

FIAP GRADUAÇÃO

# BUILDING RELATIONAL DATABASE

Prof. Diogo Alves  
profdiogo.alves@fiap.com.br

**COMANDOS DQL DRS (SQL)**  
**JUNÇÃO DE TABELAS \* JOIN**  
**PESQUISA EM MÚLTIPLAS TABELAS**

- Objetivo
- Conceitos referentes a linguagem de consulta/recuperação de dados
- Revisão dos Conceitos
- Exercícios

# Objetivos

- ❑ Aplicar os conceitos da linguagem SQL durante a implementação do banco de dados

# Conteúdo Programático referente a esta aula

- ☐ Linguagem para consulta e recuperação de dados
  - ☐ DRS DQL (SELECT)
    - ☐ Junções (consulta a duas ou mais tabelas)
  - ☐ Exercícios

# Linguagem SQL

**SQL: Structured Query Language**  
**(Linguagem Estruturada de Consulta)**

## JUNÇÕES - JOIN

# TEORIA DOS CONJUNTOS



Estudo iniciado por Georg Cantor (Matemático Alemão, 1845 - 1918);

Ramo da matemática que estuda conjuntos, ou seja, uma coleção de elementos;

- **União**

- $A \{1, 2, 3\} \text{ e } B \{4, 5\} = \{1, 2, 3, 4, 5\}$

- **Intersecção**

- $A \{1, 2, 3, 4, 5\} \text{ e } B \{1, 3, 5, 7, 9\} = \{1, 3, 5\}$

- **Diferença**

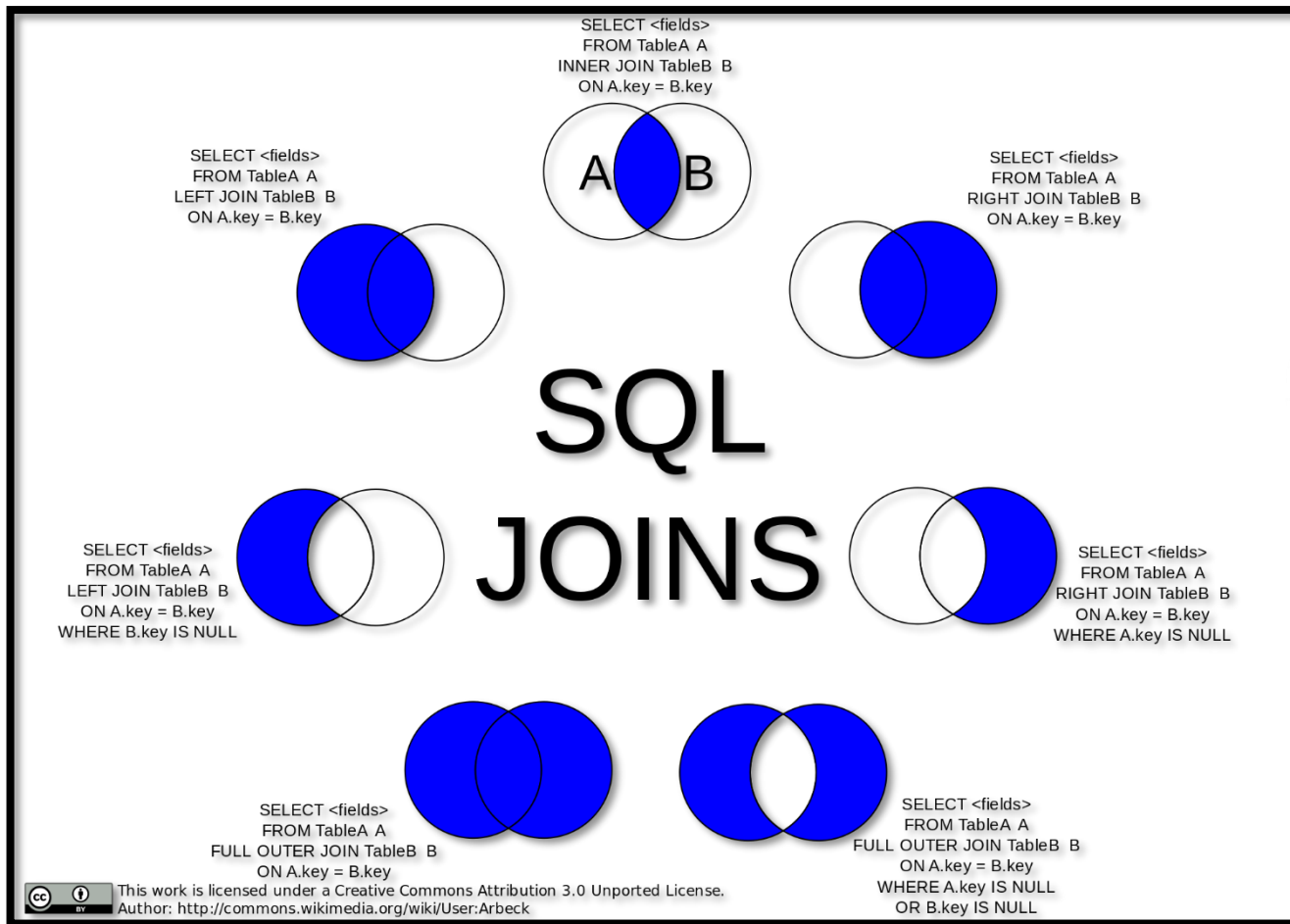
- $A \{1, 2, 3, 4, 5\} \text{ e } B \{1, 3, 5, 7, 9\} = \{2, 4\} \text{ e } \{7, 9\}$

- **Produto Cartesiano (Conjunto de todos os Pares Ordenados)**

- $A \{1, 2, 3\} \text{ e } B \{x, y\} = \{1x, 1y, 2x, 2y, 3x, 3y\}$



Diagramas utilizados para representar, de forma gráfica, as relações entre conjuntos e seus elementos.







## Consulta dos Dados utilizando a linguagem SQL

### Qualificadores de Nome

Consiste no nome da tabela seguido de um ponto e o nome da coluna da tabela, por exemplo:

```
SELECT  T_SIP_DEPARTAMENTO.NM_DEPARTAMENTO
        
        
        Nome da tabela      Nome do campo
FROM    T_SIP_DEPARTAMENTO;
```

Colocar a identificação (qualificador) é opcional, porém é uma prática recomendada para facilitar o entendimento do comando.

Porém quando estivermos recuperando colunas com mesmo nome, em tabelas diferentes, se faz necessário informar a tabela que se deseja recuperar a informação.

## Consulta dos Dados utilizando a linguagem SQL

### Pesquisa Básica em Tabelas – QUALIFICADORES DE NOME

#### Qualificadores de Nome – EXEMPLO



```
SELECT  T_SIP_FUNCIONARIO.NR_MATRICULA ,  
        T_SIP_FUNCIONARIO.CD_DEPTO ,  
        T_SIP_FUNCIONARIO.DT_ADMISSAO ,  
        T_SIP_FUNCIONARIO.VL_SALARIO  
FROM    T_SIP_FUNCIONARIO;
```

# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL Pesquisa Básica em Tabelas – ALIAS



### ALIAS (APELIDO) PARA TABELAS E QUALIFICADORES – EXEMPLO

Podemos utilizar apelidos para as tabelas, assim não teremos uma digitação redundante em relação aos nomes das tabelas.

-- APELIDOS PARA TABELAS E QUALIFICADORES

```
SELECT F.NR_MATRICULA ,  
       F.CD_DEPTO ,  
       F.DT_ADMISSAO ,  
       F.VL_SALARIO  
FROM T_SIP_FUNCIONARIO F;
```



# JUNÇÕES (JOIN)



- Utilizado quando há a necessidade de se **recuperar informações** que estão distribuída **em mais de uma tabela**;
- A operação de junção, 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).

# CONDIÇÕES DE JOIN



- **Equijoin (JOIN)**
  - Operador de Igualdade ( = );
  - Uma **junção** idêntica pode ser estabelecida **entre tabelas que possuem chave primária e estrangeira correspondente.**
- **No-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.

# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL



### Sintaxe Básica

```
SELECT tabela1.coluna, tabela2.coluna
```

```
FROM tabela1
```

```
[CROSS JOIN tabela2] |
```

```
[NATURAL JOIN tabela2] |
```

```
[JOIN tabela2 USING (nome_coluna)] |
```

```
[JOIN tabela2
```

```
ON(tabela1.nome_coluna = tabela2.nome_coluna)] |
```

```
[LEFT|RIGHT|FULL OUTER JOIN tabela2
```

```
ON (tabela1.nome_coluna = tabela2.nome_coluna)] ;
```

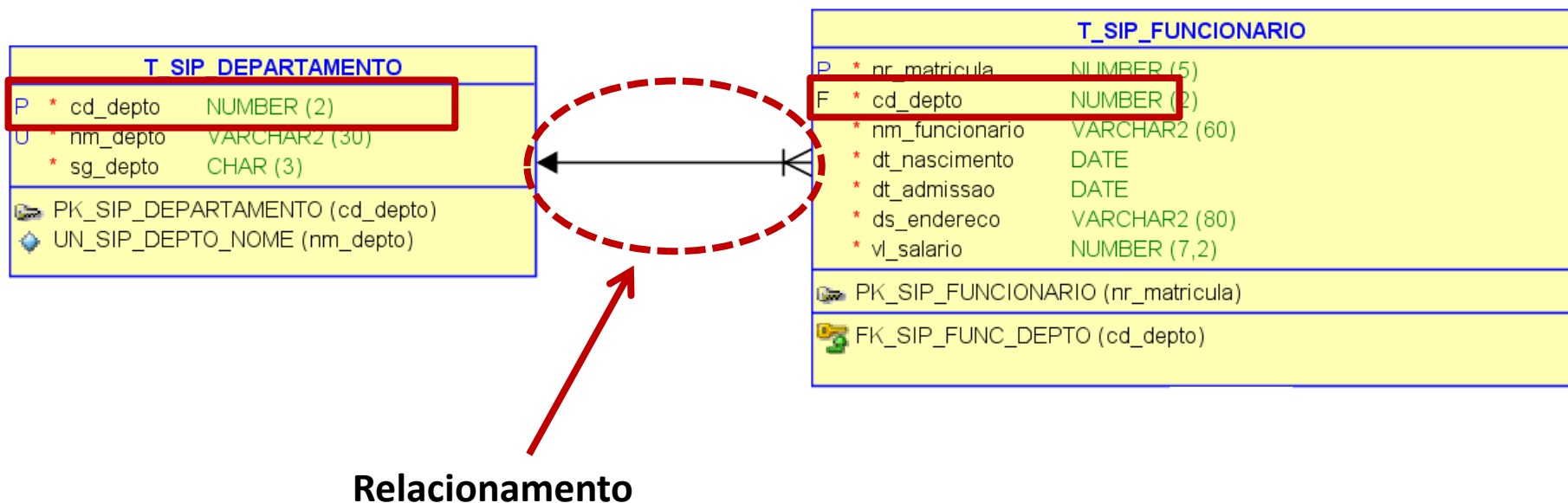
# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL



### Pesquisa em Múltiplas Tabelas – EQUIJOIN (Igualdade)

Para ilustrar, imagine que precisássemos recuperar o nome do departamento em que o funcionário está trabalhando, conforme o modelo abaixo:



# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL



### Pesquisa em Múltiplas Tabelas – EQUIJOIN (Igualdade)

Observe que o **nome do departamento** está armazenado na tabela **“DEPARTAMENTO”** que representa o cadastro dos departamentos.

Perceba que existe uma associação entre estas duas tabelas, observamos a **chave estrangeira “CD\_DEPTO”** na tabela **“FUNCIONARIO”**.



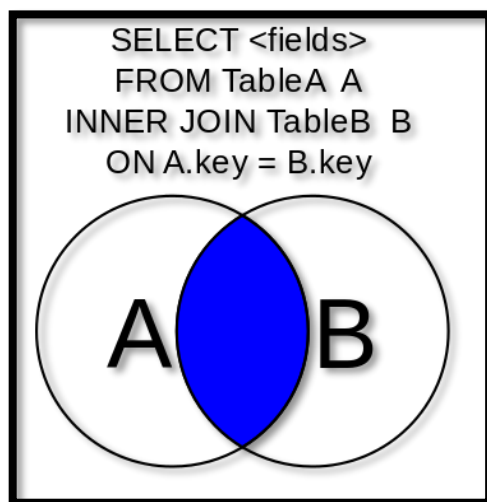
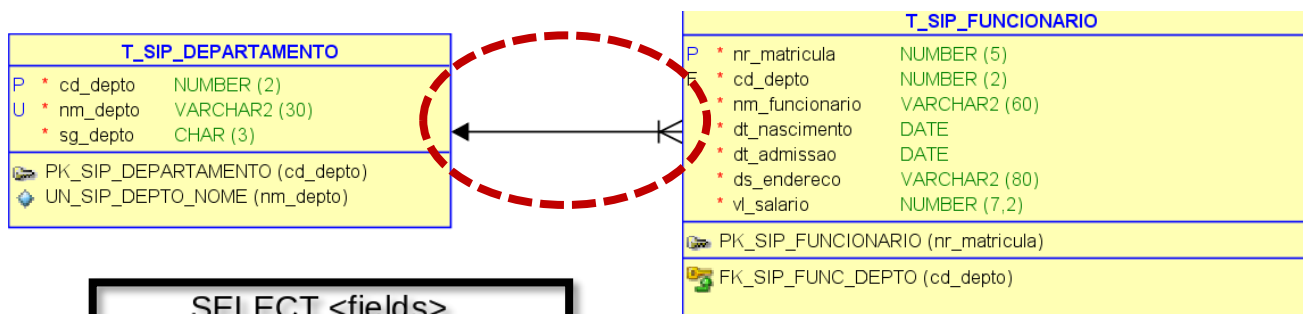
# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL



### Pesquisa em Múltiplas Tabelas – EQUIJOIN (Igualdade)

Para que possamos recuperar o nome do departamento para cada registro da tabela “FUNCIONARIO” precisaremos **combinar (JUNÇÃO)** estas duas tabela.



# Linguagem SQL

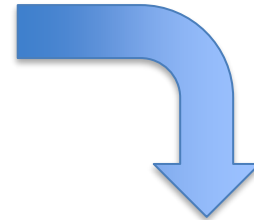
## Consulta dos Dados utilizando a linguagem SQL



### Pesquisa em Múltiplas Tabelas – EXEMPLO - EQUIJOIN

```
-- EXEMPLO - EQUIJOIN (IGUALDADE)
-- SINTAXE: SQL/99
SELECT D.NM_DEPTO, F.NR_MATRICULA, F.NM_FUNCIONARIO
  FROM T_SIP_DEPTO D INNER JOIN T_SIP_FUNCIONARIO F
    ON (D.CD_DEPTO = F.CD_DEPTO);

-- OU SINTAXE ANTERIOR AO SQL/99
SELECT D.NM_DEPTO, F.NR_MATRICULA, F.NM_FUNCIONARIO
  FROM T_SIP_DEPTO D, T_SIP_FUNCIONARIO F
 WHERE D.CD_DEPTO = F.CD_DEPTO;
```



NM_DEPTO	NR_MATRICULA	NM_FUNCIONARIO
FINANCEIRO	12345	JOAO DA SILVA
FINANCEIRO	12346	MANUEL DA SILVA
FINANCEIRO	12347	JANDIRA DA SILVA
TECNOLOGIA DA INFORMAÇÃO	12348	KATIA REGINA SOUZA
TECNOLOGIA DA INFORMAÇÃO	12349	MARIA DAS DORES SOUZA
TECNOLOGIA DA INFORMAÇÃO	12350	ALFREDO DE SOUZA
CONTAS A PAGAR	12351	GISELE DE JESUS
CONTAS A PAGAR	12352	RAFAEL DE JESUS
CONTAS A PAGAR	12353	ROSANA DE JESUS
FATURAMENTO	12354	JOSEFINA DE ALMEIDA
FATURAMENTO	12355	LUCIANA DE ALMEIDA
FATURAMENTO	12356	THIAGO DE ALMEIDA
RECURSOS HUMANOS	12357	LARISSA DE CAMARGO
RECURSOS HUMANOS	12358	ANTONIO DE CAMARGO
RECURSOS HUMANOS	12359	JOSE DE CAMARGO

# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL



Tabela: FUNCIONARIO

Tabela: DEPARAMENTO

NR_MATRICULA	CD_DEPTO	NM_FUNCIONARIO
12345	1	JOAO DA SILVA
12346	1	MANUEL DA SILVA
12347	1	JANDIRA DA SILVA
12348	2	KATIA REGINA SOUZA
12349	2	MARIA DAS DORES SOUZA
12350	2	ALFREDO DE SOUZA
12351	3	GISELE DE JESUS
12352	3	RAFAEL DE JESUS
12353	3	ROSANA DE JESUS
12354	4	JOSEFINA DE ALMEIDA
12355	4	LUCIANA DE ALMEIDA
12356	4	THIAGO DE ALMEIDA
12357	5	LARISSA DE CAMARGO
12358	5	ANTONIO DE CAMARGO
12359	5	JOSE DE CAMARGO

