# Caderno de estudos de SQL

**Anotações dia 1 – 01/05/24:**

**Introdução à origem dos DB**

Na década de 50 os primeiros computadores eram militares e acadêmicos e os dados eram guardados em papeis que eram guardadas em pastas que eram guardadas em armários. As empresas que começaram a usar computadores queriam armazenar os dados de forma computadorizada.

Inicialmente esse armazenamento era feito de forma sequencial, ou seja, cada ficha era armazenada após a outra (Arquivos Sequenciais).

Depois disso surgiram os Arquivos de Acesso Direto, arquivos indexados que, como o nome já diz, forneciam acesso direto a algum registro específico.

Chegando na década 60, o departamento de defesa dos EUA realizou o CODASYL, um evento de tecnologia de onde surgiu a linguagem COBOL e com ela, os bancos de dados.

Um banco de dados é composto por algumas partes:

Base de dados: Os dados em si.

SGBD: O sistema se gerenciamento desses dados.

Linguagem de Exploração: Linguagem usada para acessar, editar etc. o banco de dados.

A IBM também foi muito importante por ter proposto um tipo de banco de dados ligados por hierarquia (modelo hierárquico), após isso ela evoluiu o formato com o modelo em rede.

Porém esses modelos ainda não eram o suficiente, então, na década de 70, criaram o modelo relacional.

Esse modelo de banco de dados possibilitou uma melhor organização do banco de dados, consultas mais simples e é muito mais fácil de implementar.

A linguagem comum criada para esse tipo de modelo foi a linguagem Structured English Query Language ou SEQUEL (trocadilho pra seek well, procura bem em inglês). Depois de um tempo o “English” saiu da sigla e ela virou apenas SQL.

Não demorou muito pra que empresas começassem a alterar a linguagem SQL para suas próprias especificações, até a ISO e a ANSI padronizarem tudo.

Dentre os principais bancos de dados que temos hoje em dia, segundo o Stack Overflow Survey 2023, os mais usados são:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente PostgreSQL

MySQL

SQLite

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente Microsoft SQL Server

MongoDB

Redis

MariaDB

Elasticsearch

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente Dynamodb

Oracle

Diagrama

Descrição gerada automaticamente

O MySQL foi criado na Suécia, um modelo simples e gratuito de SQL com código aberto e livre. Depois de um tempo foi comprado pena Sun (criadora do Java) e a Sun foi comprada pela Oracle

**Anotações dia 2 – 14/05/24:**

Chave estrangeira: Coluna ou combinação de colunas cujos valores aparecem na chave primária da tabela referenciada, é um mecanismo que permite a implementação de relacionamentos em bancos de dados.

Tipos de integridade:

**Integridade de domínio:** Define quais valores podem ser assumidos por um determinado campo.

**Integridade de vazio:** Define se um campo pode ou não ser vazio.

**Integridade de unicidade:** Define se um campo é único.

**Integridade de chave:** Define que os valores da chave primária e alternativa devem ser únicos.

**Integridade de Entidade:** Define que nenhum valor da chave primária pode ser nulo.

**Integridade Referencial:** Define que os valores que aparecem como chave estrangeira devem aparecer na chave primária da tabela referenciada.

**Anotações dia 3 – 19/05/24:**

**PRINCIPAIS COMANDOS SQL**

**Criar banco de dados:**

CREATE DATABASE <nome do db>;

**Selecinar qual banco de dados vai usar:**

USE <nome do db>;

**Criar tabela no banco de dados:**

CREATE TABLE <nome da tabela>(

<atributo 1> <tipo do atributo>,

<atributo 1> <tipo do atributo>,

...

<atributo n> <tipo do atributo>

);

**Incluir dados na tabela:**

INSERT INTO <nome da tabela>(<nome do atributo 1>, <nome do atributo 2>, ..., <nome do atributo n>) VALUES

(<valor atributo1 registro1>, <valor atributo2 registro1,...),

(<valor atributo1 registro2>,<valor atributo2 registro2,...),...;

**Selecionar dados em uma tabela:**

**Todos os dados:**

SELECT \* FROM <tabela>;

**Atributos específicos:**

SELECT <atributo1>,<atributo2>,... FROM <tabela>;

