Banco de Dados Relacional

SELECT (MySQL)

1 SQL – Structured Query Language

SQL é uma linguagem de programação utilizada para gerenciar dados em Sistemas de Gerenciamento de Banco de Dados (**SGBD**) / Relational Database Management System (RDBMS).

Foi desenvolvida inicialmente pela IBM na década de 1970 e tornou-se um padrão para comunicação com bancos de dados.

2 Categorias

As categorias representam diferentes conjuntos de comandos ou operações que podem ser executadas em um banco de dados. Seguem algumas delas.

2.1 DDL

(Data **D**efinition **L**anguage / Linguagem de Definição de Dados**)**

Utilizada para definir a estrutura e a organização dos dados em um banco de dados. Inclui comandos como CREATE, ALTER e DROP.

CREATE TABLE: criar uma tabela;

ALTER TABLE: modificar a estrutura de uma tabela existente;

DROP TABLE: remover uma tabela.

2.2 DML

(**D**ata **M**anipulation **L**anguage/Linguagem de Manipulação de Dados)

Utilizada para manipular os dados armazenados. Exemplos:

INSERT INTO: Usado para adicionar novos registros a uma tabela.

UPDATE: Utilizado para modificar os registros existentes em uma tabela.

DELETE FROM: Remove registros de uma tabela.

2.3 DQL

(**D**ata **Q**uery **L**anguage/ Linguagem de Pesquisa de Dados)

A linguagem de consulta de dados é usada para recuperar dados de um banco de dados. Ela inclui principalmente o comando SELECT. Exemplo:

SELECT: Usado para **recuperar dados** de uma ou mais tabelas no banco de dados.

2.4 DCL

(**D**ata **C**ontrol **L**anguage/ Linguagem de Controle de Dados)

Utilizada para controlar o acesso e a segurança dos dados em um banco de dados. Ela inclui comandos como:

GRANT: Concede privilégios específicos a usuários ou funções no banco de dados.

REVOKE: Revoga os privilégios concedidos anteriormente.

3 Select, consulta para recuperar dados

3.1 Sintaxe

```
SELECT column1, column2, ...
FROM table_name;
```

Exemplos:

```
SELECT * FROM alunos;
SELECT nome, idade FROM alunos;
```

Exemplos de contexto de SELECT:

Exemplo 1: Escrever uma consulta para selecionar todos os alunos com uma idade superior a 20 anos.

Exemplo 2: Selecionar os nomes dos alunos e suas respectivas notas em ordem decrescente de notas.

Exemplo 3: Recuperar os cinco produtos mais caros da tabela de produtos.

Conceitos importantes: projeção, filtragem, ordenação, limitação e agregação.

3.2 Projeção

A projeção em uma instrução SELECT refere-se à seleção específica de colunas que serão incluídas no resultado da consulta. Isso significa que você está "projetando" ou "extraindo" apenas as colunas que são relevantes para sua necessidade de consulta, em vez de todas as colunas disponíveis na tabela. Os exemplos de SELECT sempre apresentam uma projeção de colunas.

3.3 Filtragem

Restringe os resultados de uma consulta com base em condições específicas.

cláusula **DISTINCT** - usada para retornar apenas valores distintos em uma coluna específica ou em um conjunto de colunas. Isso significa que ela remove registros duplicados do resultado da consulta.

Exemplo de contexto: Cidades em que os alunos da instituição moram.

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
SELECT DISTINCT cidade
FROM aluno;
```

cláusula **WHERE** - filtra resultados com base em condições específicas.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Para se criar a condição, se utiliza operadores. Um operador em SQL é um símbolo ou palavra-chave que é usado para realizar operações em valores ou expressões em uma consulta. Eles são usados principalmente em cláusulas WHERE, **JOINs** e outras partes de uma consulta para filtrar dados, comparar valores, combinar resultados e realizar cálculos.

Operadores Aritméticos são usados para realizar operações matemáticas em valores numéricos. Exemplos incluem adição (+), subtração (-), multiplicação (*), divisão (/) e módulo (%).

Operadores de Comparação são usados para comparar valores. Exemplos incluem igualdade (=), desigualdade (<> ou !=), maior que (>), menor que (<), maior ou igual que (>=) e menor ou igual que (<=).

Operadores Lógicos são usados para combinar condições lógicas. Exemplos incluem AND, OR e NOT. Eles são usados em cláusulas WHERE para criar condições mais complexas.

Operadores de Concatenação são usados para concatenar strings. Em muitos sistemas de banco de dados, o operador de concatenação é o símbolo de barra vertical (|) ou duas barras verticais (||).

Operadores de Pertencimento são usados para verificar se um valor está presente em um conjunto de valores. Exemplos incluem IN e NOT IN.

Operadores de Intervalo são usados para verificar se um valor está dentro de um intervalo específico. Exemplos incluem BETWEEN e NOT BETWEEN.

Operadores de Existência são usados para verificar se uma subconsulta retorna algum resultado. Exemplos incluem EXISTS e NOT EXISTS.

Operadores de União são usados para combinar resultados de duas consultas. Exemplos incluem UNION, UNION ALL, INTERSECT e EXCEPT.

Exemplo:

```
SELECT *
FROM alunos
WHERE idade > 18;
```

3.4 Ordenação

Refere-se à organização dos resultados de uma consulta de acordo com critérios específicos, como ordem alfabética, ordem numérica crescente ou decrescente, entre outros.

cláusula **ORDER BY** - ordena os resultados de uma consulta.

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1 [ASC|DESC], column2 [ASC|DESC], ...;
```

[ASC|DESC]: É opcional e indica a direção da ordenação.
ASC (Ascending) para ordenação crescente (padrão), e
DESC (Descending) para ordenação decrescente.

```
SELECT *
FROM produtos
ORDER BY preco DESC;
```

3.5 Limitação

A limitação no comando SELECT refere-se à restrição do número de linhas retornadas pelo resultado de uma consulta. Ela é útil quando se precisa restringir o número de resultados retornados por uma consulta, especialmente em casos em que há uma grande quantidade de dados na tabela e você só precisa de uma parte desses dados.

```
SELECT column1, column2, ...
FROM table_name
```

LIMIT number_of_rows;

number_of_rows: É o número máximo de linhas que você deseja retornar como resultado da consulta.

```
SELECT *
FROM alunos
LIMIT 10;
```

Um exemplo de contexto prático da necessidade de limitação no comando SELECT pode ocorrer em uma aplicação web que exibe uma lista de postagens de um blog. Considere uma situação em que o blog/ possui milhares de postagens e a página inicial precisa exibir apenas as 5 postagens mais recentes.

Sem a limitação no SELECT, a consulta poderia retornar todas as postagens do banco de dados, resultando em um grande volume de dados sendo transferido do banco de dados para o servidor web e, consequentemente, para o navegador do usuário. Isso pode causar um atraso significativo no tempo de carregamento da página e consumir muitos recursos de rede e de processamento.

Com a limitação no SELECT, a consulta pode ser ajustada para retornar apenas as 5 postagens mais recentes, reduzindo assim a quantidade de dados recuperados do banco de dados. Isso resulta em um carregamento mais rápido da página inicial do blog e em uma experiência de usuário mais adequada.

3.6 Agregação

A agregação em um SELECT refere-se à combinação de múltiplas linhas de dados em um único valor com base em alguma operação de agregação. Isso geralmente envolve o uso de funções de agregação, que operam em conjuntos de dados para calcular valores agregados, como: SUM, AVG, COUNT, MIN e MAX.

Por exemplo, ao calcular a soma total de uma coluna de valores, você está agregando esses valores em um único valor resultante. Da mesma forma, ao contar o número de registros em uma tabela, você está agregando esses registros em um único número.

Cláusula **GROUP BY** - As operações de agregação são frequentemente usadas em conjunto com a cláusula GROUP BY, que divide os dados em grupos com base em valores comuns em uma ou mais colunas. Isso permite que você agregue os dados em cada grupo separadamente.

Por exemplo, se você tem uma tabela de vendas com colunas de produto e valor de venda, você pode querer calcular a soma total de vendas para cada produto individualmente. Você faria isso usando uma operação de agregação com GROUP BY:

```
SELECT produto, SUM(valor_venda) AS total_vendas
FROM vendas
GROUP BY produto;
```

Tabela:

produto	valor_venda
A	100
B	150
A	200
C	120
B	180

Resultado:

produto	total_vendas
A	300
B	330
C	120

3.7 Funções

Em SQL, uma função é uma construção que realiza uma operação específica em um ou mais valores e retorna um resultado. Existem vários tipos de função.

3.7.1 Funções de Agregação

São usadas para realizar cálculos em conjuntos de dados (entrada) e retornar um único valor agregado (saída). Exemplos incluem:

SUM - usada para calcular a soma de valores em uma coluna específica. É útil para obter a soma total de valores numéricos em uma coluna. Exemplo:

```
SELECT SUM(valor) AS total_vendas FROM vendas;
```

Este comando retornaria a soma de todos os valores na coluna "valor" da tabela "vendas".

