Banco de Dados I

SQL

2014-1

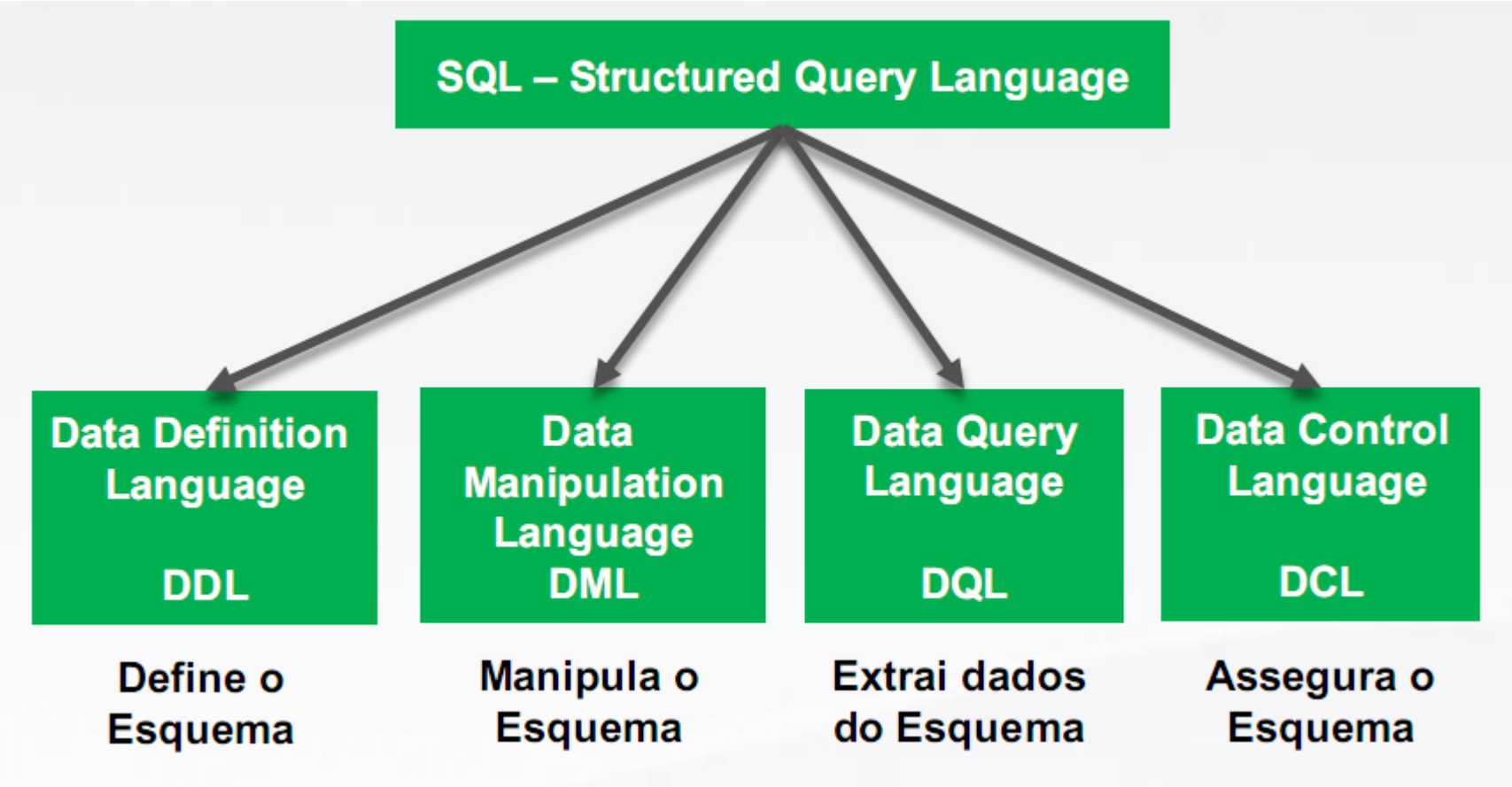
Profa.: Márcia Sampaio Lima

EST - UEA

SQL

- Structured Query Language (SQL),
- Linguagem de Consulta Estruturada
  - é uma linguagem de pesquisa declarativa para BD relacionais.
- Propósitos:
  - Servir como Linguagem de Definição de Dados (DDL), ou seja, uma linguagem que serve para informar ao SGBD qual a estrutura do BD, dando uma descrição completa dos meus metadados.
  - Servir como Linguagem de Manipulação de Dados (DML). Linguagem usada para gravar, alterar, excluir ou atualizar os dados BD.

SQL


SQL – Structured Query Language
Data Definition Language
DDL
Data Manipulation Language
DML
Data Query Language
DQL
Data Control Language
DCL
Define o Esquema
Manipula o Esquema
Extrai dados do Esquema
Assegura o Esquema

SQL

- DDL – Data Definition Language
  - Permite a criação dos componentes de BD, como tabelas, índices, etc.
  - Principais comandos:
  - CREATE TABLE
    - Cria uma nova tabela em um BD existente
  - ALTER TABLE
    - Altera uma tabela em um BD existente
  - DROP TABLE
    - Exclui uma tabela em um BD existente

SQL

- DML – Data Manipulation Language
  - Subconjunto de instruções usado para realizar inclusões, alterações e exclusões de dados presentes em registros de uma tabela.
  - Principais comandos:
  - INSERT
    - Insere novos registros em uma tabela
  - UPDATE
    - Atualiza dados já existentes
  - DELETE
    - Exclui registros de tabelas