**Seleção ordenada a partir de determinado atributo:**

SELECT \* FROM <tabela>

ORDER BY <atributo que quer usar pra ordenar>

* Da pra ordenar textos, números e datas

**Seleção de registros com valor específico:**

SELECT \* FROM <tabela>

WHERE <atributo> = <valor do atributo>;

**Alterar valor de determinados registros:**

UPDATE <tabela>

SET <atributo para alterar> = <valor desejado>

WHERE <atributo para busca> = <valor para busca>;

**Apagar registros específicos:**

DELETE FROM <tabela>

WHERE <atributo> = <valor do atributo>;

**Apagar todos os registros de uma tabela:**

TRUNCATE TABLE <tabela>

**Anotações dia 4 – 21/05/24: (Curso rápido Andre Iacono)**

NoSQL é mais rápido e escalável, SQL é mais organizado e seguro.

DBMS ou SGBD’s são basicamente ferramentas que te auxiliam no processo de lidar com o DB, seja para backups, interação com outros sistemas, etc.

PopSQL é um software para mexer em mysql bem interessante, tem como trabalhar em conjunto numa mesma query.

Temos basicamente três ferramentas importantes pra usar MySQL:

MySQL Server: O servidor que roda o MySQL localmente na máquina.

MySQL Shell: CLI (command line interface) para consultas no MySQL.

**Interface gráfica do usuário

Descrição gerada automaticamente**

Diferença entre Schema e Banco de Dados:

Schema (esquema) é a representação lógica e visual da estrutura do banco de dados.

Temos os schemas lógicos e físicos, os lógicos são justamente a estruturação através de modelos conceituais e lógicos, já os físicos são as versões finais com tabelas feitas, tipos de dados, relacionamentos, etc.

É possível “criar” novas tabelas nas consultas e ainda dar nomes à elas, nesse exemplo o terceiro atributo não existia, ele criou só pra consulta.

Tela preta com letras brancas

Descrição gerada automaticamente

As consultas usando SELECT possuem vários modificadores que devem ser listados em uma determinada ordem:

SELECT

FROM

WHERE

GROUP BY

HAVING

ORDER BY

É possível usar a palavra chave ASC ou DESC nos ORDER BY para listar em ordem crescente ou decrescente.

Strings são citadas com aspas simples ‘’.

Os operadores aritméticos, relacionais e lógicos funcionam basicamente da mesma forma que no C#.

**OPERADOR IN**

Uma forma de buscar dados utilizando o WHERE é assim:

Texto

Descrição gerada automaticamenteEmbora essa forma seja boa, quando começamos a buscar muitos dados específicos, fica ruim ter que usar o OR várias vezes, então podemos usar o operador IN.

Texto

Descrição gerada automaticamenteDessa forma fica bem mais limpo e prático.

**OPERADOR BETWEEN**

Texto

Descrição gerada automaticamenteColoca os dois valores que quer usar como início e fim de um trecho, é bem intuitivo.

**OPERADOR LIKE**

O operador LIKE é usado para buscar palavras específicas dentro de strings, ele anda lado a lado com o placeholder %.

Texto

Descrição gerada automaticamente com confiança baixaAo usar o % depois da letra “a”, estou dizendo que quero buscar registros cujo nome comece com “a”, caso eu quizesse buscar registros que terminam com “a”, eu apenas escreveria LIKE “%a”, caso eu quizesse buscar registros que contenham “a” dentro deles, eu buscaria LIKE “%a%”.

Também existe o placeholder \_, que opera diferente do %. O % não se importa com quantos caracteres há antes ou depois, apenas se estão antes ou depois, já o \_ se importa, cada \_ representa um caractere antes ou depois.

**OPERADOR IS NULL**

Interface gráfica do usuário

Descrição gerada automaticamente com confiança médiaEsse operador retorna registros cujo campo citado não está preenchido, sem muito segredo.

**OPERADOR LIMIT**

Esse operador permite que façamos uma busca por um determinado número de registros porém pulando uma certa quantidade inicial, exemplo.

Tela de computador com letras e números em fundo preto

Descrição gerada automaticamente

Como dá pra ver, o primeiro número diz quantos devem ser ignorados e o segundo é a quantidade que deve ser listada a partir do último ignorado.

