%d.log'
```

O serviço do PostgreSQL deve ser reiniciado para aplicar as novas configurações

Na console psql para verificar o formato e se os logs estão ativos

```
SHOW log_destination;

SHOW logging_collector;
```

Para ativar os logs das queries em formato CSV, ajustar as configurações no postgresql.conf

```
log_destination = 'csvlog'

log_directory = '/var/log/pg_log/'

log_rotation_size = 10MB

log_truncate_on_rotation = on
```



```
log_filename = 'postgresql-%Y-%m-%d.log'
```

```
log_statement = 'mod'
```



O consumo de espaço em disco poderá ser alto com um volume grande de transações

Tuning

O tuning do PostgreSQL envolve ajustar parâmetros de configuração para otimizar o desempenho no arquivo postgresql.conf. Os ajustes são relacionados aos recursos de CPU, memória e disco do servidor.

Exemplos de parâmetros para tuning:

```
max_connections = 250
```

```
shared_buffers = 512
```

```
work_mem = 100MB
```

```
maintenance_work_mem = 512MB
```

```
wal_buffers = 16MB
```

```
default_statistics_target = 100
```

```
max_wal_size = 1GB
```

```
min_wal_size = 80MB
```



Na console usando o comando `psql` é possível realizar alterações de parâmetros em tempo de execução

Ao reiniciar o banco de dados estas configurações serão perdidas. Para manter os parâmetros deve ser usado o arquivo `postgresql.conf`

Este comando define o tamanho de memória de trabalho para a sessão atual

```
SET work_mem = '64MB';
```

Para listar todos os parâmetros

```
SHOW all;
```

Para lista o parâmetro `work_mem`

```
SHOW work_mem;
```



Os valores dos parâmetros podem ser ajustados a partir do monitoramento do servidor. Outra alternativa é o uso de simuladores para estimar os valores tal como o Pgtune <https://pgtune.leopard.in.ua/>

Monitoramento

Mostra as atividades em todos os bancos de dados e processos do sistema

```
SELECT * FROM pg_stat_activity;
```

Mostra somente as atividades do banco de dados de nome “empresa”

```
SELECT * FROM pg_stat_activity where datname='empresa';
```

Mostra estatísticas de todos os banco de dados

```
SELECT * FROM pg_stat_database;
```



Mostra estatísticas somente do banco de dados “empresa”

```
SELECT * FROM pg_stat_database where datname = 'empresa';
```

Lista as conexões de usuários no banco de nome “empresa”

```
SELECT pid, username, client_addr FROM pg_stat_activity WHERE datname = 'empresa';
```

Encerra todos as conexões ativas no banco de nome “empresa”

```
SELECT pg_terminate_backend (pid) FROM pg_stat_activity WHERE datname = 'empresa';
```

Mostra as queries em execução ordenado pelo tempo de início

```
SELECT pid, age(clock_timestamp(), query_start), username, query FROM pg_stat_activity WHERE query != '<IDLE>' AND query NOT ILIKE '%pg_stat_activity%' ORDER BY query_start desc;
```

Encerra o processo com o PID 1817

Listar os processos

```
SELECT * FROM pg_stat_activity;
```

Encerra de forma elegante

```
SELECT pg_cancel_backend(1817);
```

Força o encerramento

```
SELECT pg_terminate_backend(1817);
```

psql - Atalhos

`\s` - histórico de comandos

`\q` - para sair

`\t` - ativar o temporizador



\d - mostrar as tabelas, views

\d+ - mostrar as tabelas, views com o tamanho em disco

\dS+ - mostrar as tabelas, views, incluindo do sistema, com o tamanho em disco

\watch [SEG] repete a execução de uma query a cada X segundos

\c - conecta em um banco

\l+ - lista banco de dados, com o tamanho em disco

\d+ tabela - mostra informações sobre uma tabela

\du - lista usuário e permissões

\i nome_do_arquivo - executa comandos contidos no arquivo

Ferramentas complementares

pg_top – ferramenta similar ao top para monitoramento de processos no PostgreSQL

Instalação: # yum install pg_top

Uso básico

Como usuário postgres, executar

pg_top

Usar a letra “h” para acessar a ajuda da ferramenta

Exemplo:

Letra “Q” e o número do processo mostra a query sendo executada

initdb: Inicializa um novo cluster de banco de dados PostgreSQL.

initdb - inicializa o ambiente de dados, similar a uma formatação.

Criação do diretório de dados em /var/banco_dados