COUNT - usada para contar o número de linhas em um conjunto de resultados ou o número de valores não nulos em uma coluna específica. Exemplo:

SELECT COUNT(*) AS total_registros FROM clientes;

Este comando retornaria o número total de registros na tabela "clientes".

MIN - usada para encontrar o menor valor em uma coluna específica. É útil para encontrar o valor mínimo em um conjunto de dados. Exemplo:

SELECT MIN(valor) AS menor_valor FROM produtos;

Este comando retornaria o menor valor na coluna "valor" da tabela "produtos".

MAX - usada para encontrar o maior valor em uma coluna específica. É útil para encontrar o valor máximo em um conjunto de dados. Exemplo:

SELECT MAX(valor) AS maior_valor FROM produtos;

Este comando retornaria o maior valor na coluna "valor" da tabela "produtos".

3.7.2 Funções de String

São usadas para manipular valores de texto. Exemplos incluem:

CONCAT - usada para concatenar duas ou mais strings em uma única string. Ela aceita múltiplos argumentos de strings e os une em ordem. Exemplo:

```
SELECT CONCAT(‘Olá’, ‘ ’, ‘mundo’);
```

Resultado: 'Olá mundo'

SUBSTRING - também conhecida como SUBSTR, é usada para extrair uma parte de uma string. Se especifica o ponto de início e a quantidade de caracteres a serem extraídos. Exemplo:

```
SELECT SUBSTRING('Hello World', 1, 5);
```

Resultado: 'Hello'

UPPER - usada para converter uma string em letras maiúsculas. Exemplo:

```
SELECT UPPER('hello');
```

Resultado: 'HELLO'

LOWER - usada para converter uma string em letras minúsculas. Exemplo:

```
SELECT LOWER('WORLD');
```

Resultado: 'world'

LENGTH - usada para retornar o número de caracteres em uma string. Exemplo:

```
SELECT LENGTH('MySQL');
```

Resultado: '5.

3.7.3 Funções Matemáticas

São usadas para realizar operações matemáticas em valores numéricos. Exemplos incluem:

ABS – retorno o valor absoluto de um número;
ROUND - arredonda um número para um número específico de casas decimais;
CEILING - arredonda um número para cima para o inteiro mais próximo;
FLOOR - arredonda um número para baixo para o inteiro mais próximo;
MOD - retorna o resto da divisão de dois números.

3.7.4 Funções de Data e Hora

São usadas para manipular valores de data e hora. Exemplos incluem:

DATE_FORMAT - usada para formatar uma data em uma string com um formato específico. Se fornece a data e o formato desejado como argumentos para a função. Exemplo:

```
SELECT DATE_FORMAT(data_venda, '%d/%m/%Y') AS data_formatada FROM vendas;
```

Tabela:

data_venda
2024-03-01
2024-03-05
2024-03-10

Resultado:

data_formatada
01/03/2024
05/03/2024
10/03/2024

Esta consulta formata a coluna data_venda no formato 'dia/mês/ano' e a renomeia como data_formatada.

DATE_ADD - usada para adicionar uma quantidade específica de tempo a uma data. Se fornece a data, o intervalo de tempo a ser adicionado e a unidade de tempo (como YEAR, MONTH, DAY, HOUR, MINUTE, SECOND). Exemplo:

```
SELECT DATE_ADD(data_venda, INTERVAL 7 DAY) AS
data_entrega FROM vendas;
```

Tabela:

data_venda
2024-03-01
2024-03-05
2024-03-10

Resultado:

data_entrega
2024-03-08
2024-03-12
2024-03-17

Esta consulta adiciona 7 dias à data de venda na coluna data_venda e renomeia o resultado como data_entrega.

DATE_SUB - é semelhante à DATE_ADD, mas é usada para subtrair uma quantidade específica de tempo de uma data. Exemplo:

SELECT DATE_SUB(data_vencimento, INTERVAL 3 MONTH) AS data_aviso FROM faturas;

Tabela:

data_vencimento
2024-05-15
2024-06-20
2024-07-05

Resultado:

data_aviso
2024-02-15
2024-03-20
2024-04-05

Esta consulta subtrai 3 meses da data de vencimento na coluna data_vencimento e renomeia o resultado como data_aviso.

