



Pontifícia Universidade Católica do Rio Grande do Sul FACULDADE DE INFORMÁTICA PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Curso Oracle 10g

Prof. MSc. Luciano Blomberg lblomberg@uol.com.br

Agenda

Introdução

- Oracle 10g overview
- Tipos de dados no Oracle
- SQL Developer

- DDL (Data Definition Language)
- DML (Data Manipulation Language)
 - Sequence
 - Visão
 - Trigger
- DQL (Data Query Language)

Agenda

PARTE I

- Introdução
- DDL, DML
- Sequence e Trigger
- Exercícios Lista I

PARTE 2

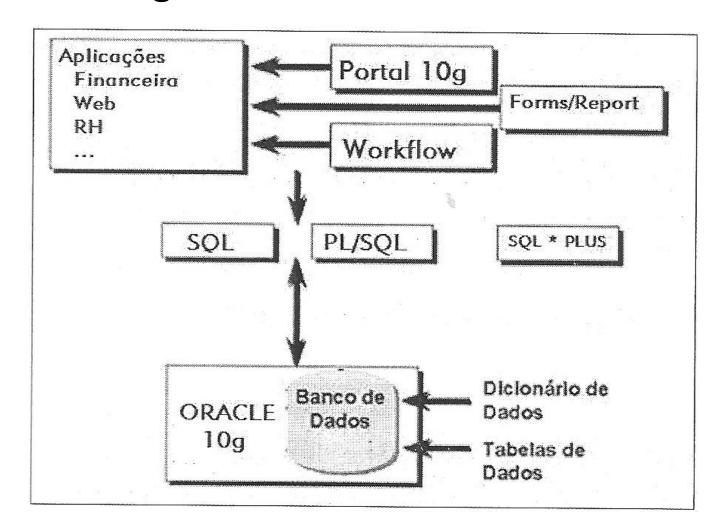
- DQL
- Junção de tabelas
- Funções de agrupamento
- Outras Funções
- Visões
- Exercícios Lista 2

Agenda

• PARTE 3

- Operações de conjunto
- Subconsultas
 - Correlacionadas
 - Não-correlacionadas
- Exercícios Lista 3

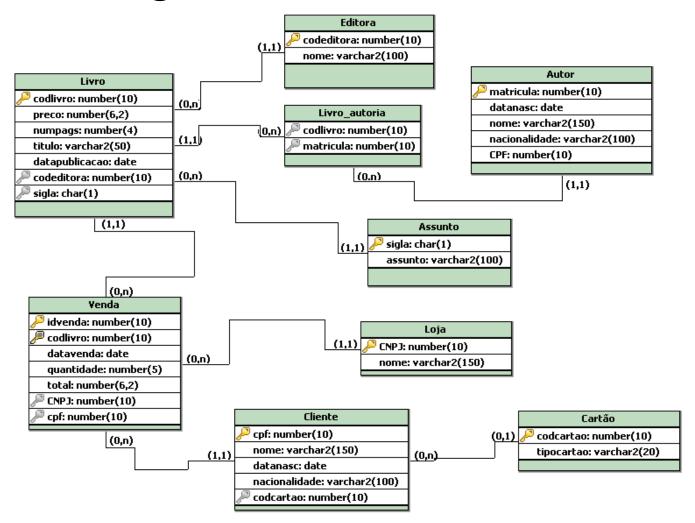
Oracle 10g Overview



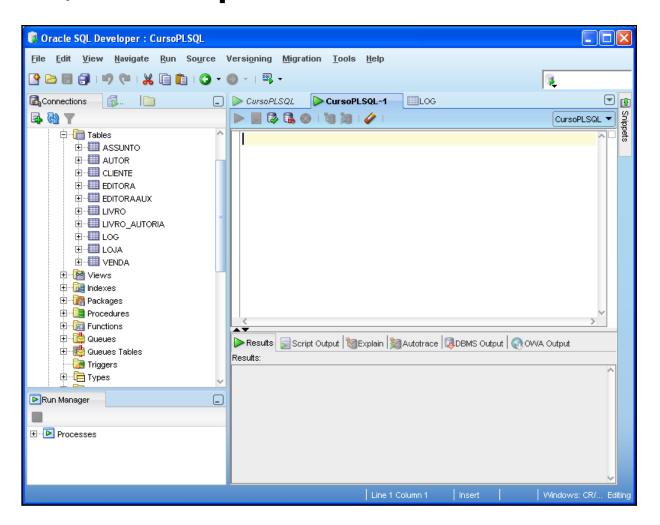
Tipos de Comandos SQL

SELECT	LINGUAGEM DE RECUPERAÇÃO DE DADOS (DQL)				
INSERT					
UPDATE	LINGUAGEM MANIPULAÇÃO DE DADOS (DML)				
DELETE					
CREATE					
ALTER	LINGUAGEM DE DEFINIÇÃO DE DADOS (DDL)				
DROP					
RENAME					
GRANT	LINCHACEM DE CONTROLE DE DADOS (DCI)				
REVOKE	LINGUAGEM DE CONTROLE DE DADOS (DCL)				
COMMIT					
ROLLBACK	CONTROLE DE TRANSAÇÕES				
SAVEPOINT	<u> </u>				

Modelo lógico – Controle de Vendas de Livros



SQL Developer



• Tipos de dados Oracle

TIPO DE DADO	DESCRIÇÃO			
VARCHAR2(size)	Dados caractere de tamanho variável. Tamanho mínimo é 1, enquanto o máximo é 4000.			
CHAR(size)	Dados de caractere de tamanho fixo. Tamanho mínimo, enquanto o máximo é 2000.			
NUMBER(p,s)	Número possuindo precisão p (número total de dígitos decimais) e escala s (número de dígitos à direita do ponto decimal. A precisão deve estar entre 1 e 38.			
BLOB	Dados binários de até 4 gigabytes * tamanho do bloco de dados.			
LONG	Dados caractere de tamanho variável de até 2 gigabytes.			
DATE	Valores de data e hora entre 1 de janeiro de 4712 A.C.à Dezembro de 9999 D.C.			

- DDL (Data Definition Language)
 - Comando CREATE TABLE

CREATE TABLE table(

column datatype [DEFAULT expr]
[column constraint],
[table constraint]);

table: é o nome da tabela.

column datatype: é o nome da coluna com o seu tipo de dado e tamanho.

DEFAULT expr: especifica um valor default se um valor for omitido no INSERT.

column constraint: restrição de integridade declarada como parte da definição da coluna.

constraint table: restrição de integridade declarada como parte da definição da tabela.