SQL

- Data Query Language – DQL
  - Permite extrair dados do BD
  - Principal comando:
  - SELECT
    - Usado para realizar consultas a dados em tabelas

SQL

- Data Control Language – DCL
  - Provê segurança interna do BD
  - Principal comando:
  - CREAT USER
  - ALTER USER
  - CREAT SCHEMA

Código Armazenado no Banco de Dados

- O Modelo Relacional não previa, originalmente, a possibilidade de armazenar trechos de código no banco de dados. No entanto, foi adaptado para permitir a definição de
  - *Stored Procedures*: trechos de código escritos em linguagem SQL, armazenados no BD, e que podem ser ativados a partir de aplicações-cliente, comandos SQL, outras *stored procedures*, etc.
  - *Triggers*: trechos de código armazenados no BD ativados automaticamente após determinados eventos

SQL

- **CREATE:**
  - Ele serve para criar um objeto no Banco de Dados.
  - Usado para criar o próprio Banco de Dados.

    - **Create database** DBemprestimo;

Nome do Banco

SQL

- Um esquema do Banco de Dados é uma coleção de objetos de um banco de dados que estão disponíveis para um determinado usuário ou grupo.
- Os objetos de um esquema:
  - Tabelas,
  - Visões,
  - Procedimentos armazenados,
  - Índices.
- Esquema é sinônimo de Banco de Dados propriamente dito.

SQL

- CREATE TABLE:
  - Usado para especificar uma nova relação:
    - Dado nome
    - Especificado os atributos
    - Especificado restrições iniciais

```
CREATE TABLE Diretor (
  idDiretor INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  nomeDiretor VARCHAR(50) NOT NULL,
  PRIMARY KEY(idDiretor)
);
```

SQL

- CREATE TABLE:
  - Definição dos atributos:
    - Dado um nome
    - Tipo que especifica domínio dos valores
    - Algumas restrições:
      - NOT NULL
      - Chave
      - Integridade de entidade
      - Integridade referencial

SQL

```
CREATE TABLE Ator (
  idAtor INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  nomeAtor VARCHAR(50) NOT NULL,
  PRIMARY KEY(idAtor)
);
```

SQL

```
CREATE TABLE Empregado (
  codigo        CHAR(9)                 NOT NULL,
  fName          VARCHAR(15)            NOT NULL,
  lName         VARCHAR(15) NOT NULL,
  dataNasc      DATE,
  salario       DECIMAL(10,2),
  depNum        INT      NOT NULL,
  PRIMARY KEY(codigo),
 FOREIGN KEY(depNum) REFERENCES DEPARTAMENTO(num)


);
```

SQL

- FOREIGN KEY(depNum):
  - Informa que o campo depNum é uma chave estrangeira.
- REFERENCES DEPARTAMENTO(num):
  - Informa que o campo num da tabela Departamento é uma chave estrangeira na tabela empregado.

SQL

- ALTER TABLE:
  - o ALTER serve para alterar determinado objeto. Exemplo: mudar a estrutura de determinada tabela, adicionando um campo extra.
  - Por exemplo:
    - Adicionar o campo nacionalidade a tabela ator.

ALTER TABLE Ator ADD COLUMN nacionalidade VARCHAR(25) NULL;

SQL

ALTER TABLE Amigo ADD COLUMN sexo CHAR NOT NULL CHECK (sexo IN ('M','F'));

- Acrescenta o campo sexo, na tabela Amigo.
- Sexo será do tipo CHAR e não aceita valores nulos.
- A cláusula CHECK:
  - Restringe **o domínio desse campo**.
  - O significado da expressão sexo IN ('M','F')
    - Só os valores 'M' ou 'F‘ são válidos para o campo sexo.
    - O domínio definido só aceita 'M' ou 'F' maiúsculos.

SQL

- DROP
- Comando na DDL usado para apagar seu BD ou tabelas.

DROP DATABASE DBemprestimo;

DROP TABLE Diretor;

SQL

- Excluir uma coluna de uma tabela:
  - Usamos o comando ALTER combinado com o DROP. Da seguinte forma:

ALTER TABLE Ator DROP COLUMN nacionalidade;

SQL

- Objetos do Banco de Dados:
  - INDEX (índices),
  - TRIGGER (gatilho),
  - FUNCTION (função),
  - PROCEDURE (procedimento) e
  - VIEW (visão).

SQL

- **Índices:**
  - Servem para tornar as consultas mais rápidas.
  - Semelhança:
    - É um processo parecido com o que fazemos quando procuramos uma informação em um livro. Qual o primeiro passo? Procuramos no índice do livro para saber em que página está a informação que queremos. E se o livro não tiver índice? Bem, ai teremos que folhear página por página até encontrar o conteúdo desejado.

SQL

- **Índices:**
- Os índices tornam a recuperação de dados mais eficiente.
- Para toda chave primária o SGBD cria automaticamente um índice para aquele(s) campo(s).
- Logo, existe um índice para o campo código na tabela Empregado.
- Funcionamento:
  - Imaginem que essa tabela tenha 5 mil Empregados cadastrados. Se pedirmos para o SGBD procurar um empregado específico, a partir do seu codigo, o SGBD não vai procurar registro por registro da tabela. O SGBD ai consultar um índice, e vai direto ao ponto na tabela, encontrando o empregado.