CD_DEPTO	NM_DEPTO	DS_SIGLA
1	FINANCEIRO	FIN
2	TECNOLOGIA DA INFORMAÇÃO	TIN
3	CONTAS A PAGAR	CPG
4	FATURAMENTO	FAT
5	RECURSOS HUMANOS	RHU

## Resultado da Junção:

NM_DEPTO	NR_MATRICULA	NM_FUNCIONARIO
FINANCEIRO	12345	JOAO DA SILVA
FINANCEIRO	12346	MANUEL DA SILVA
FINANCEIRO	12347	JANDIRA DA SILVA
TECNOLOGIA DA INFORMAÇÃO	12348	KATIA REGINA SOUZA
TECNOLOGIA DA INFORMAÇÃO	12349	MARIA DAS DORES SOUZA
TECNOLOGIA DA INFORMAÇÃO	12350	ALFREDO DE SOUZA
CONTAS A PAGAR	12351	GISELE DE JESUS
CONTAS A PAGAR	12352	RAFAEL DE JESUS
CONTAS A PAGAR	12353	ROSANA DE JESUS
FATURAMENTO	12354	JOSEFINA DE ALMEIDA
FATURAMENTO	12355	LUCIANA DE ALMEIDA
FATURAMENTO	12356	THIAGO DE ALMEIDA
RECURSOS HUMANOS	12357	LARISSA DE CAMARGO
RECURSOS HUMANOS	12358	ANTONIO DE CAMARGO
RECURSOS HUMANOS	12359	JOSE DE CAMARGO

# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL



### Tipos de Join

Há três tipos diferentes de Joins:

#### 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.

#### Autojunções (SELF-JOIN)

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

# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL

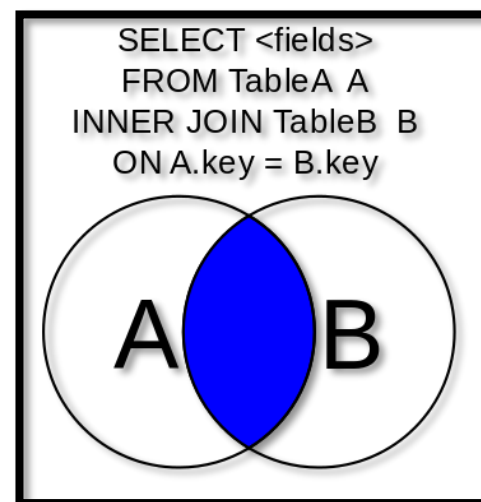


### INNER JOIN

Nesse tipo de junção, caso sejam unidas duas tabelas serão **exibidos todos os dados relacionados existentes nas duas tabelas envolvidas na consulta.**

Denomina-se **união regular** as uniões que têm a cláusula **WHERE** indicando a **chave primária à estrangeira das tabelas** afetadas pelo comando SELECT.

É um tipo de junção interna.



# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL

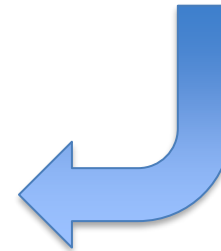


### Pesquisa em Múltiplas Tabelas – INNER JOIN

A ideia é recuperar todos os funcionários que possuam dependentes e todos os dependentes que pertencem a funcionários.

```
SELECT F.NM_FUNCIONARIO, CD_DEPENDENTE, NM_DEPENDENTE  
FROM T_SIP_FUNCIONARIO F INNER JOIN T_SIP_DEPENDENTE D  
ON (F.NR_MATRICULA = D.NR_MATRICULA);
```

NM_FUNCIONARIO	CD_DEPENDENTE	NM_DEPENDENTE
JOAO DA SILVA	1	JOANINHA
JOAO DA SILVA	2	JULINHA
JOAO DA SILVA	3	TONINHO
THIAGO DE ALMEIDA	1	JUNINHO
THIAGO DE ALMEIDA	2	ZEZINHO
THIAGO DE ALMEIDA	3	MARCELINHO
JOSE DE CAMARGO	1	MARIAZINHA
JOSE DE CAMARGO	2	LUIZINHA
JOSE DE CAMARGO	3	CARMINHA



# Linguagem SQL

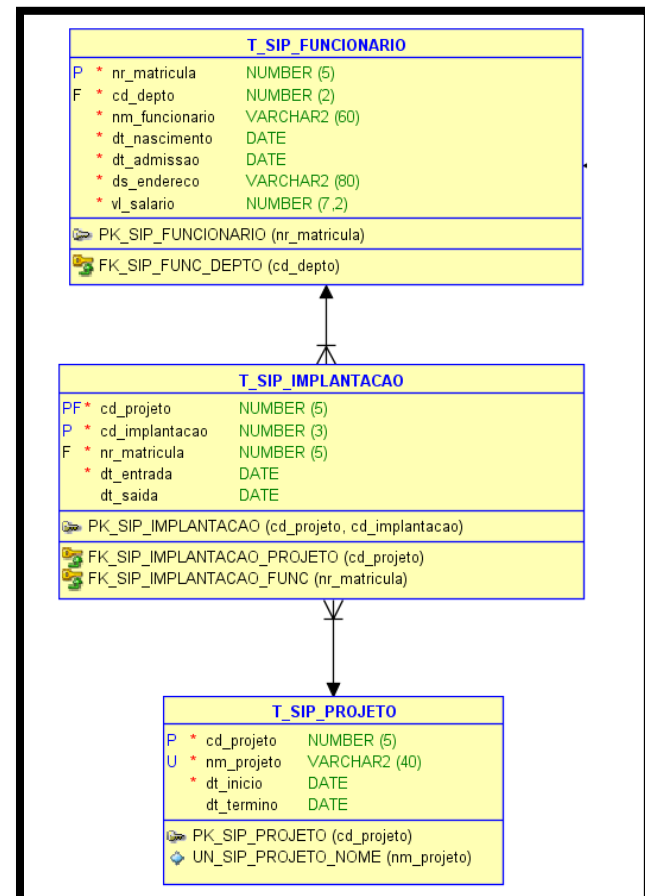
## Consulta dos Dados utilizando a linguagem SQL



### Pesquisa em Múltiplas Tabelas – INNER JOIN

A ideia é recuperar todos os projetos que tenham suas implantações iniciadas por funcionários.

Para isso teremos que consultar as tabelas: PROJETO, IMPLANTAÇÃO e FUNCIONÁRIO.



# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL



### Pesquisa em Múltiplas Tabelas – INNER JOIN

```
-- JUNÇÃO INTERNA (INNER JOIN) DE DUAS OU MAIS TABELAS
-- RECUPERAR TODOS OS PROJETOS QUE TENHAM SUAS IMPLANTAÇÕES INICIADAS
-- POR FUNCIONARIOS
-- EXEMPLO COM INNER JOIN - PADRÃO SQL/99

SELECT F.NR_MATRICULA      "MATRICULA" ,
       F.NM_FUNCIONARIO    "FUNCIONARIO" ,
       P.NM_PROJETO        "PROJETO" ,
       I.DT_ENTRADA        "ENTRADA" ,
       I.DT_SAIDA          "SAIDA"

FROM T_SIP_PROJETO P INNER JOIN T_SIP_IMPLANTACAO I
ON ( P.CD_PROJETO = I.CD_PROJETO )
INNER JOIN T_SIP_FUNCIONARIO F
ON ( F.NR_MATRICULA = I.NR_MATRICULA )

ORDER BY F.NM_FUNCIONARIO ;
```

MATRICULA	FUNCIONARIO	PROJETO	ENTRADA	SAIDA
12350	ALFREDO DE SOUZA	PROJETO 5	28/09/2014	15/09/2018
12358	ANTONIO DE CAMARGO	PROJETO 4	25/03/2017	(null)
12345	JOAO DA SILVA	PROJETO 1	11/10/2016	15/09/2017
12345	JOAO DA SILVA	PROJETO 3	28/04/2013	(null)
12345	JOAO DA SILVA	PROJETO 5	28/09/2014	15/09/2018
12345	JOAO DA SILVA	PROJETO 2	25/10/2013	15/04/2014
12354	JOSEFINA DE ALMEIDA	PROJETO 3	28/04/2013	(null)
12354	JOSEFINA DE ALMEIDA	PROJETO 4	25/03/2017	15/08/2018
12354	JOSEFINA DE ALMEIDA	PROJETO 5	28/09/2014	15/09/2018
12348	KATIA REGINA SOUZA	PROJETO 1	11/10/2016	25/06/2018
12357	LARISSA DE CAMARGO	PROJETO 4	25/03/2017	(null)
12355	LUCIANA DE ALMEIDA	PROJETO 2	25/10/2013	15/04/2015
12355	LUCIANA DE ALMEIDA	PROJETO 3	28/04/2013	(null)
12346	MANUEL DA SILVA	PROJETO 1	11/10/2016	(null)
12352	RAFAEL DE JESUS	PROJETO 2	25/10/2013	25/07/2014





# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL



### OUTER JOIN – JUNÇÕES EXTERNAS

Define-se união externa como aquela que inclui linhas no resultado da busca mesmo que não haja relação entre as duas tabelas que estão sendo unidas.