- DDL (Data Definition Language)
 - Comando CREATE TABLE (Sem Constraints)
 - Exemplos:

```
codeditora number(10),
nome varchar2(100));

CREATE TABLE TLIVRO I (
titulo varchar2(50),
codlivro number(10),
codeditora number(10));
```

- DDL (Data Definition Language)
 - Comando CREATE TABLE (Constraints)
 - Previnir entrada de dados inválidos em tabelas;
 - Garantir regras à nível de tabela sempre que uma linha for inserida, atualizada ou apagada desta tabela.
 - Previnir a exclusão de uma tabela se existitem dependências em outras tabelas;

CONSTRAINT	DESCRIÇÃO			
NOT NULL	Especifica que a coluna não pode conter valores			
	nulos.			
UNIQUE KEY	Especifica uma coluna ou combinação de			
ONIQUEREI	colunas, cujos valores devem ser únicos para			
	todas as linhas da tabela.			
PRIMARY KEY	Identifica de forma única cada linha da tabela.			
FOREIGN KEY	Estabelece e garante um relacionamento de			
	chave estrangeira entre a coluna e uma coluna			
	de tabela referenciada.			
CHECK	Especifica uma condição que deve ser			
	verdadeira.			

- DDL (Data Definition Language)
 - Comando CREATE TABLE (Constraints)
 - Exemplos:

```
CREATE TABLE TEDITORA2 (
          codeditora number(10),
          nome varchar2(100) NOT NULL,
          constraint Ted_codeditora_bk PRIMARY KEY(codeditora));
CREATE TABLE TLIVRO2 (
          titulo varchar2(50) NOT NULL,
          codlivro number(10),
          codeditora number(10),
           constraint Tliv_codlivro_pk PRIMARY KEY(codlivro),
           constraint Tliv_Ted_codeditoa_fk FOREIGN KEY(codeditora)
           REFERENCES TEDITORA2);
```

- DDL (Data Definition Language)
 - Comando CREATE TABLE (Constraints)
 - Exemplos:

```
CREATE TABLE TAUTOR2 (

CPF number(10),

datanasc date,

nome varchar2(150) NOT NULL,

nacionalidade varchar2(100),

matricula number(10),

CONSTRAINT Tau_matricula_pk PRIMARY KEY(matricula),

CONSTRAINT Tau_cpf_uk UNIQUE(cpf),

CONSTRAINT Tau_naci_ck CHECK(nacionalidade='Brasileira'));
```

- DDL (Data Definition Language)
 - Comando CREATE TABLE
 - Exemplos:

```
CREATE TABLE TEDITORA3 (
          codeditora number(10),
          nome varchar2(100) NOT NULL,
          PRIMARY KEY(codeditora));
CREATE TABLE TLIVRO3 (
          titulo varchar2(50) NOT NULL,
          codlivro number(10),
          codeditora number(10),
          PRIMARY KEY(codlivro),
          FOREIGN KEY(codeditora) REFERENCES TEDITORA3);
```

- DDL (Data Definition Language)
 - Comando CREATE TABLE
 - Criando uma tabela utilizando uma subconsulta:

```
CREATE TABLE table(
column [NOT NULL])
AS
Subquery.
```

table: é o nome da tabela.

column: é o nome da coluna.

subquery: é o comando SELECT que define o conjunto de linhas a ser inserido na nova

tabela.

- DDL (Data Definition Language)
 - Comando CREATE TABLE

Script auxiliar:

INSERT INTO TEDITORA2(codeditora, nome) VALUES (1,'Tabajara');
INSERT INTO TEDITORA2(codeditora, nome) VALUES (2,'Qualquer uma');
INSERT INTO TEDITORA2(codeditora, nome) VALUES (3,'Exemplo');

- DDL (Data Definition Language)
 - Comando CREATE TABLE
 - Criando uma tabela utilizando uma subconsulta:
 - Exemplo:

```
CREATE TABLE TNOVAEDITORA(

codeditora primary key,

nome NOT NULL)

AS
```

SELECT * FROM TEDITORA2:

- Datatype não deve ser informado junto à column.
- Não é permitido estabelecer integridade referencial.
- Pode-se alterá-las posteriormente com ALTER TABLE.

- DDL (Data Definition Language)
 - Comando ALTER TABLE

```
ALTERTABLE table
         ADD (column datatype [DEFAULT expr]
         [column constraint],
         [table constraint]);
ALTERTABLE table
         MODIFY (column datatype [DEFAULT expr]
         [column constraint],
         [table constraint]);
ALTER TABLE table
         DROP COLUMN column;
```

- DDL (Data Definition Language)
 - Comando ALTER TABLE
 - Exemplos:

ALTER TABLE TEDITORA I

ADD (endereco char(150));

ALTER TABLE TEDITORA I

MODIFY (endereco varchar2(150),
PRIMARY KEY(codeditora));

ALTER TABLE TEDITORA I

DROP COLUMN endereco;

- DDL (Data Definition Language)
 - Comando ALTERTABLE (Com Constraints)
 - Exemplos:

ALTER TABLE TEDITORA I

ADD CONSTRAINT Ted_codeditora_ck CHECK(codeditora > 0);

ALTER TABLE TEDITORA I

DROP CONSTRAINT Ted_codeditora_ck;

ALTER TABLE TEDITORA I DROP PRIMARY KEY;

- DDL (Data Definition Language)
 - Comando RENAME

RENAME old_name TO new_name;

old_name: é o nome antigo da tabela, visão, sequence ou sinônimo.

new_name: é o nome novo da tabela, visão, sequence ou sinônimo.

• Exemplo:

RENAME TNOVAEDITORA TO TEDITORA;

- DDL (Data Definition Language)
 - Comando DROPTABLE

DROP TABLE table;

- Todos os dados são apagados da tabela;
- Visões permanecem, porém inválidas;
- Somente o criador da tabela ou um usuário com o privilégio DROPANY TABLE pode remover uma tabela.
- Assim como os demais comandos DDL, o comando DROP emite um COMMIT implícito, tornando a transação permanente.

- DDL (Data Definition Language)
 - Comando DROPTABLE
 - Exemplos:

```
DROPTABLE TEDITORAI CASCADE CONSTRAINT;
DROPTABLE TLIVROI CASCADE CONSTRAINT;
DROPTABLE TEDITORA2 CASCADE CONSTRAINT;
DROPTABLE TLIVRO2 CASCADE CONSTRAINT;
DROPTABLE TAUTOR2 CASCADE CONSTRAINT;
DROPTABLE TEDITORA3 CASCADE CONSTRAINT;
DROPTABLE TLIVRO3 CASCADE CONSTRAINT;
DROPTABLE TEDITORA CASCADE CONSTRAINT;
```

- DDL (Data Definition Language)
 - Abra e execute(F5) no Developer o arquivo abaixo:
 - SCRIPT_BANCO.SQL

- DML (Data Manipulation Language)
 - Comando INSERT

INSERT INTO table[column1, column2m ,....columnN]
 VALUES(value1, value2,... valueN);

table: é o nome da tabela.

column : é o nome da coluna da tabela que receberá os valores.

value: é o valor correspondente para a coluna.