SQL

```
CREATE TABLE Empregado (
  codigo        CHAR(9)                 NOT NULL,
  fName          VARCHAR(15)            NOT NULL,
  lName         VARCHAR(15) NOT NULL,
  dataNasc      DATE,
  salario       DECIMAL(10,2),
  depNum        INT      NOT NULL,
  PRIMARY KEY(codigo),
 FOREIGN KEY(depNum) REFERENCES DEPARTAMENTO(num);


);
```

SQL

- Criando novos índices:
- Criar um índice para o campo **fName** da tabela **Empregado** , para que quando um empregado seja procurado pelo nome, o SGBD use um índice.

CREATE INDEX ID_EMP_01 ON Amigo (nome);

- CREATE INDEX cria um índice.
- ID_EMP_01 nome do índice (poderia ser qualquer nome).
- Na seqüência o nome da tabela e do(s) campo(s) entre parênteses.

SQL

```
CREATE TABLE Ator (
  idAtor INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  nomeAtor VARCHAR(50) NOT NULL,
 INDEX ind_nomeAtor(nomeAtor),
  PRIMARY KEY(idAtor)
);
```

SQL

- **TRIGGERS:**
  - São trechos de código (sequência de comando SQL) que são ativados (disparados) automaticamente.
  - Ao criamos uma trigger informando ao SGBD duas informações:
    - um evento - é o que vai ativar a trigger;
    - uma ação - são os passos que devem ser executados na trigger.

- Exemplo:
  - Um evento: é uma alteração em um registro de um tabela.
  - Ação: ser gravar um registro em outra tabela, registrando a data, hora, pessoa que alterou.
  - Objetivo → o SGBD para vigiar essas operações para você.

SQL

```
CREATE TRIGGER mytrigger
BEFORE UPDATE ON emprestimo FOR EACH ROW
BEGIN
.....
END;
```

Exemplo de *Trigger*

```
create trigger t_itens_pedidos after insert or
   update or delete on pedidos_produtos for each row
begin
    if inserting or updating then
   update pedidos
   set valor_total = valor_total + :new.valor *
   :new.quantidade
   where num_pedido = :new.num_pedido;
    endif;
    if deleting or updating then
   update pedidos
   set valor_total = valor_total - :old.valor *
   :old.quantidade
   where num_pedido = :old.num_pedido;
    endif;
end;
/
```

SQL

- **PROCEDURES:**
  - É um conjunto de instruções em SQL que serve para executar uma determinada tarefa.
  - Possibilitam guardar parte das regras de negócio da aplicação dentro do BD.
  - Podem ser usadas pelo DBA para automatizar tarefas rotineiras (ex.: backups).
  - Também conhecidas como **STORED PROCEDURES.**
  - São como pequenos trechos de programa, ou seja, seqüência de comandos, que realizam uma determinada tarefa.

SQL

- **PROCEDURES:**
  - PROCEDURES X TRIGGERS
    - Ambas têm um conjunto de instruções que executam determinada tarefa
    - TRIGGERS estão associadas a um evento e são executadas automaticamente quando esse evento ocorre.
    - PROCEDURES não estão associadas a eventos, nem a tabelas específicas, e precisam ser executadas diretamente para surtirem efeito.

SQL

CREATE PROCEDURE myProcedure ()
BEGIN

....

END;

- Pode-se passar parâmetros para a PROCEDURE.
- Exemplo: uma PROCEDURE que calcula a quantidade de dias que um cliente da nossa aplicação está atrasado na devolução de uma livro.
- Concluindo: as PROCEDURES são objetos do Banco de Dados que servem para guardar uma seqüência de comandos, usados de forma lógica para resolver um determinado problema.

Exemplo de *Stored Procedure*

```
create procedure reajusta_precos (percentual
  in number) as
begin
  update produtos
  set preco = preco * (1 + percentual/100);
end
/
```

SQL

- FUNCTIONS:
- Semelhante a uma PROCEDURE, porém retorna um valor.
- É usada para efetuar determinado conjunto de comando, e no final, ela retorna o resultado.
- Por exemplo:
  - Uma FUNCTION que recebe como parâmetro um determinado CPF, e retorna o seu dígito verificador.

```
CREATE FUNCTION digito (CPF VARCHAR(11))
RETURNS INTEGER
Begin
...
End;
```

SQL

As principais operações feitas nos Bancos de Dados, a partir desse momento são: inclusão de dados, exclusão de dados, alteração de dados e consulta a dados.

Fazemos com os comandos da DML.

SQL

- **INSERT**
  - Comando SQL que usado para inserirmos dados em determinada tabela do Banco de Dados.
  - Sintaxe:

```
INSERT INTO Ator (nomeAtor) VALUES ('KEANU REAVES');
INSERT INTO Ator (nomeAtor) VALUES ('LAURENCE');
COMMIT;
```

- Ator(IdAtor, nomeAtor).
- idAtor é AUTO INCREMENT

Ele deve ser executado depois dos comandos DML para confirmar a operação. Ele grava definitivamente a operação realizada no comando no BD.

