



SQL

Linguagem de Manipulação de Dados

Profa. Maria Camila Nardini Barioni

camila.barioni@ufu.br

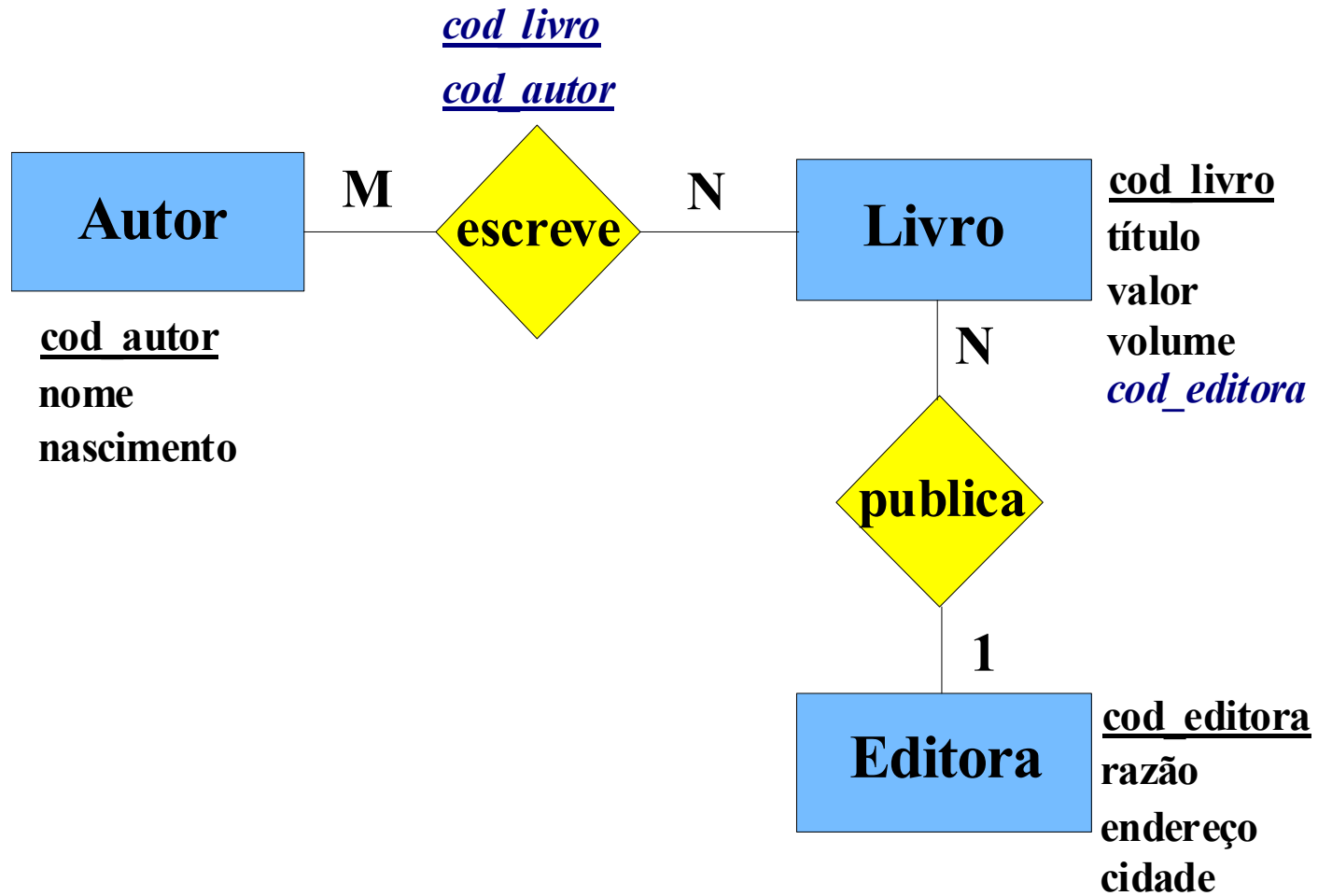
Bloco B - sala 1B137

1º semestre de 2024

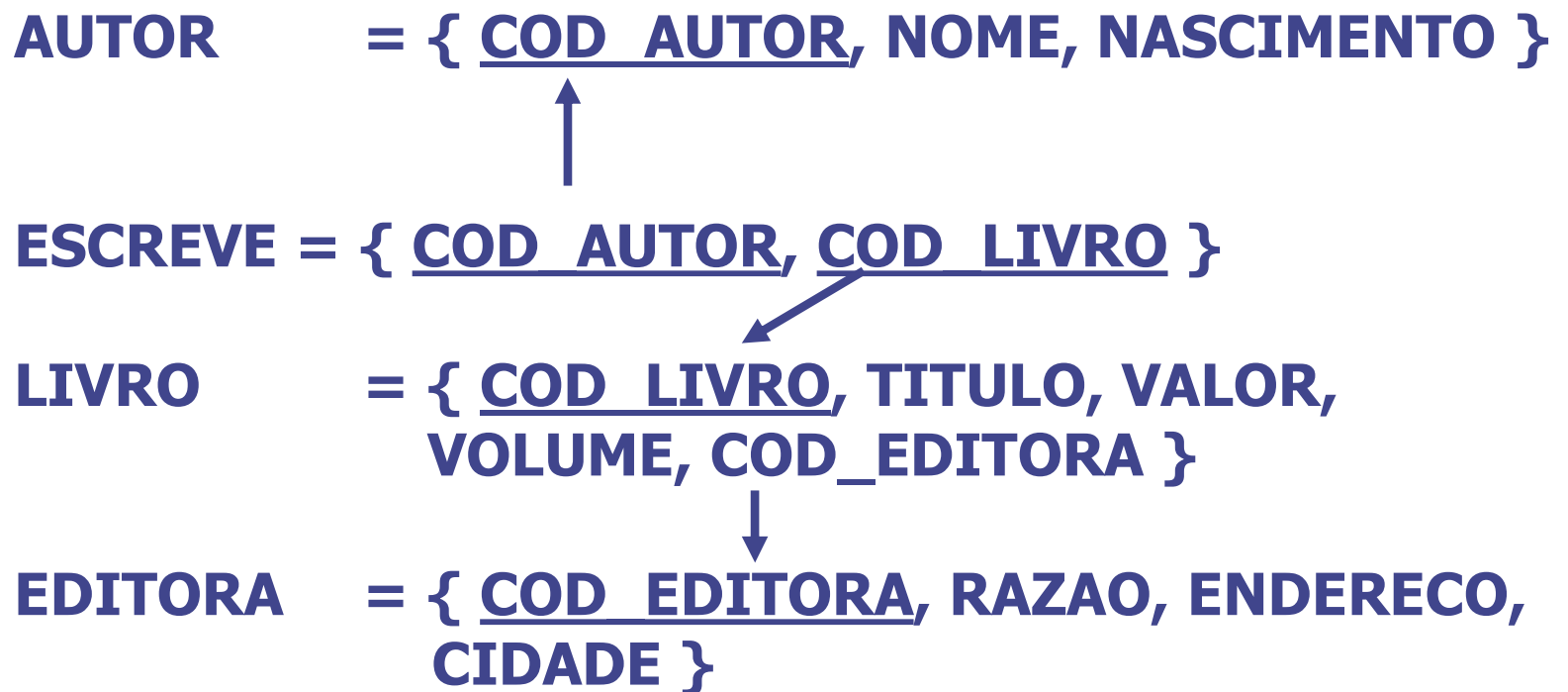
SQL

- ◆ Os comandos podem ser agrupados em 3 classes:
 - DDL - Comandos para a Definição de Dados
 - DML - Comandos para a Manipulação de Dados
 - DCL - Comandos para o controle do Gerenciador, Conexão e Usuário

Exemplo ME-R



Exemplo Modelo Relacional



Comandos de Manipulação de Dados

◆ Comandos DML

- **INSERT INTO:** adiciona uma tupla ou múltiplas tuplas a uma tabela
- **UPDATE:** altera valores de campos em uma determinada tabela com base em critérios especificados
- **DELETE:** remove tuplas de uma ou mais tabelas que satisfaçam aos critérios especificados
- **SELECT** - realiza consultas em um banco de dados

Comandos de Manipulação de Dados

◆ Comandos DML

- **INSERT INTO:** adiciona uma tupla ou múltiplas tuplas a uma tabela
- **UPDATE:** altera valores de campos em uma determinada tabela com base em critérios especificados
- **DELETE:** remove tuplas de uma ou mais tabelas que satisfaçam aos critérios especificados
- **SELECT** - realiza consultas em um banco de dados – **CONSULTAS SIMPLES**

SQL DML

Inserção

- ◆ Realizada através da especificação
 - de uma tupla particular
 - de uma consulta que resulta em um conjunto de tuplas a serem inseridas
- ◆ Valores dos atributos das tuplas inseridas
 - devem pertencer ao domínio do atributo
- ◆ Atributos sem valores
 - especificados por NULL ou valor DEFAULT

SQL DML

Inserção

```
INSERT INTO nome_tabela  
VALUES ( V1, V2, ..., VN );
```

- ◆ Ordem dos atributos deve ser mantida

SQL DML

Inserção

```
INSERT INTO nome_tabela (A1, A2, ..., An)  
VALUES ( V1, V2, ..., VN );
```

- Ordem dos atributos não precisa ser mantida

Comando INSERT INTO

- ◆ Adiciona uma tupla ou múltiplas tuplas a uma tabela
 - Para a inserção de uma única tupla é necessário especificar:
 - ◆ o nome da relação
 - ◆ a lista de valores para a tupla, **na mesma ordem** em que os atributos correspondentes foram especificados no comando CREATE TABLE

Comando INSERT INTO

◆ Exemplos:

```
CREATE SCHEMA Livraria;
SET search_path TO Livraria;

CREATE TABLE AUTOR(
    COD_AUTOR SMALLINT NOT NULL ,
    NOME VARCHAR(20) NOT NULL,
    NASCIMENTO DATE NOT NULL,
    CONSTRAINT CHAVEAUTOR PRIMARY KEY (COD_AUTOR)
);

-- Insere um registro na tabela AUTOR
INSERT INTO AUTOR
    VALUES (1, 'Paulo Coelho', '01/01/1947');
-- ok, funciona. Nesse caso tanto faz, mas
-- qual campo é o dia e qual é o mês?
```

Comando INSERT INTO

◆ Exemplos:

-- Dica manipulação de datas

```
-- descobrindo o formato atual configurado no postgresql
-- ver aquivo postgresql.conf. O meu é datestyle = 'iso,
dmy'
-- formato ISO: YYYY-MM-DD (ano,mês,dia)
-- formato dmy: DD/MM/YYYY
```

SHOW datestyle;

Comando INSERT INTO

◆ Exemplos:

-- Dica manipulação de datas

-- mudando a configuração do sistema para aceitar a data no formato anterior

```
SET datestyle TO 'mdy';
```

-- conferindo a mudança

```
SHOW datestyle;
```

-- tentando inserir a data que daria erro com a config. anterior

```
INSERT INTO AUTOR
```

```
VALUES (3, 'AUTOR FICTÍCIO', '12/30/1840');
```

Comando INSERT INTO

◆ Exemplos:

-- Dica manipulação de datas

-- É possível inserir dados independentemente do tipo
-- configurado no postgresql. Para isso use o comando to_date
-- inserindo o dia 30 de dezembro em dois formatos diferentes

```
INSERT INTO AUTOR VALUES (4,  
'AUTOR FICTÍCIO', to_date('12/30/1840', 'MM/DD/YYYY')) ;
```

```
INSERT INTO AUTOR VALUES (5,  
'AUTOR FICTÍCIO', to_date('30/12/1840', 'DD/MM/YYYY')) ;
```

Comando INSERT INTO

◆ Exemplos:

```
CREATE TABLE EDITORA(  
    COD_EDITORA SMALLINT NOT NULL,  
    RAZAO VARCHAR(20),  
    ENDERECO CHAR(20),  
    CIDADE CHAR(70),  
    CONSTRAINT CHAVEEDITORIA PRIMARY KEY  
    (COD_EDITORA)  
);  
  
-- Insere um registro na tabela EDITORA  
INSERT INTO EDITORA (COD_EDITORA, RAZAO, CIDADE)  
    VALUES (1, 'ROCCO', 'RIO DE JANEIRO');  
INSERT INTO EDITORA (COD_EDITORA, CIDADE, RAZAO)  
    VALUES (2, 'RIO DE JANEIRO', 'GLOBO');
```

Comando INSERT INTO

◆ Exemplos:

-- Insere um registro na tabela editora

```
INSERT INTO EDITORA (COD_EDITORA, RAZAO, CIDADE)
VALUES (1, 'ROCCO', 'RIO DE JANEIRO');
```

```
INSERT INTO EDITORA (COD_EDITORA, CIDADE, RAZAO)
VALUES (2, 'RIO DE JANEIRO', 'GLOBO');
```

-- Neste caso, indica-se as colunas

-- Essa sintaxe é fortemente recomendada

-- A ordem das colunas de uma tabela pode ser

-- alterada em uma eventual manutenção no BD.

Comando INSERT INTO

◆ Exemplos:

```
CREATE TABLE LIVRO (  
    COD_LIVRO SMALLINT NOT NULL,  
    TITULO CHAR(35) NOT NULL,  
    VALOR NUMERIC(7,2),  
    VOLUME SMALLINT DEFAULT 1,  
    COD_EDITORA SMALLINT NOT NULL,  
    CONSTRAINT CHAVELIVRO PRIMARY KEY (COD_LIVRO),  
    CONSTRAINT ESTRANGEIRAEDITORIA FOREIGN KEY (COD_EDITORA)  
        REFERENCES EDITORA  
);  
  
-- Pode-se omitir uma coluna  
INSERT INTO LIVRO (COD_LIVRO, TITULO, VALOR, COD_EDITORA)  
    VALUES (31, 'MAKTUB', 24, 1)  
-- Nesse caso ela é preenchida com valor DEFAULT
```

Comando INSERT INTO

◆ Exemplos:

-- Especificando um valor DEFAULT

```
INSERT INTO LIVRO (COD_LIVRO, TITULO, VALOR, VOLUME,  
COD_EDITORA)  
VALUES (55, 'BRIDA', 29.5, DEFAULT, 1)
```

Comando INSERT INTO

◆ Exemplos:

```
CREATE TABLE ESCREVE (  
    COD_LIVRO_ESC SMALLINT NOT NULL,  
    COD_AUTOR_ESC SMALLINT NOT NULL,  
    CONSTRAINT CHAVEESCREVE PRIMARY KEY (COD_LIVRO_ESC,  
    COD_AUTOR_ESC),  
    CONSTRAINT ESTRANGEIRALIVRO FOREIGN KEY (COD_LIVRO_ESC)  
    REFERENCES LIVRO,  
    CONSTRAINT ESTRANGEIRAAUTOR FOREIGN KEY (COD_AUTOR_ESC)  
    REFERENCES AUTOR  
);
```

-- Inserindo mais de uma linha

```
INSERT INTO LIVRO ( COD_LIVRO, TITULO, VALOR, COD_EDITORA )  
VALUES (55, 'BRIDA', 29.5, 1)
```

```
INSERT INTO ESCREVE VALUES (31, 1), (55, 1);
```

Comando INSERT INTO

◆ Inserção de múltiplas tuplas

- Insere múltiplas tuplas em uma tabela juntamente com a criação da tabela, preenchendo-a com o resultado de uma consulta

Comando INSERT

◆ Exemplos

```
-- Cria tabela AUTOREXPERIENTE
CREATE TABLE AUTOREXPERIENTE (
    COD_AUTOR SMALLINT NOT NULL
    CONSTRAINT CHAVEAUTOREXP
    PRIMARY KEY,
    NOME VARCHAR(30) NOT NULL,
    NASCIMENTO DATE NOT NULL
);
```

Comando INSERT

◆ Continuando...

```
-- E em seguida insere na tabela AUTOREXPERIENTE,  
-- os autores com nascimento < 01/01/1900
```

```
INSERT INTO AUTOREXPERIENTE  
SELECT *  
FROM AUTOR  
WHERE NASCIMENTO < '01/01/1900';
```

Comando INSERT

```
INSERT INTO table [ ( column [, ...] ) ]  
{ DEFAULT VALUES | VALUES ( { expression | DEFAULT } [,  
  ...] ) [, ...] | query }  
[ RETURNING * | output_expression [ [ AS ] output_name ]  
  [, ...] ]
```

<https://www.postgresql.org/docs/current/sql-insert.html>

Comando UPDATE

- ◆ É usado para modificar valores de campos de uma ou mais tuplas de uma determinada tabela, com base em critérios especificados

```
UPDATE tabela  
SET novovalor  
WHERE critérios;
```


Comando UPDATE

◆ Exemplos:

-- Alterar o endereço e a cidade da tupla da
-- tabela EDITORA com COD_EDITORA = 1

```
UPDATE EDITORA
```

```
SET ENDERECO = 'R. RODRIGO SILVA, 26',
```

```
CIDADE = 'PETRÓPOLIS'
```

```
WHERE COD_EDITORA = 1;
```

-- Reajustar o valor de todos os livros em 10%

```
UPDATE LIVRO
```

```
SET VALOR = VALOR * 1.1;
```

Comando UPDATE

```
UPDATE [ ONLY ] table [ [ AS ] alias ] SET { column
    = { expression | DEFAULT } | ( column [, ...] ) =
    ( { expression | DEFAULT } [, ...] ) } [, ...] [
    FROM fromlist ] [ WHERE condition | WHERE CURRENT
    OF cursor_name ] [ RETURNING * | output_expression
    [ [ AS ] output_name ] [, ...] ]
```

<https://www.postgresql.org/docs/current/sql-update.html>

CUIDADO: Sempre teste a clausula WHERE de um comando UPDATE antes de executá-lo. Um simples erro no WHERE pode resultar na alteração de muitas, senão todas, linhas da tabela, atribuindo a elas o mesmo valor

Comando DELETE

- ◆ Remove tuplas de uma ou mais tabelas que satisfaçam aos critérios especificados

DELETE

FROM tabela

WHERE critérios;

Comando DELETE

◆ Exemplos:

-- Apaga a tupla da tabela AUTOR com COD_AUTOR = 3

DELETE

FROM AUTOR

WHERE COD_AUTOR = 3;

-- Apaga todos os registros da tabela LIVRO

DELETE

FROM LIVRO;

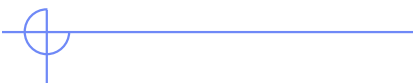
◆ **CUIDADO:** é muito fácil apagar os dados das tabelas

Comando DELETE

```
DELETE FROM [ ONLY ] table [ [ AS ] alias ]  
[ USING usinglist ]  
[ WHERE condition | WHERE CURRENT OF cursor_name ]  
[ RETURNING * | output_expression [ [ AS ]  
output_name ] [, ...] ]
```

<https://www.postgresql.org/docs/current/sql-delete.html>

SELECT



```
SELECT <lista de atributos e funções>  
FROM <lista de tabelas>  
[ WHERE predicado ]  
[ GROUP BY <atributos de agrupamento> ]  
[ HAVING <condição para agrupamento> ]  
[ ORDER BY <lista de atributos> ] ;
```

<https://www.postgresql.org/docs/current/sql-select.html>

SELECT

◆ Cláusula SELECT

- lista os atributos e/ou as funções a serem exibidos no resultado da consulta
- **corresponde à operação de projeção da álgebra relacional**

◆ Cláusula FROM

- especifica as relações a serem examinadas na avaliação da consulta
- **corresponde à operação de produto cartesiano da álgebra relacional**

SELECT

◆ Cláusula WHERE

- especifica as condições para a seleção das tuplas no resultado da consulta
 - ◆ as condições devem ser definidas sobre os atributos das relações que aparecem na cláusula FROM
- inclui condições de junção
- corresponde ao predicado de seleção da álgebra relacional
- pode ser omitida

SELECT

◆ Resultado de uma consulta

- ordem de apresentação dos atributos
 - ◆ ordem dos atributos na cláusula SELECT
- ordem de apresentação dos dados
 - ◆ ordem ascendente ou descendente de acordo com a cláusula ORDER BY
 - ◆ sem ordenação
- duas ou mais tuplas podem possuir valores idênticos de atributos
 - ◆ eliminação de tuplas duplicadas
 - SELECT DISTINCT

Comando SELECT

◆ Exemplos

-- Selecionar todos os campos da tabela autor

```
SELECT * FROM AUTOR;
```

*-- Selecionar todos os nomes e datas de nascimento
-- da tabela autor*

```
SELECT NOME, NASCIMENTO  
FROM AUTOR;
```

*-- Selecionar todos os nomes e datas de nascimento
-- da tabela autor renomeando a coluna nome para autor*

```
SELECT NOME AS AUTOR, NASCIMENTO  
FROM AUTOR;
```

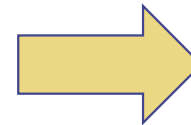
Comando SELECT

◆ Para tratar tuplas duplicadas

/ Selecionar os valores de todos os livros */*

```
SELECT ALL VALOR  
FROM LIVRO;
```

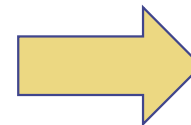
Resultado correto,
mas com tuplas duplicadas



Output pane	
Data Output Explain	
	valor numeric(7,2)
1	24.00
2	29.50
3	35.00
4	10.90
5	10.90

/ Selecionar apenas os valores de livros
distintos */*

```
SELECT DISTINCT VALOR  
FROM LIVRO;
```



Data Output Explain	
	valor numeric(7,2)
1	29.50
2	24.00
3	10.90
4	35.00

Cláusula WHERE

SELECT

FROM

WHERE <atributo> <operador>
 <valor | atributo | lista de valores>

◆ Operadores

- conjunção de condições: AND
- disjunção de condições: OR
- negação de condições: NOT

Cláusula WHERE

◆ Operadores

- conjunção de condições: AND
- disjunção de condições: OR
- negação de condições: NOT

a	b	a AND b	a OR b
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
TRUE	NULL	NULL	TRUE
FALSE	FALSE	FALSE	FALSE
FALSE	NULL	FALSE	NULL
NULL	NULL	NULL	NULL

a	NOT a
TRUE	FALSE
FALSE	TRUE
NULL	NULL

Cláusula WHERE

◆ Alguns operadores de comparação

igual a	=	diferente de	!= ou < >
maior que	>	maior ou igual a	>=
menor que	<	menor ou igual a	<=
teste de nulo	IS NULL <i>ou</i> IS NOT NULL	igual a algum de vários valores	<i>expressão</i> IN (<i>lista_valores</i>)
entre <i>dois</i> valores	BETWEEN <i>valor1</i> AND <i>valor2</i>	de cadeias de caracteres	LIKE <i>ou</i> NOT LIKE

Cláusula WHERE

- ◆ Especifica quais tuplas das tabelas listadas na cláusula FROM são afetados por uma instrução SELECT, UPDATE ou DELETE
- ◆ Se uma cláusula WHERE não for especificada, a consulta retornará todas as linhas da tabela
- ◆ Exemplo

```
/* Selecionar as editoras da cidade de São Paulo */  
SELECT RAZAO  
FROM EDITORA  
WHERE CIDADE = 'SÃO PAULO';
```

Cláusula WHERE

◆ Exemplos:

/ Selecionar os livros que não sejam da editora com
COD_EDITORA igual a 1 */*

```
SELECT *  
FROM LIVRO  
WHERE COD_EDITORA != 1;
```

/ Selecionar os livros cujo VALOR seja maior ou igual
a R\$ 24,00 */*

```
SELECT *  
FROM LIVRO  
WHERE VALOR >= 24;
```


Cláusula WHERE

◆ Exemplos

/ Selecionar todos os títulos dos livros da editora ROCCO renomeando as tabelas */*

```
SELECT LI.TITULO
FROM EDITORA ED, LIVRO LI
WHERE (ED.RAZAO = 'ROCCO') AND
      (ED.COD_EDITORA = LI.COD_EDITORA);
```

Output pane

	Data Output	Explain	Messages	History
	cod_editora smallint	razao character varying(20)	endereco character(20)	cidade character(70)
1	1	ROCCO	R. RODRIGO S	RIO DE JANEI
2	2	GLOBO		RIO DE JANEI
3	3	ATICA		SÃO PAULO
4	4	USP/ICMC		

SELECT * FROM EDITORA

SELECT * FROM LIVRO

	titulo character(35)	valor numeric(7,2)	volume smallint	cod_editora smallint
1	MAKTUB	24.00		1
2	BRIDA	29.50		1
3	HISTÓRIAS P	35.00		2
4	DOM CASMURR	10.90		3
5	SQL	10.90		4

Cláusula WHERE

◆ Exercício:

- Especifique uma consulta que selecione todos os livros escritos por Paulo Coelho. Devem ser exibidos o nome do autor e o título dos livros

Exemplo Modelo Relacional

AUTOR = { COD AUTOR, NOME, NASCIMENTO }



ESCREVE = { COD AUTOR, COD LIVRO }



LIVRO = { COD LIVRO, TITULO, VALOR, VOLUME, COD_EDITORA }



EDITORIA = { COD EDITORA, RAZAO, ENDERECO, CIDADE }

Cláusula WHERE

◆ Resolução do exercício

-- 1) Qual o resultado da execução dos comandos abaixo?
-- 2) Quantas tuplas são retornadas? Por que?
/* 3) Pensando na álgebra relacional, o que significa executar esses comandos sem a cláusula WHERE? */

```
SELECT AU.NOME, LI.TITULO  
FROM AUTOR AU, LIVRO LI, ESCREVE ES;
```

```
SELECT AU.NOME, LI.TITULO  
FROM AUTOR AU, LIVRO LI, ESCREVE ES  
WHERE (AU.NOME = 'PAULO COELHO');
```

Operador BETWEEN...AND

- ◆ Determina se o valor de uma expressão se encontra dentro de um intervalo especificado de valores
- ◆ Se o valor da expressão estiver entre valor1 e valor2 (inclusive), o operador BETWEEN...AND retornará True; caso contrário, retornará False
- ◆ Exemplo:

```
/* Selecionar livros com valores entre R$ 30,00 e  
R$ 60,00 */
```

```
SELECT TITULO, VALOR  
FROM LIVRO  
WHERE VALOR BETWEEN 30 AND 60;
```

Operador IN

- ◆ Determina se o valor de uma expressão é igual a algum dos vários valores em uma lista especificada
- ◆ Se o valor da expressão for encontrado na lista de valores, o operador IN retornará True; caso contrário, retornará False
- ◆ Exemplo:

```
/* Selecionar as editoras localizadas nas cidades de  
São Paulo, São Carlos e Rio de Janeiro */  
SELECT *  
FROM EDITORA  
WHERE UPPER(CIDADE) IN ('SÃO PAULO', 'SÃO CARLOS',  
    'RIO DE JANEIRO');  
-- UPPER transforma os caracteres do atributo Cidade  
em maiúsculo
```

Operador IS NULL

- ◆ Determina se o valor de uma expressão é nulo
- ◆ Exemplo:

```
/* Selecionar as editoras com endereço nulo */  
SELECT RAZAO  
FROM EDITORA  
WHERE ENDERECO IS NULL;
```

Operador LIKE

◆ Operador de comparação de cadeias de caracteres

- % (porcentagem): substitui qualquer ***string***
 - ◆ exemplo: LIKE 'MAK%', para 'MAKROMBOOKS'
- _ (underscore): substitui qualquer ***caractere***
 - ◆ exemplo: LIKE '_OSE', para 'JOSE' e 'ROSE'

◆ Característica

- operadores sensíveis ao caso
 - ◆ letras maiúsculas são consideradas diferentes de letras minúsculas

Operador LIKE

◆ Exemplo:

/ Selecionar nome de autor que inicia com a letra 'J' */*

```
SELECT NOME
FROM AUTOR
WHERE NOME LIKE 'J%';
```

/ Selecionar nome de todos autores ignorando as três primeiras letras e que após o 'É' seja seguido por qualquer quantidade de caracteres */*

```
SELECT NOME
FROM AUTOR
WHERE UPPER(NOME) LIKE '____É%'; -- São 3 '_'
```

Cláusula ORDER BY

- ◆ Classifica as tuplas resultantes de uma consulta em ordem crescente ou decrescente de acordo com uma lista de atributos especificada
- ◆ As tuplas são classificadas pelo primeiro atributo listado após ORDER BY
- ◆ As tuplas que possuam valores iguais no primeiro atributo são então classificadas pelo valor do segundo atributo listado e assim por diante

Cláusula ORDER BY

◆ Exemplos

/ Selecione nome e nascimento da tabela autor em ordem dos autores mais novos */*

```
SELECT NOME, NASCIMENTO  
FROM AUTOR  
ORDER BY NASCIMENTO DESC;
```

/ Selecione titulo e valor da tabela livro em ordem dos valores mais baixos */*

```
SELECT TITULO, VALOR  
FROM LIVRO  
ORDER BY VALOR ASC;
```

Operações de Conjuntos

SQL	Álgebra Relacional
UNION	União (\cup)
INTERSECT	Intersecção (\cap)
EXCEPT	Diferença ($-$)

- Observações
 - as relações participantes das operações precisam ser *compatíveis*
 - operações oferecidas dependem do SGBD

Operações de Conjuntos

Exemplos

```
/* Selecione todos os autores (autores e autores experientes) */
```

```
SELECT NOME
```

```
FROM AUTOR
```

```
UNION
```

```
SELECT NOME
```

```
FROM AUTOREXPERIENTE;
```

Data Output	Explain	Messa
	nome character varying(20)	
1	PAULO COELHO	
2	MACHADO DE ASSIS	
3	JOSÉ MARIA	

AUTOR

∪

AUTOREXPERIENTE

Data Output	Explain	Messages
	nome character varying(30)	
1	MACHADO DE ASSIS	

Operações de Conjuntos

Exemplos

/ Selecione os autores que constem em ambas tabelas (autor e autor experiente) */*

SELECT NOME

FROM AUTOR

INTERSECT

SELECT NOME

FROM AUTOREXPERIENTE;

Data Output	Explain	Messa
	nome character varying(20)	
1	PAULO COELHO	
2	MACHADO DE ASSIS	
3	JOSÉ MARIA	

AUTOR

∩

AUTOREXPERIENTE

Data Output	Explain	Messages
	nome character varying(30)	
1	MACHADO DE ASSIS	

Operações de Conjuntos

◆ Exemplos

/ Selecione os autores que constem na primeira mas não na segunda tabela (autor e autor experiente) */*

SELECT NOME

FROM AUTOR

EXCEPT

SELECT NOME

FROM AUTOREXPERIENTE;

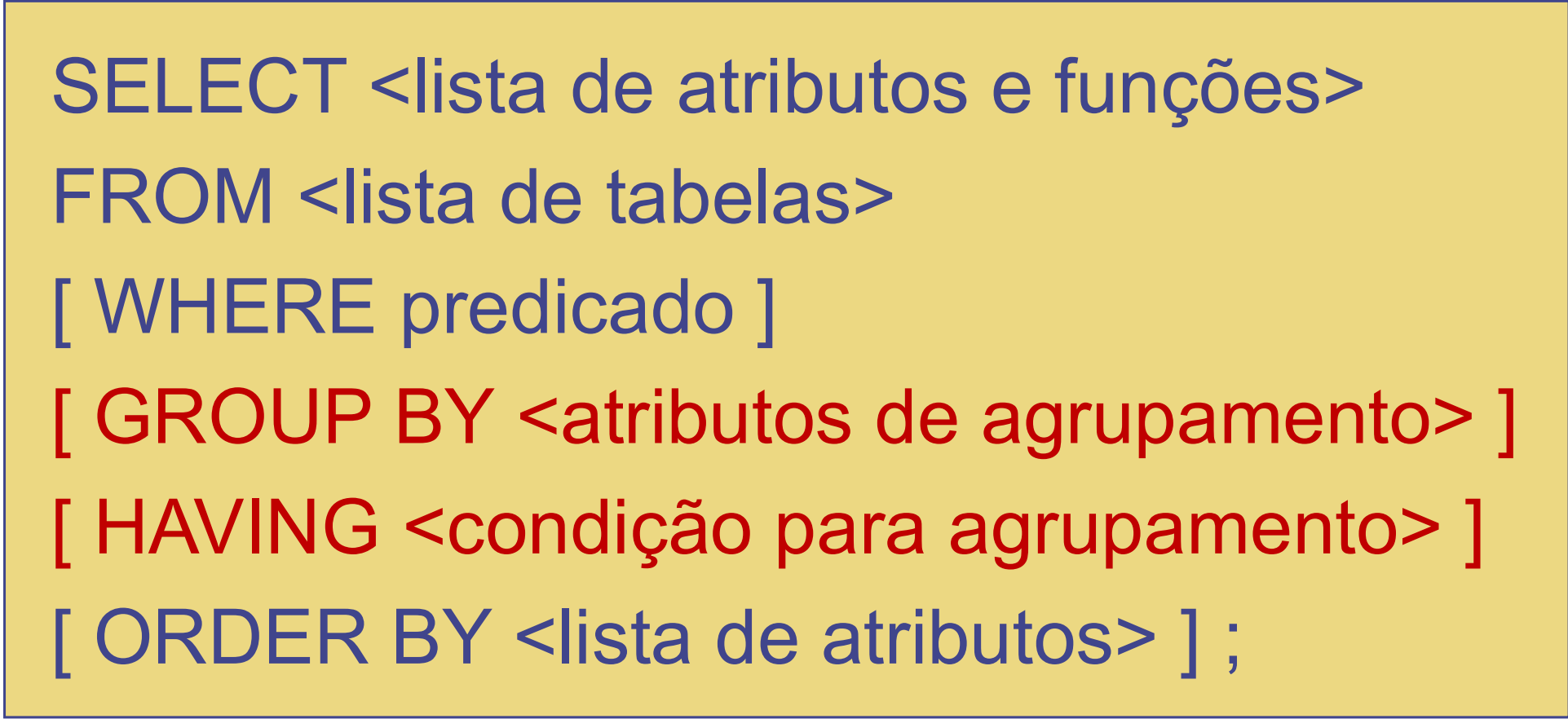
Data Output	Explain	Messa
	nome character varying(20)	
1	PAULO COELHO	
2	MACHADO DE ASSIS	
3	JOSÉ MARIA	

AUTOR

AUTOREXPERIENTE

Data Output	Explain	Messages
	nome character varying(30)	
1	MACHADO DE ASSIS	

SELECT



```
SELECT <lista de atributos e funções>  
FROM <lista de tabelas>  
[ WHERE predicado ]  
[ GROUP BY <atributos de agrupamento> ]  
[ HAVING <condição para agrupamento> ]  
[ ORDER BY <lista de atributos> ] ;
```

<https://www.postgresql.org/docs/current/sql-select.html>

Leitura complementar para casa

- ◆ Capítulo 8 do livro: Elmasri, Ramez; Navathe, Shamkant B. Sistemas de banco de dados.
- ◆ Manual do SGBD PostgreSQL
 - <http://www.postgresql.org/docs/manuals/>
 - Explorar as outras particularidades dos comandos apresentados

Sintaxe SQL PostgreSQL

- ◆ Consultar material disponível em
<http://www.postgresql.org/docs/manuals/>

Manual da ferramenta pgAdim

- ◆ Consultar material disponível em
<http://www.pgadmin.org/docs/1.4/>

Bibliografia

- ◆ Elmasri, Ramez; Navathe, Shamkant B. **Sistemas de banco de dados**. 4 ed. São Paulo: Addison Wesley, 2005, 724 p. Bibliografia: p. [690]-714.
- ◆ Material Didático produzido pelos professores Cristina Dutra de Aguiar Ciferri, Caetano Traina Júnior e Bruno Travençolo