Ciência da Computação
GBC043 Sistemas de Banco de Dados

SQL
Linguagem de Manipulação de Dados

Profa. Maria Camila Nardini Barioni
camila.barioni@ufu.br
Bloco B - sala 1B137

1° semestre de 2024

SQL DML – CONTINUAÇÃO...

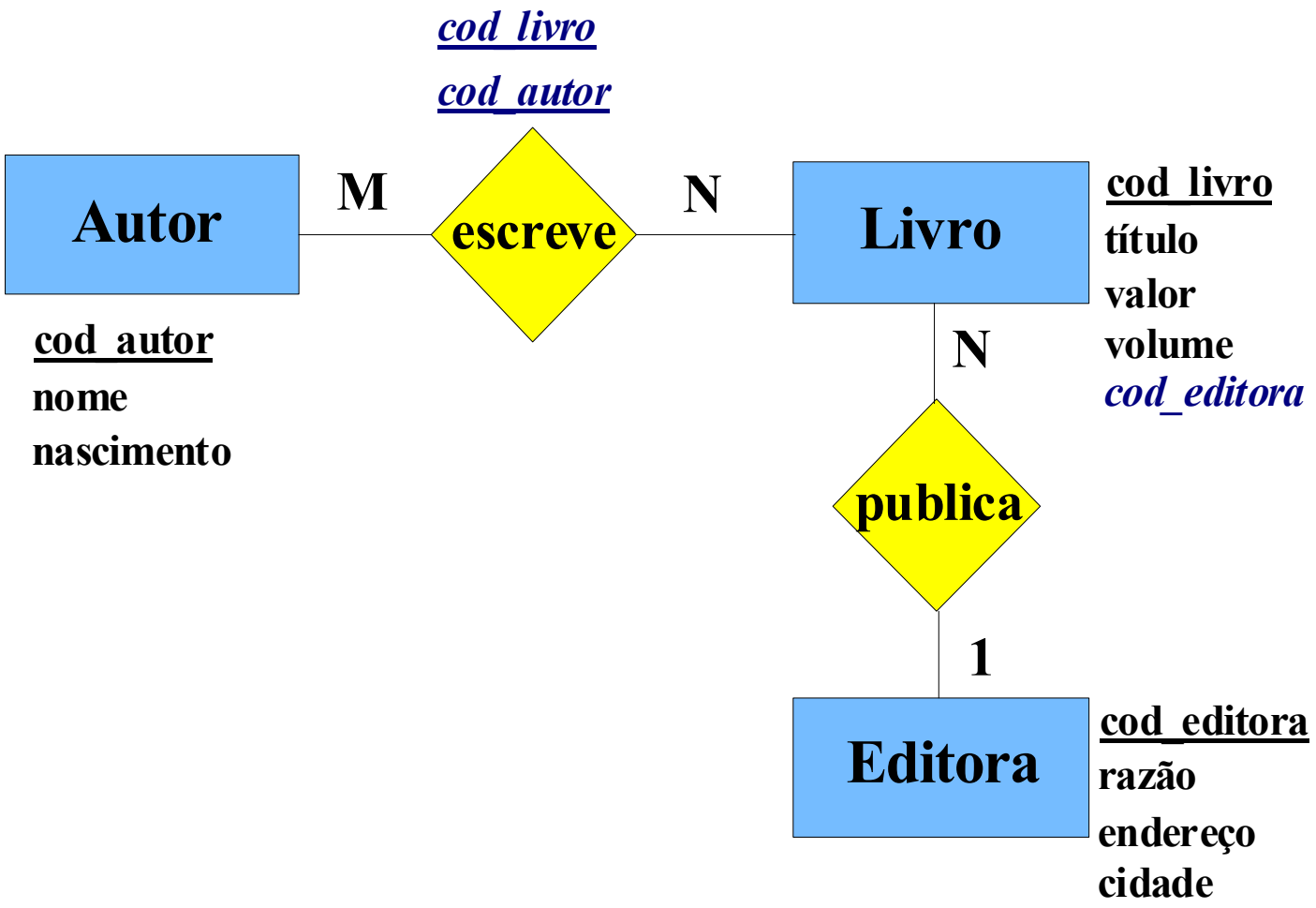
SQL DML

- SELECT ... FROM ... WHERE ...
  - lista atributos de uma ou mais tabelas de acordo com alguma condição
- INSERT INTO ...
  - insere dados em uma tabela
- DELETE FROM ... WHERE ...
  - remove dados de tabelas já existentes
- UPDATE ... SET ... WHERE ...
  - altera dados específicos de uma tabela

