Universidade Presbiteriana Mackenzie

Banco de Dados – Aula 12

Linguagem SQL

SELECT Básico

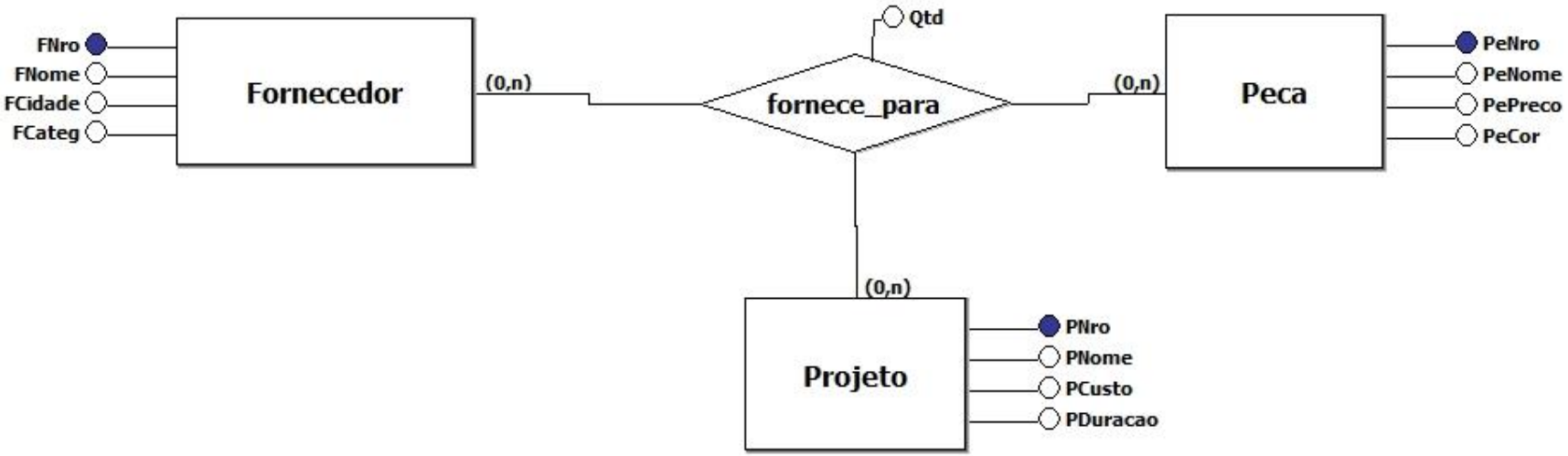
Profa. Elisângela Botelho Gracias

Faculdade de Computação e Informática

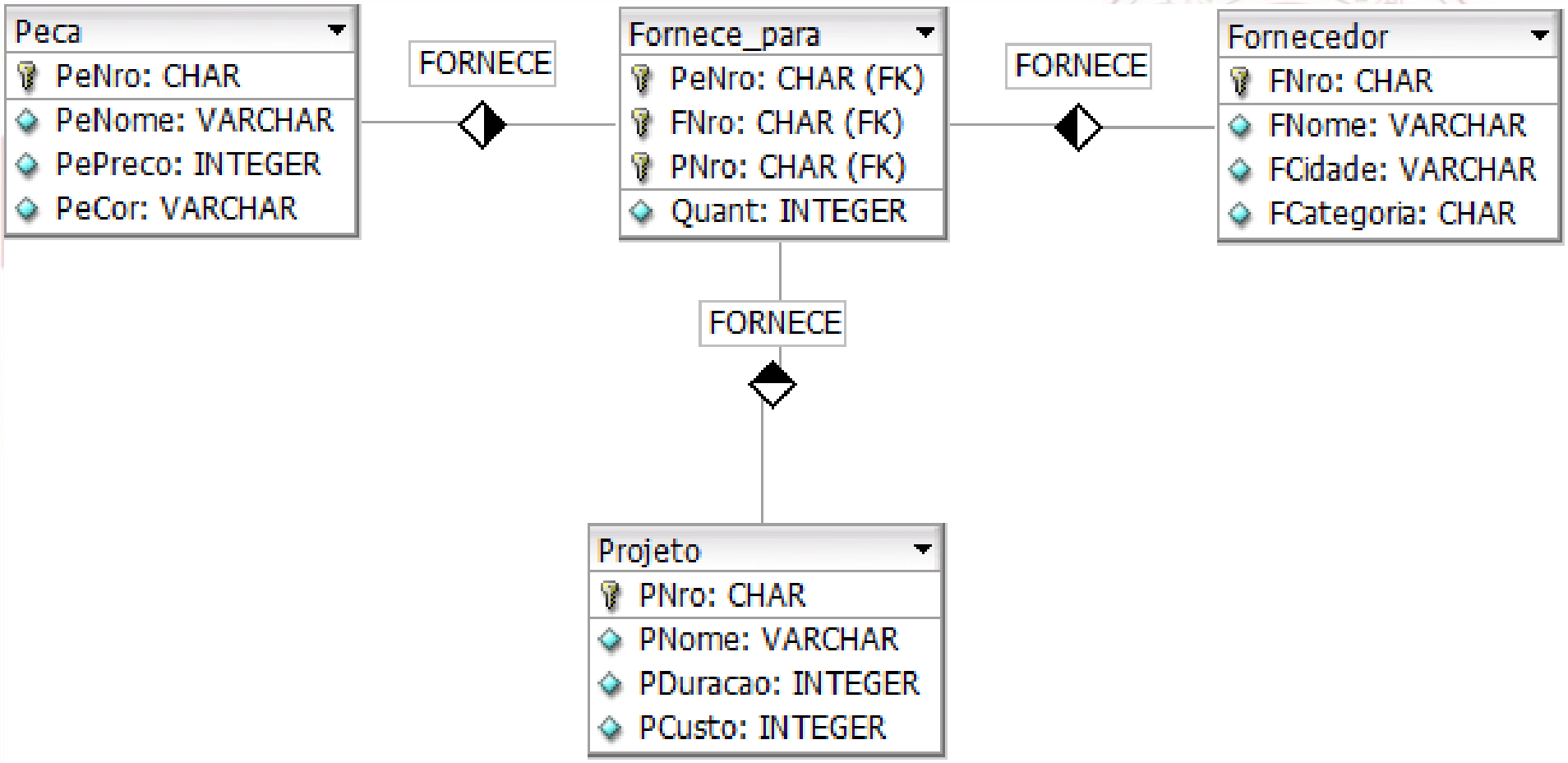
Banco de Dados Exemplo

2

Modelo Entidade-Relacionamento



Modelo Relacional



Exemplo de Banco de Dados

Considere o seguinte modelo relacional (chaves primárias estão sublinhadas:

PECA = {PeNro, PeNome, PePreco, PeCor}

FORNECEDOR = {FNro, FNome, FCidade, FCateg}

PROJETO = {PNro, PNome, PDuracao, PCusto}

FORNECE_PARA = {PeNro, FNro, PNro, Quant}

- **PeNro é chave estrangeira que referencia a tabela Peca**
- **FNro é chave estrangeira que referencia a tabela Fornecedor**
- **PNro é chave estrangeira que referencia a tabela Projeto**

5

Peca

PeNro	PeNome	PePreço	PeCor
PE1	Cinto	22	Azul
PE2	Volante	18	Vermelho
PE3	Lanterna	14	Preto
PE4	Limpador	09	Amarelo
PE5	Painel	43	Vermelho

Fornecedor

FNro	FNome	FCidade	FCateg
F1	Plastec	Campinas	B
F2	CM	São Paulo	D
F3	Kirurgic	Campinas	A
F4	Piloto	Piracicaba	A
F5	Equipament	São Carlos	C

Projeto

PNro	PNome	PDuração	PCusto
P1	Detroit	5	43000
P2	Pegasus	3	37000
P3	Alfa	2	26700
P4	Sea	3	21200
P5	Paraiso	1	17000

Fornece_para

PeNro	FNro	PNro	Quant
PE1	F5	P4	5
PE2	F2	P2	1
PE3	F3	P4	2
PE4	F4	P5	3
PE5	F1	P1	1
PE2	F2	P3	1
PE4	F3	P5	2

