COMANDOS SQL

Profa. Ana Paula Wauke

Disciplina de Banco de Dados

```
CREATE TABLE Persons
(
P_id int NOT NULL  AUTO_INCREMENT,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
PRIMARY KEY (P_id)
);
```

```
CREATE TABLE Orders
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
P_Id int,
PRIMARY KEY (O_Id),
FOREIGN KEY (P_Id) REFERENCES Persons(P_Id)
)
```

ALTER

- Adicionar novo atributo:
  - ALTER TABLE tabela ADD campo tipo;
- Excluir atributo de tabela:
  - ALTER TABLE tabela DROP campo;
- Alterar tipo e tamanho de atributo e auto_increment:
  - ALTER TABLE tabela MODIFY campo tipo(tam);
  - ALTER TABLE tabela MODIFY campo tipo(tam) not null AUTO_INCREMENT;
- Alterar nome de atributo:
  - ALTER TABLE TESTE CHANGE CAMPO1 CAMPO2;
  - ALTER TABLE tabela MODIFY campo tipo(tam);
- Adicionar *default*:
  - ALTER TABLE tabela ALTER COLUMN campo SET DEFAULT 'valor default';
- Excluir *default*:
  - ALTER TABLE tabela ALTER COLUMN campo DROP DEFAULT;

ALTER

- Alterar tipo e tamanho de atributo:
  - ALTER TABLE tabela CHANGE nome_antigo nome_novo tipo(tam);
- Alterar o nome da tabela:
  - RENAME TABLE nome_antigo_tabela TO novo_nome_tabela;

ALTER

- Alterar NULL para NOT NULL:
  - ALTER TABLE tabela MODIFY campo tipo(tam) NOT NULL;
- Alterar NOT NULL para NULL:
  - ALTER TABLE tabela MODIFY campo tipo(tam) NULL;
- Excluir uma constraint de chave estrangeira:
  - ALTER TABLE tabela DROP FOREIGN KEY constraint;
- Adicionar uma constraint de chave estrangeira:
  - ALTER TABLE tabela ADD CONSTRAINT constraint FOREIGN KEY (campo associado a constraint) REFERENCES tabela (campo chave);
- Adicionar constraint Unique:
  - ALTER TABLE tabela ADD CONSTRAINT constraint UNIQUE (campo associado a constraint);
- Excluindo constraint Unique:
  - ALTER TABLE tabela DROP INDEX constraint;

ALTER

- Alterar NULL para NOT NULL:
  - ALTER TABLE tb_nome_tabela MODIFY campo tipo(tam) NOT NULL;
- Alterar NOT NULL para NULL:
  - ALTER TABLE tb_nome_tabela MODIFY campo tipo(tam) NULL;
- Excluir uma constraint de chave estrangeira:
  - ALTER TABLE tb_nome_tabela DROP FOREIGN KEY nome_constraint;
- Adicionar uma constraint de chave estrangeira:
  - ALTER TABLE tb_nome_tabela ADD CONSTRAINT nome_constraint FOREIGN KEY (campo_chave_estrangeira) REFERENCES tabela (campo_chave);
- Adicionar constraint Unique:
  - ALTER TABLE tb_nome_tabela ADD CONSTRAINT nome_constraint UNIQUE (campo_unico);
- Excluindo constraint Unique:
  - ALTER TABLE tb_nome_tabela DROP INDEX nome_constraint;

INDEX

- Criar índice em tabela. Permite valores duplicados:
  - CREATE INDEX idx_nome_indice ON tb_nome_tabela (nome_campo);
  - ALTER TABLE tb_nome_tabela ADD INDEX idx_nome_indice (nome_campo);
- Criar índice unique em tabela. Não permite valores duplicados:
  - CREATE UNIQUE INDEX idx_nome_indice ON tb_nome_tabela (nome_campo);
- Excluir índice da tabela:
  - DROP INDEX idx_nome_indice ON tb_nome_tabela;
  - ALTER TABLE tb_nome_tabela DROP INDEX idx_nome_indice;

