Banco de Dados

Prof. Thiago Cassio Krug

thiago.krug@iffarroupilha.edu.br

SQL DQL

- A DQL possui apenas um comando: SELECT
- Entretanto é o comando mais utilizado do SQL.
- Ele é utilizado para fazer consultas no banco de dados.

SELECT

SELECT <colunas>

FROM <tabelas>

WHERE <condições>

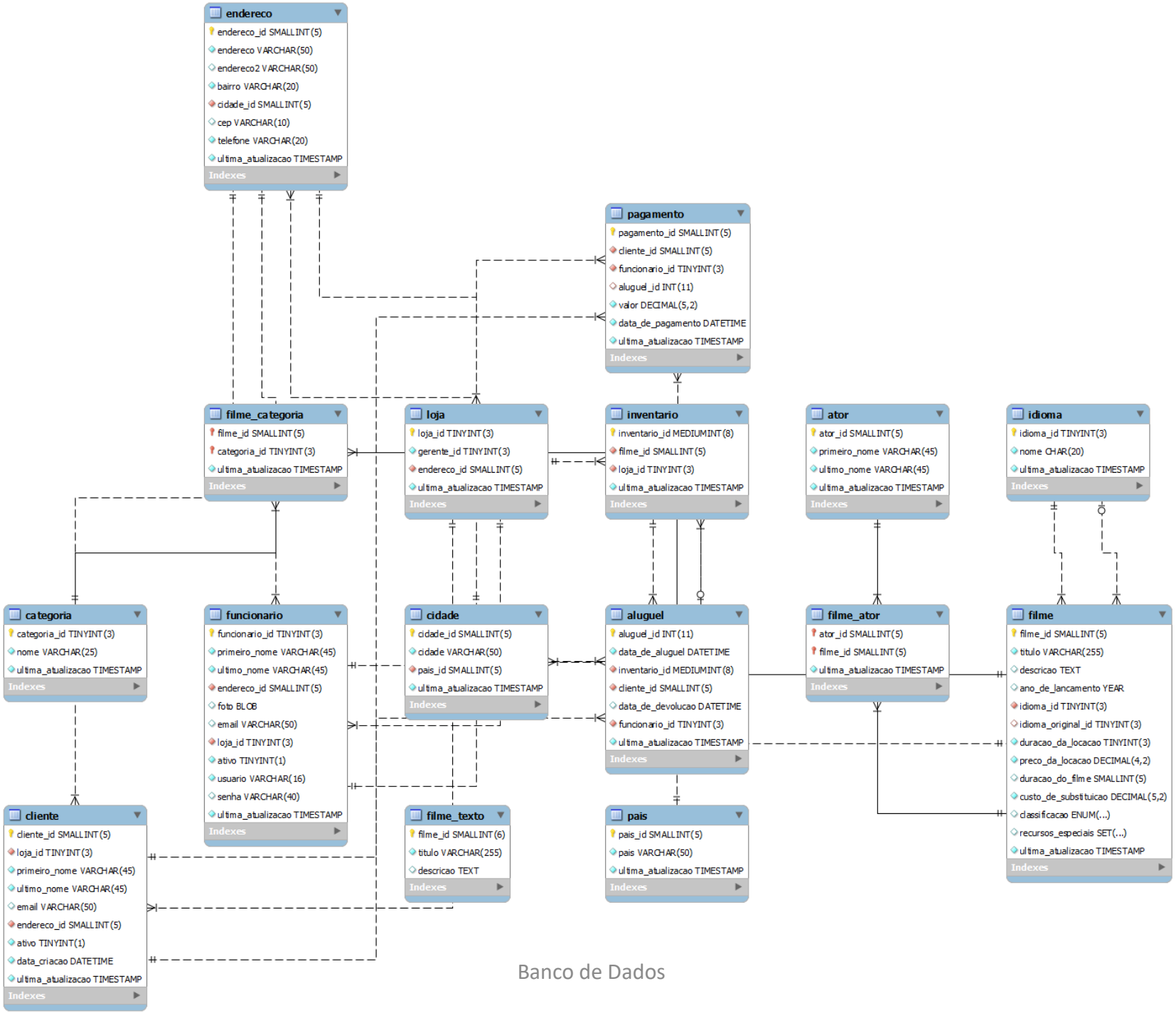
- <colunas> é a lista de colunas que devem ser exibidos pela consulta
- <tabelas> é a lista de tabelas de onde as colunas serão recuperadas
- <condições> são condições que filtram quais linhas vão ser recuperadas pela consulta

SELECT

- Para exibir, temos primeiro que possuir um banco de dados com dados.