**Anotações dia 5 – 22/05/24: (Curso rápido Andre Iacono)**

**Tipos de dados em MySQL**

**Varchar vs Char:**

Char possui tamanho máximo de 255 bytes e é fixo, ou seja, independente do tamanho usado de fato, ele ocupa 255; já o Varchar tem tamanho máximo de 65.535 e é variável, ou seja, ocupa o armazenamento que está sendo de fato utilizado.

Existem outros tipos para texto, como Tinytext, Text, Mediumtext e Longtext, mas são menos usados.

**Int e Integer:**

Existem tipos diferentes para números inteiros e todos esses tipos podem ser UNSIGNED (sem sinal), o que impede a existência de números negativos e dobra a capacidade de números positivos; dentre eles temos:

Tinyint

Smallint

Mediumint

Int/Integer

Bigint

**Float vs Double:**

Tal qual em C#, temos ambos os tiposmas double é mais preciso.

Uma particularidade do SQL é a especificação do tamanho, precisamos especificar o tamanho de um float, double ou decimal usando FLOAT(quantidade total, quantidade de decimais), um atributo com FLOAT(10,2) terá 10 casas totais, sendo duas depois da vírgula.

**Dados Temporais:**

Date: Ano, mês e dia.

Datetime: Ano, mês, dia, hora, minuto e segundo.

Timestamp: Ano, mês, dia, hora, minuto e segundo.

Year: Ano

Time: Hora, Minuto, Segundo.

**COMANDOS EXTRAS**

Para adicionar um campo na tabela utilizamos:

**ALTER TABLE** tabela

**ADD** coluna\_nova tipo\_coluna\_nova;

Para mudarmos o tipo de uma coluna, usamos:

**ALTER TABLE** tabela

**MODIFY COLUMN** coluna tipo\_coluna\_novo;

**OPERADOR REGEXP – EXPRESÕES REGULARES**

Interface gráfica do usuário

Descrição gerada automaticamente com confiança médiaInterface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamenteEsse comando busca todas as cidades que tenham ‘a’ no nome, se fossemos criar utilizando LIKE seria assim:

Agora, existem outros simbolos úteis também:

Se eu quiser a letra que **começa** a string:

**WHERE** atributo **REGEXP** **‘^a’**;

Se eu quiser **várias** letras que **começam**:

WHERE atributo REGEXP ‘^a|^b|^c’

Como dá pra ver, o | significa um “or”, tal qual na programação.

Esses simbolos com funções especiais em uma regexp são chamados de metacaracteres.

Regex realizam três coisas diferentes através de padrões:

Buscas

Validações

Substituições

Tudo digitado entre colchetes é buscado diretamente.

Se eu tenho um conjunto de 1000 números aleatórios e escrevo **[0]**, ele retornará todos os 0 presentes nesses números; se eu colocar **[02]**, ele retornará todos os 0 e todos os 2; se eu digitar **[0-9]**, ele retornará todos os números entre 0 e 9 (no caso, todos os números da lista; se eu fizer **[0][0]** ele vai encontrar locais com ‘00’ no número, outra forma de fazer isso é com o quantificador {}, se eu digitar **[0]{2}** ele busca o equivalente a **[0][0]**; se eu digitar **[0-9]{6}** ele irá retornar os primeiros seis númerais de cada número (se for 12314515, ele irá retornar **123145**15); se eu digitar $ no final da regexp **[0-9]{6}$**, ele irá destacar os seis últimos numerais do número, se eu usar **^** ele destacará todos os números com seis digitos **^[0-9]{6}$** ; se eu quiser fazer a regex de cpf, eu posso fazer assim:

[0-9]{3}[.][0-9]{3}[.][0-9]{3}[-][0-9]{2}

Se eu quiser substituir valores do padrão, posso agrupar eles em parênteses e depois usar $ para localizar cada “grupo” que criei (o que tem dentro dos parênteses).

Se eu tenho várias datas no formato AAAA-MM-DD:

2004-04-12

E quero mudar para DD/MM/AAAA:

12/04/2004

Posso montar a regexp do formato antigo:

[0-9]{4}[-][0-9]{2}[-][0-9]{2}

Adicionar parênteses agrupando os locais que quero trocar de lugar:

**(**[0-9]{4}**)**[-]**(**[0-9]{2}**)**[-]**(**[0-9]{2}**)**

Depois disso basta eu passar a substituição assim:

$3/$2/$1

Ele colocará o bloco 3 (dias), depois o 2 (meses) e depois o 1 (anos) separados por /.

