Nota: Este material representa uma cópia resumida do capítulo 5 do livro Sistemas de Gerenciamento Sistemas de Gerenciamento de Banco de Dados - Ramakrishnan Gehrke (3ª edição), versão disponível em pdf na internet.

Linguagem SQL

A Linguagem de Consulta Estruturada (SQL — Structured Query Language) é a linguagem de banco de dados relacional comercial mais amplamente utilizada. Ela foi originalmente desenvolvida pela IBM nos projetos SEQUEL-XRM e System-R (1974-1977). Quase imediatamente, outros fabricantes introduziram produtos SGBD baseados em SQL, que agora é um padrão de fato.

Visão geral sobre a linguagem SQL

A linguagem SQL é relativamente semelhante entre os principais Sistemas Gerenciadores de Banco de Dados (SGBDs) do mercado, como: Oracle, MySQL, MariaDB, PostgreSQL, Microsoft SQL Server, entre outros. De forma geral, os comandos SQL são instruções ou consultas usadas para interagir com um banco de dados relacional. Essas instruções SQL permitem que as pessoas ou aplicativos realizem várias operações, como recuperação, inserção, atualização e exclusão de dados em tabelas de banco de dados. Os comandos SQL são categorizados em várias **linguagens específicas**, cada uma com seu propósito e função. As duas principais são: **Linguagem de Definição de Dados (DDL)** e **Linguagem de**

Manipulação de Dados (DML). As DLLs são utilizadas para “montar” o banco de dados e suas tabelas, enquanto as DMLs são utilizadas para manipular os dados contidos no banco.

- **DDL (Data Definition Language):** Suporta a criação, exclusão e modificação de definições de tabelas e visões, incluindo a definição de restrições de integridade. CREATE, DROP, ALTER, TRUNCATE, COMMENT, RENAME
    - CREATE: cria um objeto (uma Tabela, por exemplo) dentro da base de dados.
    - DROP: apaga um objeto do banco de dados.
    - ALTER TABLE: Permite modificar a estrutura de uma tabela existente.
- **DML (Data Manipulation Language):** Permite formular consultas e realizar operações de inserção, exclusão e modificação de tuplas. INSERT, UPDATE, DELETE
    - INSERT INTO: Adiciona novos registros a uma tabela.
    - UPDATE: Modifica registros existentes com novos valores.
    - DELETE FROM: Remove registros de uma tabela.
    - **DQL (Data Query Language):** A Linguagem de Consulta de Dados é o comando para consulta aos dados armazenados. Ele é composto por apenas o comando SELECT. Em alguns materiais acadêmicos essa instrução aparece incorporada no conjunto DML.
        - SELECT: Usado para selecionar dados de uma tabela.
        - FROM: Especifica a tabela da qual você deseja selecionar dados.
        - WHERE: Define critérios para filtrar os resultados.
        - GROUP BY: Agrupa os resultados com base em uma ou mais colunas.
        - HAVING: Define condições para grupos criados pelo GROUP BY.
        - ORDER BY: Classifica os resultados em ordem crescente ou decrescente.
- **A DTL ou TCL (Data Transaction Language):** subconjunto do SQL para transação de dados. A DTL envolve gerenciamento e controle de transações. BEGIN/SET TRANSACTION, COMMIT, ROLLBACK, SAVEPOINT
    - BEGIN WORK - (ou BEGIN TRANSACTION, dependendo do dialeto SQL) - pode ser usado para marcar o começo de uma transação de banco de dados que pode ser completada ou não.
    - COMMIT - finaliza uma transação dentro de um sistema de gerenciamento de banco de dados.
    - ROLLBACK - faz com que as mudanças nos dados existentes desde o último COMMIT ou ROLLBACK sejam descartadas.
    - SAVEPOINT: Define um ponto de salvamento em uma transação.
- **DCL (Data Control Language):** Mecanismos para controlar o acesso dos usuários aos objetos de dados, como tabelas e visões. GRANT, REVOKE
    - GRANT - autoriza ao usuário executar ou setar operações.
    - REVOKE - remove ou restringe a capacidade de um usuário de executar operações.