```
# initdb -D /var/banco_dados
```

-D: Especifica o diretório para os arquivos de dados.

createdb - Cria um novo banco de dados no prompt do sistema operacional.

Cria o banco de dados “empresa”

```
# createdb empresa
```

dropdb - Remove um banco de dados existente no prompt do sistema operacional.

Remove o banco de dados “empresa”

```
# dropdb nome_do_banco_de_dados
```

psql - Interface de linha de comando interativa para o PostgreSQL.

Conecta no serviço do PostgreSQL no endereço de host 192.168.10.5, na porta TCP 7000, com o usuário aluno e no banco de dados empresa

```
# psql -h 192.168.10.5 -p 7000 -U aluno -d empresa
```

Conecta no serviço do PostgreSQL no endereço de host 192.168.10.5, na porta TCP 7000, com o usuário aluno e solicita a senha no banco de dados empresa

```
# psql -h 192.168.10.5 -p 7000 -U aluno -W -d empresa
```

Lista os bancos de dados do servidor local

```
# psql -l
```

Lista os bancos de dados do servidor remoto de IP 192.168.10.5 e porta padrão TCP 5432, acessando como usuário aluno e sem solicitar senha

```
# psql -l -h 192.168.10.5 -w -U aluno
```

pg_ctl - Controla o servidor de banco de dados PostgreSQL (iniciar, parar, reiniciar).

Realiza o gerenciamento do serviço onde o diretório de dados está em /var/banco_dados

```
# pg_ctl -D /var/banco_dados start
```

```
# pg_ctl -D /var/banco_dados stop
```

```
# pg_ctl -D /var/banco_dados restart
```

```
# pg_ctl -D /var/banco_dados reload
```


2) MySQL (MariaDB)

O banco de dados MySQL é um projeto em código-fonte aberto bastante popular, com a primeira versão do MySQL lançada em 1995. O nome MySQL é uma homenagem à filha de um dos criadores, Monty Widenius chamada My. Em 2008, a Sun Microsystems adquiriu a MySQL AB, empresa por trás do MySQL, solidificando ainda mais sua posição no mercado. Em 2010, o projeto MySQL mudou de proprietário com a aquisição da Sun pela empresa Oracle. Com a aquisição por empresas, o MySQL passou a ser disponibilizado em versões comercial e comunitária. O projeto MariaDB é um fork do projeto MySQL com o objetivo de ser livre para qualquer tipo de uso e compatível com o MySQL. O MariaDB é a alternativa padrão nas distribuições Linux atuais. A porta padrão de comunicação do MySQL é a TCP 3306. A versão do banco de dados MariaDB utilizada para os exemplos de comandos foi a **10.5.27** sendo executada em sistema operacional **Linux** na distribuição **CentOS Stream release 9**.

As configurações do MySQL ficam em um arquivo principal denominado **my.cnf**. Este arquivo é o principal para a configuração do banco de dados, contendo parâmetros que afetam o comportamento geral do servidor. Normalmente, é usada a diretiva **includedir** para apontar para mais arquivos de configuração. Os arquivos de configuração ficam em /etc e o diretório dos arquivos com os dados chamado datadir está localizado em /var/lib/mysql. Entretanto, os caminhos podem variar de acordo com a distribuição ou com a configuração definida pelo administrador do sistema. Por exemplo, os caminhos do arquivo my.cnf nas distribuições CentOS e Debian:

CentOS - /etc/my.cnf

Debian - /etc/mysql/my.cnf

O cliente nativo é o aplicativo mysql que funciona como uma console de gerenciamento e operação dos bancos de dados.



Os comandos precedidos de “#” devem ser executados no shell do sistema operacional. Os outros comandos são executados dentro da console do **mysql**

Criar banco de dados

Cria o banco "empresa" com as configurações padrão

```
CREATE DATABASE empresa;
```

Cria o banco "empresa", sendo o dono o usuário aluno, com a codificação UTF8 e o locale en_US (Inglês americano)

```
CREATE DATABASE empresa DEFAULT CHARACTER SET = 'utf8' DEFAULT COLLATE 'utf8_general_ci';
```

Clonar o banco empresa para empresa_bk

```
# mysqladmin create empresa_bk mysqldump --routines --triggers empresa | mysql empresa_bk
```



O banco empresa deve estar sem atividade ou em somente leitura no momento da cópia para manter a integridade da cópia.

Remover banco de dados

Remove o banco de dados "empresaX"

```
drop database empresaX;
```

```
mysqladmin drop empresaX;
```

Remove o banco empresaX, caso exista um banco com este nome