EXTRACT - A função EXTRACT é usada para extrair partes específicas de uma data, como ano, mês ou dia. Você fornece a parte da data que deseja extrair e a data como argumentos para a função. Exemplo:

SELECT EXTRACT(YEAR FROM data_nascimento) AS ano_nascimento FROM clientes;

Tabela:

data_nascimento
1990-04-15
1985-10-25
1988-12-30

Resultado:

ano_nascimento
1990
1985
1988

Esta consulta extrai o ano da data de nascimento na coluna data_nascimento e renomeia o resultado como ano_nascimento.

NOW - retorna a data e hora atuais do sistema no formato de data e hora do banco de dados. Exemplo:

```
SELECT NOW() AS data_atual;
```

Resultado:

data_atual
2024-03-03 12:30:45

3.7.5 Funções de Conversão de Dados

São usadas para converter valores de um tipo de dados para outro. Exemplos incluem: CAST, CONVERT e TO_DATE.

3.7.6 Funções de Controle de Fluxo

São usadas para controlar o fluxo de execução de uma consulta com base em condições específicas. Exemplos incluem: CASE e IF.

As funções em SQL são poderosas e flexíveis, permitindo realizar uma ampla gama de operações em dados armazenados em bancos de dados. Elas são frequentemente usadas em consultas SELECT para transformar, calcular ou formatar dados de acordo com os requisitos específicos da aplicação.

3.8 Conceitos Adicionais

AS - AS é uma palavra-chave que é comumente usada para atribuir um alias (um apelido, nome alternativo) a uma coluna, tabela ou resultado de expressão em uma consulta. O alias é utilizado para facilitar a referência a uma coluna ou tabela em consultas mais complexas ou para tornar o resultado da consulta mais legível. Seguem alguns dos principais usos de AS em SQL:

Alias de Coluna - usado para renomear o cabeçalho de uma coluna ou para criar um nome alternativo para uma coluna específica no resultado da consulta. Exemplo:

```
SELECT nome AS nome_do_aluno, idade AS idade_do_aluno FROM alunos;
```

Neste exemplo, AS é usado para renomear as colunas nome e idade como nome_do_aluno e idade_do_aluno, respectivamente, no resultado da consulta.

Alias de Tabela - usado para criar um nome alternativo para uma tabela em uma consulta, tornando mais fácil referenciar a tabela em consultas aninhadas. Exemplo:

```
SELECT a.nome, a.idade FROM alunos AS a;
```

Alias de Expressão - usado para criar um nome alternativo para o resultado de uma expressão na consulta. Exemplo:

```
SELECT idade, ano_atual - ano_nascimento AS idade_atual FROM pessoas;
```

Neste caso, AS é usado para atribuir o alias idade_atual ao resultado da expressão ano_atual - ano_nascimento, facilitando a referência a esse resultado na consulta.

Visão – Uma "visão" (ou "view" em inglês) é uma consulta armazenada que retorna uma visualização virtual de dados de uma ou mais tabelas. Ela pode ser considerada como uma tabela virtual, construída a partir de outras tabelas (ou até mesmo de outras visões) e apresentada de uma forma específica para atender a determinadas necessidades de consulta ou relatórios.

A criação de uma visão não envolve armazenar dados físicos, mas sim definir uma consulta que, quando a visão é consultada, é executada em tempo real para fornecer os resultados.

Nem todo SELECT é uma visão. Uma visão é uma consulta armazenada que cria uma tabela virtual com base nos resultados da consulta. As visões são armazenadas no banco de dados como objetos independentes e podem ser consultadas como se fossem tabelas reais.

Por outro lado, um SELECT é uma instrução SQL que recupera dados de uma ou mais tabelas ou visualizações existentes. Um SELECT pode ser usado para recuperar dados diretamente de tabelas, sem a necessidade de criar uma visão.

4 Revisão

Essas são as palavras reservadas no MySQL presentes neste texto, em ordem alfabética e sem repetição. Você lembra a utilidade de cada uma? Caso não se lembre, revise!

1. ALTER
2. AS
3. BY
4. CASE
5. CREATE
6. DELETE
7. DESC
8. DISTINCT

9. DROP
10. EXTRACT
11. FROM
12. GRANT
13. GROUP
14. INSERT
15. INTO
16. INTERVAL
17. JOIN
18. LIMIT
19. MAX
20. MIN
21. NOW
22. ORDER
23. ROUND
24. SELECT
25. SUBSTRING
26. SUM
27. TABLE
28. UNION
29. UPDATE
30. WHERE