Essas duas consultas representam a mesma coisa, cidades que começam com R ou S e depois têm A, no LIKE basta o % para dizer que é no começo e o ‘a’ separado, mas no REGEXP usamos o ^ para representar o começo e precisamos especificar acentos.

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

**Anotações dia 6 – 23/05/24: (Curso rápido Andre Iacono)**

JOIN:

**Texto

Descrição gerada automaticamente** Nesse exemplo estou pegando o nome do ator em uma tabela, o nome do filme q ele fez em outra e o ano de lançamento também, sendo que tive que passar por uma tabela auxiliar.

Uma outra forma de inserir dados em uma tabela é sem especificar quais colunas você quer mexer, mas pra isso você precisa colocar o DEFAULT no id.

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Normalmente eu faço de outra forma:

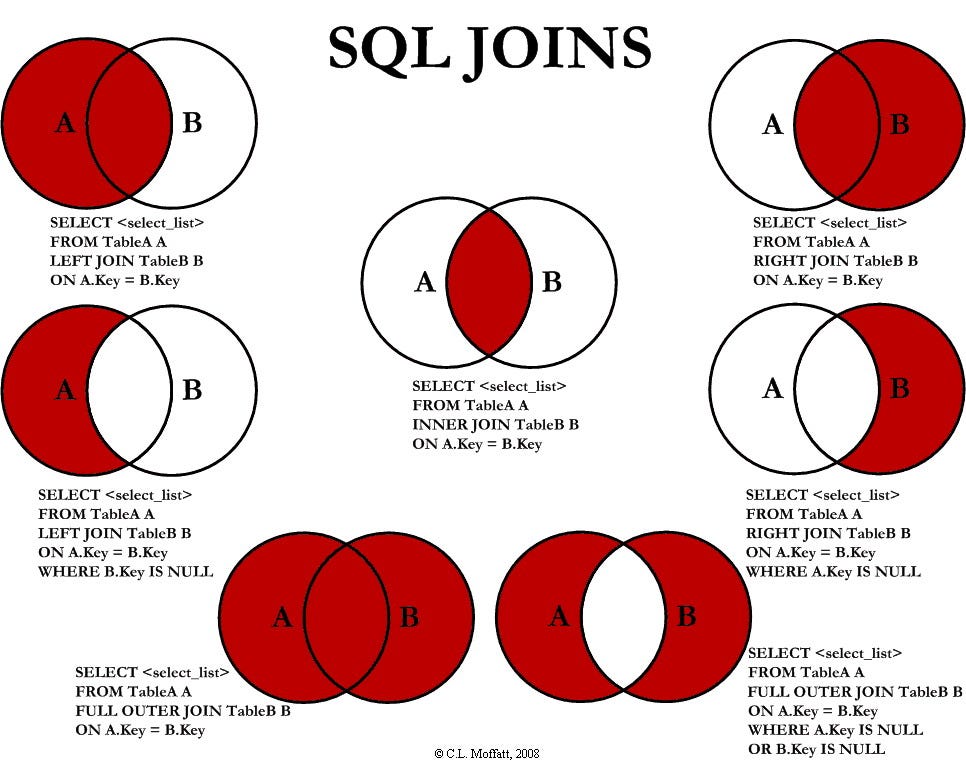
Interface gráfica do usuário, Texto

Descrição gerada automaticamente Mas é opcional.

Se eu quiser adicionar informações em múltiplas tabelas, posso usar a função last\_insert\_id(), que adicionará como chave estrangeira, o último id da outra tabela a ter sido salvo.

**JOINS**

Joins são formas diferentes de unir duas tabelas e exibir os resultados numa consulta com SELECT.



Temos diversos tipos de joins:

Inner Join – Retorna os registros que possuem relação entre as duas tabelas.

Left Join/ Left Outer Join – Retorna todos os registros da tabela à esquerda (primeira que passamos), independente de terem relação com a tabela da direita (que usamos no join), já na da direita só irão aparecer aqueles registros que possuem relação com a tabela da esquerda.

Right Join/ Right Outer Join – Retorna todos os registros da tabela da direita mas só retorna os da esquerda que possuam relação com a direita.

Full Join – Retorna todos os registros da tabela da esquerda e da tabela da direita

Semi Join – Retorna todos os registros da tabela da esquerda ou da direita que possuam interação com a tabela oposta, mas vai exibir apenas de uma das tabelas.