- Abra o phpMyAdmin
- Crie um banco de dados chamado “sakila”
- Importe o arquivo sakila_tiagopassos.sql

DER


endereco
endereco_id SMALLINT(5)
endereco VARCHAR(50)
endereco2 VARCHAR(50)
bairro VARCHAR(20)
cidade_id SMALLINT(5)
cep VARCHAR(10)
telefone VARCHAR(20)
ultima_atualizacao TIMESTAMP
Indexes
pagamento
pagamento_id SMALLINT(5)
cliente_id SMALLINT(5)
funcionario_id TINYINT(3)
aluguel_id INT(11)
valor DECIMAL(5,2)
data_de_pagamento DATETIME
ultima_atualizacao TIMESTAMP
Indexes
filme_categoria
filme_id SMALLINT(5)
categoria_id TINYINT(3)
ultima_atualizacao TIMESTAMP
Indexes
loja
loja_id TINYINT(3)
gerente_id TINYINT(3)
endereco_id SMALLINT(5)
ultima_atualizacao TIMESTAMP
Indexes
inventario
inventario_id MEDIUMINT(8)
filme_id SMALLINT(5)
loja_id TINYINT(3)
ultima_atualizacao TIMESTAMP
Indexes
ator
ator_id SMALLINT(5)
primeiro_nome VARCHAR(45)
ultimo_nome VARCHAR(45)
ultima_atualizacao TIMESTAMP
Indexes
idioma
idioma_id TINYINT(3)
nome CHAR(20)
ultima_atualizacao TIMESTAMP
Indexes
categoria
categoria_id TINYINT(3)
nome VARCHAR(25)
ultima_atualizacao TIMESTAMP
Indexes
funcionario
funcionario_id TINYINT(3)
primeiro_nome VARCHAR(45)
ultimo_nome VARCHAR(45)
endereco_id SMALLINT(5)
foto BLOB
email VARCHAR(50)
loja_id TINYINT(3)
ativo TINYINT(1)
usuario VARCHAR(16)
senha VARCHAR(40)
ultima_atualizacao TIMESTAMP
Indexes
cidade
cidade_id SMALLINT(5)
cidade VARCHAR(50)
pais_id SMALLINT(5)
ultima_atualizacao TIMESTAMP
Indexes
aluguel
aluguel_id INT(11)
data_de_aluguel DATETIME
inventario_id MEDIUMINT(8)
cliente_id SMALLINT(5)
data_de_devolucao DATETIME
funcionario_id TINYINT(3)
ultima_atualizacao TIMESTAMP
Indexes
filme_ator
ator_id SMALLINT(5)
filme_id SMALLINT(5)
ultima_atualizacao TIMESTAMP
Indexes
filme
filme_id SMALLINT(5)
titulo VARCHAR(255)
descricao TEXT
ano_de_lancamento YEAR
idioma_id TINYINT(3)
idioma_original_id TINYINT(3)
duracao_da_locacao TINYINT(3)
preco_da_locacao DECIMAL(4,2)
duracao_do_filme SMALLINT(5)
custo_de_substituicao DECIMAL(5,2)
classificacao ENUM(...)
recursos_especiais SET(...)
ultima_atualizacao TIMESTAMP
Indexes
cliente
cliente_id SMALLINT(5)
loja_id TINYINT(3)
primeiro_nome VARCHAR(45)
ultimo_nome VARCHAR(45)
email VARCHAR(50)
endereco_id SMALLINT(5)
ativo TINYINT(1)
data_criacao DATETIME
ultima_atualizacao TIMESTAMP
Indexes
filme_texto
filme_id SMALLINT(6)
titulo VARCHAR(255)
descricao TEXT
Indexes
pais
pais_id SMALLINT(5)
pais VARCHAR(50)
ultima_atualizacao TIMESTAMP
Indexes

SELECT

- Comando utilizado para pesquisar todas as cidades no banco importado:

```
SELECT *
FROM cidade;
```

- * significa tudo, no caso ele exibe todas as **colunas** da tabela cidade
- cidade é a tabela em que será feita a consulta.

SELECT

- Após o SELECT, coloca-se o nome das colunas que se deseja visualizar as informações.
- A separação das colunas deve sempre ser com vírgula, igual o exemplo:

```
SELECT cidade_id, cidade FROM cidade;
```

- Na linguagem SQL é possível utilizar ALIAS, dando um apelido fictício a tabela, exemplo:

```
SELECT cid.cidade_id, cid.cidade FROM cidade cid;
```

WHERE

- Entretanto, caso seja necessário fazer filtros na busca realizada, existe o comando WHERE no SQL:

SELECT * FROM ator WHERE primeiro_nome = ‘penelope’;

- O exemplo acima busca na tabela ator todos os atores que tem o seu primeiro nome igual a ‘Penelope’.