--**Script de criação do Banco de Dados:**

```
DROP TABLE Fornece_Para CASCADE CONSTRAINT;
DROP TABLE Projeto CASCADE CONSTRAINT;
DROP TABLE Fornecedor CASCADE CONSTRAINT;
DROP TABLE Peca CASCADE CONSTRAINT;

CREATE TABLE Peca (
PeNro CHAR(4),
PeNome VARCHAR(30) NOT NULL,
PePreco INTEGER NOT NULL,
PeCor VARCHAR(20) NOT NULL,
PRIMARY KEY(PeNro));

CREATE TABLE Fornecedor (
FNro CHAR(4),
FNome VARCHAR(30) NOT NULL,
FCidade VARCHAR(30) NOT NULL,
FCategoria CHAR(1) NOT NULL,
PRIMARY KEY(FNro));

CREATE TABLE Projeto (
PNro CHAR(4),
PNome VARCHAR(30) NOT NULL,
PDuracao INTEGER NOT NULL,
PCusto INTEGER NOT NULL,
PRIMARY KEY(PNro));

CREATE TABLE Fornece_para (
PeNro CHAR(4),
FNro CHAR(4),
PNro CHAR(4),
Quant INTEGER,
PRIMARY KEY(PeNro,FNro,PNro),
FOREIGN KEY(PeNro) REFERENCES Peca(PeNro),
FOREIGN KEY(FNro) REFERENCES Fornecedor(FNro),
FOREIGN KEY(PNro) REFERENCES Projeto(PNro));
```

```
INSERT INTO Peca VALUES ('PE1', 'Cinto', 22, 'Azul');
INSERT INTO Peca VALUES ('PE2', 'Volante', 18, 'Vermelho');
INSERT INTO Peca VALUES ('PE3', 'Lanterna', 14, 'Preto');
INSERT INTO Peca VALUES ('PE4', 'Limpador', 9, 'Amarelo');
INSERT INTO Peca VALUES ('PE5', 'Painel', 43, 'Vermelho');

INSERT INTO Fornecedor VALUES ('F1', 'Plastec', 'Campinas', 'B');
INSERT INTO Fornecedor VALUES ('F2', 'CM', 'Sao Paulo', 'D');
INSERT INTO Fornecedor VALUES ('F3', 'Kirurgic', 'Campinas', 'A');
INSERT INTO Fornecedor VALUES ('F4', 'Piloto', 'Piracicaba', 'A');
INSERT INTO Fornecedor VALUES ('F5', 'Equipament', 'Sao Carlos', 'C');

INSERT INTO Projeto VALUES ('P1', 'Detroit', 5, 43000);
INSERT INTO Projeto VALUES ('P2', 'Pegasus', 3, 37000);
INSERT INTO Projeto VALUES ('P3', 'Alfa', 2, 26700);
INSERT INTO Projeto VALUES ('P4', 'Sea', 3, 21200);
INSERT INTO Projeto VALUES ('P5', 'Paraiso', 1, 17000);

INSERT INTO Fornece_para VALUES ('PE1', 'F5', 'P4', 5);
INSERT INTO Fornece_para VALUES ('PE2', 'F2', 'P2', 1);
INSERT INTO Fornece_para VALUES ('PE3', 'F3', 'P4', 2);
INSERT INTO Fornece_para VALUES ('PE4', 'F4', 'P5', 3);
INSERT INTO Fornece_para VALUES ('PE5', 'F1', 'P1', 1);
INSERT INTO Fornece_para VALUES ('PE2', 'F2', 'P3', 1);
INSERT INTO Fornece_para VALUES ('PE4', 'F3', 'P5', 2);
COMMIT;
```

SELECT

- SELECT
  - possibilita a consulta de uma ou mais tabelas de acordo com os critérios estabelecidos e com as necessidades

SELECT

- Sintaxe do comando SELECT

SELECT [DISTINCT] nome_atributo1,... nome_atributoN

FROM nome_tabela1, ... nome_tabelaN

[WHERE (condições**)]**

[GROUP BY nome_atributo1,... nome_atributoN**]**

[HAVING (condições**)]**

[ORDER BY nome_atributo1 **{ASC | DESC}, ...**

nome_atributoN **{ASC | DESC}] ;**

Obs: *tudo que está entre [ ] é opcional, mas se for utilizar tire o [ ]*

SELECT

- Onde:
  - SELECT: o que se deseja no resultado da consulta
  - DISTINCT: não permite repetição de valores no resultado
  - FROM: de onde buscar os dados necessários
  - WHERE: condições para busca dos resultados

SELECT

- Onde (continuação):
  - GROUP BY: agrupamento de dados
  - HAVING: condições para a definição de grupos no resultado
  - ORDER BY: estabelece a ordenação lógica do resultado