SQL

- **COMMIT**
  - Confirma todas as operações DML anteriores à sua execução
- **ROLLBACK**
  - Desfaz essas operações

SQL

```
CREATE TABLE Empregado (
  codigo        CHAR(9)                 NOT NULL,
  fName          VARCHAR(15)            NOT NULL,
  lName         VARCHAR(15) NOT NULL,
  dataNasc      DATE,
  salario       DECIMAL(10,2),
  depNum        INT      NOT NULL,
  PRIMARY KEY(codigo),
 FOREIGN KEY(depNum) REFERENCES DEPARTAMENTO(num);


);
```

SQL

- INSERT INTO Empregado
  (codigo,fName,lName,dataNasc,salario,depNum)
  VALUES
  ('123456789',‘José',‘Silva',’25/09/1980',2300.00,4);

IGUAL A

- INSERT INTO Empregado
  VALUES
  ('123456789',‘José',‘Silva',’25/09/1980',2300.00,4);

SQL

- DELETE
  - Exclui tuplas de tabelas.

DELETE * FROM Amigo;

- Excluiu TODAS as tuplas da tabela Amigo.

SQL

- DELETE

DELETE FROM Amigo WHERE cpf = '12345678911';

- A cláusula WHERE seleciona o conjunto de tuplas a serem excluídas.
- O que esse comando faz ?
  - Exclui as linhas da tabela Amigo que têm o campo CPF com valor igual a '12345678911'.
- Se mais de uma linha satisfizer a condição?
  - Todas as linhas que satisfizeram a condição serão deletadas.
- Se nenhuma linha satisfizer a condição?
  - Não, simplesmente o comando não faz nada.

SQL

- TRUNCATE
  - Limpa toda a tabela, deixando-a vazia.

TRUNCATE TABLE NomeTabela;

- TRUNCATE TABLE é um comando DDL
- TRUNCATE TABLE também não aciona triggers.
- TRUNCATE TABLE não pode ser desfeito (Rollback). O DELETE pode.

SQL

- UPDATE:
  - Usado para atualizar um ou mais campos de uma ou mais linhas de determinada tabela.

UPDATE Ator SET nomeAtor = 'KEANU REEVES‘

Qual o conseqüência deste comando?

SQL

- UPDATE:
  - Usado para atualizar um ou mais campos de uma ou mais linhas de determinada tabela.

UPDATE Ator SET nomeAtor = 'KEANU REEVES‘

Qual o conseqüência deste comando?

Atualiza o nome de todos os atores que estão na tabela Ator para a string KEANU REEVES

SQL

UPDATE Ator SET nomeAtor = 'KEANU REEVES'
WHERE nomeAtor = 'KEANU REAVES';

- Composição do UPDATE:
  - Palavra chave UPDATE, seguida do nome da tabela, depois temos a palavra chave SET, seguida pelo(s) campo(s) a serem atualizados, e por fim um WHERE com a operação de seleção, para informar em que linhas (ou registros, ou tuplas) serão feitas as atualizações.

SQL

```
UPDATE Amigo SET fone = '3333-4444' ,
cidade = 'Manaus' WHERE cpf =
'111.111.111-11';
```

SQL

- SELECT:
- É o comando que recupera as informações do Banco de Dados.
- Presente nas versões de SGBD mais simples e nas mais complexas.
- Uso mais básico:
  - Corresponde a operação de projeção da álgebra relacional.

SELECT * FROM DVD;

SQL

- SELECT:

SELECT * FROM DVD;

  - Qual a conseqüência deste comando?

