Universidade Presbiteriana Mackenzie

Banco de Dados – Aula 15

Linguagem SQL - SELECT com utilização de várias tabelas

Profa. Elisângela Botelho Gracias

Faculdade de Computação e Informática

SELECT

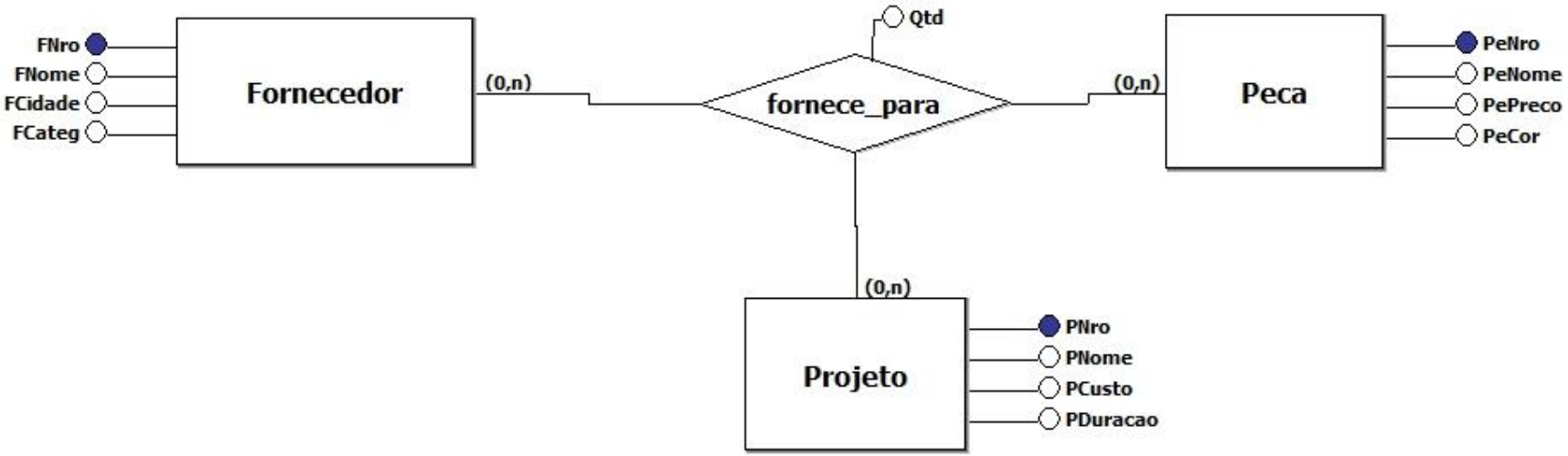
- Até o momento trabalhamos com a recuperação de dados de apenas uma única tabela, mas o conceito de banco de dados reúne várias tabelas relacionadas
- Então, muitas vezes, necessitaremos acessar dados de várias tabelas simultaneamente