INSERT

- Inserir dados quando não se conhece a ordem dos atributos:
  - INSERT INTO tb_nome_tabela (nome_campo_a, …, nome_campo_n) VALUES (valor_campo_a, …, valor_campo_n);
- Inserir dados quando se conhece a ordem dos atributos:
  - INSERT INTO tb_nome_tabela VALUES (valor_campo1, …, valor_campo_n);
- Inserir dados, explicitando *null* e *default*:
  - INSERT INTO tb_nome_tabela (campo1, campo2, …, campo_n, campo_null, campo_default) VALUES (valor_campo1, valor_campo2, …, valor_campo_n, null, default);
- Inserir dados, explicitando apenas os que serão preenchidos:
  - INSERT INTO tb_nome_tabela (campo1, campo2, …, campo_n) VALUES (valor_campo1, valor_campo2, …, valor_campo_n);

DELETE E UPDATE

- Excluir, conforme condição:
  - DELETE FROM tb_nome_tabela WHERE <<condição>>;
    - <<condição>> - podem ser usados OR, AND, NOT, <, <=, >, >=, <> ou !=, is null;
  - DELETE FROM tb_nome_tabela WHERE <<condição pode ser consulta>>;
- Excluir todos os dados da tabela:
  - DELETE FROM tb_nome_tabela;
- Atualizar, conforme condição:
  - UPDATE tb_nome_tabela SET nome_atributo = novo valor WHERE <<condição>>;
    - <<condição>> - podem ser usados OR, AND, NOT, <, <=, >, >=, <> ou !=, is null;

CONSULTAS - SQL

SELECT, FROM E WHERE – CLÁUSULAS BÁSICAS

- O comando SELECT é utilizado para realizar consultas na tabela , a estrutura básica em SQL consiste em três cláusulas:
- SELECT = atributos/colunas desejados no resultado da consulta
- FROM = Qual tabela será usada na consulta
- WHERE = Descreve a condição da consulta

- Consulta generalizada: SELECT * FROM Exemplo;
- Consulta com condição : SELECT nome_ator FROM tbAutor WHERE codigo_ator = 1;

INNER JOIN

O comando INNER JOIN - junção entre 2 ou mais tabelas.

Ex: SELECT *

FROM tbTItulo

INNER JOIN tbCategoria

ON tbCategoria.codigo_categoria = tbTitulo.codigo_categoria;

Está consulta irá retornar tudo da junção das tabela “tbTitulo” e “tbCategoria” onde o codigo_categoria da tabela “tbCategoria” for igual ao codigo_categoria da tabela “tbTitulo”.

ALIASES

- apelidos para atributos ou tabelas.
- utiliza a palavra AS (opcional);
- ex.:SELECT L.nome_lanche,G.nome_garcom FROM tbLanche L INNER JOIN tbGarcom G ON G.nome_garcom = L.nome_garcom WHERE L.preco = 2.50;

LIKE

- Comparação de Strings, diferentemente de = ou <> (comparação exata); like permite comparação com parte da string:
- '%' (porcentagem) - representa parte da string;
- '_' (underscore) – considera 1 caractere;
- Ex: // títulos que comecem com M
- SELECT *
- FROM tbTItulo T
- WHERE T.nome_titulo LIKE '%M';
- // títulos com 5 caracteres
- SELECT *
- FROM tbTItulo T
- WHERE T.nome_titulo LIKE '_ _ _ _ _';

ORDER BY

- Ordena resultado em ordem crescente (ASC) ou decrescente (DESC);
- // ordena titulo por ordem crescente e ano por decrescente;
- SELECT T.nome_titulo, T.ano
- FROM tbTItulo T
- ORDER BY T.nome_titulo ASC, T.ano DESC;

FUNÇÕES

- AVG
- SUM
- MIN E MAX
- COUNT

CRIAR TABELA E INSERIR VALORES

- Criar tabela:

tbIngresso (codigo_ingresso: inteiro, numero_sala: inteiro, codigo_filme: inteiro, data: date, horário: time, data_venda:date, preco: dinheiro);

Numero_sala referencia tbSalaFilme;

Codigo_filme referencia tbSalaFilme;

Data e horário referencia tbSalaFilme

- Inserir valores;
- Inserir estes filmes em cartaz: Jurassic Park, A Lista de Schindler, Super 8, Lincoln → sob direção de Steven Spielberg.