- DML (Data Manipulation Language)
 - Comando INSERT
 - Exemplos:

INSERT INTO EDITORA(codeditora, nome)
 VALUES(5,'NOVA EDITORA');

INSERT INTO EDITORA(codeditora, nome)

(SELECT codeditora, nome FROM EDITORAAUX);

- DML (Data Manipulation Language)
 - Comando UPDATE

```
UPDATE table
```

SET column=value [, columnN=valueN] [WHERE condition];

table: é o nome da tabela.

column: é o nome da coluna a alterar.

value: é o novo valor correspondente.

condition: identifica as linhas a serem atualizadas.

- DML (Data Manipulation Language)
 - Comando UPDATE
 - Exemplos:

```
UPDATE LIVRO SET preco=preco * 1.10
WHERE sigla='B';
```

UPDATE VENDA SET total=total * 1.10 WHERE codlivro IN(1,4);

UPDATE VENDA SET total=total * 1.10

WHERE codlivro IN(

SELECT L.codlivro

FROM ASSUNTO A, LIVRO L

WHERE L.sigla=A.sigla AND A.sigla='B');

- DML (Data Manipulation Language)
 - Comando DELETE

DELETE [FROM] table [WHERE condition];

table: é o nome da tabela.

condition: identifica as linhas a serem apagadas.

- DML (Data Manipulation Language)
 - Comando DELETE
 - Exemplos:

DELETE FROM EDITORAAUX

WHERE nome='GLOBO';

DELETE FROM EDITORA

WHERE nome='MIRANDELA EDITORA';

ERRO (RESTR. INTEG.)

- DML (Data Manipulation Language)
 - Comando MERGE

MERGE INTO tabela_destino [AS apelido_destino]

USING tabela_origem [AS apelido_origem]

ON (expr)

WHEN MATCHED THEN

operação_verdadeiro

WHEN NOT MATCHED THEN

operação_falso;

tabela_destino: tabela que receberá os dados.

tabela_origem: tabela que serve de origem para os dados.

expr: deverá conter a especificação entre as colunas da tabela

operação_verdadeiro: atualização a ser realizada nas linhas onde expr for verdadeira.

operação_falso: inclusão a ser realizada quando exp for falsa.

- DML (Data Manipulation Language)
 - Comando MERGE
 - Exemplo:

MERGE INTO CLIENTE CLI

USING AUTOR AU

ON(AU.cpf=CLI.cpf)

WHEN MATCHED THEN

UPDATE SET CLInome=AU nome

WHEN NOT MATCHED THEN

INSERT (cpf, datanasc, nome, nacionalidade)

VALUES(AU.cpf, AU.datanasc, AU.nome, AU.nacionalidade);

- DML (Data Manipulation Language)
 - Comando MERGE
 - Exemplo:

	2 CPF	2 DATANASC	2 NOME	NACIONALIDADE		2 CPF	B DATANA	SC 2 NOME	NACIONALIDADE 2	CODCARTAO
	1 920003	11/12/1981	Ana Escobar	Colombiana	1	920003	11/12/1981	Ana Silveira Escobar	Colombiana	1
	2 890006	13/08/1979	Carla Silva	Brasileira	2	890006	13/08/1979	Carla Maldonado Silva	Brasileira	1
	3 890106	15/08/1979	João Inácio Magalhães	Brasileira	3	890106	15/08/1979	João Inácio Magalhães	Brasileira	2
	4 890206	29/10/1977	César Gonçalves	Brasileira	4	890206	29/10/1977	César Gonçalves	Brasileira	2
	5 890306	30/03/1976	Flávio Aguiar Fonseca	Brasileira	5	890306	30/03/1976	Flávio Aguiar Fonseca	Brasileira	2
	6 890406	02/07/1966	Henrique Meirelles	Brasileira	6	890406	02/07/1966	Henrique Meirelles	Brasileira	2
	7 890506	01/09/1969	Cleide Castanheira	Brasileira	7	890506	01/09/1969	Cleide Castanheira	Brasileira	2
	8 890606	15/10/1982	Júlio Siqueira	Brasileira	8	890606	15/10/1982	Júlio Siqueira	Brasileira	2
	9 890706	02/09/1951	Elza Saldanha Prado	Brasileira	9	890706	02/09/1951	Elza Saldanha Prado	Brasileira	2
1	10 890806	08/08/1971	Marco Antônio Rossi	Brasileira	10	890806	08/08/1971	Marco Antônio Rossi	Brasileira	2
				<u> </u>	11	780002	23/03/1959	Carlos Padilha	Brasileira	(null)
					12	890001	08/08/1978	Luciano Blomberg	Brasileira	(null)
					13	780007	24/05/1959	Carlos Tenório	Equatoriana	(null)
					14	820005	25/04/1965	Marcos Andrade	Brasileira	(null)
					15	770008	29/11/1960	Maria Fonseca	Brasileira	(null)
					16	810004	01/11/1963	César Oliveira	Brasileira	(null)

Criando Sequence

- Uma sequence é um objeto do banco de dados criado por um usuário podendo ser compartilhado por múltiplos usuários.
- Um uso típico para sequence é criar um valor para uma chave primária, que deve ser único para cada linha.
- Os números das sequências são armazenados e gerados independentemente das tabelas. Portanto, a mesma sequence pode ser utilizada em tabelas diferentes.

```
CREATE SEQUENCE sequence

[INCREMENT BY n]

[START WITH n]

[{MAXVALUE n | NOMAXVALUE}]

[{MINVALUE n | NOMINVALUE}]

[{CYCLE | NOCYCLE}]

[{CACHE n | NOCACHE}]

[{ORDER | NOORDER}];
```

Criando Sequence

sequence: é o nome do gerador de sequência.

INCREMENT BY n: especifica o intervalo entre os geradores de sequência.

START WITH n: especifica o primeiro número da sequência.

MAXVALUE n: especifica o valor máximo que a sequência pode gerar.

NOMAXVALUE: compreende o valor -1 a 10²⁷(Padrão)

MINVALUE n: especifica o valor mínimo da sequência.

NOMINVALUE: compreende o valor $I = 10^{26}$ (Padrão)

CYCLE | NOCYCLE: continua gerando valores após atingir valor máximo / mínimo.

CACHE | NOCACHE: especifica quantos valores o servidor oracle mantém em memória.

ORDER | NOORDER: valores gerados na ordem de requisição.