SELECT

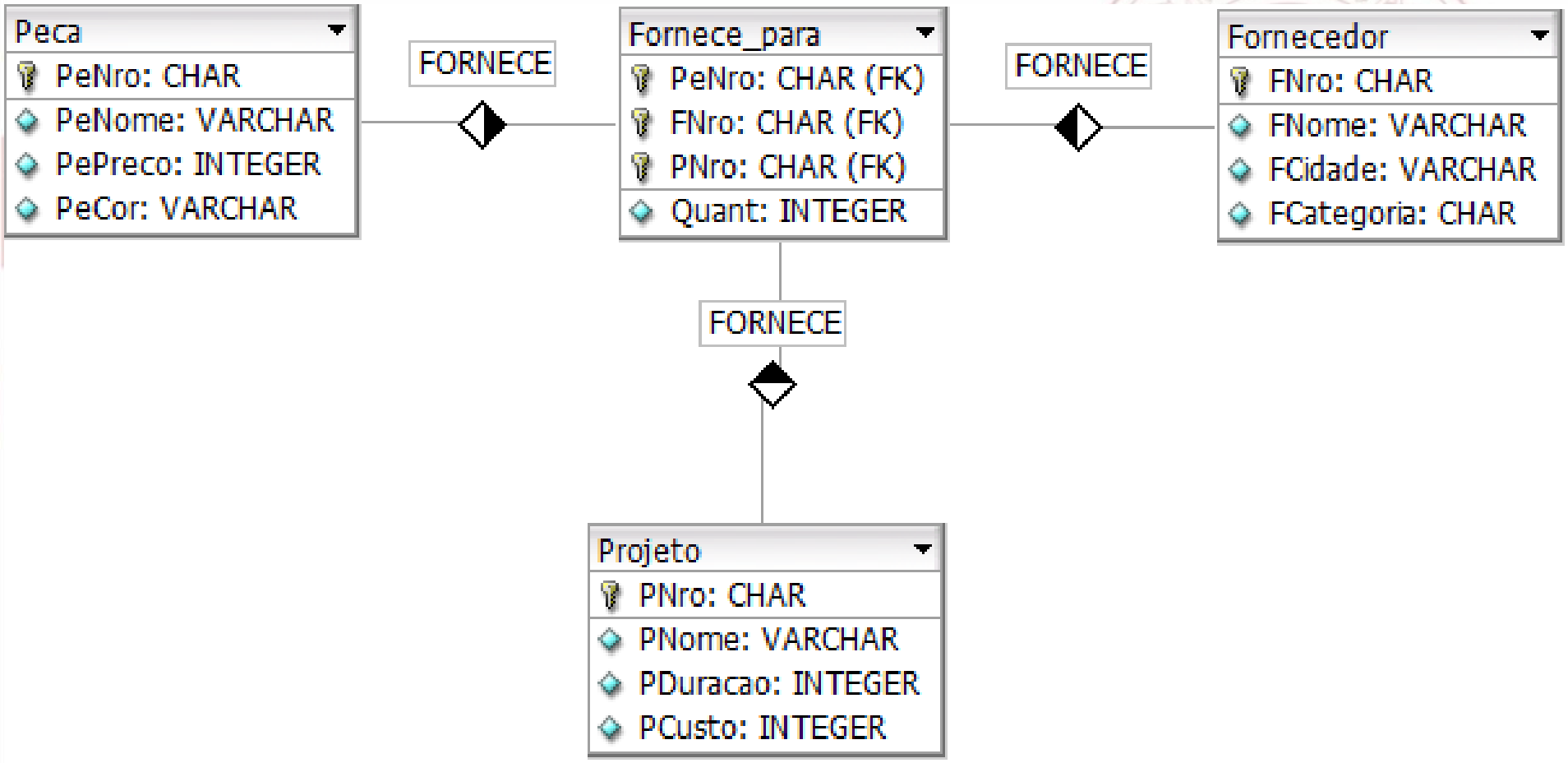
- Para trabalhar com dados de várias tabelas simultaneamente utilizamos o conceito de *join*, ou seja, **junção de tabelas que se relacionam**
- Uma das formas de trabalhar com junção de 2 ou mais tabelas é colocando na cláusula WHERE a **condição de junção entre as tabelas**

Banco de Dados Exemplo

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**

7

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

- Para fazer a **junção completa entre as tabelas Fornece_Para e Peca**, utiliza-se o seguinte comando:

```
SELECT *
FROM Fornece_para, Peca
WHERE (Fornece_para.PeNro = Peca.PeNro);
```

SELECT

- Para fazer a **junção completa entre as tabelas Fornece_Para e Peca**, utiliza-se o seguinte comando:

```
SELECT *
FROM Fornece_para, Peca
WHERE (Fornece_para.PeNro = Peca.PeNro)
```

Esta é a condição de junção entre as tabelas Fornece_para e Peca

SELECT

- Para a consulta anterior foi necessária a junção de duas tabelas – **Peca e Fornece_para,** portanto:
  - foi inserida na cláusula WHERE a **condição de junção** entre elas - **Fornece_para.PeNro = Peca.PeNro**, que combina duas tuplas, uma de Fornece_para e outra de Peca, sempre que o **valor do atributo PeNro da tabela Fornece_para for igual ao valor do atributo PeNro da tabela Peca**

SELECT

- A **condição de junção** entre tabelas combina:
  - **chave primária** de uma tabela com **chave estrangeira** da outra tabela
- A **junção de *n* tabelas** em um único SELECT obriga à colocação de, pelo menos, ***n-1*** **condições de junção**

SELECT

- Se **MAIS de UMA tabela** for especificada na cláusula FROM e **não for inserida a condição de junção entre elas**, gera-se o **PRODUTO CARTESIANO:**
  - o PRODUTO CARTESIANO, como na teoria de conjuntos, gera **todas as COMBINAÇÕES POSSÍVEIS entre os elementos das tabelas** que estão na cláusula FROM
  - neste caso, não serão respeitados os relacionamentos existentes entre as tabelas

SELECT

- Um **mesmo nome de atributo** pode ser utilizado em **tabelas diferentes**, portanto:
  - quando uma **consulta envolver duas ou mais tabelas e fizer referência a dois ou mais atributos com o mesmo nome, é preciso dizer de qual tabela ele pertence**
- Isso é feito **prefixando o nome da tabela ao nome do atributo** e separando os dois por um ponto – **nome_tabela.nome_atributo**