```
drop database IF EXISTS empresaX;
```

Criar tabela

Cria a tabela fornecedores

```
CREATE TABLE fornecedores (  
    cod_forn      integer,  
    nome  varchar(80)  
);
```

Cria a tabela adesivoB a partir do conteúdo da tabela adesivo

```
CREATE TABLE adesivoB SELECT * FROM adesivo;
```

Remover tabela

Remove as tabelas adesivoB e adesivoC

```
DROP TABLE adesivoB, adesivoC;
```

Limpa o conteúdo da tabela adesivoB

```
TRUNCATE adesivoB;
```

Limpa o conteúdo da tabela adesivoB, zerando os contadores de sequência

```
TRUNCATE adesivoB;
```

```
ALTER TABLE adesivoB AUTO_INCREMENT = 1
```

Criar usuários

Cria o usuário aluno

```
CREATE USER 'aluno' IDENTIFIED BY '@P4ssW0rd$';
```

Cria o usuário aluno10 com restrições de acesso somente para localhost

```
CREATE USER 'aluno10'@'127.0.0.1' IDENTIFIED BY '@P4ssW0rd$';
```

Cria o usuário aluno11 com permissão de acesso somente da rede 192.168.10.0/24

```
CREATE USER 'aluno11'@'192.168.10.0/24' IDENTIFIED BY '@P4ssW0rd$';
```

Cria o usuário aluno12 com permissão de acesso de qualquer endereço

```
CREATE USER 'aluno11'@'%' IDENTIFIED BY '@P4ssW0rd$';
```



Nas versões mais recentes o usuário sem definição de origem de acesso assume “%” por padrão

Operações com usuários

Para trocar a senha do usuário aluno3

```
ALTER USER 'aluno13' IDENTIFIED BY '@P4ssW0rd$';
```

Para trocar a senha do usuário aluno13 para @P4ssW0rd\$ e habilitar as conexões de qualquer origem

```
ALTER USER 'aluno13'@ '%' IDENTIFIED BY '@P4ssW0rd$';
```

Remover um usuário aluno3

```
DROP USER 'aluno13';
```

Listar o usuário atual

```
SELECT CURRENT_USER();
```

Faz um “sudo”, assumindo a identidade de outro usuário

```
system mysql -u aluno13 -p
```

Listar os usuários existentes

```
SELECT user FROM mysql.user;
```

Permissões de usuários

Concede todos os privilégios para o usuário "aluno" no banco de dados “video1” vindo de qualquer origem

```
grant all privileges on video1.* to aluno@ '%' ;
```

Concede ao usuário “aluno13” o privilégio de SELECT na tabela “usuarios” no banco “video1” com acesso de localhost

```
grant select on video1.usuarios to 'aluno13'@'localhost';
```

Concede permissões de SELECT, INSERT e UPDATE no banco “video1” para o usuário “aluno13” com acesso de localhost e renova as permissões

```
GRANT SELECT, INSERT, UPDATE ON video1.* TO 'aluno13'@'localhost';
```

```
FLUSH PRIVILEGES;
```

Concede todos os privilégios para o usuário “aluno14” na tabela “usuarios” do banco “video1” com origem da rede 192.168.254.0/24

```
grant all privileges on video1.usuarios to 'aluno14'@'192.168.254.0/24';
```

Revoga todos os privilégios do usuário aluno no banco "video1" originado de localhost

```
REVOKE all privileges ON video1.* FROM 'aluno'@'localhost';
```

Revoga o privilégio de DELETE do usuário aluno13 no banco "video1" vindo que qualquer origem

```
REVOKE DELETE ON video1.* FROM 'aluno13'@'%';
```

Mostra os usuários e as origens permitidas

```
SELECT user,host FROM mysql.user;
```

Mostra as permissões do usuário aluno com qualquer origem

```
SHOW GRANTS FOR 'aluno'@'%';
```

Trocar senha

Para trocar a senha do usuário aluno3

```
ALTER USER 'aluno3' IDENTIFIED BY '@P4ssW0rd$';
```

Para trocar a senha do usuário "aluno13" para "@P4ssW0rd\$" e habilitar as conexões de qualquer origem

```
ALTER USER 'aluno13'@'%' IDENTIFIED BY '@P4ssW0rd$';
```



Um usuário poderá ter senhas e permissões diferentes de acordo com a origem

Em caso de ser perdida a senha do usuário root

Parar o serviço do MySQL

```
# systemctl stop mariadb.service
```

Executar o serviço em modo seguro (ignorar os grants) e colocar o processo em segundo plano