FUNÇÕES - AVG

- Calcula Média de um conjunto de valores.
- Ex.: cálculo da média de valores de ingressos vendidos

```
SELECT AVG (tbIngresso.preco) AS 'Media de Preço'

FROM tbIngresso;
```

FUNÇÕES - SUM

- Calcula Soma de conjunto de valores.
- Ex.: Valor total arrecadado com venda de ingressos num período de 2010

```
SELECT SUM (tbFilme.preco) AS 'Total Arrecadado'

FROM tbFilme

WHERE tbFilme.data_venda BETWEEN '2010-01-01'
AND '2010-12-31';
```

FUNÇÕES – MIN E MAX

- MAX e MIN – valor máximo e mínimo, respectivamente
- Ex.: últimos filmes lançados

```
SELECT F.nome_filme, MAX (F.ano_lançamento) AS
'Lançamento Recente'
FROM tbFilme F
```

- Ex.: filmes de lançamento mais antigo

```
SELECT F.nome_filme, MIN (F.ano_lançamento) AS
'Lançamento Antigo'
FROM tbFilme F
```

FUNÇÕES - COUNT

- Conta o número de ocorrências de um atributo
- Ex.: contar a quantidade de filmes cadastrados

```
SELECT count (F.codigo_filme) AS 'Quantidade de Filmes Cadastrados'
FROM tbFilme as F
```

- Ex.: contar a quantidade de filmes cadastrados dirigidos pelo diretor 'Steven Spielberg'

```
SELECT count (F.codigo_filme) AS 'Quantidade de Filmes dirigidos por Spielberg'
FROM tbFilme as F
INNER JOIN tbDiretor D ON F.codigo_diretor = D.codigo_diretor
WHERE D.nome_diretor = 'Steven Spielberg';
```

GROUP BY

- Usado para formar grupos e categorizar os resultados de grupos.
- Ex.: Quantidade de Filmes por diretor

```
SELECT COUNT (DISTINCT F.codigo_filme) AS
'Quantidade de Filmes', D.nome_diretor

FROM tbFilme F

INNER JOIN tbDiretor D ON
F.codigo_diretor=D.codigo_diretor

GROUP BY D.nome_diretor;
```

HAVING

- Condição em grupos formados pelo GROUP BY.
- Ex.: Imaginem no exemplo anterior para listar Quantidade de Filmes por diretor, apenas para os que tiverem mais do que 2 filmes:

```
SELECT COUNT (DISTINCT F.codigo_filme) AS
'Quantidade de Filmes', D.nome_diretor

FROM tbFilme F

INNER JOIN tbDiretor D ON
F.codigo_diretor=D.codigo_diretor

GROUP BY D.nome_diretor

HAVING COUNT (F.codigo_filme) > 2;
```

FUNÇÕES DE DATA ATUAL

- Caso queira utilizar data atual pode-se usar: CURDATE(); SYSDATE(); NOW();
- Ex.: listar filmes que ainda serão exibidos, junto com o número da sala, data e horário de exibição:

```
SELECT F.nome_filme, SF.numero_sala, SF.data, SF.horario

FROM tbFilme F

INNER JOIN tbSalaFilme SF ON SF.codigo_filme=F.codigo_filme

WHERE SF.data > curdate(); {Sysdate (); ou now(); - retornam horário também.}
```

VALORES NULOS

- Para Nulos usa-se “IS NULL” e para os não nulos “IS NOT NULL”.

TABELAS DERIVADAS

- Pode-se criar novas tabelas em outras cláusula e manipulá-las, por exemplo no FROM
- Listagem de categorias e quantidade de filmes por categoria

```
SELECT tbNova.cat, tbNova.Quantidade AS 'Quantidade de filmes por categoria'
FROM (SELECT categoria_filme cat, COUNT (codigo_filme) AS Quantidade
      FROM tbFilme
      GROUP BY categoria_filme) AS tbNova
```