- Criando Sequence
 - Exemplo

CREATE SEQUENCE IDEDITORA

INCREMENT BY I

START WITH 10

MAXVALUE 5000

NOCACHE

NOCYCLE;

CREATE SEQUENCE IDLIVRO

INCREMENT BY I

START WITH 6

MAXVALUE 5000

NOCACHE

NOCYCLE;

Criando Sequence

Exemplo

INSERT INTO EDITORA(codeditora, nome)

VALUES(IDEDITORA.NEXTVAL, 'EDITORA SEQUENCE');

INSERT INTO LIVRO(codlivro, titulo, sigla, preco, numpags, datapublicacao, codeditora)

VALUES(IDLIVRO.NEXTVAL,'BANCO DE DADOS

TEMPORAIS','B',50.00, I 02,TO_DATE(' I 0/02/200 I ',
'DD/MM/YYYY'),IDEDITORA.CURRVAL);

INSERT INTO LIVRO_AUTORIA(codlivro, matricula)

VALUES(IDLIVRO.CURRVAL, I I I);

ALTER SEQUENCE IDEDITORA

MAXVALUE 300;

DROP SEQUENCE IDEDITORA;

Criando Visões

CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW view

[alias I, alias 2,...alias N]

AS subquery

[WITH CHECK OPTION [CONSTRAINT constraint]

[WITH READ ONLY];

OR REPLACE: recria a visão caso já exista.

FORCE: cria a visão mesmo que as tabelas básicas não existam.

NOFORCE: cria a visão somente se as tabelas básicas existem (default).

view restrição de integridade declarada como parte da definição da coluna.

alias: restrição de integridade declarada como parte da definição da tabela.

WITH CHECK OPTION: especifica que somente as linhas acessíveis para a visão podem ser

inseridas ou atualizadas.

WITH READ ONLY: nenhuma operação DML possa ser executada nesta visão.

Criando Visões

Exemplo

CREATE VIEW VISAO_VENDAS

AS

SELECT CL.nome AS NOME_CLIENTE,

li.titulo AS TITULO_LIVRO,

ed.nome AS NOME_EDITORA,

VE.QUANTIDADE

FROM LIVRO LI INNER JOIN EDITORA ED ON LI.codeditora=ED.codeditora

INNER JOIN VENDA VE ON LI.codlivro=VE.codlivro

INNER JOIN CLIENTE CL ON CL.cpf=VE.cpf;

CREATE OR REPLACE....

DROP VIEW VISAO_VENDAS;

Criando Triggers

```
CREATE [OR REPLACE] TRIGGER nome_trigger
{BEFORE | AFTER | INSTEAD OF}
evento_disparador | OR evento_disparador | ON tabela_evento
[nível_gatilho]
[condição_execução]
Bloco_de_comandos_SQL;
```

Exemplo

```
CREATE OR REPLACE TRIGGER gera_log_alteracoes

AFTER UPDATE of preco
ON LIVRO

BEGIN

INSERT INTO LOG(usuario, datahora)

VALUES (user, TO_CHAR(sysdate, 'DD/MM/YY HH24:MI:SS'));

END;
```

Criando Triggers

END;

Exemplo

```
CREATE OR REPLACE TRIGGER teste
AFTER UPDATE OF preco
ON LIVRO
REFERENCING NEW AS NI OLD AS OI
FOR EACH ROW
WHEN (NI.PRECO > 01.PRECO *1.2)
BEGIN
  INSERT INTO auditoria(codlivro, precoantigo, preconovo, datahora)
  VALUES(:O1.codlivro, :O1.preco, :N1.preco, TO_CHAR(sysdate, 'DD/MM/YY
  HH24:MI:SS'));
                                                        Aumentar o preco
```

em 21%, ver a tabela auditoria e

depois dar ROLLBACK

- DQL (Data Query Language)
 - Comando SELECT

SELECT [DISTINCT] {*, column [alias], columnN [aliasN]} FROM table;

Exemplo

```
SELECT codlivro CODIGO_LIVRO ,
titulo TÍTULO ,
numpags "NÚMERO DE PÁGINAS"
FROM LIVRO;
```

	A	CODIGO_LIVRO	A	TÍTULO	A	NÚMERO DE PÁGINAS
1		1	ВА	NCO DE DADOS PARA A WEB		55
2		2	PRO	OGRAMANDO EM LINGUAGEM C		47
3		3	PRO	OGRAMANDO EM LINGUAGEM C++		103
4		4	ВА	NCO DE DADOS NA BIOINFORMATICA		66
5		5	RED	DES DE COMPUTADORES		93
6		6	ВА	NCO DE DADOS TEMPORAIS		102

- DQL (Data Query Language)
 - Comando SELECT Restringindo e ordenando dados
 - Exemplo

```
SELECT codlivro CODIGO_LIVRO,
titulo TÍTULO,
numpags "NÚMERO DE PÁGINAS"

FROM LIVRO
WHERE numpags > 70 Operador de comparação
ORDER BY 2 ASC; --OU DESC
```

	A	CODIGO_LIVRO	₹ ¶ TÍTULO	A	NÚMERO DE PÁGINAS
1		6	BANCO DE DADOS TEMPORAIS		102
2		3	PROGRAMANDO EM LINGUAGEM C++		103
3		5	REDES DE COMPUTADORES		93

- DQL (Data Query Language)
 - Comando SELECT Operadores de comparação

OPERADOR	SIGNIFICADO
=	Igual a
>	Maior que
	A A cric v. cu. ro. cr. ri cu. rol
>=	Maior que ou igual
<	Menor que
<=	·
	Menor que ou igual
<> OU !=	Diferente de
BETWEEN	Entre dois valores (inclusive)
IN	Igual a um dos valores da lista
LIKE	Igual a um padrão de caracteres
IS NULL	Possui um valor nuno

- DQL (Data Query Language)
 - Comando SELECT Restringindo e ordenando dados
 - Exemplo

```
SELECT codlivro CODIGO_LIVRO,
titulo TÍTULO,
numpags "NÚMERO DE PÁGINAS"
FROM LIVRO
WHERE numpags BETWEEN 55 AND 90
ORDER BY 2 ASC;
```

A	CODIGO_LIVRO 🛭 TÍTULO	Ħ	NÚMERO DE PÁGINAS
	4 BANCO DE DADOS NA BIOINFORMATICA		66
	1 BANCO DE DADOS PARA A WEB		55

- DQL (Data Query Language)
 - Comando SELECT Junção de Tabelas

INNER JOIN

SELECT CL.nome, CA.tipocartao
FROM CLIENTE CL INNER JOIN CARTÃO CA
ON CL.codcartao=ca.codcartao
ORDER BY 2 DESC;