```
# mysqld_safe --skip-grant-tables &
```

Entrar na console (mysql -u root)

Executar os comandos

```
use mysql;
```

```
FLUSH PRIVILEGES;
```

```
ALTER USER 'root'@'localhost' IDENTIFIED BY '@P4ssW0rd$';
```



Parar o serviço em modo seguro: `mysqladmin shutdown`

Tamanho do banco

Para listar todos os bancos de dados com o tamanho ocupado em disco

```
SELECT table_schema AS 'Nome do Banco', ROUND(SUM(data_length + index_length) /  
1024 / 1024, 2) AS 'Tamanho (MB)' FROM information_schema.tables GROUP BY table_schema;
```

Tamanho de tabela

Para listar o tamanho em disco das tabelas do banco de dados empresa



```
SELECT table_name AS 'Tabela', ROUND((data_length + index_length) / 1024 / 1024, 2) AS 'Tamanho (MB)' FROM information_schema.tables WHERE table_schema = 'empresa' ORDER BY (data_length + index_length) DESC;
```

Modo somente leitura (manutenção)

Bloqueia as tabelas para operações de leitura e escrita, permitindo operações de manutenção (somente leitura)

```
FLUSH TABLES WITH READ LOCK;
```

Para reativar as transações no banco (leitura/escrita)

```
UNLOCK TABLES;
```

Rotinas de checagem

Para verificar a integridade da tabela funcionarios

```
check table funcionarios;
```

O comando analyze verifica se as estatísticas para o otimizar de queries do MySQL estão atualizadas na tabela

```
ANALYZE table funcionarios;
```

Executa uma checagem geral e otimização em todos os bancos de dados

```
# mysqlcheck --auto-repair -o --all-databases
```



Estes procedimentos podem causar indisponibilidade de acesso ao banco durante a execução (poderá levar horas, depende da complexidade do banco)

Modo Monousuário

Uma das formas de ativar algo parecido com um modo monousuário é ignorando as permissões e o acesso por rede

```
# mysqld_safe --skip-grant-tables --skip-networking
```



O modo Monousuário é indicado para casos onde há corrompimento de índices ou tabelas.

Backup

Para realizar um backup (dump) do banco de dados “empresa” no arquivo “backup_empresa_01032025_0915.sql”

```
# mysqldump -u root -p empresa > backup_empresa_01032025_0915.sql
```

Para realizar o backup de todos os bancos de dados como usuário root e solicitando senha

```
# mysqldump -u root -p --all-databases > backup_todos_01032025_0915.sql
```

Para realizar o backup da tabela “funcionarios” do banco “empresa”

```
# mysqldump -u root -p empresa funcionarios > backup_tabela_funcionarios.sql
```



Script com rotinas completas de backup para Postgresql e MySQL disponível em https://github.com/emmonks/bkp_bancos

Restore

Para realizar a restauração do backup (dump) do banco de dados “empresa” para um novo servidor com IP 192.168.10.3

```
# mysql --host=192.168.10.3 --user=root --port=3306 -p empresa < empresa.sql
```

Para restaurar a tabela “funcionarios” do banco “empresa”

```
# mysql --user=root -p empresa < backup_tabela_funcionarios.sql
```

Para restaurar todos os banco de dados

```
# mysql -u root -p < backup_todos_01032025_0915.sql
```



Script com rotinas completas de backup para PostgreSQL e MySQL disponível em https://github.com/emmonks/bkp_bancos

Logs

Para ativar os logs, ajustar as configurações no my.cnf (50-server.cnf)

```
[mysqld]
```

```
log-error=/var/log/mysql/error.log
```

```
general-log-file=/var/log/mysql/mysql.log
```

```
slow-query-log-file=/var/log/mysql/slow.log
```

O serviço do MySQL deve ser reiniciado para aplicar as novas configurações

Na console mysql para verificar o formato e se os logs estão ativos

```
SHOW VARIABLES LIKE 'log_error';
```

```
SHOW VARIABLES LIKE 'log%';
```

Para desabilitar os logs

```
SET GLOBAL general_log = 'OFF';
```

```
SET GLOBAL slow_query_log = 'OFF';
```

Para reativar os logs

```
SET GLOBAL general_log = 'ON';
```

```
SET GLOBAL slow_query_log = 'ON';
```

Para definir o tempo considerado para uma “slow query” 0 a 10 segundos

```
SET GLOBAL long_query_time = 8;
```

Utilitário mysqldumpslow apresenta um resumo dos logs de queries lentas