Recursos Avançados: Incluem recursos de orientação a objetos, consultas recursivas, consultas de apoio à decisão e áreas emergentes como mineração de dados, dados espaciais e gerenciamento de texto e dados XML.

- **Gatilhos e Restrições de Integridade Avançadas:** O padrão SQL 1999 inclui suporte a gatilhos e permite o uso de consultas para criar restrições de integridade complexas.
- **SQL Embutida e Dinâmica:** Recursos que permitem a chamada de código SQL por meio de uma linguagem hospedeira e a construção de consultas em tempo de execução.

- **Execução Cliente-Servidor e Acesso a Banco de Dados Remoto:** Comandos que controlam a conexão de um programa cliente a um servidor de banco de dados e o acesso a dados por meio de uma rede.

Alguns dos padrões SQL mais conhecidos são:

- **SQL-86**: Foi o primeiro padrão SQL oficialmente adotado em 1986. Este padrão estabeleceu muitos dos conceitos fundamentais da linguagem, como comandos SELECT, FROM e WHERE.
- **SQL-89**: Trata-se de uma revisão do padrão SQL que fez pequenas melhorias e correções em relação ao SQL-86.
- **SQL-92:** Este é um dos padrões mais influentes e amplamente adotados. Ele introduziu várias melhorias significativas, como suporte para junções externas, subconsultas e alterações na manipulação de datas e horas.
- **SQL-1999** (SQL-99): Também conhecido como SQL3, este padrão introduziu recursos avançados, como suporte a tipos de dados complexos (objetos), gatilhos, procedimentos armazenados e recursão.
- **SQL-2003:** Continuou a expansão do SQL introduzindo recursos como janelas (WINDOW functions), melhorias na manipulação de XML e suporte a expressões regulares.
- **SQL-2008:** Este padrão continuou a adicionar recursos avançados, como suporte a tipos de dados espaciais para geolocalização, cláusulas MERGE e melhorias na manipulação de data e hora.
- **SQL-2011:** Introduziu recursos adicionais, como comandos temporais (para rastrear alterações no tempo), um suporte a procedimentos SQL com múltiplos idiomas e melhorias nas sequências.
- **SQL-2016:** Este padrão concentrou-se em melhorias na produtividade do desenvolvedor, com recursos como consultas de tabela temporal, JSON e XML, entre outros.
- **SQL-2019:** Introduziu recursos como gráficos e dados semi-estruturados, suporte a consultas aprimoradas em JSON, extensões em análise e melhorias no desempenho de consultas.
- **SQL-2022:** Este é o padrão mais recente, que continua a expandir o SQL com recursos adicionais, incluindo aprimoramentos nas capacidades de gráficos e consultas.

Quais são as vantagens e desvantagens da linguagem SQL?

Vantagens

Dentre todas as vantagens do SQL, as principais são:

- Padronização: trata-se de uma linguagem padronizada, o que significa que, independentemente do sistema de gerenciamento de banco de dados (SGBD) que você está usando (por exemplo, MySQL, PostgreSQL, Oracle, SQL Server), a linguagem em si é a mesma. Isso facilita a portabilidade e a aprendizagem.
- Facilidade de Consulta: oferece uma maneira intuitiva de consultar e recuperar dados de bancos de dados. É uma linguagem declarativa que permite que você especifique o que deseja, em vez de como obtê-lo, tornando as consultas mais compreensíveis.
- Integridade de Dados: permite a definição de restrições de integridade, como chaves primárias e estrangeiras, para garantir que os dados sejam consistentes e confiáveis.