SELECT

- **Precedência de operadores:** se vários operadores aparecerem em uma consulta, eles serão executados na seguinte sequência:
  1. Parênteses ( )
  2. Multiplicação / Divisão (* /)
  3. Adição / Subtração (+ -)
  4. NOT
  5. AND
  6. OR

SELECT

- Exemplo1 (DISTINCT): Obtenha o código de todas as peças fornecidas para projetos (**sem repetição de códigos**)

```
SELECT DISTINCT PeNro
FROM Fornece_para;
```

PeNro
PE1
PE2
PE3
PE4
PE5

SELECT

- Observe, no exemplo anterior, que **DISTINCT** traz o resultado da consulta, **eliminando as linhas duplicadas**, ou seja, se mais de uma linha do resultado da consulta contém valores iguais, ele só trará uma linha com estes valores

SELECT

- Exemplo2 (WHERE): Obtenha o nome e o código dos fornecedores da cidade de 'Campinas '

SELECT FNome, FNro

FROM Fornecedor

WHERE (FCidade = 'Campinas');

FNome	FNro
Plastec	F1
Kirurgic	F3

SELECT

- Observe, no exemplo anterior, que a cláusula **WHERE** seleciona somente as linhas da tabela que obedecerem às condições contidas nela, ou seja, na cláusula **WHERE** tem-se a condição de linha da tabela

Operadores

- Operadores Aritméticos
  - **+** (*adição*)
  - **-** (subtração)
  - ***** (*multiplicação*)
  - **/** (*divisão*)

SELECT

- Exemplo3 (Multiplicação): Obtenha o nome e a duração, em DIAS, de cada projeto

SELECT PNome, (PDuracao * 30) **AS** Dias

FROM Projeto;

PNome	Dias
Detroit	150
Pegasus	90
Alfa	60
Sea	90
Paraiso	30

SELECT

- Observe que é possível utilizar os **operadores aritméticos (+ - * /)** em uma consulta, já que pode-se formatar o resultado da consulta da maneira que desejar
- O **AS** permite alterar o nome de um atributo/expressão somente no resultado da consulta

Operadores

- Operadores Relacionais
  - **<** **e** **<=** (*menor e menor ou igual, respectivamente*)
  - **>** **e** **>=** (*maior e maior ou igual, respectivamente*)
  - **<>** **e** **=** (*diferente e igual, respectivamente*)
  - **LIKE** (*especifica um padrão de comparação*)
  - **BETWEEN** (*especifica um intervalo de valores*)

SELECT

- Exemplo4 (Menor que): Obtenha o nome dos projetos com custo menor que 28000

```
SELECT PNome
FROM Projeto
WHERE (PCusto < 28000);
```

PNome
Alfa
Sea
Paraiso

Operadores

- Operadores Lógicos
  - **AND**
  - **OR**
  - **NOT**

SELECT

- Exemplo5 (AND): Obtenha o nome das peças de cor vermelha e com preço maior que 25

PeNome
Painel

SELECT PeNome

FROM Peca

WHERE (PeCor = 'Vermelho') **AND** (PePreco > 25)**;**

SELECT

- Observe, no exemplo anterior, que só serão retornados os nomes das peças que obedecerem às duas condições da cláusula **WHERE**, já que tem-se o operador **AND**
- Ou seja, as duas condições devem ser verdadeiras

SELECT

- Exemplo6 (ORDER BY): Obtenha, em ordem decrescente de preço, o nome das peças de cor vermelha e com preço maior que 15.

```
SELECT PeNome
FROM Peca
WHERE (PeCor = 'Vermelho') AND (PePreco > 15)
ORDER BY PePreco DESC;
```

PeNome
Painel
Volante

SELECT

- Observe, no exemplo anterior, que a cláusula **ORDER BY** traz o resultado da consulta ordenado, em **ordem decrescente**, pelo preço da peça

SELECT

- Operadores Conjunturais
  - **= ANY e <> ANY**
  - **> ANY e >= ANY**
  - **< ANY e <= ANY**
  - **< ALL**

SELECT

- Operadores Conjunturais (continuação)
  - **> ALL**
  - **<> ALL**
  - **EXISTS**
  - **NOT EXISTS**
  - **IN**
  - **NOT IN**

SELECT

- Exemplo7 (IN): Obtenha, em ordem crescente de preço, o nome das peças de cor vermelha OU amarela E com preço de 9, 18, 22, 40 ou 90

PeNome
Limpador
Volante

```
SELECT PeNome
FROM Peca
WHERE ((PeCor = 'Vermelho') OR (PeCor = 'Amarelo'))
AND (PePreco IN (09, 18, 22, 40, 90))
ORDER BY PePreco ASC;
```

SELECT

- Uma outra forma de fazer a consulta do exemplo7, **sem utilizar o operador IN** (que é igualdade para um conjunto de valores):

```
SELECT PeNome
FROM Peca
WHERE ((PeCor = 'Vermelho') OR (PeCor = 'Amarelo'))
AND ((PePreco = 09) OR (PePreco = 18) OR (PePreco = 22)
OR (PePreco = 40) OR (PePreco = 90))
ORDER BY PePreco ASC;
```

SELECT

- Exemplo8 (NOT IN): Obtenha o nome das peças cujo preço NÃO SEJA 9, 14 E nem 60

```
SELECT PeNome
FROM Peca
WHERE (PePreco NOT IN (09, 14, 60));
```

PeNome
Cinto
Volante
Painel

SELECT

- Uma outra forma de fazer a consulta do exemplo8, **sem utilizar o operador NOT IN** (que é desigualdade para um conjunto de valores):

```
SELECT PeNome
FROM Peca
WHERE (PePreco <> 09) AND (PePreco <> 14)
AND (PePreco <> 60);
```

Operadores

- Operadores Relacionais
  - **< e <=** (*menor e menor ou igual, respectivamente*)
  - **> e >=** (*maior e maior ou igual, respectivamente*)
  - **<> e =** (*diferente e igual, respectivamente*)
  - **LIKE** (*especifica um padrão de comparação*)
  - **BETWEEN** (*especifica um intervalo de valores*)

SELECT

- Exemplo9 (LIKE): Obtenha o nome dos fornecedores residentes em cidades iniciadas com a letra S

```
SELECT FNome
FROM Fornecedor
WHERE (FCidade LIKE 'S%');
```

FNome
CM
Equipament

SELECT

- Exemplo10 (NOT LIKE): Obtenha o nome dos fornecedores cujo nome do fornecedor NÃO se INICIE com a letra P, em ordem crescente do nome do fornecedor.

```
SELECT FNome
FROM Fornecedor
WHERE (FNome NOT LIKE 'P%')
ORDER BY FNome ASC;
```

FNome
CM
Equipament
Kirurgic

SELECT

- Exemplo11 (LIKE): Obtenha o nome das peças cuja quarta letra do nome da peça seja 't'.

```
SELECT PeNome
FROM Peca
WHERE (PeNome LIKE '___t%');
```

PeNome
Cinto
Lanterna

Observações sobre o LIKE:

- % *representa nenhum ou muitos caracteres;*
- _ *representa um único caracter.*

SELECT

- Exemplo12 (BETWEEN): Obtenha os nomes dos projetos com preço entre 20000 e 30000.

```
SELECT PNome
FROM Projeto
WHERE (PCusto BETWEEN 20000 AND 30000);
```

PNome
Alfa
Sea

SELECT

- Uma outra forma de fazer a consulta do exemplo12, **sem utilizar o operador BETWEEN**:

```
SELECT PNome
FROM Projeto
WHERE (PCusto >= 20000) AND (PCusto <=30000);
```

SELECT

- Exemplo13 (IS NULL): Obtenha o nome dos projetos que estão sem valor para duração, ou seja, a duração está nula.

```
SELECT PNome
FROM Projeto
WHERE (PDuracao IS NULL);
```

Observe que o resultado desta consulta NÃO traria nenhum projeto, pois todos projetos possuem um valor para o atributo PDuracao

SELECT

- Exemplo14 (IS NOT NULL): Obtenha o nome dos projetos que tem um valor para duração, ou seja, a duração não está nula.

```
SELECT PNome
FROM Projeto
WHERE (PDuracao IS NOT NULL);
```

Observe que o resultado desta consulta retorna TODOS os projetos, pois todos os projetos possuem um valor para o atributo PDuracao

Obrigado

Elisângela Botelho Gracias
elisangela.botelho@mackenzie.br