```
# mysqldumpslow
```



O diretório dos logs deve possuir como dono o usuário mysql
O consumo de espaço em disco poderá ser alto com um volume grande de transações

Tuning

O tuning do PostgreSQL envolve ajustar parâmetros de configuração para otimizar o desempenho no arquivo postgresql.conf. Os ajustes são relacionados aos recursos de CPU, memória e disco do servidor.

Exemplos de parâmetros para tuning:

```
[mysqld]
```

```
innodb_buffer_pool_size=1G
```



```
query_cache_size=64M
```

```
max_connections=200
```

Na console usando o comando `mysql` é possível realizar alterações de parâmetros em tempo de execução. Ao reiniciar o banco de dados estas configurações serão perdidas. Para manter os parâmetros deve ser usado o arquivo `my.cnf`

Este comando define o número máximo de conexões para 500

```
SET GLOBAL max_connections = 500;
```

Para listar todos os parâmetros

```
SHOW variables;
```

Para listar parâmetros relacionados a conexões

```
show variables like '%connections%';
```



Os valores podem ser estimados com o uso de simuladores tal como o MySQLTuner - <https://github.com/major/MySQLTuner-perl>

Monitoramento

Mostra as atividades em todos os bancos de dados e processos do sistema



```
SHOW PROCESSLIST;
```

Mostra somente as atividades do banco de dados de nome “empresa”

```
SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST where DB='empresa';
```

Mostra estatísticas de todos os banco de dados

```
SHOW STATUS;
```

Para listar o número de queries executadas

```
show status where variable_name='Queries';
```

Lista as conexões de usuários no banco de nome “empresa”

```
SELECT pid, username, client_addr FROM pg_stat_activity WHERE datname ='empresa';
```

Mostra as queries em execução no banco "empresa" ordenado pelo tempo em execução

```
SELECT id,time FROM INFORMATION_SCHEMA.PROCESSLIST WHERE DB ='empresa'
order by time desc;
```

Encerra todas as conexões ativas no banco “empresa”

```
select concat('KILL ',id,';') from information_schema.processlist where db='empresa' into
outfile '/tmp/t_empresa.txt';
```

```
source /tmp/t_empresa.txt;
```

Encerra o processo com o PID 1812

Listar as threads ordenado por tempo em execução

```
SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST order by time desc;
```

Encerra a conexão e as threads

```
kill connection 1812;
```

Encerra apenas a thread da query

```
kill query 1812;
```

mysql - Atalhos

connect (\r) - Conecta em um servidor, parâmetros são nome do banco e endereço do host

```
\r empresa localhost
```

edit (\e) - Edita o último comando com o editor definido na variável do sistema \$EDITOR (editor padrão é o vi).

Para alterar para o nano:

```
EXPORT EDITOR="nano"
```

ego (\G) - Apresenta os resultados verticalmente

```
select * from funcionarios limit 5 \G
```

exit (\q) - Mesma função do comando quit

help (\h) - Apresenta as opções de ajuda para comandos e parâmetros

```
help show grants
```

pager (\P) - Define o paginador para a saída dos comandos

```
\P less utilitário "less" como paginador
```

nopager (\n) - Desabilita a saída pelo pager e retorna para a saída padrão (stdout)

notee (\t) - Desabilita o redirecionamento dos comandos e saídas para um arquivo

prompt (\R) - Modifica o prompt padrão. Ao executar sem parâmetros retorna para o prompt padrão.

```
prompt (\u@\h) [\d]>_ --> (root@localhost) [empresa]>
```

quit (\q) - Encerra a sessão

source (!.) - Executa um arquivo com comandos sql

```
source /tmp/comandos.sql
```

status (!s) - Apresenta informações sobre o status do servidor

system (!!) - Executa um comando no shell do sistema operacional

```
system grep "show" ~/.mysql_history
```

tee (!T) - Direciona a saída dos comandos para um arquivo

```
\T /tmp/comandos.log
```

use (!u) - Acessa um banco de dados

```
use empresa;
```

SHOW databases; - lista banco de dados

SHOW tables; - lista as tabelas do banco de dados corrente

DESC nome_da_tabela; - mostra a estrutura de uma tabela

DESC funcionarios;



Mais opções de **prompt** -

<https://dev.mysql.com/doc/refman/8.4/en/mysql-commands.html>

Ferramentas complementares

mytop – ferramenta similar ao top para monitoramento de processos no MySQL



Instalação: yum install mytop

Uso básico

Para analisar os processos do banco empresa em localhost

mytop -h 127.0.0.1 -d empresa

Atalhos

d – troca de banco de dados

k – encerra uma thread

h – lista threads de um host específico

mysqlshow - lista banco de dados

mysqlshow

Lista as tabelas do banco “empresa”

mysqlshow empresa

mysqladmin - Utilitário para tarefas administrativas no shell do sistema operacional.

Cria o banco de dados “empresa”

mysqladmin create empresa

Remove o banco de dados “empresa”

mysqladmin drop empresa

Lista os processos em execução

mysqladmin processlist

Termina o processo com o ID 1812

mysqladmin kill 1812

Troca a senha do usuário root (MySQL)