Desvantagens

Mas tudo tem dois lados. E o SQL também tem alguns pontos não tão bons assim:

- Complexidade: as consultas SQL mais complexas podem se tornar difíceis de escrever e entender, principalmente para iniciantes, tornando-as mais suscetíveis a erros.
- Desempenho Variável: o desempenho das consultas SQL pode variar dependendo da estrutura do banco de dados, da indexação e do volume de dados. Consultas mal otimizadas podem ser lentas.
- Falta de Suporte para dados não estruturados: o SQL é otimizado para dados estruturados, o que significa que não é a melhor escolha para armazenar e consultar dados não estruturados, como documentos de texto, áudio ou vídeo

Onde o SQL é usado?

O SQL e os bancos de dados relacionais têm amplas aplicações em diferentes áreas:

- **Desenvolvimento de Aplicativos Web:** Utilizados para armazenar e manipular dados em tempo real em aplicativos web.
- **Análise de Dados e Business Intelligence (BI):** Cruciais para executar consultas, gerar relatórios e criar painéis de controle em ferramentas de BI.
- **Ciência de Dados e Mineração de Dados:** essenciais para limpar, transformar e analisar dados complexos em bancos de dados.
- **Aplicações Móveis:** Empregados para acessar bancos de dados embutidos em aplicativos móveis, permitindo armazenamento local de dados.
- **Educação e Pesquisa:** Utilizados em instituições acadêmicas e de pesquisa para armazenar e analisar uma variedade de dados, contribuindo para avanços em diversas disciplinas.
- **Gestão de Finanças e Contabilidade:** Utilizados por empresas financeiras e departamentos de contabilidade para gerenciar e analisar dados financeiros, incluindo transações e demonstrações financeiras.

Comparação entre SQL e NoSQL:

- SQL (Bancos de Dados Relacionais):
    - Modelo de dados relacional em tabelas.
    - Esquema fixo.
    - Transações ACID para garantir integridade.
    - Flexibilidade limitada para alterações de esquema.
    - Suporte a consultas complexas usando SQL.
- NoSQL (Bancos de Dados Não Relacionais):
    - Modelo de dados flexível, suportando diversos tipos como documentos, chave-valor, etc.
    - Esquema dinâmico.
    - Transações BASE para disponibilidade e desempenho.
    - Escalabilidade horizontal.
    - Simplicidade em grandes volumes de dados, adequado para alto tráfego.

A escolha entre SQL e NoSQL depende dos requisitos do projeto, incluindo estrutura dos dados, complexidade das consultas, volume de dados, necessidades de escalabilidade e desempenho. Sistemas híbridos também são comuns para aproveitar vantagens de ambos.

Principais comandos SQL

Comandos em SQL: DML

Seguem os principais comandos DML. Para fins didáticos, vamos considerar os comandos de consulta como DQL e, por conseguinte, iremos apresentá-los na próxima seção

Comando	**Descrição**	**Sintaxe**	**Exemplo**
INSERT	Inserção dos dados na tabela (padrão).	INSERT INTO tabela (coluna1, coluna2, …) VALUES (valor1, valor2, …)	INSERT INTO Colaborador (nome, cidade) VALUES (“Cristiane”, “Rio de Janeiro”)
	Inserção dos dados na tabela, omitindo o nome das colunas (considerando os dados inseridos na mesma ordem).	INSERT INTO tabela VALUES (valor1, valor2, …)	INSERT INTO Colaborador VALUES (1, “Cristiane”, 18/01/1985, “Rio de Janeiro”)
UPDATE	Atualização dos dados na tabela. Caso a condição seja omitida, todas as linhas da tabela serão atualizadas.	UPDATE tabela SET coluna1 = valor 1, coluna 2 = valor2 … [WHERE condição]	UPDATE Colaborador SET cidade = “Florianópolis” WHERE nome = “Cristiane”
DELETE	Remoção dos dados da tabela. Caso a condição seja omitida, todas as linhas da tabela serão removidas.	DELETE FROM tabela [WHERE condição]	DELETE FROM Colaborador

Tabela 1 – Principais Comandos em SQL do Tipo DML.