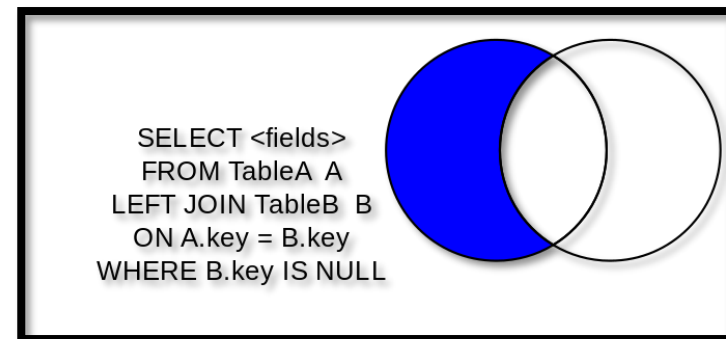
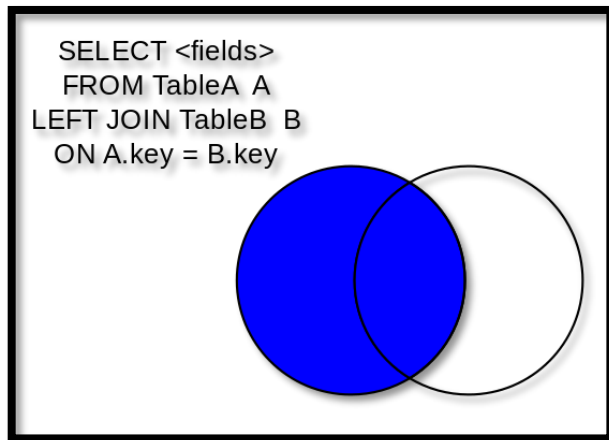
Onde não há informação o banco de dados insere NULL.

- Há três tipos de junções externas:
  - **LEFT JOIN**: junção externar 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 direita e à esquerda da operação.

## Consulta dos Dados utilizando a linguagem SQL

### LEFT JOIN

- Todas as linhas da tabela a 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).



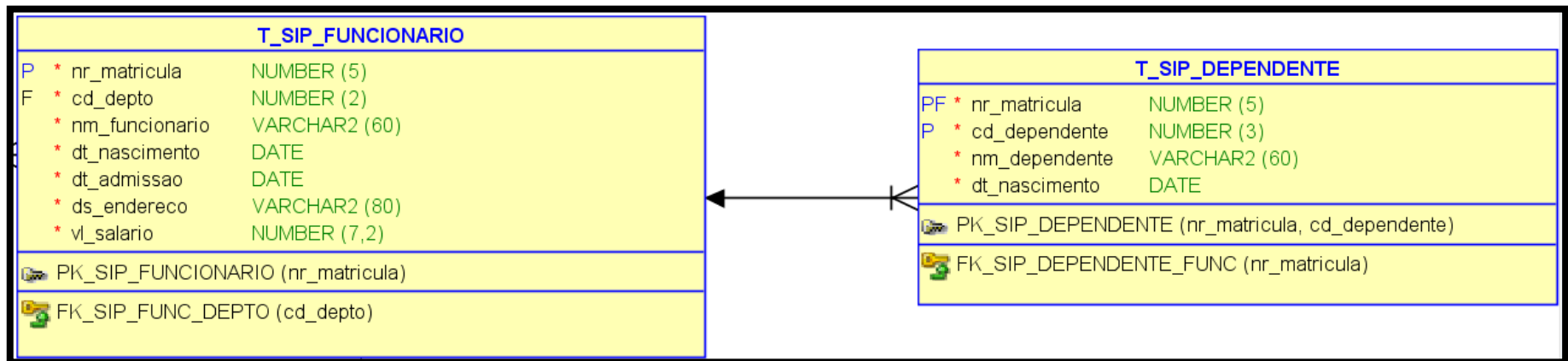
# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL



### Pesquisa em Múltiplas Tabelas – LEFT OUTER JOIN

A ideia é recuperar todos os Funcionários que não possuem Dependentes.



# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL

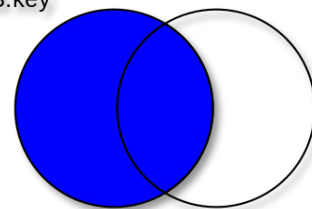
FIAP



### Pesquisa em Múltiplas Tabelas – LEFT OUTER JOIN

```
-- LEFT OUTER JOIN
-- A ideia é recuperar todos os Funcionários que não possuem Dependentes
SELECT F.NM_FUNCIONARIO, CD_DEPENDENTE, NM_DEPENDENTE
FROM T_SIP_FUNCIONARIO F LEFT OUTER JOIN T_SIP_DEPENDENTE D
ON (F.NR_MATRICULA = D.NR_MATRICULA);
-- OU
SELECT F.NM_FUNCIONARIO, CD_DEPENDENTE, NM_DEPENDENTE
FROM T_SIP_FUNCIONARIO F LEFT JOIN T_SIP_DEPENDENTE D
ON (F.NR_MATRICULA = D.NR_MATRICULA);
```

```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
```



Observe que são recuperados todos os funcionários, aqueles que possuem dependentes e aqueles que não possuem.

Veja que para os funcionários que não possuem dependentes, os campos: código e nome do dependente são apresentados com NULL.



NM_FUNCIONARIO	CD_DEPENDENTE	NM_DEPENDENTE
JOAO DA SILVA	1	JOANINHA
JOAO DA SILVA	2	JULINHA
JOAO DA SILVA	3	TONINHO
THIAGO DE ALMEIDA	1	JUNINHO
THIAGO DE ALMEIDA	2	ZEZINHO
THIAGO DE ALMEIDA	3	MARCELINHO
JOSE DE CAMARGO	1	MARIAZINHA
JOSE DE CAMARGO	2	LUIZINHA
JOSE DE CAMARGO	3	CARMINHA
GISELE DE JESUS	(null)	(null)
MARIA DAS DORES SOUZA	(null)	(null)
ALFREDO DE SOUZA	(null)	(null)
RAFAEL DE JESUS	(null)	(null)
ROSANA DE JESUS	(null)	(null)
LUCIANA DE ALMEIDA	(null)	(null)
MANUEL DA SILVA	(null)	(null)
KATIA REGINA SOUZA	(null)	(null)
JANDIRA DA SILVA	(null)	(null)
JOSEFINA DE ALMEIDA	(null)	(null)
ANTONIO DE CAMARGO	(null)	(null)
LARISSA DE CAMARGO	(null)	(null)

# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL

FIAP

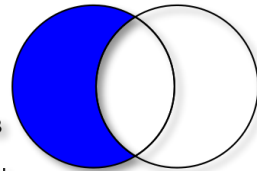


### Pesquisa em Múltiplas Tabelas – LEFT OUTER JOIN (IS NULL)

```
-- LEFT OUTER JOIN
-- A ideia é recuperar todos os Funcionários que não possuem Dependentes
SELECT F.NM_FUNCIONARIO, CD_DEPENDENTE, NM_DEPENDENTE
  FROM T_SIP_FUNCIONARIO F LEFT OUTER JOIN T_SIP_DEPENDENTE D
    ON (F.NR_MATRICULA = D.NR_MATRICULA)
 WHERE D.NR_MATRICULA IS NULL;

-- OU
SELECT F.NM_FUNCIONARIO, CD_DEPENDENTE, NM_DEPENDENTE
  FROM T_SIP_FUNCIONARIO F LEFT JOIN T_SIP_DEPENDENTE D
    ON (F.NR_MATRICULA = D.NR_MATRICULA)
 WHERE D.NR_MATRICULA IS NULL;
```

```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
  ON A.key = B.key
WHERE B.key IS NULL
```



Para recuperar apenas os funcionários que não possuem dependentes, é necessário acrescentar a condição: **CHAVE ESTRANGEIRA IS NULL**, desta forma as linhas correspondentes a intersecção dos conjuntos são excluídas da apresentação.