```
# mysqladmin password
```

Mostra a versão do banco e outras informações

```
# mysqladmin version
```

Mostra informações resumidas sobre o estado do banco

```
# mysqladmin status
```

Para o serviço do banco de dados

```
# mysqladmin shutdown
```

Faz um reload na tabelas de permissões (grants)

```
# mysqladmin reload
```

mysql - Interface de linha de comando interativa para o MySQL.

Conecta no serviço do MySQL no endereço de host 192.168.10.5, na porta TCP 7000, com o usuário aluno e no banco de dados empresa (será solicitada a senha)

```
# mysql -h 192.168.10.5 --port=7000 -u aluno -D empresa
```

Conecta no serviço do PostgreSQL no endereço de host 192.168.10.5, na porta TCP 7000, com o usuário aluno e envia a senha no banco de dados empresa


```
# mysql -h 192.168.254.34 -p'senac2010' -u aluno10 -D empresa
```

Lista os bancos de dados do servidor local, parâmetro “-e” executa um comando SQL

```
# mysqlshow mysql -h 127.0.0.1 -u root -e "SHOW databases;"
```

Lista os bancos de dados do servidor remoto de IP 192.168.10.5 e porta padrão TCP 7000, acessando como usuário aluno10 e será pedida a senha

```
# mysql -h 192.168.10.5 --port=7000 -u aluno10 -p'@P4ssW0rd$' -s -e "SHOW databases;"
```



```
# mysqlshow -h192.168.10.5 -P7000 -u aluno10 -p'@P4ssW0rd$'
```



3) Exemplos de queries SQL

Estes são alguns exemplos de consultas SQL simples para entendimento da sintaxe do tratamento de dados armazenados em sistemas de bancos de dados.

Conta o número de funcionários

```
select count(*) from funcionarios;
```

Seleciona os funcionários que o nome contenha “ed”

```
select * from funcionarios where nome LIKE '%ed%';
```

Calcula a média de salário de todos os funcionários

```
select avg(salario) from funcionarios;
```

Seleciona os dez maiores salários

```
SELECT * FROM funcionario ORDER BY salario limit 10;
```

Soma o total de salários

```
SELECT sum(salario) FROM funcionarios;
```

Seleciona todos os dados dos funcionários os quais possuam “ed” no nome e salário maior do que 99999

```
SELECT * FROM funcionarios WHERE nome LIKE '%ed' AND salario < 99999;
```

Atualiza em 10% os salários menores do que 99999

```
update funcionarios SET salario = (salario * 1.1) where salario < 99999;
```

Atualiza em 20% os salários dos funcionários do sexo feminino

```
UPDATE funcionarios SET salario = (salario * 1.2) where sexo = 'F';
```

Troca o time dos funcionários com salário maior do que 4555555

```
update funcionarios set time='GE Brasil' where salario > 4555555;
```

Atualiza o ranking para “9” de todos os funcionários que começam com a letra “A” no nome

```
update funcionarios set ranking='9' where nome LIKE 'A%';
```

Lista todos os nomes dos times

```
select distinct(time) from funcionarios;
```

Conta todos os funcionários que possuem o nome “Silva”

```
select count(*) from funcionarios where nome LIKE '%Silva%';
```

Mostra a quantidade de torcedores de cada time

```
select count(nome) as torcedores, time from funcionarios group by time;
```

Mostra o número de funcionários por sexo

```
select count(nome) as Qte, sexo from funcionarios group by sexo;
```

Insere na tabela funcionários o nome, o time e o sexo de um funcionário

```
INSERT INTO funcionarios (nome, time, sexo) VALUES ('Fulano de Tal', 'EC Pelotas', 'M');
```

Insere na tabela funcionários o nome, o time e o sexo de três funcionários