Comandos em SQL: DQL

Conforme explicado anteriormente, vamos considerar os comandos de consulta nesta seção. Você vai observar que a consulta é representada pelo *SELECT*:

Comando	**Descrição**	**Sintaxe**	**Exemplo**
SELECT	Consulta a todos os campos da tabela (padrão). Caso a condição seja omitida, todas as linhas da tabela serão consultadas.	SELECT * FROM tabela [WHERE condição]	SELECT * FROM Colaborador WHERE nome = “Cristiane”

	Consulta a alguns campos discriminados da tabela. Caso a condição seja omitida, todas as linhas da tabela serão consultadas.	SELECT campo1, campo2 FROM tabela [WHERE condição]	SELECT nome, cidade FROM Colaborador

Tabela 2 – Principais Comandos em SQL Padrão do Tipo DQL.

Funções de Agregação

Conforme o próprio nome diz, essas funções agregam, agrupam um conjunto de valores em um único resultado, após a realização de uma determinada operação por um comando em SQL. O agrupamento é dado pelo *GROUP BY*.

Em seguida, após agrupá-los, você poderá efetuar uma nova filtragem (opcional). Essa filtragem sobre o agrupamento realizado é feita com o *HAVING*. Só para ilustrar, vamos esquematizar para você entender melhor:

Figura 1 – Passo a passo das Funções de Agregação em SQL.

Dessa forma, as principais funções de agregação são:

Função de Agregação	**Descrição**
COUNT	Total de linhas no resultado.
SUM	Somatório de valores da coluna.
AVG	Média aritmética de valores da coluna.
MAX	Maior valor da coluna.
MIN	Menor valor da coluna.

Tabela 3 – Principais Funções de Agregação.

Em seguida, vamos ver o SQL esquematizado:

Comando	**Descrição**	**Sintaxe**	**Exemplo**

SELECT	Consulta aos dados da tabela, aplicando função de agregação. Se a função for aplicada, o agrupamento é obrigatório.	SELECT função (campo1), campo2 FROM tabela [WHERE condição] GROUP BY campo2	SELECT COUNT(*), cidade FROM Colaborador GROUP BY cidade
	Consulta aos dados da tabela, aplicando função de agregação e filtragem dos dados agregados. Se a função for aplicada, o agrupamento é obrigatório. Entretanto, a filtragem do agrupamento é opcional.	SELECT função (campo1), campo2 FROM tabela [WHERE condição1] GROUP BY campo2 [HAVING condição2]	SELECT COUNT(*), cidade FROM Colaborador GROUP BY cidade HAVING COUNT(*) > 10

Tabela 4 – Principais Comandos em SQL do Tipo DQL com Funções de Agregação.

Ordenação

Por outro lado, caso você deseje, o resultado da consulta poderá vir ordenado. A ordenação poderá ser crescente ou decrescente. A fim de evitar confusão, lembre-se sempre de que ela é a última coisa a ser aplicada na consulta elaborada com um comando SQL, independente se ela tenha funções de agregação ou não:

Comando	**Descrição**	**Sintaxe**	**Exemplo**
SELECT	Consulta padrão aos dados da tabela, aplicando ordenação crescente (ASC) ou decrescente (DESC).	SELECT * FROM tabela [WHERE condição] ORDER BY campo1, campo2 [ASC | DESC]	SELECT * FROM Colaborador ORDER BY nome, cidade ASC
	Consulta aos dados da tabela, aplicando função de agregação e ordenação crescente (ASC) ou decrescente (DESC) combinadas.	SELECT função (campo1), campo2 FROM tabela [WHERE condição1] GROUP BY campo2 [HAVING condição2] ORDER BY função (campo1), campo2 [ASC | DESC]	SELECT COUNT(*), cidade FROM Colaborador GROUP BY cidade HAVING COUNT(*) > 10 ORDER BY cidade DESC

Tabela 5 – Principais Comandos em SQL do Tipo DQL com Ordenação.