2 NOME	TIPOCARTAO		
Ana Silveira Escobar	SPECIAL		
Carla Maldonado Silva	SPECIAL		
Marco Antônio Rossi	PREMIUM		
Henrique Meirelles	PREMIUM		
Cleide Castanheira	PREMIUM		
Júlio Siqueira	PREMIUM		
Elza Saldanha Prado	PREMIUM		
César Gonçalves	PREMIUM		
João Inácio Magalhães	PREMIUM		
Flávio Aguiar Fonseca	PREMIUM		

- DQL (Data Query Language)
 - Comando SELECT Junção de Tabelas

LEFT OUTER JOIN

SELECT CL.nome, CA.tipocartao

FROM CLIENTE CL LEFT OUTER JOIN CARTÃO CA

ON CL.codcartao=ca.codcartao

ORDER BY 2 DESC;

NOME	TIPOCARTAC
César Oliveira	(null)
Carlos Padilha	(null)
Luciano Blomberg	(null)
Carlos Tenório	(null)
Marcos Andrade	(null)
Maria Fonseca	(null)
Carla Maldonado Silva	SPECIAL
Ana Silveira Escobar	SPECIAL
César Gonçalves	PREMIUM
João Inácio Magalhães	PREMIUM
Henrique Meirelles	PREMIUM
Cleide Castanheira	PREMIUM
Júlio Siqueira	PREMIUM
Elza Saldanha Prado	PREMIUM
Marco Antônio Rossi	PREMIUM
Flávio Aguiar Fonseca	PREMIUM

- DQL (Data Query Language)
 - Comando SELECT Junção de Tabelas

RIGHT OUTER JOIN

SELECT CL.nome, CA.tipocartao

FROM CLIENTE CL RIGHT OUTER JOIN CARTÃO CA

ON CL.codcartao=ca.codcartao

ORDER BY 2 DESC;

2 NOME	TIPOCARTAO		
Ana Silveira Escobar	SPECIAL		
Carla Maldonado Silva	SPECIAL		
Flávio Aguiar Fonseca	PREMIUM		
César Gonçalves	PREMIUM		
João Inácio Magalhães	PREMIUM		
Henrique Meirelles	PREMIUM		
Marco Antônio Rossi	PREMIUM		
Cleide Castanheira	PREMIUM		
Júlio Siqueira	PREMIUM		
Elza Saldanha Prado	PREMIUM		
(null)	GOLD PLUS		
(null)	GOLD		

- DQL (Data Query Language)
 - Comando SELECT Junção de Tabelas

FULL OUTER JOIN

SELECT CL.nome, CA.tipocartao

FROM CLIENTE CL FULL OUTER JOIN CARTÃO CA

ON CL.codcartao=ca.codcartao

ORDER BY 2 DESC;

NOME	TIPOCARTAO		
Luciano Blomberg	(null)		
Carlos Tenório	(null)		
Marcos Andrade	(null)		
Maria Fonseca	(null)		
César Oliveira	(null)		
Carlos Padilha	(null)		
Carla Maldonado Silva	SPECIAL		
Ana Silveira Escobar	SPECIAL		
João Inácio Magalhães	PREMIUM		
Flávio Aguiar Fonseca	PREMIUM		
Henrique Meirelles	PREMIUM		
Cleide Castanheira	PREMIUM		
Júlio Siqueira	PREMIUM		
Elza Saldanha Prado	PREMIUM		
Marco Antônio Rossi	PREMIUM		
César Gonçalves	PREMIUM		
(null)	GOLD PLUS		
(null)	GOLD		

- DQL (Data Query Language)
 - Comando SELECT Junção de Tabelas

CROSS JOIN

SELECT CL.nome, CA.tipocartao
FROM CLIENTE CL CROSS JOIN CARTÃO CA
ORDER BY 2 DESC;

	2 NOME	TIPOCARTAO			
1		_			
1	Maria Fonseca	SPECIAL			
2	César Oliveira	SPECIAL			
3	Carlos Tenório	SPECIAL			
4	Luciano Blomberg	SPECIAL			
5	Carlos Padilha	SPECIAL			
6	Marco Antônio Rossi	SPECIAL			
7	Elza Saldanha Prado	SPECIAL			
8	Júlio Siqueira	SPECIAL			
9	Cleide Castanheira	SPECIAL			
10	Henrique Meirelles	SPECIAL			
11	Flávio Aguiar Fonseca	SPECIAL			
	:				
61	César Gonçalves	GOLD			
62	João Inácio Magalhães	GOLD			
63	Ana Silveira Escobar	GOLD			
64	Carla Maldonado Silva	GOLD			

- DQL (Data Query Language)
 - Comando SELECT Junção de Tabelas

NATURAL JOIN

SELECT CL.nome, CA.tipocartao FROM CLIENTE CL NATURAL JOIN CARTÃO CA ORDER BY 2 DESC;

NOME	TIPOCARTAO		
Ana Silveira Escobar	SPECIAL		
Carla Maldonado Silva	SPECIAL		
Marco Antônio Rossi	PREMIUM		
Henrique Meirelles	PREMIUM		
Cleide Castanheira	PREMIUM		
Júlio Siqueira	PREMIUM		
Elza Saldanha Prado	PREMIUM		
César Gonçalves	PREMIUM		
João Inácio Magalhães	PREMIUM		
Flávio Aguiar Fonseca	PREMIUM		

- DQL (Data Query Language)
 - Comando SELECT Junção de Tabelas

SELECT CL.nome, CA.tipocartao

FROM CLIENTE CL, CARTÃO CA

WHERE CL.codcartao=CA.codcartao

ORDER BY 2 DESC:

SELECT CL.nome, CA.tipocartao

FROM CLIENTE CL, CARTÃO CA

WHERE CL.codcartao=CA.codcartao(+)

ORDER BY 2 DESC;

SELECT CL.nome, CA.tipocartao

FROM CLIENTE CL, CARTÃO CA

WHERE CL.codcartao(+)=CA.codcartao

ORDER BY 2 DESC:

INNER JOIN

LEFT OUTER JOIN

RIGHT OUTER JOIN

- DQL (Data Query Language)
 - Comando SELECT Funções Básicas

SELECT sigla, assunto

FROM ASSUNTO

WHERE sigla='b';

SELECT sigla, assunto

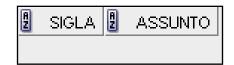
FROM ASSUNTO

WHERE **LOWER**(sigla)='b';

SELECT sigla, **UPPER**(assunto) AS ASSUNTO

FROM ASSUNTO

WHERE **LOWER**(sigla)='b';







- DQL (Data Query Language)
 - Comando SELECT Funções Básicas

SELECT sigla, **INITCAP**(assunto) AS ASSUNTO FROM ASSUNTO WHERE **LOWER**(sigla)='b';



SELECT sigla, **INITCAP**(**SUBSTR**(assunto, 1, 5)) AS ASSUNTO FROM ASSUNTO WHERE **LOWER**(sigla)='b';