- A busca estre aspas ‘‘ porque se trata de uma string.

WHERE

WHERE

- O operador IN permite especificar vários valores em uma cláusula WHERE:

SELECT * FROM ator WHERE primeiro_nome IN (‘joe’, ‘ed’);

- O comando SQL acima busca na tabela ator todos os atores que tem seu primeiro_nome o valor ‘joe’ ou ‘ed’.

WHERE

- O operador BETWEEN seleciona valores dentro de uma faixa. Os valores podem ser números, texto ou datas:

SELECT * FROM pagamento WHERE valor BETWEEN 2 AND 8;

- No comando SQL acima, busca-se na tabela pagamento todos os registros cujo valor ficou entre 2 e 8 reais.

UNION

- O operador UNION é usado para combinar o resultado-conjunto de duas ou mais instruções SELECT.

```
SELECT primeiro_nome FROM funcionario
UNION
SELECT primeiro_nome FROM ator
```

- No comando SQL acima, uni-se o resultado de dois campos como mesmo tipo de dado e de tabelas diferentes.
- Nota: O operador UNION seleciona apenas valores distintos por padrão. Para permitir valores duplicados, use a palavra-chave ALL com UNION.

DISTINCT

- A instrução SELECT DISTINCT é usada para retornar apenas valores distintos (diferentes):

```
SELECT DISTINCT(loja_id) FROM cliente;
```

- No comando SQL acima, busca-se apenas resultados diferentes da coluna loja_id referente a tabela cliente.

SELECT

- A cláusula SELECT pode conter expressões aritméticas envolvendo as operações +, –, *, e /, com argumentos constantes ou atributos.
- Por exemplo, a consulta abaixo devolve todos os títulos, custos_de_substituicao acrescentando + 3 reais ao custo.

SELECT titulo, custo_de_substituicao + 3 FROM filme

- Geralmente, os bancos de dados definem uma biblioteca de funções que podem ser utilizadas com a cláusula SELECT.

FROM

- A cláusula FROM permite que pesquisamos em mais de uma tabela também.

SELECT * FROM filme, filme_ator

- O que acontece com os campos iguais como filme_id?

FROM

- Na grande maioria das consultas SQL, quando mais de uma tabela está na cláusula FROM é necessário a criação de junções na cláusula WHERE.
- Por exemplo: Encontrar todos os clientes que tem pedido:

SELECT * FROM cliente, pagamento WHERE cliente.cliente_id = pagamento.cliente_id;

- Uma junção corresponde a um relacionamento entre duas ou mais tabelas.

FROM

- Exemplo (SELECT para buscar o nome e a categoria de filmes de tabelas distintas):

SELECT filme.titulo, categoria.nome

FROM categoria, filme_categoria, filme

WHERE filme_categoria.categoria_id = categoria.categoria_id AND filme_categoria.filme_id = filme.filme_id

- Porque precisamos da segunda cláusula no WHERE?

WHERE

- Os resultados de comparações podem ser combinados por intermédio dos conectivos lógicos AND, OR, e NOT;
- Exemplo:

SELECT filme_categoria.filme_id, categoria.nome

FROM filme_categoria, categoria

WHERE filme_categoria.categoria_id = categoria.categoria_id AND (categoria.nome = ‘action’ OR categoria.nome=‘comedy’)

- Porque precisamos dos () ao final do SQL?

LIKE (Outros Exemplos)

- LIKE ‘A%’ – Todas as palavras que iniciem com a letra A;
- LIKE ‘%A’ – Todas que terminem com a letra A;
- LIKE ‘%A%’ – Todas que tenham a letra A em qualquer posição;
- LIKE ‘A_’ – String de dois caracteres que tenham a primeira letra A e o segundo caractere seja qualquer outro;

LIKE (Outros Exemplos)

- LIKE ‘_A’ – String de dois caracteres cujo primeiro caractere seja qualquer um e a última letra seja A;
- LIKE ‘_A_’ – String de três caracteres cuja segunda letra seja A, independentemente do primeiro ou do último caractere;
- LIKE ‘%A_’ – Todos que tenham a letra A na penúltima posição e a última seja qualquer outro caractere;
- LIKE ‘_A%’ – Todos que tenham a letra A na segunda posição e o primeiro caractere seja qualquer um.

Operação de ordenação – ORDER BY

- A palavra-chave ORDER BY é usada para classificar a consulta por uma ou mais colunas de acordo com o conjunto de resultados.
- A palavra-chave ORDER BY classifica os registros em ordem crescente por padrão.
- Para classificar os registros em ordem decrescente, você pode usar a palavra-chave DESC.