Junções

Uma junção em SQL é uma consulta especial que envolve duas ou mais tabelas. Dentre os principais tipos cobrados em prova, podemos destacar a junção interna (*inner join*), a junção externa (*outer join*) e o produto cartesiano (*cross join*). Vamos esquematizar tudo nas próximas seções.

Junção Interna (*Inner Join*) em SQL

Descrição	Sintaxe	Exemplo
– Compara cada linha da tabela A com as linhas da tabela B, verificando se o valor de um determinado campo de uma é igual ao de outra. – Se for, os valores do campo em comum das linhas correspondentes das tabelas A e B serão combinados e incluídos no conjunto de resultados.	SELECT * FROM tabela A INNER JOIN tabela B ON A.campo = B.campo	SELECT A.nome, B.cidade FROM Colaborador A INNER JOIN Cidade B ON A.idCidade = B.idCidade

Tabela 2 – Resumo de Junção Interna em SQL.

nome	cidade
Cristiane	Rio de Janeiro
Felipe	Campinas
Raphael	Brasília

Figura 1 – Exemplo da Junção Interna.

Junção Externa (*Outer Join*) em SQL

Normalmente, a cobrança de junção externa divide-se em três tipos: junção externa à esquerda (*left outer join*), junção externa à direita (*right outer join*) e junção externa completa (*full outer join*).

Junção Externa à Esquerda (*Left Outer Join*) em SQL

Descrição	Sintaxe	Exemplo
– Compara cada linha da tabela A com as linhas da tabela B, verificando se o valor de um determinado campo de uma é igual ao de outra. – Se for, os valores do campo em comum das linhas correspondentes da tabela A serão incluídos no conjunto de resultados. – Caso o valor do campo de A não tenha nenhum valor correspondente em B, o nulo será incluído no conjunto de resultados.	SELECT * FROM tabela A LEFT OUTER JOIN tabela B ON A.campo = B.campo	SELECT A.nome, B.cidade FROM Colaborador A LEFT OUTER JOIN Cidade B ON A.idCidade = B.idCidade

Tabela 3 – Resumo de Junção Externa à Esquerda em SQL.

nome	cidade
Cristiane	Rio de Janeiro
Felipe	Campinas
Raphael	Brasília
Mohammed	NULL
Steve	NULL

Figura 2 – Exemplo da Junção Externa à Esquerda.

Junção Externa à Direita (*Right Outer Join*) em SQL

Descrição	**Sintaxe**	**Exemplo**
– Compara cada linha da tabela A com as linhas da tabela B, verificando se o valor de um determinado campo de uma é igual ao de outra. – Se for, os valores do campo em comum das linhas correspondentes da tabela B serão incluídos no conjunto de resultados. – Caso o valor do campo de B não tenha nenhum valor correspondente em A, o nulo será incluído no conjunto de resultados.	SELECT * FROM tabela A RIGHT OUTER JOIN tabela B ON A.campo = B.campo	SELECT A.nome, B.cidade FROM Colaborador A RIGHT OUTER JOIN Cidade B ON A.idCidade = B.idCidade

Tabela 4 – Resumo de Junção Externa à Direita em SQL.

nome	cidade
Cristiane	Rio de Janeiro
Felipe	Campinas
Raphael	Brasília
NULL	Florianópolis
NULL	São Paulo

Figura 3 – Exemplo da Junção Externa à Direita.