SQL DML

- SELECT ... FROM ... WHERE ...
  - lista atributos de uma ou mais tabelas de acordo com alguma condição – consultas avançadas
- INSERT INTO ...
  - insere dados em uma tabela
- DELETE FROM ... WHERE ...
  - remove dados de tabelas já existentes
- UPDATE ... SET ... WHERE ...
  - altera dados específicos de uma tabela

Exemplo ME-R


cod_livro
cod_autor
Autor
M
escreve
N
Livro
cod_livro
título
valor
volume
cod_editora
cod_autor
nome
nascimento
N
publica
1
Editora
cod_editora
razão
endereço
cidade

Exemplo Modelo Relacional

AUTOR = { COD_AUTOR, NOME, NASCIMENTO }

ESCREVE = { COD_AUTOR, COD_LIVRO }

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

EDITORA = { COD_EDITORA, RAZAO, ENDERECO, CIDADE }

SQL DML

- SELECT ... FROM ... WHERE ...
  - lista atributos de uma ou mais tabelas de acordo com alguma condição
  - Detalhamento sobre JUNÇÕES
- INSERT INTO ...
  - insere dados em uma tabela
- DELETE FROM ... WHERE ...
  - remove dados de tabelas já existentes
- UPDATE ... SET ... WHERE ...
  - altera dados específicos de uma tabela

Junção Natural

- SQL (primeiras versões)
  - não tem uma representação para a operação de junção
- Definida em termos de
  - um produto cartesiano
  - uma seleção
  - uma projeção

Junção

- Não é representada explicitamente
- Cláusulas SELECT e WHERE
  - especificam atributos com mesmo nome usando o nome da tabela e o nome do atributo (nome_tabela.nome_atributo)
- Cláusula FROM
  - possui mais do que uma tabela
- Cláusula WHERE
  - inclui as condições de junção

Junção
Relembrando os tipos

- Existem dois tipos de condição de junção
  - **Equi-junções** empregam o operador de igualdade
  - **Junções theta** empregam outros operadores como <, >, BETWEEN, etc.
- Existem três tipos de junção
  - **Junção interna:** retornam uma linha somente quando os atributos na junção contêm valores que satisfazem a condição de junção
  - **Junção externa:** retornam uma linha mesmo quando um dos atributos (ou ambas) na condição de junção contém um valor nulo
  - **Auto junção:** retornam o resultado da junção de uma tabela com ela mesma

Nota: exemplos de junção interna

Exemplos

```
-- Selecionar todos os títulos dos livros da editora ROCCO
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

xplain | Messages | History

		titulo character(35)	valor numeric(7,2)	volume smallint	cod_editora smallint
1	31	MAKTUB	24.00		1
2	55	BRIDA	29.50		1
3	63	HISTÓRIAS PA	35.00		2
4	14	DOM CASMURRO	10.90		3
5	13	SQL	10.90		4

Nota: exemplos de junção interna

Exemplos

```
/* Selecionar os títulos da tabela livro e os nomes da tabela autor */
SELECT AU.NOME, LI.TITULO
FROM AUTOR AU, LIVRO LI, ESCREVE ES
WHERE (AU.NOME = 'PAULO COELHO') AND
      (AU.COD_AUTOR = ES.COD_AUTOR_ESC) AND
(LI.COD_LIVRO = ES.COD_LIVRO_ESC);
```

Data Output | Explain | Messages | History

	cod_autor smallint	nome character varying(20)	nascimento date
1	1	PAULO COELHO	1947-01-01
2	2	MACHADO DE ASSIS	1839-06-21
3	3	JOSÉ MARIA	1960-08-25

AUTOR

Output pane

Data Output | Explain | Messages | History

	cod_livro smallint	titulo character(35)	valor numeric(7,2)	volume smallint	cod_editora smallint
1	31	MAKTUB	24.00		1
2	55	BRIDA	29.50		1
3	63	HISTÓRIAS PA	35.00		2
4	14	DOM CASMURRO	10.90		3
5	13	SQL	10.90		4

LIVRO

Data Output | Explain | Messages

	cod_livro_esc smallint	cod_autor_esc smallint
1	31	1
2	55	1
3	63	1
4	14	2
5	13	3

ESCREVE

Auto Junção

- É necessário empregar um apelido (*alias*) para identificar cada referencia para a tabela na consulta

Cláusula AS

- Renomeia
  - atributos
    - deve aparecer na cláusula SELECT
    - **útil para a visualização das respostas na tela**
  - relações
    - deve aparecer na cláusula FROM
    - **útil quando a mesma relação é utilizada mais do que uma vez na mesma consulta**
- Sintaxe
  - nome_antigo AS nome_novo

Auto Junção
Auxiliada pelo uso de apelidos

Nota: exemplo de auto junção

- Exemplo:
  - Para selecionar o nome de todos os funcionários de uma empresa juntamente com o nome de seus gerentes:
  - EMPREGADO = (id_funcionario, nome, data_nascimento, endereço, cpf, id_gerente)

```
SELECT t1.nome funcionario,
       t2.nome  gerente
FROM   empregado t1, empregado t2
WHERE t1.id_gerente = t2.id_funcionario;
```

Auto Junção
Auxiliada pelo uso de apelidos

Nota: exemplo de auto junção

- Exemplo auto junção

Empregado

id	nome	datanasc	end	cpf	gerente
1	José	05/01/1977	R. X, 111	111.111.111-11	NULL
2	Maria	07/07/1978	R. Y, 22	222.222.222-22	1
3	João	11/01/1984	R. Z, 78	333.333.333-33	2
4	Pedro	03/08/1954	R. W, 10	444.444.444-44	1

Auto Junção
Auxiliada pelo uso de apelidos

Nota: exemplo de auto junção

- Exemplo auto junção

Junção de Empregado t1 e Empregado t2

t1.id	t1.nome	t1.datanasc	...	t1.gerente	t2.id	t2.nome	t2.datanasc	...	t2.gerente
2	Maria	07/07/1978	...	1	1	José	05/01/1977	...	NULL
3	João	11/01/1984	...	2	2	Maria	07/07/1978	...	1
4	Pedro	03/08/1954	...	1	1	José	05/01/1977	...	NULL

Junção na cláusula `FROM`

- SQL-92
  - inclusão de operações adicionais na cláusula FROM
- Operações adicionais no PostgreSQL
  - … [INNER] JOIN … ON …
  - … LEFT [OUTER] JOIN … ON …
  - … RIGHT [OUTER] JOIN … ON …
  - … FULL [OUTER] JOIN … ON …
  - CROSS JOIN

Nota: exemplos de junção interna

Exemplos

```
-- Selecionar todos os títulos dos livros da editora ROCCO
SELECT LI.TITULO
FROM EDITORA ED INNER JOIN LIVRO LI ON
ED.COD_EDITORA = LI.COD_EDITORA
WHERE (ED.RAZAO = 'ROCCO');
```

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

xplain | Messages | History

		titulo character(35)	valor numeric(7,2)	volume smallint	cod_editora smallint
1	31	MAKTUB	24.00		1
2	55	BRIDA	29.50		1
3	63	HISTÓRIAS PA	35.00		2
4	14	DOM CASMURRO	10.90		3
5	13	SQL	10.90		4

Nota: exemplos de junção interna

Exemplos

```
/* Selecionar os títulos da tabela livro e os nomes da tabela autor */
```

?

Data Output | Explain | Messages | History

	cod_autor smallint	nome character varying(20)	nascimento date
1	1	PAULO COELHO	1947-01-01
2	2	MACHADO DE ASSIS	1839-06-21
3	3	JOSÉ MARIA	1960-08-25

AUTOR

LIVRO

Output pane

Data Output | Explain | Messages | History

	cod_livro smallint	titulo character(35)	valor numeric(7,2)	volume smallint	cod_editora smallint
1	31	MAKTUB	24.00		1
2	55	BRIDA	29.50		1
3	63	HISTÓRIAS PA	35.00		2
4	14	DOM CASMURRO	10.90		3
5	13	SQL	10.90		4

Data Output | Explain | Messages

	cod_livro_esc smallint	cod_autor_esc smallint
1	31	1
2	55	1
3	63	1
4	14	2
5	13	3

ESCREVE

Nota: exemplos de junção interna

Exemplos

```
/* Selecionar os títulos da tabela livro e os nomes da
tabela autor */
    SELECT AU.NOME, LI.TITULO
    FROM AUTOR AU INNER JOIN
                    (LIVRO LI INNER JOIN ESCREVE ES
                     ON LI.COD_LIVRO = ES.COD_LIVRO_ESC)
                  ON AU.COD_AUTOR = ES.COD_AUTOR_ESC
    WHERE (AU.NOME = 'PAULO COELHO');
```

Data Output | Explain | Messages | History

	cod_autor smallint	nome character varying(20)	nascimento date
1	1	PAULO COELHO	1947-01-01
2	2	MACHADO DE ASSIS	1839-06-21
3	3	JOSÉ MARIA	1960-08-25

AUTOR

LIVRO

Output pane

Data Output | Explain | Messages | History

	cod_livro smallint	titulo character(35)	valor numeric(7,2)	volume smallint	cod_editora smallint
1	31	MAKTUB	24.00		1
2	55	BRIDA	29.50		1
3	63	HISTÓRIAS P/	35.00		2
4	14	DOM CASMURRO	10.90		3
5	13	SQL	10.90		4

Data Output | Explain | Messages

	cod_livro_esc smallint	cod_autor_esc smallint
1	31	1
2	55	1
3	63	1
4	14	2
5	13	3

ESCREVE

Junção externa na cláusula `FROM`

- Variante da operação de JOIN que baseia-se em valores NULL. O resultado de um OUTER JOIN é igual a de um INNER JOIN mas com a inclusão das tuplas que não satisfazem a condição de JOIN.
- Três variantes:
  - LEFT OUTER JOIN
    - As tuplas da tabela à esquerda que não obedecem a condição do JOIN aparecem na resposta
  - RIGHT OUTER JOIN
    - As tuplas da tabela à direita que não obedecem a condição do JOIN aparecem na resposta
  - FULL OUTER JOIN
    - As tuplas das duas tabelas que não obedecem a condição do JOIN aparecem na resposta

Junção externa na cláusula FROM


```
/* Criando as tabelas*/
CREATE TABLE orientador (
   id INT PRIMARY KEY,
   nome VARCHAR(255)
);
CREATE TABLE aluno (
   matricula INT PRIMARY KEY,
   nome VARCHAR(255),
   orientador_id INT REFERENCES orientador(id)
);
```

Junção externa na cláusula FROM


```
-- Povoando as tabelas
INSERT INTO orientador VALUES (1,'Prof. Luiz'),
                              (2,'Profa. Maria');

INSERT INTO aluno VALUES (1,'Alessandra',NULL),
                         (2,'Walter',2),
                         (3,'Jéssica',NULL);
```

Junção externa na cláusula FROM


```
/* execute as seguintes junções nessas duas tabelas
e observe os resultados */
```

```
INNER JOIN
LEFT OUTER JOIN
RIGHT OUTER JOIN
FULL OUTER JOIN
CROSS JOIN
```

Data Output | Explain | Messages | History

	id integer	nome character varying(255)
1	1	Prof. Luiz
2	2	Profa. Maria

Data Output | Explain | Messages | History

	matricula integer	nome character varying(255)	orientador_id integer
1	1	Alessandra	
2	2	Walter	2
3	3	Jéssica	

```
SELECT *
FROM orientador INNER JOIN aluno
ON aluno.orientador_id = orientador.id;
```

Data Output | Explain | Messages | History

	id integer	nome character varying(255)	matricula integer	nome character varying(255)	orientador_id integer
1	2	Profa. Maria	2	Walter	2

Data Output | Explain | Messages | History

	id integer	nome character varying(255)
1	1	Prof. Luiz
2	2	Profa. Maria

Data Output | Explain | Messages | History

	matricula integer	nome character varying(255)	orientador_id integer
1	1	Alessandra	
2	2	Walter	2
3	3	Jéssica	

```
SELECT *
FROM orientador LEFT OUTER JOIN aluno
ON aluno.orientador_id = orientador.id;
```

Data Output | Explain | Messages | History

	id integer	nome character varying(255)	matricula integer	nome character varying(255)	orientador_id integer
1	1	Prof. Luiz			
2	2	Profa. Maria	2	Walter	2

Data Output | Explain | Messages | History

	id integer	nome character varying(255)
1	1	Prof. Luiz
2	2	Profa. Maria

Data Output | Explain | Messages | History

	matricula integer	nome character varying(255)	orientador_id integer
1	1	Alessandra	
2	2	Walter	2
3	3	Jéssica	

```
SELECT *
FROM orientador RIGHT OUTER JOIN aluno
ON aluno.orientador_id = orientador.id;
```

	id integer	nome character varying(255)	matricula integer	nome character varying(255)	orientador_id integer
1			1	Alessandra	
2	2	Profa. Maria	2	Walter	2
3			3	Jéssica	

Data Output | Explain | Messages | History

	id integer	nome character varying(255)
1	1	Prof. Luiz
2	2	Profa. Maria

Data Output | Explain | Messages | History

	matricula integer	nome character varying(255)	orientador_id integer
1	1	Alessandra	
2	2	Walter	2
3	3	Jéssica	

```
SELECT *
FROM orientador FULL OUTER JOIN aluno
ON aluno.orientador_id = orientador.id;
```

Data Output | Explain | Messages | History

	id integer	nome character varying(255)	matricula integer	nome character varying(255)	orientador_id integer
1	1	Prof. Luiz			
2	2	Profa. Maria	2	Walter	2
3			3	Jéssica	
4			1	Alessandra	

Data Output | Explain | Messages | History

	id integer	nome character varying(255)
1	1	Prof. Luiz
2	2	Profa. Maria

Data Output | Explain | Messages | History

	matricula integer	nome character varying(255)	orientador_id integer
1	1	Alessandra	
2	2	Walter	2
3	3	Jéssica	

```
SELECT *
FROM orientador CROSS JOIN aluno;
```

Data Output | Explain | Messages | History

	id integer	nome character varying(255)	matricula integer	nome character varying(255)	orientador_id integer
1	1	Prof. Luiz	1	Alessandra	
2	1	Prof. Luiz	2	Walter	2
3	1	Prof. Luiz	3	Jéssica	
4	2	Profa. Maria	1	Alessandra	
5	2	Profa. Maria	2	Walter	2
6	2	Profa. Maria	3	Jéssica	

SQL DML
CONTINUAÇÃO CONSULTAS AVANÇADAS

Suporte de SQL para OLAP

- OLAP SQL
  - Artigo de pesquisadores do Microsoft Research
    - Gray et al., "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab and Sub-Totals". Data Mining and Knowledge Discovery, 1(1):29-54, 1997. http://dx.doi.org/10.1023/A:1009726021843
      - implementação no MS SQL Server
  - Proposta ainda não foi padronizada
    - A seguir, proposta do PostgreSQL implementada a partir da versão 9.5.6

SELECT

```
SELECT <lista de atributos e funções>
FROM <lista de tabelas>
[ WHERE predicado ]
[ GROUP BY grouping_element [, ...] ]
[ HAVING <condição para agrupamento> ]
[ ORDER BY <lista de atributos> ]
[ LIMIT { count | ALL } ] ;
```

https://www.postgresql.org/docs/current/queries-table-expressions.html#QUERIES-GROUPING-SETS

Detalhando GROUP BY

- [GROUP BY grouping_element [, ...]]
  - Agrupa em uma única linha todas as linhas selecionadas que compartilham os mesmos valores para as expressões de agrupamento
  - grouping_element :
    - ( )
    - expression
    - (expression [, ...])
    - ROLLUP ({ expression | (expression [, ...]) } [, ...])
    - CUBE ({ expression | (expression [, ...]) } [, ...])

Detalhando GROUP BY

- ROLLUP e CUBE
  - Operações de agrupamento mais complexas
  - Os dados selecionados pelas cláusulas FROM e WHERE são agrupados separadamente por cada conjunto de agrupamentos especificado, agregados calculados para cada grupo, assim como para as cláusulas GROUP BY simples

Detalhando GROUP BY

- Exemplo: Pedidos

Cod_cliente	marca	tamanho	qtde
1	Adidas	M	1
1	Nike	M	2
3	Nike	G	3
4	Nike	M	3
3	Adidas	M	4
4	Adidas	G	6
5	Nike	G	11

Detalhando GROUP

- Exemplo: Pedidos

```
SELECT marca, SUM(qtde)
FROM pedidos
GROUP BY marca
```

marca	SUM(qtde)
Adidas	10
Nike	20

Cod_cliente	marca	tamanho	qtde
1	Adidas	M	1
1	Nike	M	2
3	Nike	G	4
4	Nike	M	3
3	Adidas	M	4
4	Adidas	G	5
5	Nike	G	11

Detalhando GROU

- Exemplo: Pedidos

```
SELECT marca, SUM(qtde)
FROM pedidos
GROUP BY ROLLUP(marca)
```

marca	SUM(qtde)
Adidas	10
Nike	20
	30

Cod_cliente	marca	tam...	qtde
1	Adidas	M	1
1	Nike	M	2
3	Nike	G	4
4	Nike	M	3
3	Adidas	M	4
4	Adidas	G	5
5	Nike	G	11

Detalhando G

- Exemplo: Pedi

Cod_cliente	marca		
1	Adidas		
1	Nike		
3	Nike		
4	Nike		
3	Adidas	M	4
4	Adidas	G	5
5	Nike	G	11

```
SELECT marca, tamanho, SUM(qtde)
FROM pedidos
GROUP BY ROLLUP(marca, tamanho)
```

marca	tamanho	SUM(qtde)
Adidas	M	5
Adidas	G	5
Adidas		10
Nike	M	5
Nike	G	15
Nike		20
		30

Detalhando G

- Exemplo: Pedi

Cod_cliente	marca		
1	Adidas		
1	Nike		
3	Nike		
4	Nike		
3	Adidas		
4	Adidas		
5	Nike	G	11

```
SELECT marca, tamanho, SUM(qtde)
FROM pedidos
GROUP BY CUBE(marca, tamanho)
```

marca	tamanho	SUM(qtde)
Adidas	M	5
Adidas	G	5
Adidas		10
Nike	M	5
Nike	G	15
Nike		20
M	10	
G	20	
		30

Detalhando LIMIT

- Permite recuperar apenas uma parte das linhas geradas pelo resto da consulta

Detalhando LIMIT

- Exemplo

Cod_cliente	marca	tamanho	qtde
1	Adidas	M	1
1	Nike	M	2
3	Nike	G	3
4	Nike	M	3
3	Adidas	M	4
4	Adidas	G	6
5	Nike	G	11

Detalhando L

- Exemplo

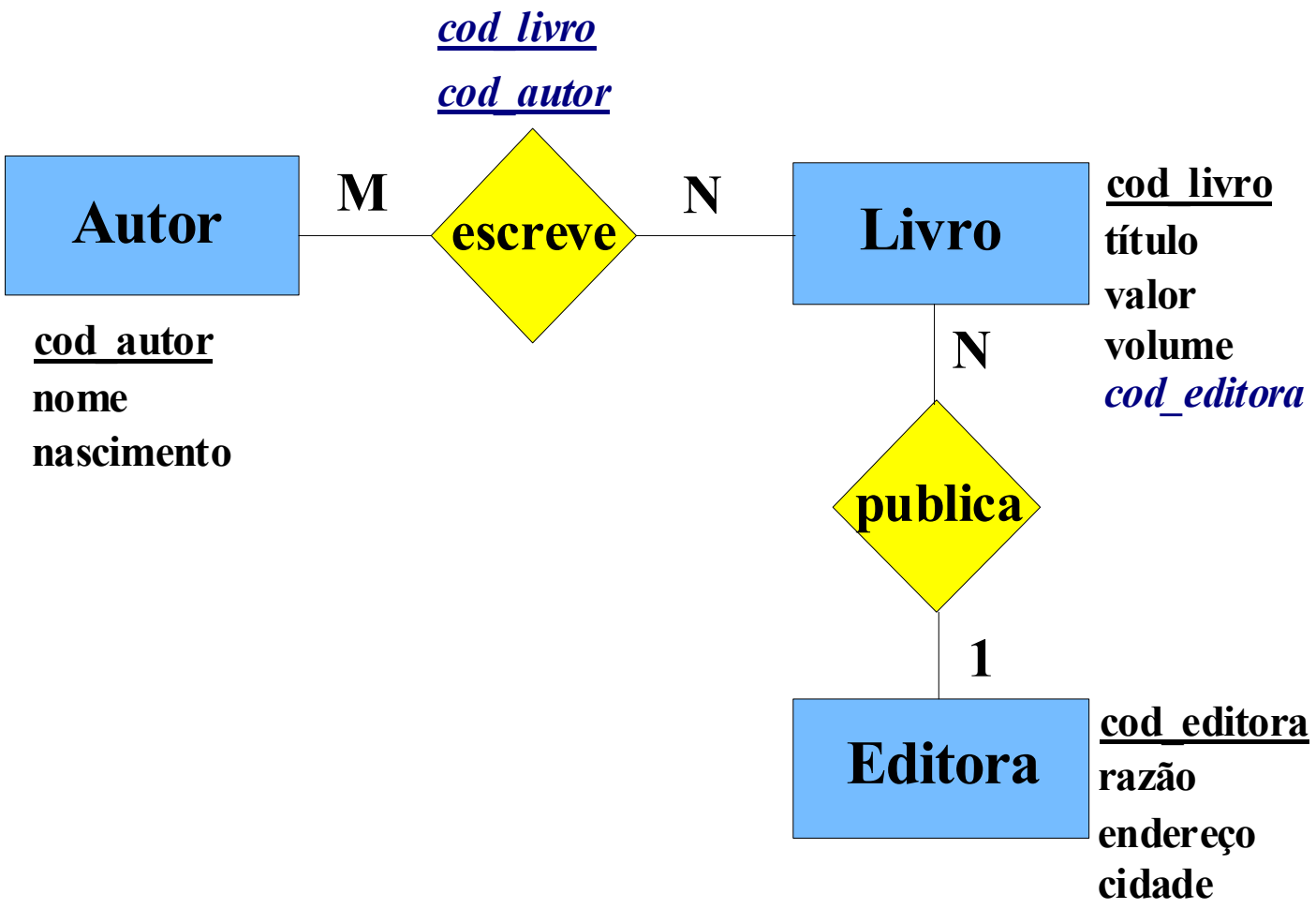
```
/* Selecionar os três clientes que
mais compraram */
SELECT cod_cliente, SUM(qtde)
FROM pedidos
GROUP BY cod_cliente
ORDER BY SUM(qtde) DESC
LIMIT 3

cod_cliente   SUM(qtde)
-----------   ---------
5             11
4             9
3             7
```

Cod_cliente	marca		
1	Adidas		
1	Nike		
3	Nike		
4	Nike	M	3
3	Adidas	M	4
4	Adidas	G	6
5	Nike	G	11

VISÕES

Exemplo ME-R


cod_livro
cod_autor
Autor
M
escreve
N
Livro
cod_livro
título
valor
volume
cod_editora
cod_autor
nome
nascimento
N
publica
1
Editora
cod_editora
razão
endereço
cidade

Exemplo Modelo Relacional

AUTOR = { COD_AUTOR, NOME, NASCIMENTO }

ESCREVE = { COD_AUTOR, COD_LIVRO }

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

EDITORA = { COD_EDITORA, RAZAO, ENDERECO, CIDADE }

Visão

- Em SQL
  - tabela simples que é derivada de outras tabelas
  - as tabelas base podem ser tabelas ou outras visões
  - não existe necessariamente em sua forma física → tabela virtual
  - a definição de uma visão é armazenada no dicionário de dados que guarda a consulta que gerou a tabela

Visão

- Utilidade
  - forma de se especificar uma tabela que precisa ser acessada freqüentemente, embora essa tabela não exista fisicamente
  - facilita a escrita de consultas complexas
  - Pode restringir a visualização do conteúdo de uma tabela por meio da limitação das colunas que são exibidas e das linhas que são filtradas

CREATE VIEW

CREATE VIEW nome_visão (lista_de_atributos)
AS <expressão_da_consulta>

- Especifica uma visão
- Características
  - lista_de_atributos: opcional (função de renomeação)
  - expressão_da_consulta: consulta para especificar o conteúdo da visão (SELECT ... FROM ... WHERE ...)

- https://www.tutorialspoint.com/postgresql/postgresql_views.htm

Exemplo

- Criação da visão

```
CREATE VIEW livro_autor
    AS SELECT nome, titulo
        FROM autor, escreve, livro
      WHERE autor.cod_autor = cod_autor_esc AND
            cod_livro_esc = cod_livro
```

- Perguntas
  - quantos atributos a visão possui?
  - qual a ordem desses atributos?

Exemplo

Criação da visão

```
CREATE VIEW livro_editora (titulo, editora)
  AS SELECT titulo, razao
       FROM livro, editora
    WHERE livro.cod_editora = editora.cod_editora
```

Perguntas

- quantos atributos a visão possui?
- qual a ordem desses atributos?

DROP VIEW

DROP VIEW nome_visão

- Remove a definição de uma visão
- As tabelas base não são afetadas por esse comando
- Exemplos:
  - DROP VIEW livro_autor;
  - DROP VIEW livro_editora;

Visão – Mais detalhes

- É possível executar operações de seleção, junção, inserção, atualização e remoção sobre uma visão, como se ela fosse uma tabela regular da base de dados, com apenas poucas restrições:
  - Existem visões atualizáveis e visões "read-only"
  - No PostgreSQL elas podem ser dos dois tipos
  - http://www.postgresqltutorial.com/postgresql-views/

Visão – Mais detalhes

- Visões atualizáveis no PostgreSQL
  - A consulta que define a visão possui apenas uma tabela na cláusula FROM
  - A consulta que define a visão **não** possui as cláusulas: GROUP BY, HAVING, LIMIT, OFFSET, DISTINCT, WITH, UNION, INTERSECT, and EXCEPT
  - A consulta que define a visão não possui nenhuma função de agregação, como: SUM, COUNT, AVG, MIN, MAX
  - http://www.postgresqltutorial.com/postgresql-updatable-views/

Visão – Mais detalhes

- Visões materializadas no PostgreSQL
  - Permite que os dados da visão sejam armazenados fisicamente
  - São úteis para aplicações que necessitam de rápido acesso aos dados
    - Data Warehouses
    - Business Intelligence

Visão – Mais detalhes

- Visões materializadas no PostgreSQL

```
CREATE MATERIALIZED VIEW nome_visão
AS consulta
WITH [NO] DATA;
```

`WITH DATA` → para carregar os dados no momento da criação da visão

`WITH NO DATA` → não carrega os dados no momento da criação

Visão – Mais detalhes

- Visões materializadas no PostgreSQL
  - Para carregar os dados em uma visão materializada

```
REFRESH MATERIALIZED VIEW nome_visão;
```

- http://www.postgresqltutorial.com/postgresql-materialized-views/

Usando o comando `SELECT` em uma visão

- O comando `SELECT` funciona da maneira usual com uma visão

Usando o comando SELECT em uma visão - Exemplos

```
CREATE VIEW autor_editora (autor, titulo, editora)
    AS SELECT nome, titulo, razao
        FROM autor, escreve, livro, editora
WHERE autor.cod_autor = cod_autor_esc AND
      cod_livro_esc = cod_livro AND
      livro.cod_editora = editora.cod_editora


SELECT * FROM autor_editora

SELECT editora, COUNT(*)
FROM autor_editora
GROUP BY (editora)
```

Visão – Comandos adicionais

- Para visualizar no dicionário de dados
  - Selecionar a tabela `pg_views`
- Para substituir uma visão
  - Eliminar com o comando `DROP` e depois empregar o comando `CREATE VIEW`
  - Usar a opção `OR REPLACE` do comando `CREATE VIEW`
    - Essa opção preserva as autorizações de segurança existentes

Bibliografia

- Elmasri, Ramez; Navathe, Shamkant B. **Sistemas de banco de dados.** 4 ed. São Paulo: Addison Wesley, 2005, 724 p. Bibliografia: p. [690]-714.

Leitura complementar para casa

- Capítulos 4 e 5 do livro: Elmasri, Ramez; Navathe, Shamkant B. Sistemas de banco de dados. 6ª edição
- Manual do SGBD PostgreSQL
  - http://www.postgresql.org/docs/manuals/
  - Explorar as outras particularidades dos comandos apresentados

Dica de estudo complementar