UNIP
UNIVERSIDADE PAULISTA
Interativa

Unidade III

BANCO DE DADOS

Prof. Luiz Fernando

Tipos de dados

- **Um Banco de Dados existe para armazenar dados de forma efetiva, para posteriormente recuperá-los.**
- **Para auxiliar nessa tarefa, existem tipos diferentes de dados.**

Principais tipos de dados

- Numérico.
- Texto.
- Data.

Observação: os tipos e subtipos podem variar um pouco de banco para banco, mas esses grupos principais nunca se alteram.

Numérico

São subtipos do tipo numérico

- **INT**
- **NUMERIC**
- **MONEY**

Texto

São subtipos do tipo texto

- **CHAR**
- **VARCHAR**
- **TEXT**

Data

São subtipos do tipo data

- **DATE**
- **TIME**
- **DATETIME**

Observação: até a versão 2005 do SQL havia apenas o tipo DATETIME, o DATE e o TIME surgiram na versão 2008.

Tipos de dados

Números entram normalmente:

- **1, 2, 3 etc.**

Texto está sempre entre aspas simples:

- **‘Luiz’**
- **‘a’**

Tipos de dados

- O formato padrão de uma data no sistema é YYYY-MM-DD.
- Quando inseridas diretamente, devem estar apresentadas entre aspas simples:
  - ‘2011-07-25’

Tipos de dados

Qualquer outro formato diferente desse deve vir com tratamento, no ORACLE utilizando o comando TO_DATE, e no SQL Server CONVERT:

- **TO_DATE('25/07/2011', 'DD/MM/YYYY')**
- **CONVERT(DATETIME, '25/07/2011',103)**

Outros tipos de dados

Existem outros tipos de dados. Conheça alguns deles:

- **BLOB – pode armazenar qualquer tipo de dado.**
- **IMAGE – armazena figuras.**
- **XML – um documento XML.**

Interatividade

Por que ao armazenar dados como o CPF ou o número do telefone, usamos dados do tipo CHAR?

a) Porque é mais bonito.

b) Porque é mais cômodo.

c) Porque não serão usados em cálculos e possuem tamanho fixo.

d) Porque precisam ser calculados.

e) Porque precisam ter tamanhos variados.

Resposta

Por que ao armazenar dados como o CPF ou o número do telefone, usamos dados do tipo CHAR?

a) Porque é mais bonito.

b) Porque é mais cômodo.

c) **Porque não serão usados em cálculos e possuem tamanho fixo.**

d) Porque precisam ser calculados.

e) Porque precisam ter tamanhos variados.

Linguagem SQL

- A linguagem SQL é a linguagem padrão para qualquer sistema de Banco de Dados.
- É padrão porque é mantida pela ANSI, órgão do governo americano equivalente ao ISO aqui no Brasil.
- A linguagem SQL é a mesma para todos os bancos, com pouca ou nenhuma variação entre eles.

Tipos de linguagem SQL

A linguagem SQL é dividida em TIPOS, de acordo com a funcionalidade dos comandos que a compõe. São tipos de linguagem SQL:

- **DDL = *Data Definition Language*.**
- **DML = *Data Manipulation Language*.**
- **DCL = *Data Control Language*.**
- **DTL = *Data Transaction Language*.**
- **DQL = *Data Query Language*.**

DDL

- É o tipo de linguagem que interage com os objetos do Banco.
- São objetos do Banco: tabelas, procedures, functions, views, entre outros.

São comandos DDL:

- CREATE
- ALTER
- DROP

Exemplo DDL – CREATE

```
CREATE TABLE ALUNO(
RA CHAR(7),
NOME VARCHAR(100)
)
```

Exemplo DDL – ALTER

Inclui uma nova coluna na tabela aluno

- ALTER TABLE ALUNO
- ADD DT_NASCIMENTO DATE

Altera a propriedade do campo RA para NOT NULL

- ALTER TABLE ALUNO
- ALTER COLUMN RA CHAR(7) NOT NULL

Define / cria a chave primária da tabela de aluno

- ALTER TABLE ALUNO
- ADD CONSTRAINT PK_ALUNO
- PRIMARY KEY(RA)

Exemplo DDL – DROP

Exclui a tabela ALUNO

- DROP TABLE ALUNO

Usando ALTER com DROP para excluir uma coluna

- ALTER TABLE ALUNO
- DROP COLUMN DT_NASCIMENTO

DML

- **É o tipo de linguagem que interage com os dados armazenados dentro das tabelas do Banco de Dados.**

São comandos DML:

- **INSERT**
- **DELETE**
- **UPDATE**
- **TRUNCATE**

Exemplo DML – INSERT

Inserir dados na tabela ALUNO

- INSERT INTO ALUNO
- (RA, NOME, DT_NASCIMENTO)
- VALUES
- (‘1234567’,’LUIZ FERNANDO’,’1978-08-28’)

Exemplo DML – UPDATE

Altera um ou mais dados dentro de uma tabela

- **UPDATE ALUNO**
- **SET NOME = ‘LUIZ FERNANDO’**
- **WHERE RA = ‘1234567’**

Atenção: o comando UPDATE deve vir sempre acompanhado da cláusula WHERE, sob risco de se atualizar todos os registros da tabela.

Exemplo DML – DELETE

Exclui um ou mais registros de uma tabela

- **DELETE FROM TABELA**
- **WHERE RA = ‘1234567’**

Atenção: o comando DELETE deve vir sempre acompanhado da cláusula WHERE, sob risco de se excluir todos os registros da tabela.