Junção Externa Completa (*Full Outer Join*) em SQL

Descrição	**Sintaxe**	**Exemplo**
– Compara cada linha da tabela A com as linhas da tabela B, verificando se o valor de um determinado campo de uma é igual ao de outra. – Se for, os valores do campo em comum das linhas correspondentes das tabelas A e B serão combinados e incluídos no conjunto de resultados. – Caso o valor do campo de A não tenha nenhum valor correspondente em B, o nulo será incluído no conjunto de resultados. – Caso o valor do campo de B não tenha nenhum valor correspondente em A, o nulo será incluído no conjunto de resultados. – Em síntese, trata-se de um *left outer join* e *right outer join* juntos.	SELECT * FROM tabela A FULL OUTER JOIN tabela B ON A.campo = B.campo	SELECT A.nome, B.cidade FROM Colaborador A FULL OUTER JOIN Cidade B ON A.idCidade = B.idCidade

Tabela 5 – Resumo de Junção Externa Completa em SQL.

nome	cidade
Cristiane	Rio de Janeiro
Felipe	Campinas
Raphael	Brasília
NULL	Florianópolis
NULL	São Paulo
Mohammed	NULL
Steve	NULL

Figura 4 – Exemplo da Junção Externa Completa.

Produto Cartesiano (*Cross Join*) em SQL

Descrição	**Sintaxe**	**Exemplo**
– Cada linha da tabela A é combinada com as linhas da tabela B. – Não há comparação de valores.	SELECT * FROM tabela A CROSS JOIN tabela B	SELECT A.nome, B.cidade FROM Colaborador A CROSS JOIN Cidade B

Tabela 6 – Resumo de Produto Cartesiano em SQL.

nome	cidade
Cristiane	Rio de Janeiro
Cristiane	Campinas
Cristiane	Brasília
Cristiane	Florianópolis
Cristiane	São Paulo
Felipe	Rio de Janeiro
Felipe	Campinas
Felipe	Brasília
Felipe	Florianópolis
Felipe	São Paulo
Raphael	Rio de Janeiro
Raphael	Campinas
Raphael	Brasília
Raphael	Florianópolis
Raphael	São Paulo
Mohammed	Rio de Janeiro
Mohammed	Campinas
Mohammed	Brasília
Mohammed	Florianópolis
Mohammed	São Paulo
Steve	Rio de Janeiro
Steve	Campinas
Steve	Brasília
Steve	Florianópolis
Steve	São Paulo

Figura 5 – Exemplo do Produto Cartesiano.

Resumo Esquematizado dos Tipos de Junção

Em suma, selecionamos um resumo esquematizado dos tipos de junção em SQL. Sem dúvida, as figuras são emblemáticas e costumam ajudar. Se você gostou, guarde-as em meio físico ou digital, pois elas poderão contribuir com os seus estudos.

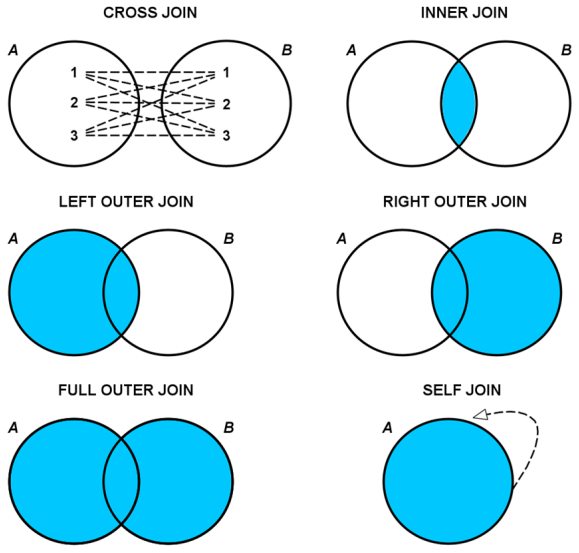

Figura 6 – Resumo Esquematizado das Junções (Fonte: DataQuest. Disponível em: https://www.dataquest.io/blog/sql-joins-interview-questions-and-answers/. Acesso em: 30 out. 2023).