SELECT

- **Observe o resultado do SELECT anterior**

PeNro	FNro	PNro	Quant	PeNro	PeNome	PePreco	PeCor
PE1	F5	P4	5	PE1	Cinto	22	Azul
PE2	F2	P2	1	PE2	Volante	18	Vermelho
PE2	F2	P3	1	PE2	Volante	18	Vermelho
PE3	F3	P4	2	PE3	Lanterna	14	Preto
PE4	F3	P5	2	PE4	Limpador	9	Amarelo
PE4	F4	P5	3	PE4	Limpador	9	Amarelo
PE5	F1	P1	1	PE5	Painel	43	Vermelho

SELECT

- **Será que vamos querer fazer sempre a junção COMPLETA entre as tabelas?**
  - Na grande maioria das vezes NÃO
  - Vamos precisar fazer junção SIM, mas iremos inserir restrições de linhas e colunas

SELECT

- **Exemplo1** (junção): Obtenha o nome das peças utilizadas no projeto P4

```
SELECT Peca.PeNome
FROM Fornece_para, Peca
WHERE (Fornece_para.PNro = 'P4')
AND (Fornece_para.PeNro = Peca.PeNro);
```

PeNome
Cinto
Lanterna

SELECT

- **Exemplo1** (junção): Obtenha o nome das peças utilizadas no projeto P4

```
SELECT Peca.PeNome
FROM Fornece_para, Peca
WHERE (Fornece_para.PNr
AND (Fornece_para.PeNro = Peca.PeNro);
```

Esta é a condição de junção entre as tabelas Fornece_para e Peca

SELECT

- A condição da cláusula WHERE - **Fornece_para.PNro = 'P4'** - é uma **condição de seleção** que busca na tabela Fornece_Para somente as tuplas (registros) onde os **valores para o atributo PNro = 'P4'**
- Quando tem-se condições de seleção, a **condição de junção deve, obrigatoriamente, ser precedida pelo operador AND**, já que essa **condição de junção deve ser verdadeira**

SELECT

- Exemplo2 (junção): Obtenha o nome dos fornecedores que forneceram peças para o projeto P5

```
SELECT F.FNome
FROM Fornece_para FP, Fornecedor F
WHERE (FP.PNro = 'P5')
AND (FP.FNro = F.FNro)
ORDER BY  FNome ASC;
```

FNome
Kirurgic
Piloto

SELECT

- Exemplo2 (junção): Obtenha o nome dos fornecedores que forneceram peça

```
SELECT F.FNome
FROM Fornece_para
WHERE (FP.PNro = 'P3')
AND (FP.FNro = F.FNro)
ORDER BY  FNome ASC;
```

Esta é a condição de junção entre as tabelas Fornece_para e Fornecedor

SELECT

- Observe que foram dados **apelidos às tabelas** da consulta anterior, ou seja, a tabela **Func_Proj tem o apelido FP** e a tabela **Fornecedor tem o apelido F**
- Dessa forma, quando for referenciar um atributo de uma tabela com apelido, deve-se utilizar **apelido_tabela.nome_atributo**

SELECT

- Observe que no exemplo2:
  - a **condição de seleção** é **FP.PNro = 'P5'**
  - a **condição de junção**, que vem precedida pelo operador **AND**, é **FP.FNro = F.FNro**

SELECT

- Exemplo3 (junção): Obtenha o nome das peças fornecidas para o projeto P5 pelo fornecedor F4

```
SELECT Pe.PeNome
FROM Peca Pe, Fornece_para FP
WHERE ((FP.PNro = 'P5') AND (FP.FNro = 'F4'))
AND (FP.PeNro = Pe.PeNro);
```

PeNome
Limpador

SELECT

- Exemplo3 (junção): Obtenha o nome das peças fornecidas para o projeto P5 pelo fornecedor F4

```
SELECT Pe.PeNome
FROM Peca Pe, Fornec
WHERE ((FP.PNro = 'P5
AND (FP.PeNro = Pe.PeNro);
```

Esta é a condição de junção entre as tabelas Fornece_para e Peca

SELECT

- Observe que no exemplo3:
  - temos duas **condições de seleção -** Fornece_para.PNro = 'P5' **AND** Fornece_para.FNro = 'F4‘
  - e a **condição de junção**, que vem precedida pelo operador **AND**, é **FP.PeNro = Pe.PeNro**

SELECT

- Exemplo4 (junção): Obtenha o nome dos projetos que utilizaram a peça 'PE2' do fornecedor 'F2'

```
SELECT P.PNome
FROM Fornece_para FP, Projeto P
WHERE ((FP.PeNro = 'PE2') AND (FP.FNro = 'F2'))
AND (FP.PNro = P.PNro);
```