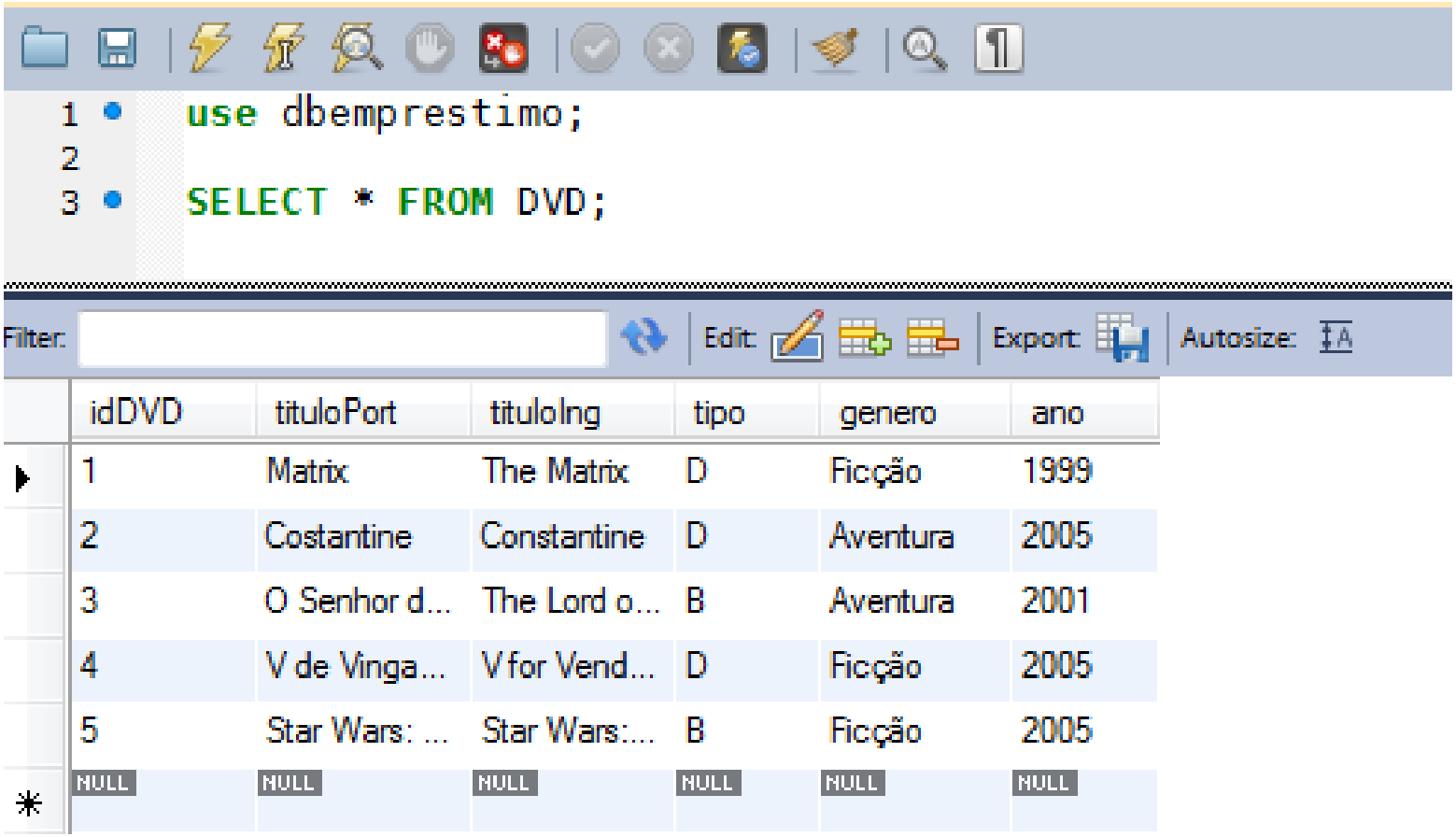
SQL

- SELECT:

SELECT * FROM DVD;

  - Qual a conseqüência deste comando?
- Esse comando seleciona todas as tuplas da tabela DVD. Isso porque o WHERE não foi especificado, ou seja, a seleção.
- O * depois do SELECT é a nossa projeção. Informa ao BD que eu quero todas as colunas dessa relação.

SQL



```
1  use dbemprestimo;
2
3  SELECT * FROM DVD;
```

Filter: Edit: Export: Autosize:

idDVD	tituloPort	tituloIng	tipo	genero	ano
1	Matrix	The Matrix	D	Ficção	1999
2	Costantine	Constantine	D	Aventura	2005
3	O Senhor d...	The Lord o...	B	Aventura	2001
4	V de Vinga...	V for Vend...	D	Ficção	2005
5	Star Wars: ...	Star Wars:...	B	Ficção	2005
NULL	NULL	NULL	NULL	NULL	NULL

SQL

- Objetivo: projetar somente alguns campos e ver todas as tuplas.

SELECT tituloPort, genero, ano FROM DVD;

- Objetivo: fazer um SELECT usando projeção e seleção.

SELECT tituloPort, genero, ano FROM DVD

WHERE ano > 2000;

- Projeção → lista os campos desejados.
- SELECT → seleciona um conjunto de tuplas.

SQL

SELECT tituloPort, genero, ano FROM DVD WHERE ano > 2000;

```
7 •  SELECT tituloPort, genero, ano FROM DVD WHERE ano > 2000;
```

Filter: Export: Autosize:

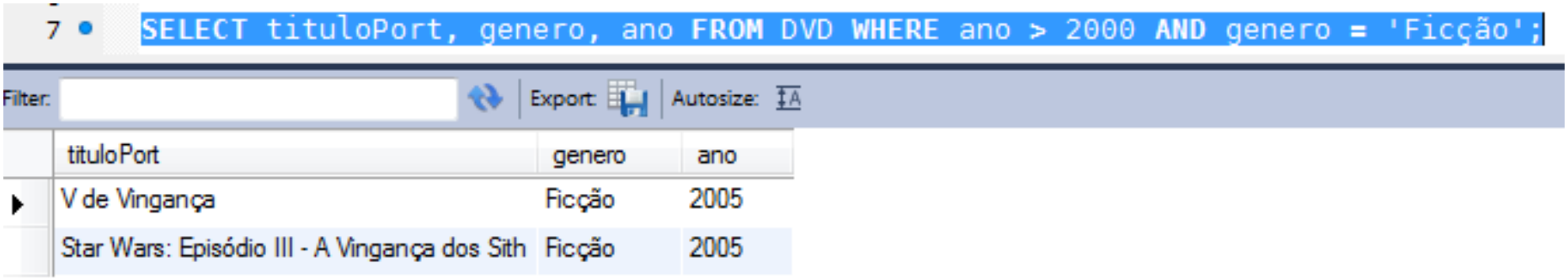
tituloPort	genero	ano
Costantine	Aventura	2005
O Senhor dos Anéis: A Sociedade do Anel	Aventura	2001
V de Vingança	Ficção	2005
Star Wars: Episódio III - A Vingança dos Sith	Ficção	2005

Operadores: < (menor), >= (maior ou igual), <= (menor ou igual), <> (diferente).

SQL

- Objetivo: Recuperar as tuplas em que os filmes tiverem o campo ano maior que 2000, e ao mesmo tempo sejam filmes de ficção?
  - Operadores lógicos: AND, OR e NOT.