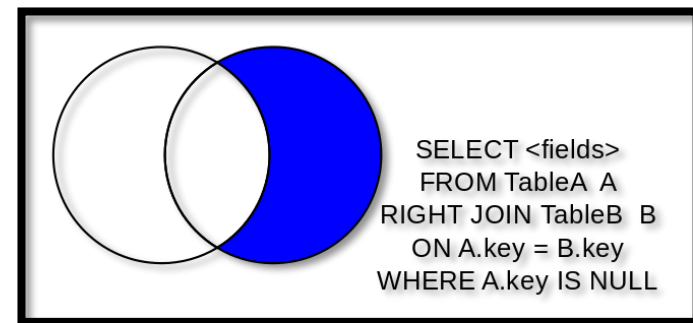
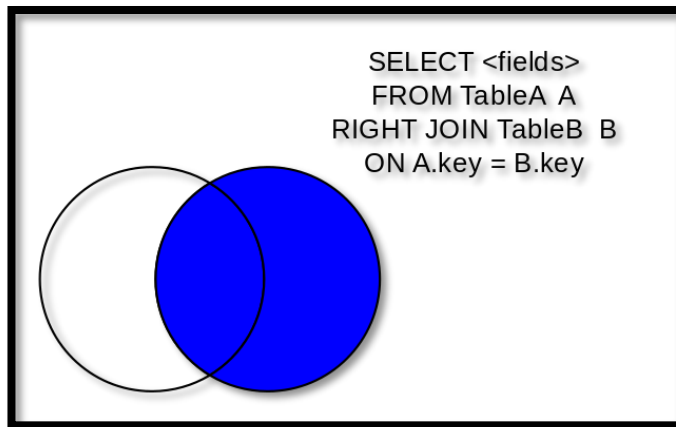
NM_FUNCIONARIO	CD_DEPENDENTE	NM_DEPENDENTE
GISELE DE JESUS	(null)	(null)
MARIA DAS DORES SOUZA	(null)	(null)
ALFREDO DE SOUZA	(null)	(null)
RAFAEL DE JESUS	(null)	(null)
ROSANA DE JESUS	(null)	(null)
LUCIANA DE ALMEIDA	(null)	(null)
MANUEL DA SILVA	(null)	(null)
KATIA REGINA SOUZA	(null)	(null)
JANDIRA DA SILVA	(null)	(null)
JOSEFINA DE ALMEIDA	(null)	(null)
ANTONIO DE CAMARGO	(null)	(null)
LARISSA DE CAMARGO	(null)	(null)



## Consulta dos Dados utilizando a linguagem SQL

### RIGHT JOIN

- Todas as linhas da tabela a 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).



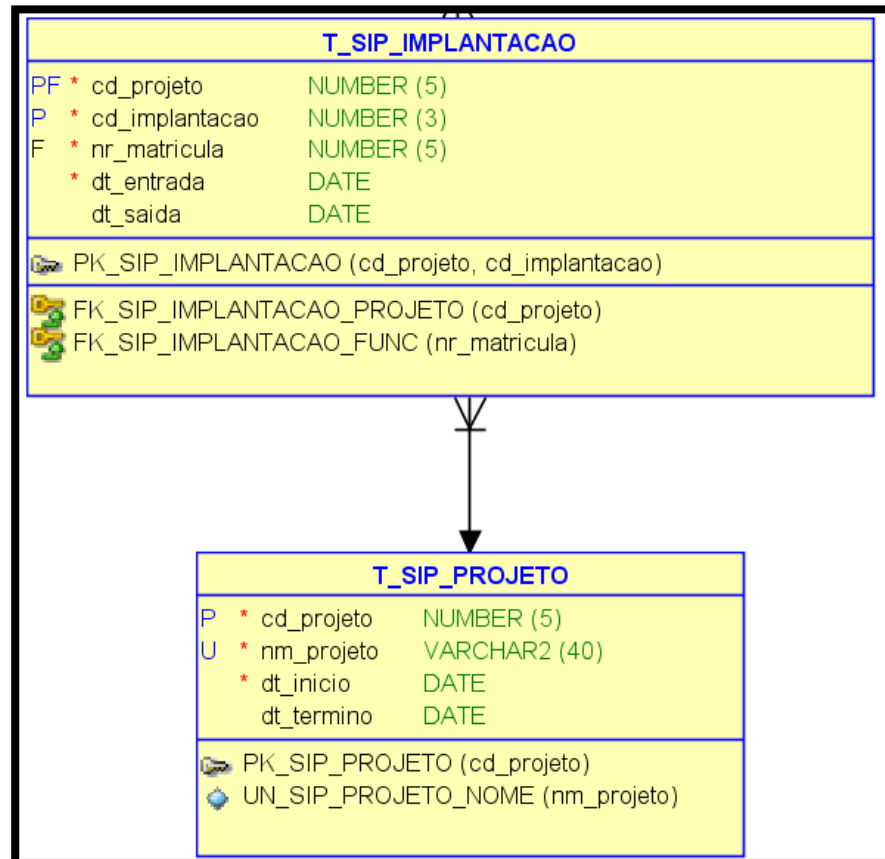
# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL



### Pesquisa em Múltiplas Tabelas – RIGHT OUTER JOIN

A ideia é recuperar todos os Projetos, que ainda foram implantados.



# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL

FIAP

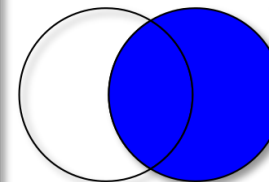


### Pesquisa em Múltiplas Tabelas – RIGHT OUTER JOIN

```
-- RIGHT OUTER JOIN
-- A ideia é recuperar todos os Projetos, que ainda foram implantados.
SELECT P.NM_PROJETO, P.DT_INICIO, I.DT_ENTRADA, I.DT_SAIDA
FROM T_SIP_IMPLANTACAO I RIGHT OUTER JOIN T_SIP_PROJETO P
ON (P.CD_PROJETO = I.CD_PROJETO);

-- OU
SELECT P.NM_PROJETO, P.DT_INICIO, I.DT_ENTRADA, I.DT_SAIDA
FROM T_SIP_IMPLANTACAO I RIGHT JOIN T_SIP_PROJETO P
ON (P.CD_PROJETO = I.CD_PROJETO);
```

```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
```



Observe que são recuperados todos os projetos, aqueles que possuem implantações e aqueles que não possuem. Veja que para os projetos que não possuem implantações, os campos: data da entrada e data da saída são apresentados com NULL.

NM_PROJETO	DT_INICIO	DT_ENTRADA	DT_SAIDA
PROJETO 1	11/10/2016	11/10/2016	15/09/2017
PROJETO 1	11/10/2016	11/10/2016	(null)
PROJETO 1	11/10/2016	11/10/2016	25/06/2018
PROJETO 2	25/10/2013	25/10/2013	15/04/2014
PROJETO 2	25/10/2013	25/10/2013	25/07/2014
PROJETO 2	25/10/2013	25/10/2013	15/04/2015
PROJETO 3	28/04/2012	28/04/2013	(null)
PROJETO 3	28/04/2012	28/04/2013	(null)
PROJETO 3	28/04/2012	28/04/2013	(null)
PROJETO 4	25/03/2017	25/03/2017	(null)
PROJETO 4	25/03/2017	25/03/2017	(null)
PROJETO 4	25/03/2017	25/03/2017	15/08/2018
PROJETO 5	28/09/2014	28/09/2014	15/09/2018
PROJETO 5	28/09/2014	28/09/2014	15/09/2018
PROJETO 5	28/09/2014	28/09/2014	15/09/2018
PROJETO 6	05/04/2020	(null)	(null)





# Linguagem SQL

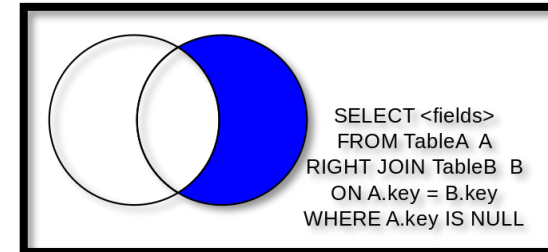
## Consulta dos Dados utilizando a linguagem SQL

FIAP



### Pesquisa em Múltiplas Tabelas – RIGHT OUTER JOIN (IS NULL)

```
-- RIGHT OUTER JOIN
-- A ideia é recuperar todos os Projetos, que ainda foram implantados.
SELECT P.NM_PROJETO, P.DT_INICIO, I.DT_ENTRADA, I.DT_SAIDA
  FROM T_SIP_IMPLANTACAO I RIGHT OUTER JOIN T_SIP_PROJETO P
    ON (P.CD_PROJETO = I.CD_PROJETO)
 WHERE I.CD_PROJETO IS NULL;
-- OU
SELECT P.NM_PROJETO, P.DT_INICIO, I.DT_ENTRADA, I.DT_SAIDA
  FROM T_SIP_IMPLANTACAO I RIGHT JOIN T_SIP_PROJETO P
    ON (P.CD_PROJETO = I.CD_PROJETO)
 WHERE I.CD_PROJETO IS NULL;
```



Para recuperar apenas os projetos que não possuem implantações, é necessário acrescentar a condição: CHAVE ESTRANGEIRA IS NULL, desta forma as linhas correspondentes a intersecção dos conjuntos são excluídas da apresentação.