PNome
Pegasus
Alfa

SELECT

- Exemplo5 (junção com função agregada): Obtenha a quantidade total da peça 'Limpador' que foi utilizada em todos os projetos.

```
SELECT SUM(FP.Quant) AS Total

FROM Peca Pe, Fornece_para FP

WHERE (Pe.PeNome = 'Limpador')

AND (FP.PeNro = Pe.PeNro);
```

Total
5

SELECT

- Exemplo6 (junção com agrupamento): Obtenha o nome de cada peça e a quantidade total de cada uma delas utilizada em todos os projetos.

```
SELECT Pe.PeNome, SUM(FP.Quant) AS Soma
FROM Peca Pe, Fornece_para FP
WHERE  (FP.PeNro = Pe.PeNro)
GROUP BY Pe.PeNome
ORDER BY SUM(FP.Quant) DESC,
                Pe.PeNome ASC;
```

PeNome	Soma
Cinto	5
Limpador	5
Lanterna	2
Volante	2
Painel	1

SELECT

- Exemplo6 (junção com agrupamento): Obtenha o nome de cada peça e a quantidade total de cada uma delas utilizada em todos os projetos.

```
SELECT Pe.PeNome, SUM(FP.Quant) AS Soma
FROM Peca Pe, Fornece_para FP
WHERE (FP.PeNro = Pe.PeNro)
GROUP BY Pe.PeNome
ORDER BY SUM(FP.Quant) DESC,
         Pe.PeNome ASC;
```

PeNome	Soma
Cinto	5
Limpador	5
Lanterna	2
Volante	2
Painel	1

SELECT

- Observe, nos dois exemplos anteriores, que:
  - pode-se utilizar em uma única consulta, com junção de tabelas, **funções agregadas, GROUP BY/HAVING, DISTINCT, etc**.

SELECT

- Exemplo7 (junção com 3 tabelas): Obtenha o nome dos fornecedores que trabalharam no projeto 'Pegasus'.

```
SELECT F.FNome
FROM Projeto P, Fornece_para FP, Fornecedor F
WHERE (P.PNome = 'Pegasus')
AND (P.PNro = FP.PNro)
AND (FP.FNro = F.FNro);
```

FNome
CM

SELECT

- Exemplo7 (junção com 3 tabelas): Obtenha o nome dos fornecedores que trabalharam no projeto 'Pegasus'.

```
SELECT F.FNome
FROM Projeto P, Fornece_
WHERE (P.PNome = 'Pegasus')
AND (P.PNro = FP.PNro)
AND (FP.FNro = F.FNro);
```

Nesta consulta, com 3 tabelas, temos **DUAS condições de junção**

SELECT

- Observe, no exemplo7, que foi necessária a junção de três tabelas, portanto, tem-se duas **condições de junção:**
  - **P.PNro = FP.PNro**
  - **FP.FNro = F.FNro**
- Se pelo menos uma das condições de junção acima não aparecer na cláusula WHERE, o **produto cartesiano** irá ocorrer

SELECT

- Para este próximo exemplo, considere que a tabela **Fornecedor foi alterada**
- Ou seja, agora a tabela **Fornecedor se relaciona com ela mesma**, ou seja, todo fornecedor tem um gerente que é um fornecedor também.

FORNECEDOR = {FNro, FNome, FCidade, FCateg, FNro_Gerente}

  - **FNro_Gerente é chave estrangeira que referencia FNro da tabela Fornecedor**

SELECT

- Exemplo8 (junção com a mesma tabela): Obtenha o nome de cada fornecedor e o nome do seu gerente.

```
SELECT F.FNome, G.FNome
FROM Fornecedor F, Fornecedor G
WHERE (F.FNro_Gerente = G.FNro);
```

SELECT

- Exemplo8 (junção com a mesma tabela): Obtenha o nome de cada fornecedor e o nome do seu gerente.

```
SELECT F.FNome, G.FNome
FROM Fornecedor F, Fornecedor G
WHERE (F.FNro_Gerente = G.FNro);
```

SELECT

- O tipo de junção utilizado anteriormente é necessário quando uma **tabela está se relacionando com ela própria – auto-relacionamento**
- Neste caso, **colocou-se duas vezes a mesma tabela logo após o FROM**. Por isso, deve-se, obrigatoriamente, criar um apelido para cada uma delas, a fim de evitar ambiguidade que vai existir em todas as referências aos atributos desta tabela
- Ou seja, quando deseja-se referir ao **fornecedor** utiliza o **apelido F**, e quando ao **gerente do fornecedor**, utiliza-se **G**

Obrigado

Elisângela Botelho Gracias
elisangela.botelho@mackenzie.br