SELECT nome FROM categoria ORDER BY nome DESC;

Funções de Agregação

- Estas funções aplicam-se a multiconjuntos de valores de uma coluna de uma relação, devolvendo um único valor como resultado.

- Funções básicas:
  - **avg**: calcula o valor médio;
  - **min**: calcula o valor mínimo;
  - **max**: calcula o valor máximo;
  - **sum**: calcula a soma dos valores;
  - **count**: calcula o número de valores;

Funções de Agregação

- SELECT avg(preco_da_locacao) FROM `filme`; -- Faz uma média do preço da locação de todos os filmes da tabela.
- SELECT min(preco_da_locacao) FROM `filme`; -- Mostra qual é o menor valor de locação de um filme.
- SELECT max(preco_da_locacao) FROM `filme`; -- Mostra qual é o maior valor de locação de um filme.
- SELECT sum(duracao_do_filme) FROM `filme`; -- Mostra o total de minutos em filmes que a locadora possui.

GROUP BY

- Usa-se a cláusula GROUP BY em conjunto com as funções de agregação para agrupar registros em subgrupos baseados em colunas ou valores retornados por uma expressão.

```
SELECT cliente.primeiro_nome, aluguel.cliente_id,
count(aluguel.cliente_id) AS contagem

FROM aluguel, cliente WHERE cliente.cliente_id = aluguel.cliente_id

GROUP BY aluguel.cliente_id

ORDER BY contagem DESC
```

HAVING

- A cláusula HAVING é usada para especificar condições de filtragem em grupos de registros ou agregações.

- É frequentemente usada em conjunto com a cláusula GROUP BY para filtrar as colunas agrupadas.

Exercícios

1. Liste todos os aluguéis realizados pela locadora
2. Liste todos os atores cadastrados no banco de dados
3. Liste todos os filmes cadastrados no banco de dados
4. Liste todos os clientes cadastrados no banco de dados
5. Busque se na entidade pais existe o brazil
6. Busque na tabela filme todos os filmes que tem a data de 2006
7. Buscar todos os clientes que tem seu nome iniciado pela letra S
8. Buscar todos os clientes que tem seu nome iniciado pela letra J
9. Buscar todos os clientes que tem seu nome iniciado pela letra B

Exercícios

10. Buscar todos os clientes que tem seu nome iniciado pela letra D
11. Buscar o endereço com id igual a 34
12. Buscar todos os clientes que tenham e-mail
13. Buscar todos os filmes que tem valor de locação inferior a R$ 3,00
14. Buscar todos os filmes que tem duração acima de 60 min
15. Buscar todos os clientes que tem seu nome terminado pela letra A
16. Buscar todos os filmes que tem valor de locação entre 2 e 4 reais
17. Buscar todos os filmes que tem o nome que inicia com a letra P
18. Buscar todos os filmes que tem recursos especiais como comentários

Exercícios

19. Buscar todos os filmes que tem recursos especiais como trailers
20. Descobrir qual autor mais atuou em um filme
21. Descobrir qual filme teve mais atores atuando
22. Descobrir qual categoria teve mais qualificações
23. Descobrir quantos filmes tem na categoria 9
24. Descobrir o último nome dos funcionários Mike e Jon
25. Descobrir os países que tem no nome a última letra H
26. Descobrir qual cliente alugou mais filmes
27. Descobrir qual filme foi mais alugado
28. Buscar todas as locações que tem o valor inferior a 4 reais

Referências

- Notas de Aula, prof. Romulo Vanzin, disciplina Banco de Dados.
- Notas de Aula, prof. Rafhael Cunha, disciplina Banco de Dados.
- HEUSER, C. A.; **Projeto de Banco de Dados**. 6ª edição. Editora Artmed, 2009.
- SILBERCHATZ, A.; KORTH, H. F.; SUDARSHA, S.; **Sistema de Banco de Dados**. 6ª edição. Editora Campus, 2012.
- AGELOTTI, E. S. **Banco de Dados**. Curitiba: Editora do Livro Técnico, 2010.
- RAMAKRISHNAN, R.; GEHRKE, J.; **Sistemas de Gerenciamento de Banco de Dados**. 3ª edição. Editora Mc Graw-Hill, 2008.
- DATE, C. J.; **Introdução a Sistemas de Bancos de Dados**. 8ª edição. Editora Campus, 2004.
- ELMASRI, R.; NAVATHE S. B.; **Sistemas de Banco de Dados**. 4ª edição. Editora Pearson, 2005.