DQL (Data Query Language)

Comando SELECT – Funções Básicas

SELECT titulo, preco PRECO_NORMAL

FROM LIVRO

WHERE LOWER(titulo) LIKE 'b%';

SELECT titulo, ROUND(preco) PRECO_ARRED

FROM LIVRO

WHERE LOWER(titulo) LIKE 'b%';

SELECT titulo, TRUNC(preco) PRECO_TRUNC

FROM LIVRO

WHERE LOWER(titulo) LIKE 'b%';

SELECT titulo, MOD (preco, preco) PRECO_RESTDI

FROM LIVRO

WHERE LOWER(titulo) LIKE 'b%';

2 TITULO	A	PRECO_NORMAL
BANCO DE DADOS PARA A WEB		37,82
BANCO DE DADOS NA BIOINFORMATICA		59,28
BANCO DE DADOS TEMPORAIS		55

2 TITULO	A	PRECO_ARRED
BANCO DE DADOS PARA A WEB		38
BANCO DE DADOS NA BIOINFORMATICA		59
BANCO DE DADOS TEMPORAIS		55

TITULO	A	PRECO_TRUNC
BANCO DE DADOS PARA A WEB		37
BANCO DE DADOS NA BIOINFORMATICA		59
BANCO DE DADOS TEMPORAIS		55

2 TITULO	A	PRECO_RESTDI
BANCO DE DADOS PARA A WEB		0
BANCO DE DADOS NA BIOINFORMATICA		0
BANCO DE DADOS TEMPORAIS		0

- DQL (Data Query Language)
 - Comando SELECT Funções de Conversão de Tipos de dados

FUNÇÃO	PROPÓSITO
TO_CHAR(number date,['fmt'])	Converte um valor numérico ou data para uma string de caracteres do tipo VARCHAR2 no formato fmt.
TO_NUMBER (char)	Converte uma string de caracteres contendo apenas dígitos para um número.
TO_DATE(char,['fmt'])	Converte uma string de caracteres representando uma data para um valor de data de acordo com o formato fmt especificado.

- DQL (Data Query Language)
 - Comando SELECT Funções de Conversão de Tipos de dados

ELEMENTO	DESCRIÇÃO
YY / YYYY	Número do ano com 2/4 dígitos
YEAR	Ano soletrado
MM	Número do mês com 2 dígitos
MONTH	Nome completo do mês
DY	Abreviação de 3 letras do nome do dia
	da semana.
DAY	Nome completo do dia.
HH OU HH12 OU HH24	Hora do dia
MI	Minuto (0-59)
SS	Segundo (0-59)

- DQL (Data Query Language)
 - Comando SELECT Funções de Conversão de Tipos de dados

SELECT titulo, **TO_CHAR**(datapublicacao, 'DD/MON/YY') DATA_PUBLICAÇÃO FROM LIVRO;

DATA_PUBLICAÇÃO
2 DATA_PUBLICAÇÃO
10/JAN/99
01/OUT/97
01/NOV/98
(null)
01/SET/96
10/FEV/01

- DQL (Data Query Language)
 - Comando SELECT Funções de Conversão de Tipos de dados

SELECT titulo, **TO_CHAR**(preco,'L99G999D99') PRECO_LIVRO FROM LIVRO;

TITULO	PRECO_LIVRO
BANCO DE DADOS PARA A WEB	R\$37,82
PROGRAMANDO EM LINGUAGEM C	R\$40,33
PROGRAMANDO EM LINGUAGEM C++	R\$147,20
BANCO DE DADOS NA BIOINFORMATICA	R\$59,28
REDES DE COMPUTADORES	R\$46,42
BANCO DE DADOS TEMPORAIS	R\$55,00

- DQL (Data Query Language)
 - Comando SELECT Funções de Conversão de Tipos de dados

SELECT titulo, **TO_NUMBER**('50') ESTOQUE FROM LIVRO;

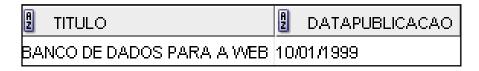
2 TITULO	A	ESTOQUE
BANCO DE DADOS PARA A WEB		50
PROGRAMANDO EM LINGUAGEM C		50
PROGRAMANDO EM LINGUAGEM C++		50
BANCO DE DADOS NA BIOINFORMATICA		50
REDES DE COMPUTADORES		50
BANCO DE DADOS TEMPORAIS		50

- DQL (Data Query Language)
 - Comando SELECT Funções de Conversão de Tipos de dados

SELECT titulo, datapublicacao

FROM LIVRO

WHERE datapublicacao= **TO_DATE**('Janeiro 10, 1999', 'Month dd, YYYY');



- DQL (Data Query Language)
 - Comando SELECT Funções de Agrupamento
 - QUANTAS VENDAS FORAM REALIZADAS POR LOJA?
 - QUAL FOI O VALOR TOTAL DE VENDAS POR LOJA?
 - QUAL FOI O VALOR MÉDIO DE VENDAS POR LOJA?
 - QUAL FOI A MAIOR VENDA REALIZADA POR LOJA?
 - QUAL FOI A MENOR VENDA REALIZADA POR LOJA?

- DQL (Data Query Language)
 - Comando SELECT Funções de Agrupamento
 - QUANTAS VENDAS FORAM REALIZADAS POR LOJA?

SELECT L.nome, **COUNT**(DISTINCT(V.idvenda)) QUANT_VENDAS FROM VENDA V INNER JOIN LOJA L ON V.cnpj=L.cnpj GROUP BY L.nome;

NOME	A	QUANT_VENDAS
Tabajara 1		12
Tabajara 2		11

- DQL (Data Query Language)
 - Comando SELECT Funções de Agrupamento
 - QUAL FOI O VALOR TOTAL DE VENDAS POR LOJA?

SELECT L.nome, **SUM**(V.total) TOTAL_VENDAS FROM VENDA V INNER JOIN LOJA L ON V.cnpj=L.cnpj GROUP BY L.nome;

NOME	A	TOTAL_VENDAS
Tabajara 1		2267,21
Tabajara 2		1658,99

- DQL (Data Query Language)
 - Comando SELECT Funções de Agrupamento
 - QUAL FOI O VALOR MÉDIO DE VENDAS POR LOJA?

SELECT

L.nome,

TRUNC(SUM(V.total) / COUNT(DISTINCT(V.idvenda)),2) MEDIA_VENDAS

FROM VENDA V INNER JOIN LOJA L ON V.cnpj=L.cnpj GROUP BY L.nome;

NOME	A	MEDIA_VENDAS
Tabajara 1		188,93
Tabajara 2		150,81

Valor médio de vendas != Valor médio de produtos vendidos

- DQL (Data Query Language)
 - Comando SELECT Funções de Agrupamento
 - QUAL FOI A MAIOR VENDA REALIZADA POR LOJA?