Anti Join – O contrário do Semi Join, retorna apenas os registros que não possuem relação com a outra tabela.

**CÓPIA DE UMA TABELA PARA OUTRA**

Há situações onde vou querer copiar uma tabela para outra dentro do banco de dados, para fazer isso é bem simples:

CREATE TABLE tabela\_backup AS

SELECT \* FROM tabela;

Pronto.

**REMOVER UMA TABELA E/OU SEUS DADOS**

Temos duas possibilidades no que tange a remover tabelas: remover seus dados mas deixar ela ou remover os dados e ela também.

O comando para remover os dados é o TRUNCATE, para remover ela é o DROP. Ambos ficariam assim:

DROP TABLE tabela;

TRUNCATE TABLE tabela;

Também temos o DELETE, que **apaga registros específicos**:

**DELETE FROM tabela**

**WHERE <condicao>;**

Se usarmos o \* no WHERE, ele apaga tudo também, mas é mais lento.

**ATUALIZAR CAMPO DA TABELA**

UPDATE tabela

SET

<atributo> = <novo valor>

WHERE

<condicao>

**FUNÇÕES SQL**

Há diversas funções prontas que realizam várias coisas de forma prática, um bom site para ver elas é o W3-Schools.

Algumas das mais comuns são:

AVG() – Faz uma Média

COUNT() – Faz uma contagem

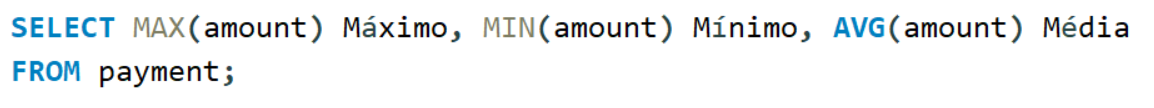
FIRST() – Primeiro valor

LAST() – Último valor

MAX() – Maior valor

MIN() – Menor valor

SUM() – Faz uma soma



**EXEMPLO DE COMANDO PARA CONSULTAR QUANTAS VENDAS E QUANTO VALOR CADA FUNCIONÁRIO ARRECADOU:**

Texto

Descrição gerada automaticamente

**EXEMPLO DE COMANDO PARA CONSULTAR QUANTAS COMPRAS CADA CLIENTE FEZ E QUANTO GASTOU EM ORDEM:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**HAVING**

HAVING é uma palavra chave similar ao WHERE, ela tem o papel de colocar uma condição para os dados aparecerem na tela.

Sabendo disso, agora temos todos os comandos do SELECT e a ordem deles:

**SELECT** – colunas que aparecerão

**FROM** – tabelas de onde as colunas serão tiradas

**WHERE** – filtros para os registros

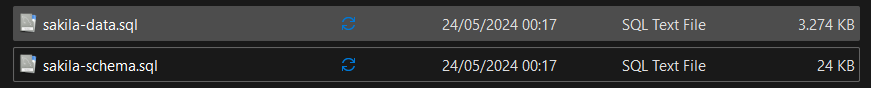
**GROUP BY** – divide os registros em grupos

**HAVING** – filtros para os registros divididos em grupos

**ORDER BY** – padrão para ordenar os registros de forma específica

**APAGAR E IMPORTAR BANCOS DE DADOS**

Apagar um banco de dados é simples, basta dar DROP DATABASE.

Importar um banco de dados pode ser feito de diversas formas, seja baixando um arquivo . sql, copiando os comandos SQL, etc.

**Anotações dia 7 – 28/05/24: (Testes práticos)**

**COMO LIDAR COM OBRIGATORIEDADE EM SQL**

Em SQL a obrigatoriedade será manipulada única e exclusivamente através do modificador “NOT NULL” ou “NULL”, se um relacionamento é obrigatório, será NOT NULL, se não, o contrário.

Exemplo prático:

Criei uma tabela aluno e professor, onde o aluno recebe a chave do professor, se eu permitir que o campo ‘id\_professor’ seja NULL, então é o mesmo que eu dizer que o mínimo de professores é zero.

Texto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente

Gráfico

Descrição gerada automaticamenteGráfico

Descrição gerada automaticamente com confiança baixa

Caso eu queira o contrário, é através de Triggers, mas vamos evitar por enquanto.