Comandos em SQL: DDL

Seguem os principais comandos DDL. Nesta seção, vamos considerar apenas tabelas nos exemplos, pois são os objetos mais cobrados em prova pelas bancas:

Comando	**Descrição**	**Sintaxe**	**Exemplo**
CREATE	Criação de uma tabela no banco, em sua versão padrão.	CREATE TABLE tabela (coluna1 tipo1, coluna 2 tipo2…)	CREATE TABLE Colaborador (id int,nome varchar, dataNascimento date,cidade varchar)
	Criação de uma nova tabela no banco, a partir de uma consulta a uma tabela já existente (antiga). A condição é opcional.	CREATE TABLE tabelaNova AS SELECT coluna1, coluna2… FROM tabelaAntiga [WHERE condição]	CREATE TABLE Colaborador AS SELECT * FROM Pessoa WHERE status = “aprovado”
ALTER	Alteração de uma tabela para adição, exclusão ou alteração de campos	ALTER TABLE tabela [ADD coluna tipo] | [DROP COLUMN coluna] | [RENAME COLUMN colunaAntiga TO colunaNova]	ALTER TABLE Colaborador ADD experiencia int
TRUNCATE	Remoção dos dados de uma tabela, mantendo a sua estrutura.	TRUNCATE TABLE tabela	TRUNCATE TABLE Colaborador
DROP	Remoção dos dados e da estrutura de uma tabela.	DROP TABLE tabela	DROP TABLE Colaborador

Tabela 6 – Principais Comandos em SQL do Tipo DDL.

Atenção: Observe que o *TRUNCATE* apaga apenas os dados de uma tabela, mantendo a sua estrutura. Contudo, ele não é considerado um comando DML, mas sim DDL. Cuidado para não confundir com o *DELETE*.

Comandos em SQL: DCL

Seguem os principais comandos DCL. Da mesma forma que na seção anterior, vamos considerar tabelas nos exemplos, pois são os objetos mais cobrados pelas bancas em provas:

Comando	**Descrição**	**Sintaxe**	**Exemplo**
GRANT	Concede privilégio a um determinado usuário ou papel, sobre uma tabela (padrão).	GRANT privilégio ON tabela TO [usuário | papel]	GRANT SELECT ON Colaborador TO estrategia
	Concede privilégio a um determinado usuário ou papel, sobre uma tabela, e permite que o beneficiado estenda o privilégio a outros.	GRANT privilégio ON tabela TO [usuário | papel] [WITH GRANT OPTION]	GRANT SELECT ON Colaborador TO estrategia WITH GRANT OPTION
REVOKE	Revoga privilégio concedido a um determinado usuário ou papel, sobre uma tabela.	REVOKE privilégio ON tabela FROM [usuário | papel]	REVOKE SELECT ON Colaborador FROM estrategia

Tabela 7 – Principais Comandos em SQL do Tipo DCL.

Comandos em SQL: DTL

Antes de mais nada, parabéns por ter chegado até aqui! SQL não é um assunto tão difícil, basta treinar. Se ainda está com dificuldade, tenha certeza de que ela é passageira. Sem dúvida, o sucesso virá com a persistência. Por fim, para fechar o artigo, seguem os principais comandos DTL. Você vai reparar que a sintaxe é igual ao nome do comando, o que dispensa exemplos adicionais:

Comando / Sintaxe	**Descrição**
COMMIT	Confirma as operações realizadas na base de dados.
ROLLBACK	Desfaz as operações realizadas na base de dados.

Tabela 8 – Principais Comandos em SQL do Tipo DTL.

Referencias:

- Fundamentos de Sistemas de Gerenciamento de Banco de Dados - Elmasri & Navathe (6ª edição)

- Sistemas de Gerenciamento Sistemas de Gerenciamento de Banco de Dados - Ramakrishnan Gehrke (3ª edição)
- https://www.sqltutorial.org/

Isenção de Responsabilidade: