SHIFT

FIAP


MASTERING C#
CODE EVERYTHING
SHIFT
FIAP

BANCO **DE DADOS**

BANCO DE DADOS

1	**AULA 1**	CONTEÚDO – Introdução ao Contexto de Dados e a Linguagem SQL
2	**AULA 2**	CONTEÚDO – Criação de Objetos (DDL)
3	**AULA 3**	CONTEÚDO – Alteração e Exclusão de Objetos (DDL)
4	**AULA 4**	CONTEÚDO – Inserção, Atualização e Exclusão de Dados (DML)

BANCO DE DADOS

5	AULA 5	CONTEÚDO – Seleção e Manipulação de Dados (DML)
6	AULA 6	CONTEÚDO – Seleção e Junção de Dados (JOIN)
7	AULA 7	CONTEÚDO – Funções e Cálculos
8	AULA 8	CONTEÚDO – Funções e Agrupamentos

AULA 5

SELEÇÃO E JUNÇÃO DE DADOS **(JOIN)**

Introdução a Junções (JOIN)

Condições de JOIN

Tipos de JOIN

CROSS JOIN

Exercícios

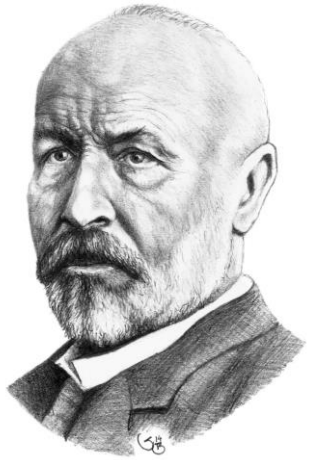
INTRODUÇÃO A **JUNÇÕES (JOIN)**

TEORIA
DOS CONJUTOS

- Estudo iniciado por Georg Cantor (Matemático Alemão, 1845 - 1918).
- Ramo da matemática que estuda conjuntos, ou seja, uma coleção de elementos.
- União
  - A {1, 2, 3} e B {4, 5} = {1, 2, 3,4, 5}
- Intersecção
  - A {1, 2, 3, 4, 5} e B {1, 3, 5, 7, 9} = {1, 3, 5}

TEORIA
DOS CONJUTOS

- Diferença
  - A {1, 2, 3, 4, 5} e B {1, 3, 5, 7, 9} = {2, 4} e {7, 9}
- Produto Cartesiano (Conjunto de todos os Pares Ordenados)
  - A {1, 2, 3} e B {x, y} = {1x, 1y, 2x, 2y, 3x, 3y}



TEORIA
DOS CONJUTOS

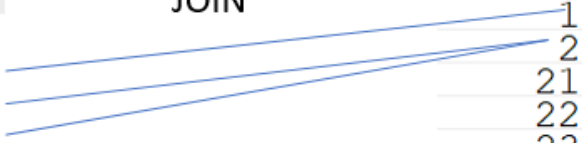
- Desenvolvido por John Venn (Matemático Inglês, 1834 - 1923).
- Diagramas utilizados para representar, de forma gráfica, as relações entre conjuntos e seus elementos.

TEORIA
DOS CONJUTOS

LEFT

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
178	Grant	(null)
200	Whalen	10
201	Hartstein	20
202	Fay	20

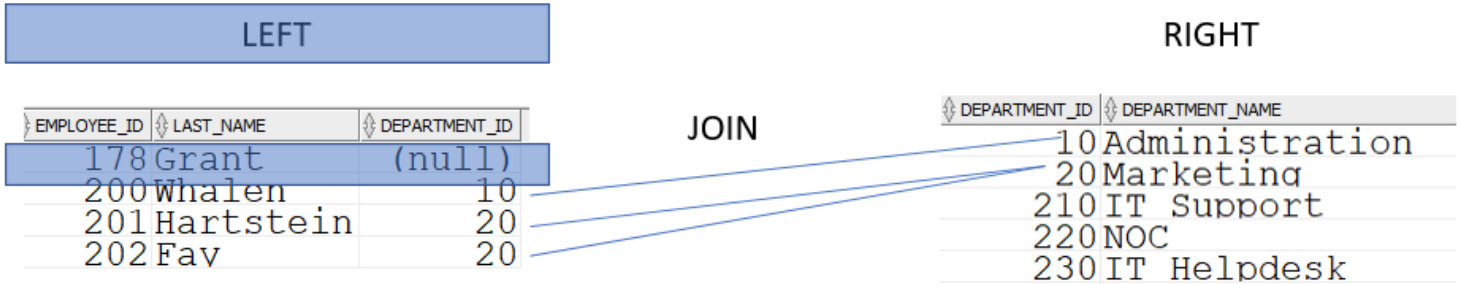
JOIN



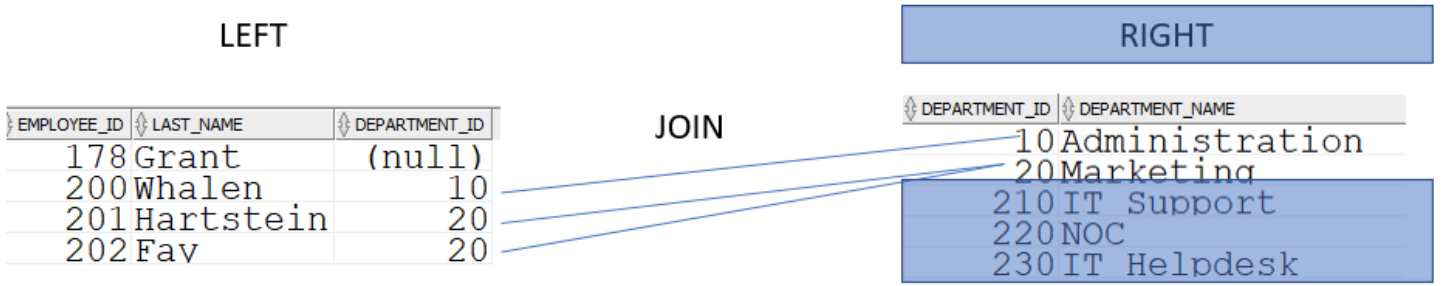
RIGHT

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
210	IT Support
220	NOC
230	IT Helpdesk

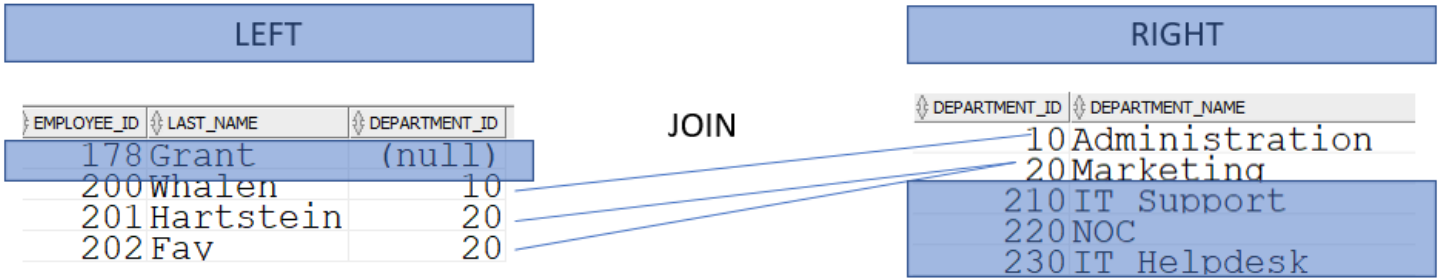
TEORIA
DOS CONJUTOS



TEORIA
DOS CONJUTOS



TEORIA
DOS CONJUTOS



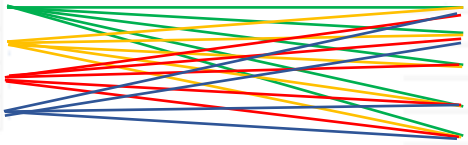
TEORIA
DOS CONJUTOS

LEFT

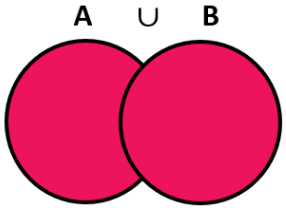
EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
178	Grant	(null)
200	Whalen	10
201	Hartstein	20
202	Fay	20

RIGHT

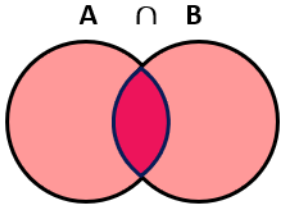
DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
210	IT Support
220	NOC
230	IT Helpdesk



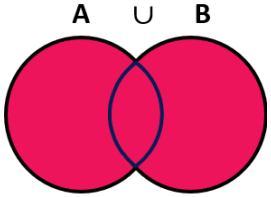
TEORIA
DOS CONJUTOS



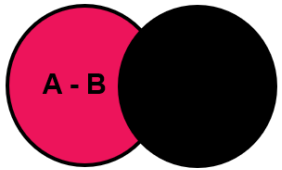

```
SELECT employee_id, job_id
FROM   employees
UNION
SELECT employee_id, job_id
FROM   job_history;
```




```
SELECT employee_id, job_id
FROM   employees
INTERSECT
SELECT employee_id, job_id
FROM   job_history;
```




```
SELECT employee_id, job_id
FROM   employees
UNION ALL
SELECT employee_id, job_id
FROM   job_history;
```




```
SELECT employee_id, job_id
FROM   employees
MINUS
SELECT employee_id, job_id
FROM   job_history;
```

JUNÇÕES (JOIN)

- A função é utilizada quando há a necessidade de **recuperar** informações que estão distribuídas em **mais de uma tabela**.
- Mais **conhecida como JOIN,** é usada para **combinar registros**, de **diferentes tabelas,** e **que estão relacionados**.
- O critério de relacionamento estabelecido pode ser baseado em junções idênticas (equijoins), não idênticas (non-equijoins), internas (inner join), externas (left join, right join ou full join) ou autojunções (self-join).

FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

CONDIÇÕES **DE JOIN**

CONDIÇÕES DE JOIN

- Equijoin (JOIN)
  - Operador de Igualdade (=);
  - Uma junção idêntica pode ser estabelecida entre tabelas que possuem chave primária e estrangeira correspondente.

FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

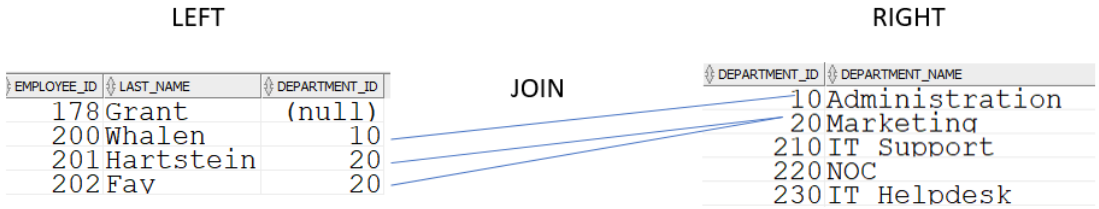
CONDIÇÕES DE JOIN

- Non-Equijoin
  - Operador que não é de Igualdade (<, >, BETWEEN, etc);
  - A junção não idêntica é utilizada para obter dados de tabelas que não possuem relacionamentos preestabelecidos. Esse tipo de junção não requer a comparação entre chaves primária e estrangeira, uma vez que é feita ao comparar as faixas de valores, por exemplo.

FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

EXEMPLO: EQUIJOIN (=)

- SELECT para mostrar os nomes dos funcionários e seus respectivos departamentos.




```
SELECT e.last_name, d.department_name
FROM   employees e INNER JOIN departments d
ON     e.department_id = d.department_id;
```

FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

EXEMPLO: NON-EQUIJOIN
(BETWEEN, <, >...)

- SELECT para mostrar a grade salarial que cada funcionário pertence.

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e JOIN job_grades j
ON     e.salary BETWEEN j.lowest_sal AND j.highest_sal;
```

FONTE:https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

TIPOS **DE JOIN**

TIPO DE JOIN

Internas (INNER)

- A junção interna é utilizada para **recuperar somente** as **linhas** de uma equijunção onde existam **registros correspondentes** entre as **tabelas.**

Externas (LEFT, RIGHT e FULL)

- A junção externa é **utilizada para recuperar** as **linhas de uma equijunção**, **sendo** que, em **alguma** das **tabelas**, pode **não haver registro** correspondente.

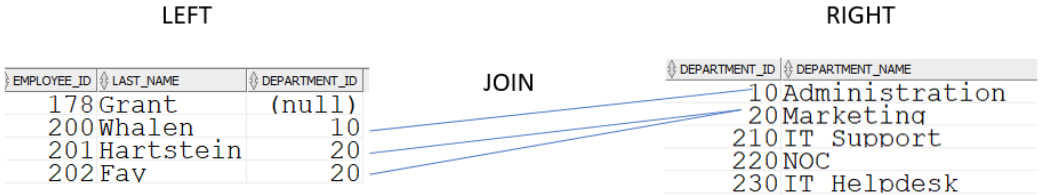
Auto-junções (SELF-JOIN)

- A auto-junção é uma operação **de junção entre colunas da mesma tabela**.

FONTE:https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

EXEMPLO: INNER JOIN

- Neste tipo de junção, a consulta retornará todas as linhas que possuam registros relacionados entre as tabelas envolvidas.
- SELECT para mostrar os nomes dos funcionários e seus respectivos departamentos.




```
SELECT e.last_name, d.department_name
FROM   employees e INNER JOIN departments d
ON     e.department_id = d.department_id;
```

FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

TIPOS DE JOIN:
OUTER JOIN

- Há três tipos de junções externas:

LEFT JOIN: junção externa esquerda recupera todas as linhas da equijunção, além das que não possuem correspondentes na tabela à esquerda da operação.

RIGHT JOIN: junção externa direita recupera todas as linhas da equijunção, além das que não possuem correspondentes na tabela à direita da operação.

FULL JOIN: junção externa completa recupera todas as linhas da equijunção, além das que não possuem correspondentes na tabela à direta e à esquerda da operação.

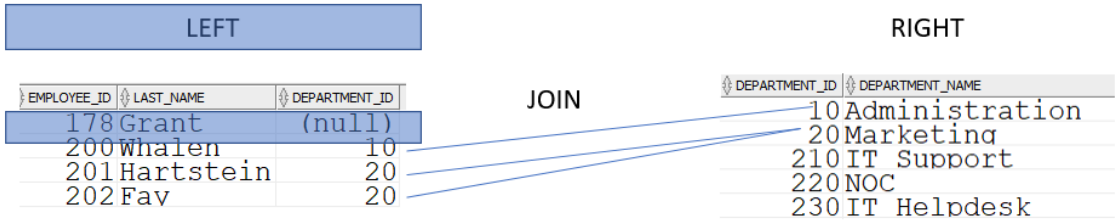
FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

EXEMPLO:
LEFT JOIN

- Todas as linhas da tabela à esquerda serão recuperadas, independentemente da existência de ocorrências relacionadas na tabela da direita.
- Preserva as linhas sem correspondência da primeira tabela (esquerda), juntando-as com uma linha nula na forma da segunda tabela (direita).

FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

EXEMPLO:
LEFT JOIN




```
SELECT e.last_name, d.department_name
FROM   employees e LEFT OUTER JOIN departments d
ON     e.department_id = d.department_id;

SELECT e.last_name, d.department_name
FROM   employees e LEFT OUTER JOIN departments d
ON     e.department_id = d.department_id
WHERE  d.department_id is null;
```

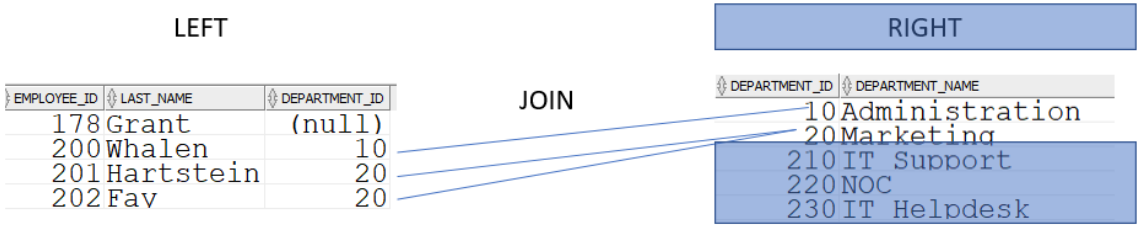
FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

EXEMPLO:
RIGHT JOIN

- Todas as linhas da tabela à direita serão recuperadas, independentemente da existência de ocorrências relacionadas na tabela da esquerda.
- Preserva as linhas sem correspondência da segunda tabela (direita), juntando-as com uma linha nula na forma da primeira tabela (esquerda).

FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

EXEMPLO:
RIGHT JOIN




```
SELECT e.last_name, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     e.department_id=d.department_id;

SELECT e.last_name, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     e.department_id = d.department_id
WHERE  e.department_id is null;
```

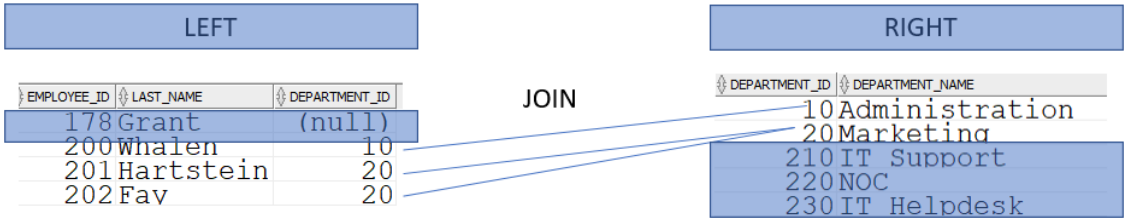
FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

EXEMPLO:
FULL JOIN

- Todas as linhas da tabela da direita e da esquerda serão recuperadas, independentemente da existência de ocorrências relacionadas na tabela da esquerda ou da direita.
- Para um melhor entendimento, o FULL JOIN retorna o resultado do LEFT e do RIGHT ao mesmo tempo.

FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

EXEMPLO:
FULL JOIN




```
SELECT e.last_name, d.department_name
FROM   employees e FULL OUTER JOIN departments d
ON     e.department_id=d.department_id;
```

FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

AUTO - JUNÇÃO
(SELF-JOIN)

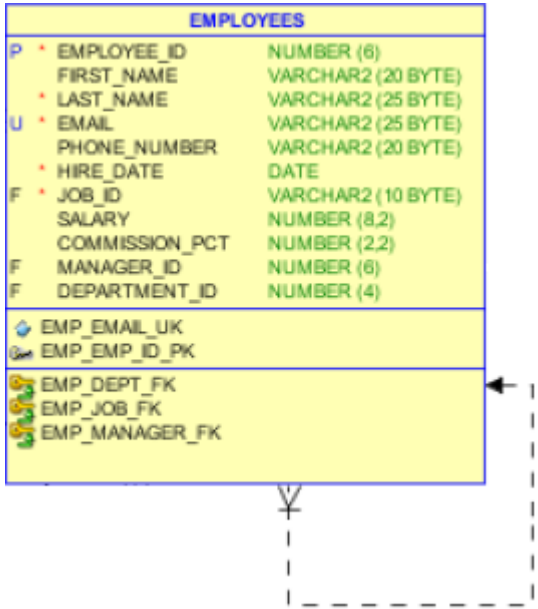
- As junções do tipo SELF-JOIN são **utilizadas em** casos em que há, no modelo de dados, a figura do **autorrelacionamento** (relacionamento recursivo);
- Nesse caso, é preciso **acessar a mesma tabela duas vezes**, uma para recuperar os registros e outra para buscar os dados relacionados (autorrelacionamento).

OBS: Para o SELF-JOIN é **necessário** o **uso de apelido** nas tabelas e nos nomes das colunas.

FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

EXEMPLO
(SELF-JOIN)

- SELECT para exibir o código e o nome do funcionário e o código e o nome do gerente.



FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

EXEMPLO
(SELF-JOIN)

```
SELECT m.last_name manager, m.job_id
manager_job,
       w.last_name worker,  w.job_id worker_job
FROM   employees w join employees m
ON    (m.employee_id = w.manager_id)
order by m.department_id,m.job_id ;
```

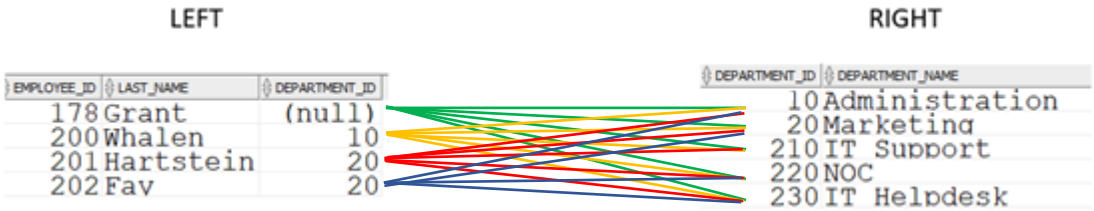
FONTE: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Joins.html#

CROSS JOIN

PRODUTO
CARTESIADO

- Criado por René Descartes (1596 – 1650), filósofo, físico e matemático francês.
- O **produto cartesiano**, também conhecido como produto direto, é o **conjunto de todos os pares ordenados de dois conjuntos**.
- Em banco de dados, um produto cartesiano é obtido após o **relacionamento entre duas tabelas sem um critério de ligação**.
- A quantidade de linhas resultantes de um produto cartesiano é exatamente o produto entre a quantidade de registros das tabelas envolvidas.

CROSS JOIN
(PRODUTO CARTESIADO)




```
SELECT e.last_name, d.department_name
FROM   employees e CROSS JOIN departments d;
```

DB_FUNCIONARIO X DB_DEPTO

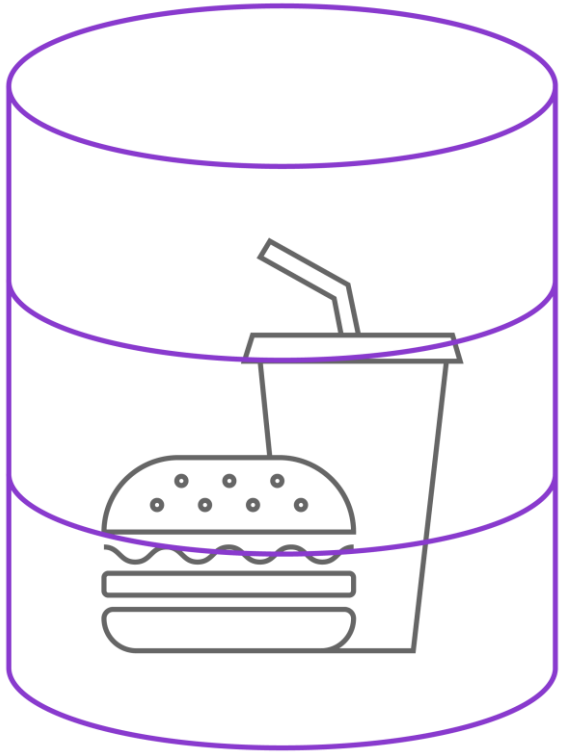
- As **linhas da primeira tabela** (de funcionários) foram **combinadas com** as linhas da **segunda tabela** (de departamentos), demonstrando um resultado indesejado.
- Veja que a tabela de departamentos possui 5 linhas e a de funcionários 6, o resultado foram **(6 * 5 = 30) 30 linhas** exibidas na **busca**.
- Existe a **necessidade** de colocar uma **"ligação" entre** as duas **tabelas**, associando a chave primária e estrangeira.
- **Normalmente**, esse tipo de resultado, um **produto cartesiano**, **não traz** qualquer **utilidade e é custoso para** o **banco** de dados.

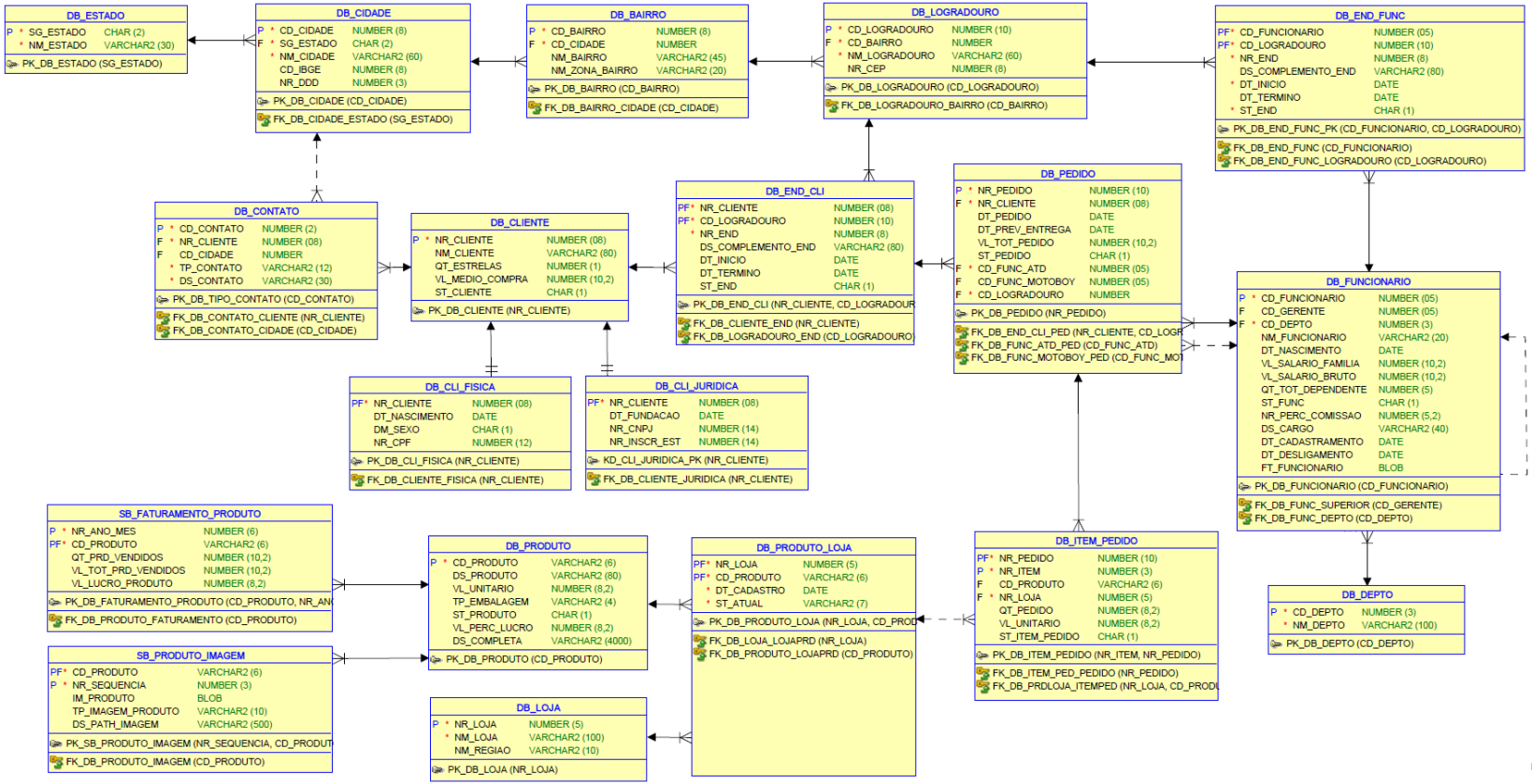
EXECÍCIOS

PROJETO:
DBURGER

Instruções para Iniciar o Projeto

1. Execute o bloco DDL para a criação da estrutura do Banco.
2. Execute o bloco DML para popular as tabelas criadas.
3. O MER (modelo entidade relacionamento) do projeto está no Slide seguinte (32).




DB_ESTADO
P * SG_ESTADO CHAR (2)
* NM_ESTADO VARCHAR2 (30)
PK_DB_ESTADO (SG_ESTADO)
DB_CIDADE
P * CD_CIDADE NUMBER (8)
F * SG_ESTADO CHAR (2)
* NM_CIDADE VARCHAR2 (60)
CD_IBGE NUMBER (8)
NR_DDD NUMBER (3)
PK_DB_CIDADE (CD_CIDADE)
FK_DB_CIDADE_ESTADO (SG_ESTADO)
DB_BAIRRO
P * CD_BAIRRO NUMBER (8)
F * CD_CIDADE NUMBER
NM_BAIRRO VARCHAR2 (45)
NM_ZONA_BAIRRO VARCHAR2 (20)
PK_DB_BAIRRO (CD_BAIRRO)
FK_DB_BAIRRO_CIDADE (CD_CIDADE)
DB_LOGRADOURO
P * CD_LOGRADOURO NUMBER (10)
F * CD_BAIRRO NUMBER
* NM_LOGRADOURO VARCHAR2 (60)
NR_CEP NUMBER (8)
PK_DB_LOGRADOURO (CD_LOGRADOURO)
FK_DB_LOGRADOURO_BAIRRO (CD_BAIRRO)
DB_END_FUNC
PF* CD_FUNCIONARIO NUMBER (05)
PF* CD_LOGRADOURO NUMBER (10)
* NR_END NUMBER (8)
DS_COMPLEMENTO_END VARCHAR2 (80)
* DT_INICIO DATE
DT_TERMINO DATE
* ST_END CHAR (1)
PK_DB_END_FUNC_PK (CD_FUNCIONARIO, CD_LOGRADOURO)
FK_DB_END_FUNC (CD_FUNCIONARIO)
FK_DB_END_FUNC_LOGRADOURO (CD_LOGRADOURO)
DB_CONTATO
P * CD_CONTATO NUMBER (2)
F * NR_CLIENTE NUMBER (08)
F CD_CIDADE NUMBER
* TP_CONTATO VARCHAR2 (12)
* DS_CONTATO VARCHAR2 (30)
PK_DB_TIPO_CONTATO (CD_CONTATO)
FK_DB_CONTATO_CLIENTE (NR_CLIENTE)
FK_DB_CONTATO_CIDADE (CD_CIDADE)
DB_CLIENTE
P * NR_CLIENTE NUMBER (08)
NM_CLIENTE VARCHAR2 (80)
QT_ESTRELAS NUMBER (1)
VL_MEDIO_COMPRA NUMBER (10,2)
ST_CLIENTE CHAR (1)
PK_DB_CLIENTE (NR_CLIENTE)
DB_END_CLI
PF* NR_CLIENTE NUMBER (08)
PF* CD_LOGRADOURO NUMBER (10)
* NR_END NUMBER (8)
DS_COMPLEMENTO_END VARCHAR2 (80)
DT_INICIO DATE
DT_TERMINO DATE
ST_END CHAR (1)
PK_DB_END_CLI (NR_CLIENTE, CD_LOGRADOUR
FK_DB_CLIENTE_END (NR_CLIENTE)
FK_DB_LOGRADOURO_END (CD_LOGRADOURO
DB_PEDIDO
P * NR_PEDIDO NUMBER (10)
F * NR_CLIENTE NUMBER (08)
DT_PEDIDO DATE
DT_PREV_ENTREGA DATE
VL_TOT_PEDIDO NUMBER (10,2)
ST_PEDIDO CHAR (1)
F * CD_FUNC_ATD NUMBER (05)
F CD_FUNC_MOTOBOY NUMBER (05)
F * CD_LOGRADOURO NUMBER
PK_DB_PEDIDO (NR_PEDIDO)
FK_DB_END_CLI_PED (NR_CLIENTE, CD_LOGR
FK_DB_FUNC_ATD_PED (CD_FUNC_ATD)
FK_DB_FUNC_MOTOBOY_PED (CD_FUNC_MO
DB_FUNCIONARIO
P * CD_FUNCIONARIO NUMBER (05)
F CD_GERENTE NUMBER (05)
F * CD_DEPTO NUMBER (3)
NM_FUNCIONARIO VARCHAR2 (20)
DT_NASCIMENTO DATE
VL_SALARIO_FAMILIA NUMBER (10,2)
VL_SALARIO_BRUTO NUMBER (10,2)
QT_TOT_DEPENDENTE NUMBER (5)
ST_FUNC CHAR (1)
NR_PERC_COMISSAO NUMBER (5,2)
DS_CARGO VARCHAR2 (40)
DT_CADASTRAMENTO DATE
DT_DESLIGAMENTO DATE
FT_FUNCIONARIO BLOB
PK_DB_FUNCIONARIO (CD_FUNCIONARIO)
FK_DB_FUNC_SUPERIOR (CD_GERENTE)
FK_DB_FUNC_DEPTO (CD_DEPTO)
DB_CLI_FISICA
PF* NR_CLIENTE NUMBER (08)
DT_NASCIMENTO DATE
DM_SEXO CHAR (1)
NR_CPF NUMBER (12)
PK_DB_CLI_FISICA (NR_CLIENTE)
FK_DB_CLIENTE_FISICA (NR_CLIENTE)
DB_CLI_JURIDICA
PF* NR_CLIENTE NUMBER (08)
DT_FUNDACAO DATE
NR_CNPJ NUMBER (14)
NR_INSCR_EST NUMBER (14)
KD_CLI_JURIDICA_PK (NR_CLIENTE)
FK_DB_CLIENTE_JURIDICA (NR_CLIENTE)
SB_FATURAMENTO_PRODUTO
P * NR_ANO_MES NUMBER (6)
PF* CD_PRODUTO VARCHAR2 (6)
QT_PRD_VENDIDOS NUMBER (10,2)
VL_TOT_PRD_VENDIDOS NUMBER (10,2)
VL_LUCRO_PRODUTO NUMBER (8,2)
PK_DB_FATURAMENTO_PRODUTO (CD_PRODUTO, NR_ANO
FK_DB_PRODUTO_FATURAMENTO (CD_PRODUTO)
SB_PRODUTO_IMAGEM
PF* CD_PRODUTO VARCHAR2 (6)
P * NR_SEQUENCIA NUMBER (3)
IM_PRODUTO BLOB
TP_IMAGEM_PRODUTO VARCHAR2 (10)
DS_PATH_IMAGEM VARCHAR2 (500)
PK_SB_PRODUTO_IMAGEM (NR_SEQUENCIA, CD_PRODUT
FK_DB_PRODUTO_IMAGEM (CD_PRODUTO)
DB_PRODUTO
P * CD_PRODUTO VARCHAR2 (6)
DS_PRODUTO VARCHAR2 (80)
VL_UNITARIO NUMBER (8,2)
TP_EMBALAGEM VARCHAR2 (4)
ST_PRODUTO CHAR (1)
VL_PERC_LUCRO NUMBER (8,2)
DS_COMPLETA VARCHAR2 (4000)
PK_DB_PRODUTO (CD_PRODUTO)
DB_LOJA
P * NR_LOJA NUMBER (5)
* NM_LOJA VARCHAR2 (100)
NM_REGIAO VARCHAR2 (10)
PK_DB_LOJA (NR_LOJA)
DB_PRODUTO_LOJA
PF* NR_LOJA NUMBER (5)
PF* CD_PRODUTO VARCHAR2 (6)
* DT_CADASTRO DATE
* ST_ATUAL VARCHAR2 (7)
PK_DB_PRODUTO_LOJA (NR_LOJA, CD_PROD
FK_DB_LOJA_LOJAPRD (NR_LOJA)
FK_DB_PRODUTO_LOJAPRD (CD_PRODUTO)
DB_ITEM_PEDIDO
PF* NR_PEDIDO NUMBER (10)
P * NR_ITEM NUMBER (3)
F CD_PRODUTO VARCHAR2 (6)
F * NR_LOJA NUMBER (5)
QT_PEDIDO NUMBER (8,2)
VL_UNITARIO NUMBER (8,2)
ST_ITEM_PEDIDO CHAR (1)
PK_DB_ITEM_PEDIDO (NR_ITEM, NR_PEDIDO)
FK_DB_ITEM_PED_PEDIDO (NR_PEDIDO)
FK_DB_PRDLOJA_ITEMPED (NR_LOJA, CD_PROD
DB_DEPTO
P * CD_DEPTO NUMBER (3)
* NM_DEPTO VARCHAR2 (100)
PK_DB_DEPTO (CD_DEPTO)

CONSULTA
UTILIZANDO JOIN

1. Trazer todos os funcionários listando código, nome, data de nascimento e nome do departamento. Filtrar somente os moradores da Vila Mariana.

2. Selecionar todos os funcionários listando, código, nome e endereço completo. Filtrar somente os moradores da Saúde e que possuam 4 estrelas ou mais.

CONSULTA UTILIZANDO
JOIN E FILTROS

1. Trazer todos os clientes que possuem 4 ou mais estrelas.
2. Selecionar todos os clientes que possuem 3 estrelas ou mais e que tenham o valor médio de compra maior que R$ 100.
3. Trazer todos os clientes com o valor médio de compra acima de R$ 100, informando o nome e a quantidade de estrelas.
4. Listar todos os produtos que possuem um valor unitário maior que R$ 30.

DESAFIO

DESAFIO

1. Listar o valor total dos pedidos de 2019 por mês.

2. Listar o valor e a quantidade total dos pedidos de 2019 por mês.

BIBLIOGRAFIA
BÁSICA

- **TAYLOR, A.** SQL para Leigos. São Paulo: Alta Books, 2016.
- **PUGA, S.; FRANÇA, E; GOYA, M.** Banco de Dados: Implementação em SQL, PL/SQL e Oracle 11g. New Jersey: Pearson Universities, 2013.
- **ORACLE LIVE SQL.** Learn and share SQL: Running on Oracle Database 19c. [s.d.]. Disponível em: <https://livesql.oracle.com/apex/f?p=590:1000>. Acesso em: 02/12/2020.
- **ORACLE HELP CENTER.** Oracle Database 19c Get Started. [s.d.]. Disponível em: <https://docs.oracle.com/en/database/oracle/oracle-database/19/index.html>. Acesso em: 02/12/2020.

OBRIGADO

/alexandrebarcelos

FIAP

SHIFT

FIAP