```
INSERT INTO funcionarios (nome, time, sexo) VALUES
```

```
    ('Ciclana de Tal', 'EC Pelotas', 'F'),
```

```
    ('Beltrano de Tal', 'Farroupilha', 'M');
```

```
    ('Fulano de Tal', 'EC Pelotas', 'M');
```

4) Dicas de Gerenciamento de Bancos de Dados

O gerenciamento de bancos de dados envolve a infraestrutura dimensionada de forma adequada e os ajustes dos parâmetros disponíveis para melhor uso dos recursos do hardware. Os recursos computacionais necessários para obtenção de melhor desempenho dependerá do tipo de uso do banco de dados. Em sistemas de bancos de dados com milhares de transações por segundo os recursos necessários serão bem maiores do que em um sistema de cadastro simples com poucas transações por segundo. Como regra geral, a utilização de servidores com processadores velozes, memória RAM abundante e armazenamento de alta velocidade, como SSDs, são essenciais para lidar com as demandas de um banco de dados. Em um ambiente virtualizado ou com o uso de storages em rede, a largura de banda disponível para acesso ao armazenamento poderá comprometer o desempenho do banco de dados. Uma das estratégias utilizadas para melhorar o desempenho é o uso de *cache* em memória.

A rede desempenha um papel crucial, exigindo alta velocidade e baixa latência para garantir a comunicação eficiente entre servidores e clientes, com segmentação e redundância para evitar gargalos e interrupções. O consumo de largura de banda não costuma ser alto em cada conexão, normalmente, as queries que cabem em um pacote e as respostas consome alguns KB. O tempo de atraso na espera das respostas das queries, na grande parte das vezes, está no tempo de processamento no servidor do banco de dados. Este tipo de problema pode ser demonstrado facilmente com o uso de ferramentas tais como Tcpcat ou Wireshark, onde poderá ser analisado o tráfego na rede. No vídeo disponível em [O problema é na rede?](#) é mostrado um exemplo bem comum de atraso ocasionado no servidor do banco de dados na qual a rede leva a culpa.

A escolha de um sistema operacional estável e seguro, otimizado para bancos de dados, e a manutenção de software atualizado com patches de segurança são práticas indispensáveis. A primeira opção é o sistema operacional Linux para uso de banco de dados MySQL/MariaDB e PostgreSQL.

A segurança é primordial, principalmente, no gerenciamento de permissões de acesso por usuário e aplicações em bancos de dados e tabelas. Aliado a isto com a implementação de firewalls, sistemas de detecção e prevenção de intrusão, atualizações e auditorias em logs. O monitoramento contínuo da infraestrutura, com ferramentas que acompanham o desempenho de

CPU, memória, disco e rede, permite identificar e resolver problemas proativamente. Alguns parâmetros importantes a serem monitorados em sistemas de bancos de dados são listados na Tabela 1.

Tabela 1. Parâmetros importantes a serem monitorados

Parâmetro	Observação
Uso de CPU (carga)	Monitorar o consumo de CPU do sistema operacional e dos processos do banco de dados
Uso de memória	Monitorar o consumo de memória do sistema operacional e dos processos do banco de dados
Número de processos	Monitorar o número de processos do sistema operacional e do banco de dados
Espaço em disco	Monitorar o espaço em disco do sistema operacional e das partições do banco de dados
Porta do serviço	Monitorar se a porta de comunicação do serviço do banco de dados está ativa
Uptime	Monitorar o tempo de atividade do sistema operacional e do banco de dados (importante para reinícios inesperados do processo do banco)
Validação de permissões e autenticação dos usuário no banco	Monitorar se as permissões dos usuários estão corretas e a autenticação no banco está funcional
Número de conexões	Monitorar a quantidade de conexões no banco de dados
Espaço em disco	Monitorar o espaço utilizado pelo banco e por tabelas
Transações por Segundo	Monitorar o número de transações por segundo realizadas no banco

Tempo de resposta - Query SQL	Monitorar por meio de alguma query conhecida o tempo de resposta
Número de locks em tabelas	Monitorar o número de locks, que são bloqueios em tabelas que impedem a escrita. Em locks de maior duração poderá acarretar em bloqueios de conexões para os usuários e enfileiramento de processos.
Queries Lentas	Monitorar a quantidade slow queries (consultas lentas) (MySQL)
Taxa de acertos no cache em memória	Monitorar a porcentagem de acertos de dados que estão no cache da memória