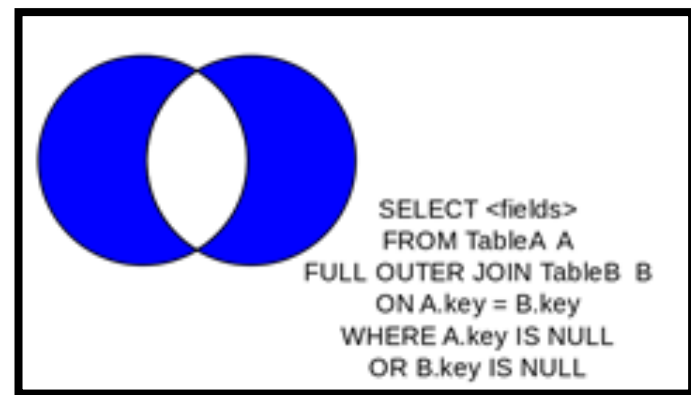
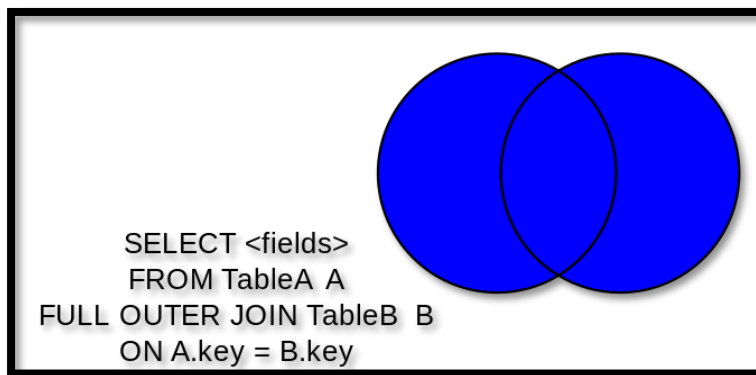
NM_PROJETO	DT_INICIO	DT_ENTRADA	DT_SAIDA
PROJETO 6	05/04/2020	(null)	(null)



## Consulta dos Dados utilizando a linguagem SQL

### 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.



# Linguagem SQL

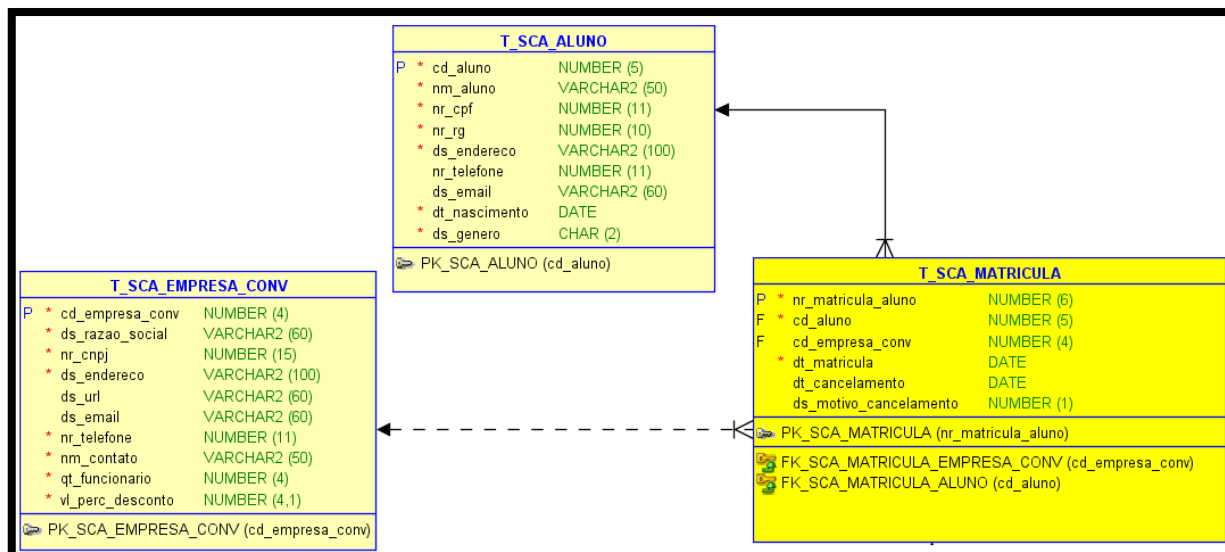
## Consulta dos Dados utilizando a linguagem SQL



### Pesquisa em Múltiplas Tabelas – FULL OUTER JOIN

O exemplo abaixo, apresenta um relacionamento totalmente opcional, entre uma empresa conveniada e os alunos matriculados em uma academia.

A ideia é recuperar todas as empresas que não possuem nenhum funcionário matriculado na academia e todos os alunos matriculados que não pertencem a nenhuma empresa conveniada.



# Linguagem SQL

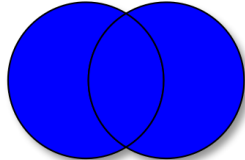
## Consulta dos Dados utilizando a linguagem SQL

FIAP



### Pesquisa em Múltiplas Tabelas – FULL OUTER JOIN

```
-- FULL OUTER JOIN
SELECT E.CD_EMPRESA_CONV ,
       E.DS_RAZAO_SOCIAL ,
       M.NR_MATRICULA_ALUNO
FROM T_SCA_EMPRESA_CONV E FULL OUTER JOIN T_SCA_MATRICULA M
ON ( E.CD_EMPRESA_CONV = M.CD_EMPRESA_CONV );
```



```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
```

Observe que são recuperados todos as matrículas, aquelas que tem origem de empresas conveniadas e as que os alunos não pertencem a nenhuma instituição, ou ainda as empresas conveniadas, que não possuem nenhum aluno matriculado na academia.

Veja que os campos: cd\_empresa\_conv, ds\_razão\_social e nr\_matricula\_aluno são apresentados com NULL.

CD_EMPRESA_CONV	DS_RAZAO_SOCIAL	NR_MATRICULA_ALUNO
1	EMPRESA XYZ LTDA.	1001
1	EMPRESA XYZ LTDA.	1002
2	EMPRESA ABC LTDA.	1003
(null)	(null)	1004
(null)	(null)	1005
4	EMPRESA EFG LTDA.	(null)
3	EMPRESA BCD LTDA.	(null)



# Linguagem SQL

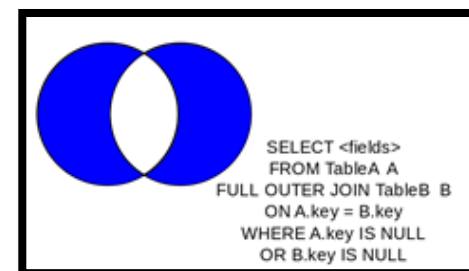
## Consulta dos Dados utilizando a linguagem SQL

FIAP



### Pesquisa em Múltiplas Tabelas – FULL OUTER JOIN (IS NULL)

```
-- FULL OUTER JOIN
SELECT E.CD_EMPRESA_CONV ,
       E.DS_RAZAO_SOCIAL ,
       M.NR_MATRICULA_ALUNO
FROM T_SCA_EMPRESA_CONV E FULL OUTER JOIN T_SCA_MATRICULA M
ON ( E.CD_EMPRESA_CONV = M.CD_EMPRESA_CONV )
WHERE E.CD_EMPRESA_CONV IS NULL OR
      M.NR_MATRICULA_ALUNO IS NULL ;
```



Para recuperar apenas os alunos matriculados que não pertencem a empresas conveniadas e as empresas conveniadas que não possuem alunos matriculados, é necessário acrescentar a condição: CHAVE ESTRANGEIRA IS NULL, desta forma as linhas correspondentes a intersecção dos conjuntos são excluídas da apresentação.