SELECT L.nome,

MAX(R.TOT) MAIOR_VENDA

FROM LOJA L INNER JOIN VENDA V ON I.cnpj= v.cnpj

INNER JOIN (SELECT I.nome nome, SUM(V.total) TOT

FROM LOJA L, VENDA V

WHERE L.cnpj= v.cnpj

GROUP BY I.nome, v.idvenda) R ON I.nome=R.nome

GROUP BY L.NOME;

NOME	A	MAIOR_VENDA
Tabajara 1		443,4
Tabajara 2		715,2

- DQL (Data Query Language)
 - Comando SELECT Funções de Agrupamento
 - QUAL FOI A MENOR VENDA REALIZADA POR LOJA?

SELECT L.nome,

MIN(R.TOT) MENOR_VENDA

FROM LOJA L INNER JOIN VENDA V ON I.cnpj= v.cnpj

INNER JOIN (SELECT I.nome nome, SUM(V.total) TOT

FROM LOJA L, VENDA V

WHERE L.cnpj= v.cnpj

GROUP BY I.nome, v.idvenda) R ON I.nome=R.nome

GROUP BY L.NOME;

NOME	MAIOR_VENDA
Tabajara 1	50,82
Tabajara 2	36

- DQL (Data Query Language)
 - Comando SELECT Funções de Agrupamento

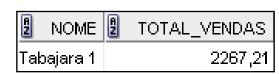
SELECT L.nome, **SUM**(V.total) TOTAL_VENDAS FROM VENDA V INNER JOIN LOJA L ON V.cnpj=L.cnpj GROUP BY L.nome;

B NOME Z	TOTAL_VENDAS
Tabajara 1	2267,21
Tabajara 2	1658,99

Total de vendas por loja

SELECT L.nome, **SUM**(V.total) TOTAL_VENDAS
FROM VENDA V INNER JOIN LOJA L ON V.cnpj=L.cnpj
GROUP BY L.nome

HAVING SUM(V.total) > 2100;



Acima da meta de 2100

- DQL (Data Query Language)
 - Comando SELECT Outras Funções
 - ATENÇÃO!

SELECT COUNT(V.total) REGISTROS_COM_TOTAL_INFORMADO FROM VENDA V;



SELECT COUNT(**NVL**(V.total,0)) REGISTROS_COM_TOTAL_INFORMADO FROM VENDA V;



- DQL (Data Query Language)
 - Comando SELECT Outras Funções

AN	CODLIVRO	A	TOTAL
	5		126
	5		42
	5		42
	5		42
	5		84
	5		84
	5		(null)
	5		(null)
	5		(null)

SELECT codlivro, total FROM VENDA
WHERE codlivro=5;

SELECT codlivro, NVL(total,0)
FROM VENDA
WHERE codlivro=5

Ą	CODLIVRO	2 TOTAL
	5	126
	5	42
	5	42
	5	42
	5	84
	5	84
	5	0
	5	0
	5	0

- DQL (Data Query Language)
 - Comando SELECT Outras Funções

SELECT nome,

TO_CHAR(datanasc,'yyyy')ANO_NASC,

CASE LOWER(nacionalidade) WHEN 'colombiana' THEN 'COL'

WHEN 'brasileira' THEN 'BRA'

WHEN 'equatoriana' THEN 'EQU'

ELSE 'NÃO INFORMADA' END CASE

FROM cliente

WHERE TO_CHAR(datanasc,'yyyy') > 1977;

2 NOME	ANO_NASC	2 CASE
Ana Silveira Escobar	1981	COL
Carla Maldonado Silva	1979	BRA
João Inácio Magalhães	1979	BRA
Júlio Siqueira	1982	BRA
Luciano Blomberg	1978	BRA

- DQL (Data Query Language)
 - Comando SELECT Outras Funções

```
SELECT nome,
```

TO_CHAR(datanasc,'yyyy')ANO_NASC,
DECODE(LOWER(nacionalidade),'colombiana','COL',

'brasileira', 'BRA',

'equatoriana', 'EQU',

'NÃO INFORMADA') NACILONALIDADE

FROM cliente

WHERE TO_CHAR(datanasc,'yyyy') > 1977;

NOME NOME	ANO_NASC	NACILONALIDADE
Ana Silveira Escobar	1981	COL
Carla Maldonado Silva	1979	BRA
João Inácio Magalhães	1979	BRA
Júlio Siqueira	1982	BRA
Luciano Blomberg	1978	BRA

- DQL (Data Query Language)
 - Comando SELECT Operações de Conjunto

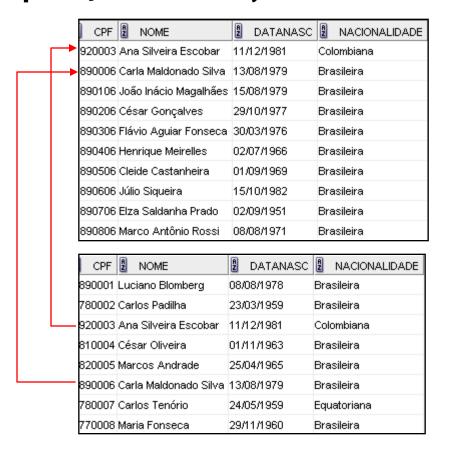
EXECUTAR COMANDO:

DELETE FROM CLIENTE C
WHERE C.CODCARTAO IS NULL;

- DQL (Data Query Language)
 - Comando SELECT Operações de Conjunto

SELECT cpf,
nome,
datanasc,
nacionalidade
FROM CLIENTE;

SELECT cpf,
nome,
datanasc,
nacionalidade
FROM AUTOR:



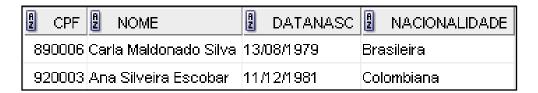
- DQL (Data Query Language)
 - Comando SELECT Operações de Conjunto

SELECT cpf, nome, datanasc, nacionalidade

FROM CLIENTE

INTERSECT

SELECT cpf, nome, datanasc, nacionalidade FROM AUTOR



- DQL (Data Query Language)
 - Comando SELECT Operações de Conjunto

SELECT cpf, nome, datanasc, nacionalidade FROM CLIENTE

MINUS

SELECT cpf, nome, datanasc, nacionalidade FROM AUTOR

2 CPF	NOME	2 DATANASC	NACIONALIDADE
_	João Inácio Magalhães	<u> </u>	Brasileira
	César Gonçalves	29/10/1977	Brasileira
	Flávio Aguiar Fonseca	30/03/1976	Brasileira
890406	Henrique Meirelles	02/07/1966	Brasileira
890506	Cleide Castanheira	01/09/1969	Brasileira
890606	Júlio Siqueira	15/10/1982	Brasileira
890706	Elza Saldanha Prado	02/09/1951	Brasileira
890806	Marco Antônio Rossi	08/08/1971	Brasileira

- DQL (Data Query Language)
 - Comando SELECT Operações de Conjunto

SELECT cpf, nome, datanasc, nacionalidade FROM CLIENTE

UNION

SELECT cpf, nome, datanasc, nacionalidade FROM AUTOR

2 CPF	2 NOME	2 DATANASC	NACIONALIDADE
770008	Maria Fonseca	29/11/1960	Brasileira
780002	Carlos Padilha	23/03/1959	Brasileira
780007	Carlos Tenório	24/05/1959	Equatoriana
810004	César Oliveira	01/11/1963	Brasileira
820005	Marcos Andrade	25/04/1965	Brasileira
890001	Luciano Blomberg	08/08/1978	Brasileira
890006	Carla Maldonado Silva	13/08/1979	Brasileira
890106	João Inácio Magalhães	15/08/1979	Brasileira
890206	César Gonçalves	29/10/1977	Brasileira
890306	Flávio Aguiar Fonseca	30/03/1976	Brasileira
890406	Henrique Meirelles	02/07/1966	Brasileira
890506	Cleide Castanheira	01/09/1969	Brasileira
890606	Júlio Siqueira	15/10/1982	Brasileira
890706	Elza Saldanha Prado	02/09/1951	Brasileira
890806	Marco Antônio Rossi	08/08/1971	Brasileira
920003	Ana Silveira Escobar	11/12/1981	Colombiana

- DQL (Data Query Language)
 - Subconsultas Não-Correlacionadas
 - Single-Row
 - Multiple-Row
 - Multiple-Column

Subconsultas Correlacionadas

- Multiple-Row
- Multiple-Column

- DQL (Data Query Language)
 - Subconsultas Não-Correlacionadas
 - Single-Row

Consulta Externa SELECT *
FROM LIVRO
WHERE preco <

A subconsulta é independente da consulta externa

Consulta Interna (SELECT preco FROM LIVRO WHERE codlivro=4);

2 TITULO	2 DATAPUBLICACAO	🖁 CODLIVRO 🖁	NUMPAGS	PRECO	2 CODEDITORA 2	SIGLA
BANCO DE DADOS PARA A WEB	10/01/1999	1	55	37,82	1 B	
PROGRAMANDO EM LINGUAGEM C	01/10/1997	2	47	40,33	1 P	
REDES DE COMPUTADORES	01/09/1996	5	93	46,42	2 R	
BANCO DE DADOS TEMPORAIS	10/02/2001	7	102	55	10 B	

- DQL (Data Query Language)
 - Subconsultas Não-Correlacionadas
 - Multiple-Row

Consulta Externa SELECT *
FROM LIVRO
WHERE PRECO IN

(SELECT preco

Consulta Interna FROM LIVRO

WHERE TO_CHAR(datapublicacao, 'YYYY') < '1998')

TITULO	2 DATAPUBLICACAO	2 CODLIVRO	NUMPAGS	PRECO	2 CODEDITORA	SIGLA
PROGRAMANDO EM LINGUAGEM C	01/10/1997	2	47	40,33	1 P	
REDES DE COMPUTADORES	01/09/1996	5	93	46,42	2 R	:

A subconsulta é independente da consulta externa

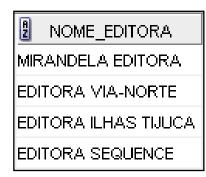
- DQL (Data Query Language)
 - Subconsultas Correlacionadas
 - Multiple-Row

SELECT nome as NOME_EDITORA
FROM EDITORA E
WHERE EXISTS
(SELECT L.titulo

A subconsulta possui dependência direta com a consulta externa

FROM LIVRO I

WHERE E.codeditora = L.codeditora AND L.datapublicacao IS NOT NULL);



- DQL (Data Query Language)
 - Subconsultas Correlacionadas
 - Nos campos de saída

SELECT E.nome, (SELECT COUNT(*)
FROM LIVRO L
WHERE L.numpags > 60 AND
L.codeditora=E.codeditora)

"LIVROS+60PAGS" FROM EDITORA E;

2 NOME	LIVROS+60PAGS
MIRANDELA EDITORA	0
EDITORA VIA-NORTE	2
EDITORA ILHAS TIJUCA	1
MARIA JOSÉ EDITORA	0
NOVA EDITORA	0
EDITORA MACEDO	0
EDITORA SILVA	0
GLOBO	0
EDIPUCRS	0
EDITORA SEQUENCE	1

- DQL (Data Query Language)
 - Subconsultas Correlacionadas
 - Na cláusula FROM

SELECT LIVRO_AUTOR.nome AUTOR,

LIVRO_AUTOR.titulo,

LIVRO_AUTOR.preco,

E.nome EDITORA

FROM EDITORA E,

(SELECT I.titulo, L.datapublicacao, A.nome, I.preco, I.codeditora

FROM LIVRO L INNER JOIN LIVRO_AUTORIA LA ON L.codlivro= LA.codlivro

INNER JOIN AUTOR A ON A.matricula=LA.matricula) LIVRO_AUTOR

WHERE E.codeditora=LIVRO_AUTOR.codeditora AND LIVRO_AUTOR.datapublicacao IS NOT NULL

ORDER BY 2, I ASC;

- DQL (Data Query Language)
 - Subconsultas Correlacionadas
 - Na cláusula FROM

2 AUTOR	1 TITULO	2 PRECO	2 EDITORA
Luciano Blomberg	BANCO DE DADOS TEMPORAIS	55	EDITORA SEQUENCE
Ana Silveira Escobar	BANCO DE DADOS PARA A WEB	37,82	MIRANDELA EDITORA
Carlos Padilha	BANCO DE DADOS PARA A WEB	37,82	MIRANDELA EDITORA
Luciano Blomberg	BANCO DE DADOS PARA A WEB	37,82	MIRANDELA EDITORA
César Oliveira	PROGRAMANDO EM LINGUAGEM C	40,33	MIRANDELA EDITORA
César Oliveira	PROGRAMANDO EM LINGUAGEM C++	147,2	EDITORA ILHAS TIJUCA
Marcos Andrade	PROGRAMANDO EM LINGUAGEM C++	147,2	EDITORA ILHAS TIJUCA
Carlos Tenório	REDES DE COMPUTADORES	46,42	EDITORA VIA-NORTE
Maria Fonseca	REDES DE COMPUTADORES	46,42	EDITORA VIA-NORTE