**COMO LIDAR COM 1:1 EM SQL**

Os relacionamentos em SQL são por padrão 1:N, caso não especifiquemos nada. Para fazermos um relacionamento 1:1, basta adicionar o UNIQUE na chave estrangeira de uma das tabelas, pois antes cada registro de, por exemplo, aluno, teria um professor, mas cada professor poderia aparecer mais vezes na tabela aluno; já com o UNIQUE, ele só pode aparecer uma vez, sendo então 1:1, a obrigatoriedade nesse contexto opera da mesma forma, os alunos podem ser obrigados a ter professor através do NOT NULL, mas o contrário é só com Trigger.

**Anotações dia 7 – 31/05/24: (Curso Rápido)**

**SUBQUERIES**

Uma subquery é uma query (consulta) feita dentro de outra consulta. Um exemplo bom é quando precisamos fazer uma consulta para retornar o valor médio de uma coluna e depois outra para exibir os registros que estão acima dessa média, para isso podemos fazer uma query consultando todos os registros e, no WHERE, passar a query da média entre parênteses.

SELECT \*

FROM payment

WHERE AMOUNT > (

SELECT AVG(amount)

FROM payment);

**VIEWS**

Views são uma espécie de função que podemos criar que armazena uma determinada consulta sob um nome específico para ser usada em um banco de dados, com ela podemos reduzir uma consulta de várias linhas a apenas uma linha.

O comando para criar é simples:

CREATE VIEW <nome da view> AS

<consulta>.

Inclusive é possível modificar a view adicionando cláusulas extras, exemplo:

SELECT \* FROM <view que não tem WHERE>

WHERE <condicao>

Também é possível dar DROP ou atualizar Views;

Dar drop:

DROP VIEW <nome da view>

Atualizar:

CREATE OR REPLACE VIEW <nome> AS

<consulta>

**SEGURANÇA E CRIAÇÃO DE USUÁRIOS**

Existe um banco de dados dentro do MySQL chamado justamente de ‘mysql’ que contém diversas informações sobre o software, dentre essas informações temos a tabela de usuários e quais autorizações eles possuem, o nome da tabela é ‘user’.

**Criando um usuário:**

CREATE USER <nome> IDENTIFIED BY <senha>;

Se colocarmos o nome sozinho, vai ser definido que aquele usuário pode acessar o db de qualquer local, caso queiramos especificar de qual máquina o usuário pode acessar, podemos usar um @ e o local, por exemplo: <nome>@localhost, ou <nome>@<ip da máquina>, ou ainda <nome>@<domínio>.

**Removendo um usuário:**

Para remover um usuário o comando também é bem simples.

DROP USER <nome>@<host>;

Se não foi especificado um host, deve-se colocar @%.

**Trocar senha de um usuário:**

Para trocar a senha de um usuário, pode-se fazer isso via código SQL ou, no caso do Workbench, via interface. Se for via código fica assim:

Para definir a senha do próprio usuário:



Para definir a senha de outro usuário:



**Aplicando privilégios a usuários:**

Ao criar um usuário, ele não possui nenhuma autorização, é preciso dar elas manualmente.

Para ver quais são os privilégios do seu próprio usuário:



Para ver os privilégios de usuários específicos:



Agora, para adicionar privilégios a um usuário, podemos usar o comando GRANT;

GRANT SELECT, INSERT, UPDATE, DELETE, <outros comandos>

ON <banco\_de\_dados>.<tabela>

TO <usuário>;

Vale lembrar que os comandos mostrados ali são exemplos e que caso queira permitir determinadas permissões para todos os bancos de dados, basta usar \*, caso queira para todas as tabelas de um banco de dados, basta usar <banco\_de\_dados>.\*

**Aplicando privilégios de adminstrador:**

Para aplicar privilégios de admin em um usuário, seja em determinado DB ou para todo o servidor, basta trocar os comandos por ALL.

Uma imagem contendo Texto

Descrição gerada automaticamente ou Uma imagem contendo Texto

Descrição gerada automaticamentee Texto

Descrição gerada automaticamente

**Removendo privilégios:**

É bem simples, basta trocar o GRANT para REVOKE e também o TO para FROM.

Uma imagem contendo Texto

Descrição gerada automaticamentePor algum motivo ele diz que tá errado, mas o comando rodou .