CD_EMPRESA_CONV	DS_RAZAO_SOCIAL	NR_MATRICULA_ALUNO
(null)	(null)	1004
(null)	(null)	1005
4	EMPRESA EFG LTDA.	(null)
3	EMPRESA BCD LTDA.	(null)

## Consulta dos Dados utilizando a linguagem SQL



### SELF JOIN ou AUTOJUNÇÃO

As junções do tipo **SELF JOIN** são utilizadas em casos onde temos no modelo de dados, a figura do **auto-relacionamento (relacionamento recursivo)**. Nesse caso precisamos acessar a mesma tabela duas vezes: uma para recuperar os registros e a outra para buscar os dados relacionados a mesma (auto-relacionamento).

**Nesse caso se apenas indicarmos a tabela duas vezes sem usar um apelido (alias) resultará em erro.**

Portanto é necessário o uso de apelido nas tabelas e nomes de colunas sempre que tratarmos de SELF JOIN.

## Consulta dos Dados utilizando a linguagem SQL

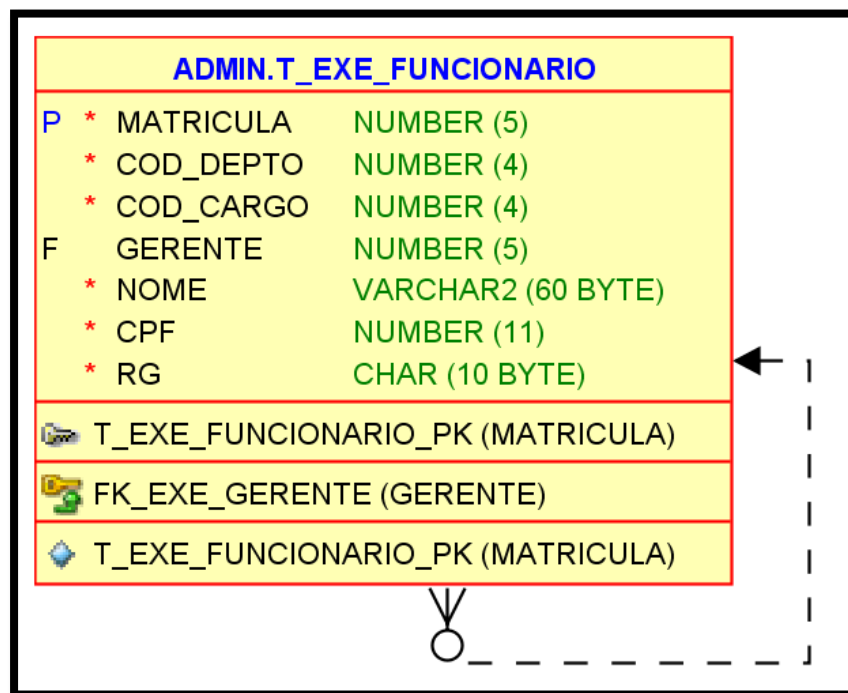
### SELF JOIN ou AUTOJUNÇÃO



#### EXEMPLO:

A ideia é recuperar o nome dos gerentes de todos os funcionários cadastrados.

#### Modelagem de Dados:



# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL

### SELF JOIN ou AUTOJUNÇÃO



Conteúdo da tabela para visualização:

Tabela: FUNCIONARIO

⚡ MATRICULA	⚡ COD_DEPTO	⚡ COD_CARGO	⚡ GERENTE	⚡ NOME	⚡ CPF	⚡ RG
1	1	1	3	RITA	12345678901	11222333X
2	1	1	3	JOAO	12342233445	11333444X
3	1	1	(null)	ROSA	12234543441	11444555X
4	1	1	(null)	MARIA	1555566651	116667778
5	1	1	3	ANA	1343322901	119764333



## Consulta dos Dados utilizando a linguagem SQL

### SELF JOIN ou AUTOJUNÇÃO



### Pesquisa em Múltiplas Tabelas – SELF JOIN (AUTOJUNÇÃO)

```
-- EXEMPLO AUTOJUNÇÃO (SELF JOIN) - PADRÃO SQL/99
SELECT FUNC.Matricula ,
       FUNC.NOME "FUNCIONÁRIO",
       FUNC.GERENTE "COD. GERENTE",
       GER.NOME "GERENTE"
FROM T_EXE_FUNCIONARIO FUNC INNER JOIN
     T_EXE_FUNCIONARIO GER
ON ( GER.MATRICULA = FUNC.GERENTE );
```

Observe que o funcionário “ROSA”  
gerencia os funcionários: Rita, João  
e Ana.

### Resultado da consulta AUTOJUNÇÃO

MATRICULA	FUNCIONÁRIO	COD. GERENTE	GERENTE
1	RITA	3	ROSA
2	JOAO	3	ROSA
5	ANA	3	ROSA

# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL



### CROSS JOIN (Junção Cruzada)

Também conhecida por **Produto Cartesiano**.

**Ocorrerá um produto cartesiano sempre que:**

- ☐ Não houver uma condição para a união (ausência da cláusula WHERE);
- ☐ Condição de união entre as tabelas inválida (cláusula WHERE incorreta);
- ☐ Todas as linhas da primeira tabela estiverem unidas a todas as linhas da segunda tabela.

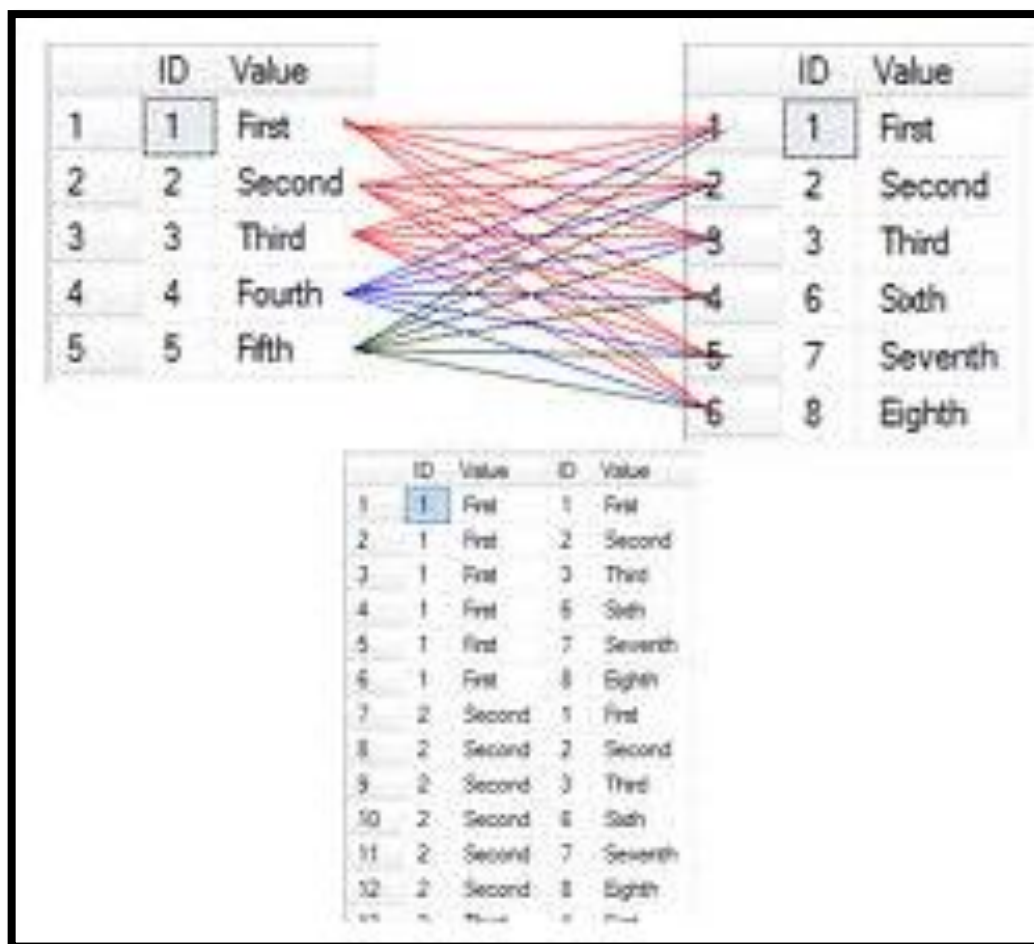
Para evitar um produto cartesiano, utilize uma condição válida para junção na cláusula WHERE.

# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL

### CROSS JOIN (Junção Cruzada)

Exemplo:



# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL

### CROSS JOIN (Junção Cruzada)



Faça uma junção cruzada entre as tabelas: “DEPARTAMENTO” e “FUNCIONARIO”.

Consulte as tabelas “DEPARTAMENTO” e “FUNCIONARIO” separadamente. Observe a tabela “DEPARTAMENTO” possui 5 linhas e a tabela “FUNCIONARIO”, possui 7 linhas:

Utilize o comando SELECT abaixo:

```
-- CROSS JOIN
-- EXEMPLO - CROSS JOIN (PRODUTO CARTESIANO)
SELECT d.nm_depto, f.nm_funcionario
  FROM T_SIP_DEPTO D,
       T_SIP_FUNCIONARIO F;
-- UTILIZANDO O PADRÃO SQL/99
SELECT d.nm_depto, f.nm_funcionario
  FROM T_SIP_DEPTO D CROSS JOIN T_SIP_FUNCIONARIO F;
```



# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL

### CROSS JOIN (Junção Cruzada)

Executando o exemplo anterior teremos o seguinte resultado:



SQL   50 linhas extraídas em 0,022 segundos		
	NM_DEPTO	NM_FUNCIONARIO
1	CONTAS A PAGAR	JOAO DA SILVA
2	CONTAS A PAGAR	MANUEL DA SILVA
3	CONTAS A PAGAR	JANDIRA DA SILVA
4	CONTAS A PAGAR	KATIA REGINA SOUZA
5	CONTAS A PAGAR	MARIA DAS DORES SOUZA
6	CONTAS A PAGAR	ALFREDO DE SOUZA
7	CONTAS A PAGAR	GISELE DE JESUS
8	CONTAS A PAGAR	RAFAEL DE JESUS
9	CONTAS A PAGAR	ROSANA DE JESUS
10	CONTAS A PAGAR	JOSEFINA DE ALMEIDA
11	CONTAS A PAGAR	LUCIANA DE ALMEIDA
12	CONTAS A PAGAR	THIAGO DE ALMEIDA
13	CONTAS A PAGAR	LARISSA DE CAMARGO
14	CONTAS A PAGAR	ANTONIO DE CAMARGO
15	CONTAS A PAGAR	JOSE DE CAMARGO
16	FATURAMENTO	JOAO DA SILVA
17	FATURAMENTO	MANUEL DA SILVA
18	FATURAMENTO	JANDIRA DA SILVA
19	FATURAMENTO	KATIA REGINA SOUZA
20	FATURAMENTO	MARIA DAS DORES SOUZA
21	FATURAMENTO	ALFREDO DE SOUZA
22	FATURAMENTO	GISELE DE JESUS
23	FATURAMENTO	RAFAEL DE JESUS

# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL



### CROSS JOIN (Junção Cruzada)

O que ocorreu no exemplo foi que as **linhas** da primeira tabela foram combinadas com as linhas da segunda tabela, demonstrando um resultado na maior parte das vezes indesejado.

Veja que a tabela “DEPARTAMENTO”, possuía 5 linhas e a tabela “FUNCIONARIO” possuía 7 linhas, o resultado foram  $(5 \times 7 = 35)$  **35 linhas exibidas como resultado da nossa busca.**

Percebemos então a **necessidade de colocar a condição após a cláusula WHERE associando a chave primária e estrangeira das tabelas.** Utilize o modelo de dados para facilitar a criação de sua consulta SQL.

## Consulta dos Dados utilizando a linguagem SQL



### União de Tabelas sem colunas em comum (no-equijoin)

Existem situações onde mesmo não havendo um relacionamento entre as tabelas, há o relacionamento de uma coluna com o intervalo de outras colunas em outras tabelas.



#### EXEMPLOS:

- Os empregados estão associados a uma faixa salarial. Caso criemos as tabelas relacionadas quando houver mudança de faixa teremos de ir na tabela de empregados para alterar a faixa salarial (FK) do empregado e vice versa, quando alterar o salário do empregado precisamos buscar a nova faixa e associá-la novamente ao empregado.

# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL

### União de Tabelas sem colunas em comum (no-equijoin)



#### EXEMPLO:

Exemplos como faixa de imposto de renda, alíquotas de impostos em geral, enquadramentos baseados em valores e ou quantidades, são exemplos típicos de consultas non equijoin e nesse caso as tabelas não seriam relacionadas através do modelo e sim através da aplicação.





# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL


### União de Tabelas sem colunas em comum (no-equijoin)




Visualizando a modelagem observamos que a tabela FAIXA\_PRECO possui a categoria ou classificação de preços dos CD's.

#### ADMIN.T\_SCC\_CD


P	* CD_CD	NUMBER (5)
	* CD_GRAVADORA	NUMBER (5)
	* NM_TITULO	VARCHAR2 (30 BYTE)
	* DT_LANCAMENTO	DATE
	* VL_PRECO_VENDA	NUMBER (6,2)


 PK\_SCC\_CD (CD\_CD)

 PK\_SCC\_CD (CD\_CD)

#### ADMIN.T\_SCC\_FAIXA\_PRECO

P	* CD_FAIXA_PRECO	NUMBER (5)
	* DS_FAIXA_PRECO	VARCHAR2 (30 BYTE)
	* VL_INICIAL	NUMBER (6,2)
	* VL_FINAL	NUMBER (6,2)

 PK\_SCC\_FAIXA\_PRECO (CD\_FAIXA\_PRECO)

 PK\_SCC\_FAIXA\_PRECO (CD\_FAIXA\_PRECO)

# Linguagem SQL

## Consulta dos Dados utilizando a linguagem SQL

### União de Tabelas sem colunas em comum (no-equijoin)



Conteúdo das tabelas para visualização:

Tabela: FAIXA\_PRECO

CD_FAIXA_PRECO	DS_FAIXA_PRECO	VL_INICIAL	VL_FINAL
1	SELO VERDE	5	15,99
2	SELO AZUL	16	25,99
3	SELO AMARELO	26	45,99
4	SELO VERMELHO	46	65,99
5	SELO PRETO	66	100

Tabela: CD

CD_CD	CD_GRAVADORA	NM_TITULO	DT_LANCAMENTO	VL_PRECO_VENDA
1	101	CD ABC	10/07/15	15,5
2	101	CD DEF	21/03/16	23,4
3	101	CD GHI	19/10/16	56,75
4	101	CD KLM	13/03/15	33,9

# Linguagem SQL

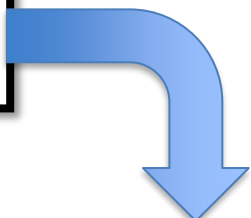
## Consulta dos Dados utilizando a linguagem SQL



### União de Tabelas sem colunas em comum (no-equijoin)

```
-- NO-EQUIJOIN
SELECT C.NM_TITULO ,
       C.VL_PRECO_VENDA ,
       F.DS_FAIXA_PRECO
FROM   T_SCC_CD C, T_SCC_FAIXA_PRECO F
WHERE  C.VL_PRECO_VENDA BETWEEN F.VL_INICIAL AND
                                   F.VL_FINAL;
```

**Nota:** Veja que a condição está utilizando a coluna preço de venda da tabela “CD”, comparando com a faixa de preços da tabela “FAIXA\_PRECO”.



NM_TITULO	VL_PRECO_VENDA	DS_FAIXA_PRECO
CD ABC	15,5	SELO VERDE
CD DEF	23,4	SELO AZUL
CD GHI	56,75	SELO VERMELHO
CD KLM	33,9	SELO AMARELO

# Próxima aula estudaremos

- ☐ Revisão de conceitos através de exercícios

## REFERÊNCIAS



- MACHADO, Felipe Nery R. Banco de Dados - Projeto e Implementação. Érica, 2004.
- Páginas: 330, 331.
- ELMASRI, R.; NAVATHE, S.B. Sistemas de Banco de Dados: Fundamentos e Aplicações. Pearson, 2005. Páginas: 153, 154.
- PRICE, JASON, ORACLE DATABASE 11 g – SQL Domine SQL e PL-SQL no banco de Dados Oracle, Bookman, 2008. Capítulos: 2.

### Outros:

- Manual Oficial Oracle – Introdução ao Oracle 9i (SQL) - Oracle Corporation, 2000, 2001.
- <https://www.w3schools.com/sql/>