Número de arquivos abertos	Monitorar o número de arquivos abertos pelos processos do banco de dados

5) Considerações Finais


Neste livro foram explorados os fundamentos da administração de bancos de dados MySQL/MariaDB e PostgreSQL, duas das ferramentas mais poderosas e versáteis disponíveis para profissionais de TI. Dominar os comandos SQL básicos e as rotinas de administração é, sem dúvida, uma competência crucial para administradores de rede e responsáveis pela infraestrutura de TI. As dicas disponibilizadas aqui são a base para garantir a eficiência, segurança e confiabilidade dos sistemas de informação, que tem como fundamento o uso de sistemas de banco de dados.

É importante reconhecer que a administração de bancos de dados compreende mais do que a mera execução de comandos; ela exige uma compreensão profunda da infraestrutura de TI, otimização de recursos e a capacidade de resolver problemas complexos. As rotinas apresentadas nesta obra são essenciais para a manutenção da integridade dos bancos de dados, desde a configuração inicial até o monitoramento contínuo e a implementação de backups e restaurações.

A distinção entre as áreas de infraestrutura e desenvolvimento tem se tornado cada vez mais reduzida, especialmente com a ascensão do conceito de DevOps. Este livro busca prepará-lo não apenas para administrar bancos de dados, mas também para colaborar efetivamente com desenvolvedores, integrando as melhores práticas de ambas as áreas.

Este material, no entanto, é apenas o ponto de partida. Os bancos de dados MySQL/MariaDB e PostgreSQL são vastos e estão em constante evolução, com novas funcionalidades e otimizações sendo introduzidas regularmente. É importante explorar a documentação oficial, participar de comunidades online e sempre estar atento às novas funcionalidades.

Realizar experimentos com os comandos e técnicas apresentadas neste livro em ambientes de teste, simulação de cenários de falha e análises para diagnosticar e resolver problemas. É fundamental manter um ambiente de testes o mais próximo possível ao ambiente de produção para que sejam válidas simulações realizadas.



Finalmente, espero que este livro tenha sido útil e de aplicação direta. Que ele sirva como um guia confiável no caminho para dominar a administração de bancos de dados MySQL/MariaDB e PostgreSQL.



Referências bibliográficas

Estas referências bibliográficas aqui listadas foram consultadas para validar comandos entre versões diferentes do MySQL e PostgreSQL e para obtenção de exemplos com parâmetros mais comuns dos comandos. As principais referências utilizadas foram as documentações oficiais disponibilizadas pelos desenvolvedores dos sistemas de bancos de dados PostgreSQL, MySQL e MariaDB.

PostgreSQL

PostgreSQL Documentation

<https://www.postgresql.org/docs/>

PostgreSQL ALTER DEFAULT PRIVILEGES - permissions explained

<https://www.cybertec-postgresql.com/en/postgresql-alter-default-privileges-permissions-explained/>

PGTune

<https://pgtune.leopard.in.ua/>

PostgreSQL Administration

<https://neon.tech/postgresql/postgresql-administration>

pg_activity

https://github.com/dalibo/pg_activity

pg_view

https://github.com/zalando/pg_view/

PostgreSQL - How to grant access to users?

<https://tableplus.com/blog/2018/04/postgresql-how-to-grant-access-to-users.html>

change_obj_ownership



https://github.com/trraghav/change_obj_ownership

User Management in PostgreSQL

<https://hostman.com/tutorials/user-management-in-postgresql/>

MySQL

MySQL Documentation

<https://dev.mysql.com/doc/>

MariaDB Server Documentation

<https://mariadb.com/kb/en/documentation/>

mysqldumpslow — Summarize Slow Query Log Files

<https://dev.mysql.com/doc/refman/8.4/en/mysqldumpslow.html>

MySQL Administration

<https://www.mysqltutorial.org/mysql-administration/>

mysqlcheck: Check and Repair Tables & Databases

<https://hevodata.com/learn/mysqlcheck/>

How To Reset Your MySQL or MariaDB Root Password

<https://www.digitalocean.com/community/tutorials/how-to-reset-your-mysql-or-mariadb-root-password>

MySQLTuner

<https://github.com/major/MySQLTuner-perl>

mysqlconfigurer

<https://github.com/Releem/mysqlconfigurer>





tuning-primer

<https://github.com/mattiabasone/tuning-primer>

mysqlmonitor-script

<https://github.com/haydenjames/mysqlmonitor-script>

Configuring MySQL to Use Encrypted Connections

<https://dev.mysql.com/doc/refman/8.3/en/using-encrypted-connections.html>





GUIA BÁSICO PARA ADMINISTRAÇÃO DE BANCOS DE DADOS

**POSTGRESQL
E
MYSQL (MARIADB)**

PROF. EDUARDO MAROÑAS MONKS