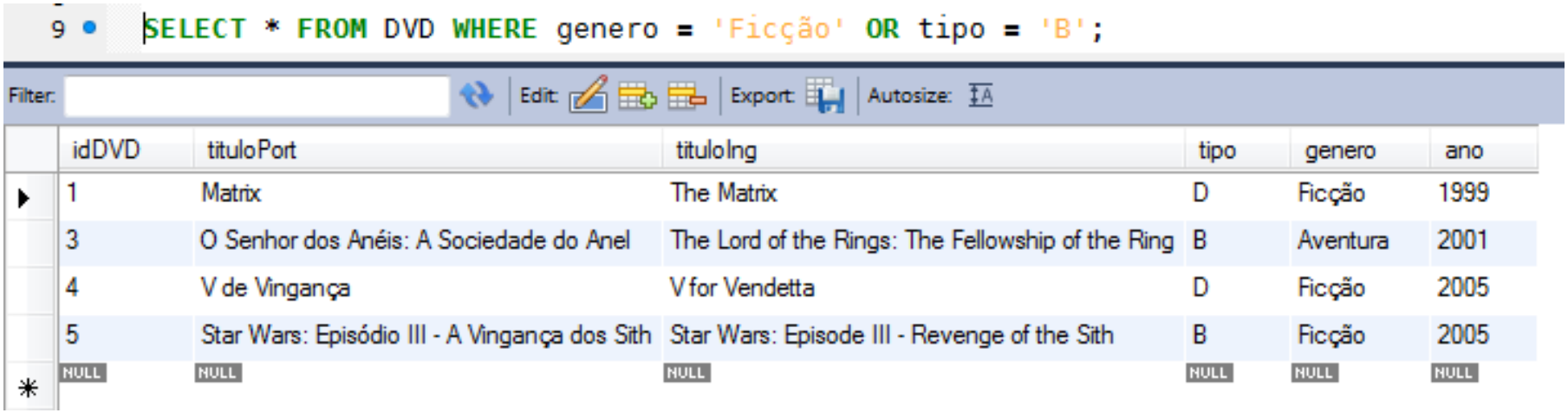
SELECT tituloPort, genero, ano FROM DVD
WHERE ano > 2000 AND genero = 'Ficção';



SQL

- Objetivo: Filmes que sejam ou de ficção, ou que sejam do tipo B (Blu-ray) .

```
SELECT * FROM DVD
 WHERE genero = 'Ficção' OR tipo = 'B';
```



idDVD	tituloPort	tituloIng	tipo	genero	ano
1	Matrix	The Matrix	D	Ficção	1999
3	O Senhor dos Anéis: A Sociedade do Anel	The Lord of the Rings: The Fellowship of the Ring	B	Aventura	2001
4	V de Vingança	V for Vendetta	D	Ficção	2005
5	Star Wars: Episódio III - A Vingança dos Sith	Star Wars: Episode III - Revenge of the Sith	B	Ficção	2005
NULL	NULL	NULL	NULL	NULL	NULL

SQL

- Objetivo: testar o operador NOT. Recuperar todos os filmes que não são do gênero aventura.

```
SELECT * FROM DVD
WHERE genero <> 'Aventura';
```

```
SELECT * FROM DVD
WHERE NOT genero ='Aventura';
```

- Os dois comandos se equivalem.

SQL - IN, BETWEEN, LIKE

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

```
SELECT * FROM Customers
WHERE City IN ('Paris','London');
```

SQL

Customers

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

SQL

- Objetivo: descobrir o código dos atores que atuaram nos filmes **Matrix** e **V de vingança**. Preciso primeiro: descobrir qual o código desses filmes.

```
SELECT * FROM DVD
WHERE tituloPort = 'Matrix' OR tituloPort = 'V
   de Vingança';
```

SQL

```
14
15 •  SELECT * FROM DVD WHERE tituloPort = 'Matrix' OR tituloPort = 'V de Vingança';
```

Filter: | Edit: | Export: | Autosize:

idDVD	tituloPort	tituloIng	tipo	genero	ano
1	Matrix	The Matrix	D	Ficção	1999
4	V de Vingança	V for Vendetta	D	Ficção	2005
NULL	NULL	NULL	NULL	NULL	NULL

- Sabendo que os códigos desses filmes são 1 e 4, faço a consulta na tabela que relaciona DVDs e Atores:

SELECT idAtor FROM AtorParticipa

WHERE idDVD IN (1,4);

SQL

```
17 •   SELECT idAtor FROM AtorParticipa WHERE idDVD IN (1,4);
18
```

Filter: Export: Autosize:

idAtor
1
2
3
4
4
11
12
13
14

SQL

- Temos o idAtor dos atores que participam dos filmes 1 ou 4.
- Note que:
  - O idAtor 4 aparece duas vezes, pois o mesmo participa dos dois filmes.
  - **DISTINCT**: faz com que elementos repetidos apareçam só uma vez.

```
SELECT DISTINCT(idAtor) FROM AtorParticipa
WHERE idDVD in (1,4);
```

SQL

```
17 •  SELECT DISTINCT(idAtor) FROM AtorParticipa WHERE idDVD IN (1,4);
18
```

idAtor
1
2
3
4
11
12
13
14

SQL

- Operador BETWEEN
  - Checa um intervalo de valores, sendo verdadeiro quando o valor testado está nesse intervalo.

- Objetivo: Encontrar os filmes lançados entre os anos de 1990 e 2001.

```
SELECT * FROM DVD
WHERE ano BETWEEN 1990 AND 2001;
```

- BETWEEN é usado da seguinte forma:
  - BETWEEN limite_inferior AND limite_superior.

SQL

```
19 •  SELECT * FROM DVD WHERE ano BETWEEN 1990 AND 2001;
```

Filter: | Edit: | Export: | Autosize:

idDVD	tituloPort	tituloIng	tipo	genero	ano
1	Matrix	The Matrix	D	Ficção	1999
3	O Senhor dos Anéis: A Sociedade do Anel	The Lord of the Rings: The Fellowship of the Ring	B	Aventura	2001
NULL	NULL	NULL	NULL	NULL	NULL

SQL

- Operador LIKE
  - Serve para comparar strings.
  - Com o operador = não é possível fazer essa comparação.
- Objetivo: obter da tabela DVD os filmes cujos campos tituloPort possuam a palavra Vingança.

SELECT * FROM DVD WHERE tituloPort LIKE '%Vingança%';

- Filmes cujo título têm, em qualquer posição, a string Vingança.
- Filmes que começam com a string Vingança: Vingança%.
- Filmes que terminam com a string Vingança: %Vingança.

SQL

```
21 •  SELECT * FROM DVD WHERE tituloPort LIKE '%Vingança%';
```

Filter: | Edit: | Export: | Autosize:

idDVD	tituloPort	tituloIng	tipo	genero	ano
4	V de Vingança	V for Vendetta	D	Ficção	2005
5	Star Wars: Episódio III - A Vingança dos Sith	Star Wars: Episode III - Revenge of the Sith	B	Ficção	2005
NULL	NULL	NULL	NULL	NULL	NULL

SQL

- Objetivo: Retornar todos os filmes que estão entre A e T, ou seja, que começam da letra A em diante, até a letra T e filmes que o campo gênero comece com F.

```
SELECT * FROM DVD
WHERE (tituloPort BETWEEN 'A' AND 'T') AND
   (genero LIKE 'F%');
```

SQL

```
23 •  SELECT * FROM DVD WHERE (tituloPort BETWEEN 'A' AND 'T') AND (genero LIKE 'F%');
```

Filter: Edit: Export: Autosize:

idDVD	tituloPort	tituloIng	tipo	genero	ano
1	Matrix	The Matrix	D	Ficção	1999
5	Star Wars: Episódio III - A Vingança dos Sith	Star Wars: Episode III - Revenge of the Sith	B	Ficção	2005
NULL	NULL	NULL	NULL	NULL	NULL

SQL

- ORDER BY
  - Serve para ordenar nossa resposta segundo algum critério.
- Objetivo: todos os filmes com ano de lançamento maior que 2000, mas quero a resposta ordenada pelo título em português:

```
SELECT * FROM DVD WHERE ano > 2000
ORDER BY tituloPort;
```

  - ORDER BY não altera as tuplas que serão retornadas, altera apena a ordem de apresentação delas.

SQL

```
27 •  SELECT * FROM DVD WHERE ano > 2000 ORDER BY tituloPort;
```

Filter: Edit: Export: Autosize:

idDVD	tituloPort	tituloIng	tipo	genero	ano
2	Costantine	Constantine	D	Aventura	2005
3	O Senhor dos Anéis: A Sociedade do Anel	The Lord of the Rings: The Fellowship of the Ring	B	Aventura	2001
5	Star Wars: Episódio III - A Vingança dos Sith	Star Wars: Episode III - Revenge of the Sith	B	Ficção	2005
4	V de Vingança	V for Vendetta	D	Ficção	2005
NULL	NULL	NULL	NULL	NULL	NULL

SQL

- Ordenação descendente:
  - A ordenação por padrão é na ordem ascendente, mas podemos ordenar também de forma descendente.
  - Usar a palavra chave DESC.

SELECT DISTINCT(idAtor) FROM AtorParticipa
WHERE idDVD IN (1,4) ORDER BY idAtor DESC;

SQL

- Ordenação por dois campos:
  - Quando escolhemos mais de um campo para ordenar, podemos definir para cada um se a ordem é ascendente (ASC) ou descendente (DESC).
  - Ausência de ASC ou DESC é considerada a ordem ascendente.

```
SELECT * FROM DVD
ORDER BY genero DESC, tituloPort ASC;
```

  - Mostra todos os filmes ordenando primeiro pelo campo gênero em ordem descente, depois por tituloPort, em ordem ascendente.

SQL

- Funções :
  - Ajudam a extrair informações das tabelas.

  - COUNT(): Serve para contar quantas tuplas são retornadas em uma query.
  - Primeiro caso: quantidade de tuplas de uma tabela.

SELECT COUNT(*) FROM DVD;

SQL

- COUNT()
  - Segundo caso: quantidade de tuplas de uma seleção.
  - Uso do COUNT par determinar quantas tuplas de uma tabela obedecem a uma determinada condição:

SELECT COUNT(*) FROM DVD WHERE ano = 2005;

  - Temos agora como resultado o valor 3, que é o número de filmes lançados no ano de 2005.

SQL

```
SELECT COUNT(*) NumeroDeLinhas
FROM WORLD.COUNTRY;
```

- Para fazer uma query em uma tabela de outro esquema, basta colocar o nome do esquema na frente.
- Assim tenho nessa query a quantidade de tuplas que a tabela Country do esquema World possui.
- NumeroDeLinhas é um apelido (alias) usado para ser exibido como nome do campo na resposta.

SQL

```
35 •   SELECT COUNT(*) NumeroDeLinhas FROM WORLD.COUNTRY;
```

SQL

- SUM().
  - Serve para somar os valores de determinada coluna.
  - Deve ter um campo numérico, que faça sentido somar.
- Objetivo: Temos na tabela Country um campo chamado population, que tem o número de habitantes de um país, e temos também o campo continent. Vou somar as populações dos países que estão na Ásia:

```
SELECT SUM(population) SOMA FROM
WORLD.COUNTRY WHERE continent = 'Asia';
```

SQL

```
39 •  SELECT SUM(population) SOMA FROM WORLD.COUNTRY WHERE continent = 'Asia';
```

Filter: Export: Autosize:

SOMA
3705025700

SQL

- AVG()
  - Calcula a média aritmética de um conjunto de valores.

```
SELECT AVG(population) MEDIA FROM
WORLD.COUNTRY WHERE continent = 'Asia';
```

SQL

- MAX():
  - Retorna o maior dos valores.
  - Descobrir qual a maior população que um país da Europa tem:

```
SELECT MAX(population) FROM
WORLD.COUNTRY WHERE continent =
  'Europe';
```

SQL

- MAX() e MIN()
  - Obtem o valor máximo e o valor mínimo de um conjunto.

```
SELECT MAX(population) MAIOR,
   MIN(population) MENOR
FROM WORLD.COUNTRY
WHERE continent = 'Europe';
```

SQL

SELECT now() DATA_HORA,
(150 * 6) Multiplicação,
(1975 - 520) Subtração;

```
48 •  SELECT now() DATA_HORA, (150 * 6) Multiplicação, (1975 - 520) Subtração;
```

Filter: Export: Autosize:

DATA_HORA	Multiplicação	Subtração
2012-05-27 10:22:24	900	1455

SQL

- Cláusula GROUP BY
  - Usada geralmente em conjunto com as funções.
  - Objetivo: Saber a soma da população de cada continente.

SELECT continent CONTINENTE, SUM(population) SOMA

FROM WORLD.COUNTRY GROUP BY continent;

- Esse comando agrupa as tuplas pelo campo continent (GROUP BY), e faz a soma da população de cada continente.

SQL

```
50 •  SELECT continent CONTINENTE, SUM(population) SOMA FROM WORLD.COUNTRY GROUP BY continent;
```

Filter: | Export: | Autosize:

CONTINENTE	SOMA
Asia	3705025700
Europe	730074600
North America	482993000
Africa	784475000
Oceania	30401150
Antarctica	0
South America	345780000

SQL

```
SELECT continent CONTINENTE, SUM(population)
   SOMA
FROM WORLD.COUNTRY GROUP BY continent
HAVING SUM(population) > 0;
```

- Agora só interessam continentes que a soma da população seja maior que zero.
- Sempre que usamos o GROUP BY e queremos fazer um teste com nossa função que conta, soma, calcula média usamos a cláusula HAVING, da seguinte forma.

SQL

```
SELECT genero, COUNT(*) FROM DVD WHERE
    ano > 2000
GROUP BY genero ORDER BY 1;
```

- Seleciona as tuplas que tem ano maior que 2000.
- Agrupa por gênero e conta (COUNT).
- No final, mostra o resultado ordenado pelo primeiro campo (1).

SQL

```
55 •  SELECT genero, COUNT(*) FROM DVD WHERE ano > 2000
56    GROUP BY genero ORDER BY 1;
```

Filter: | Export: | Autosize:

genero	COUNT(*)
Aventura	2
Ficção	2

SQL

- Junções: INNER JOIN
  - Retorna linhas quando existe pelo menos uma combinação em ambas as tabelas.
  - Temos no esquema world, uma tabela chamada city. Essa tabela tem um campo chamado countrycode (chave estrangeira), que é o código do país, e que está relacionado com a tabela country. Para listar as cidades contidas em city, mas mostrando o nome do país ao invés do código, executo a seguinte query:

```
SELECT city.id, city.name Cidade, country.name Pais,
city.population Populacao
FROM city INNER JOIN country
ON city.countrycode = country.code;
```

SQL

```
3 •  select city.id, city.name Cidade, country.Name Pais, city.population Populacao
4    from city
5    inner join country
6    on city.countrycode = country.code;
```

Filter: Export: Autosize: Fetch rows:

id	Cidade	Pais	Populacao
1	Kabul	Afghanistan	1780000
2	Qandahar	Afghanistan	237500
3	Herat	Afghanistan	186800
4	Mazar-e-Sharif	Afghanistan	127800
5	Amsterdam	Netherlands	731200
6	Rotterdam	Netherlands	593321
7	Haag	Netherlands	440900
8	Utrecht	Netherlands	234323
9	Eindhoven	Netherlands	201843
10	Tilburg	Netherlands	193238
11	Groningen	Netherlands	172701
12	Breda	Netherlands	160398
13	Apeldoorn	Netherlands	153491

SQL

- Sintaxe INNER JOIN

```
select <campos> from <tabela 1>
        inner join <tabela 2>
        on <campo tabela 1> = <campo tabela 2>
```

SQL

- Outro exemplo:

```
SELECT
   Category.CategoryID,Category.CategoryNam
   e,Category.Description,
   Product.ProductID,Product.ProductName,Pro
   duct.UnitPrice
FROM Category INNER JOIN Product ON
(Category.CategoryID = Product.CategoryID)
```

SQL

```
SELECT c.CategoryID, c.CategoryName,
   c.Description, p.ProductID, p.ProductName,
   p.UnitPrice
FROM Category c INNER JOIN Product p ON
  (c.CategoryID = p.CategoryID)
```

SQL

- A diferença das duas queries anteriores é que na primeira utilizamos o nome da tabela seguido do ponto mais o campo desejado e no segundo exemplo define um álias que se coloca logo após as tabelas.

SQL