Exemplo DML – TRUNCATE

O comando TRUNCATE exclui todos os dados de uma tabela

- **TRUNCATE TABLE ALUNO**

Atenção: o comando TRUNCATE sempre exclui todos os dados de uma tabela.

Interatividade

Assinale a alternativa correta.

a) O comando INSERT obrigatoriamente necessita da cláusula WHERE.

b) A ausência da cláusula WHERE no comando UPDATE faz com que todos os registros da tabela sejam excluídos.

c) Usamos o comando DROP para excluir todos os registros de uma tabela.

d) Usamos a cláusula WHERE nos comandos DELETE e UPDATE para restringir a ação destes comandos.

e) Todas estão erradas.

Resposta

Assinale a alternativa correta.

a) O comando INSERT obrigatoriamente necessita da cláusula WHERE.

b) A ausência da cláusula WHERE no comando UPDATE faz com que todos os registros da tabela sejam excluídos.

c) Usamos o comando DROP para excluir todos os registros de uma tabela.

d) Usamos a cláusula WHERE nos comandos DELETE e UPDATE para restringir a ação destes comandos.

e) Todas estão erradas.

DCL

- É o tipo de linguagem que interage com a parte de segurança do Banco de Dados.

São comandos DCL:

- GRANT
- REVOKE

Exemplo DCL – GRANT

- **Para permissão de execução em todas as Procedures do Banco de Dados**
- **GRANT EXECUTE TO USUARIO**

- **Para permissão de execução em apenas uma Procedure específica**
- **GRANT EXECUTE ON USP_MINHA_PROC TO USUARIO**

Exemplo DCL – REVOKE

- **Remove a permissão de execução em todas as Procedures do Banco de Dados**
- **REVOKE EXECUTE TO USUARIO**

- **Remove a permissão de execução em apenas uma Procedure específica**
- **REVOKE EXECUTE ON USP_MINHA_PROC TO USUARIO**

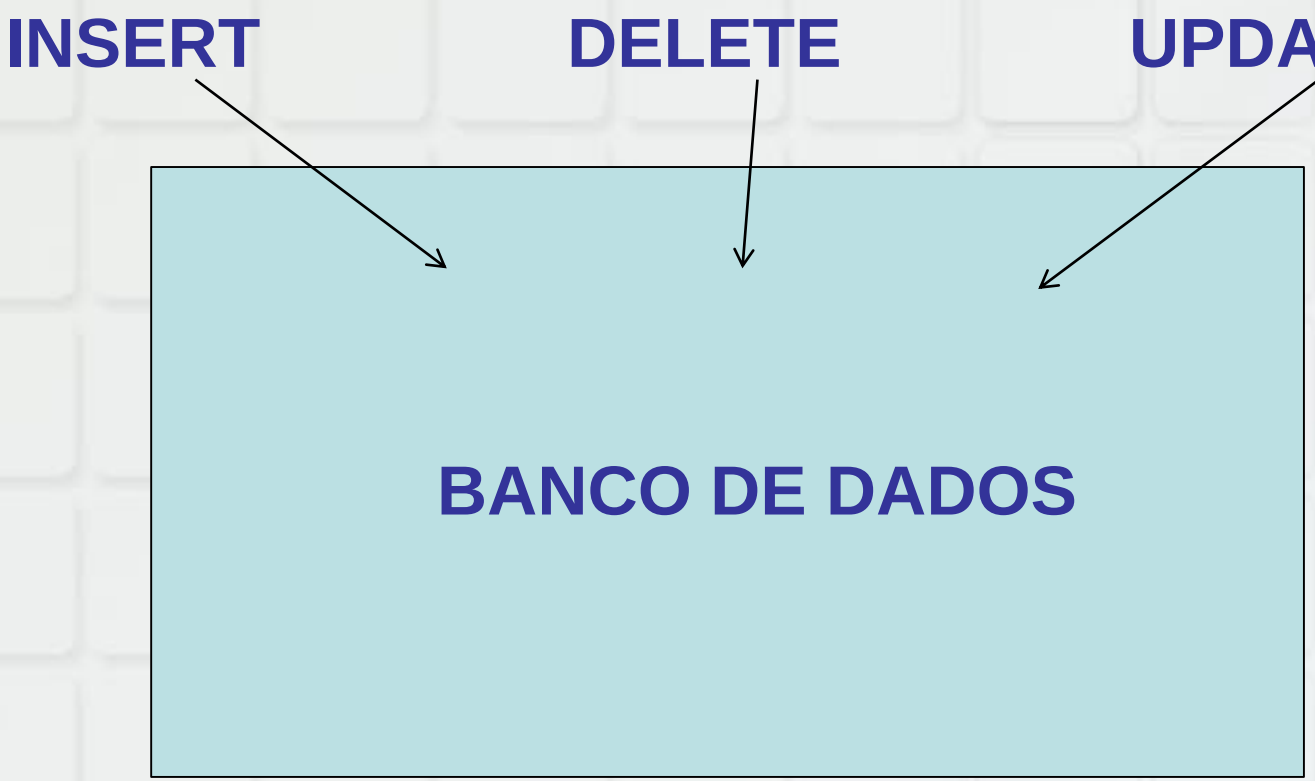
DTL

- É o tipo de linguagem que cuida da parte de transação.
- São comandos da DTL:
- BEGIN TRANSACTION
- COMMIT
- ROLLBACK

Exemplo DTL

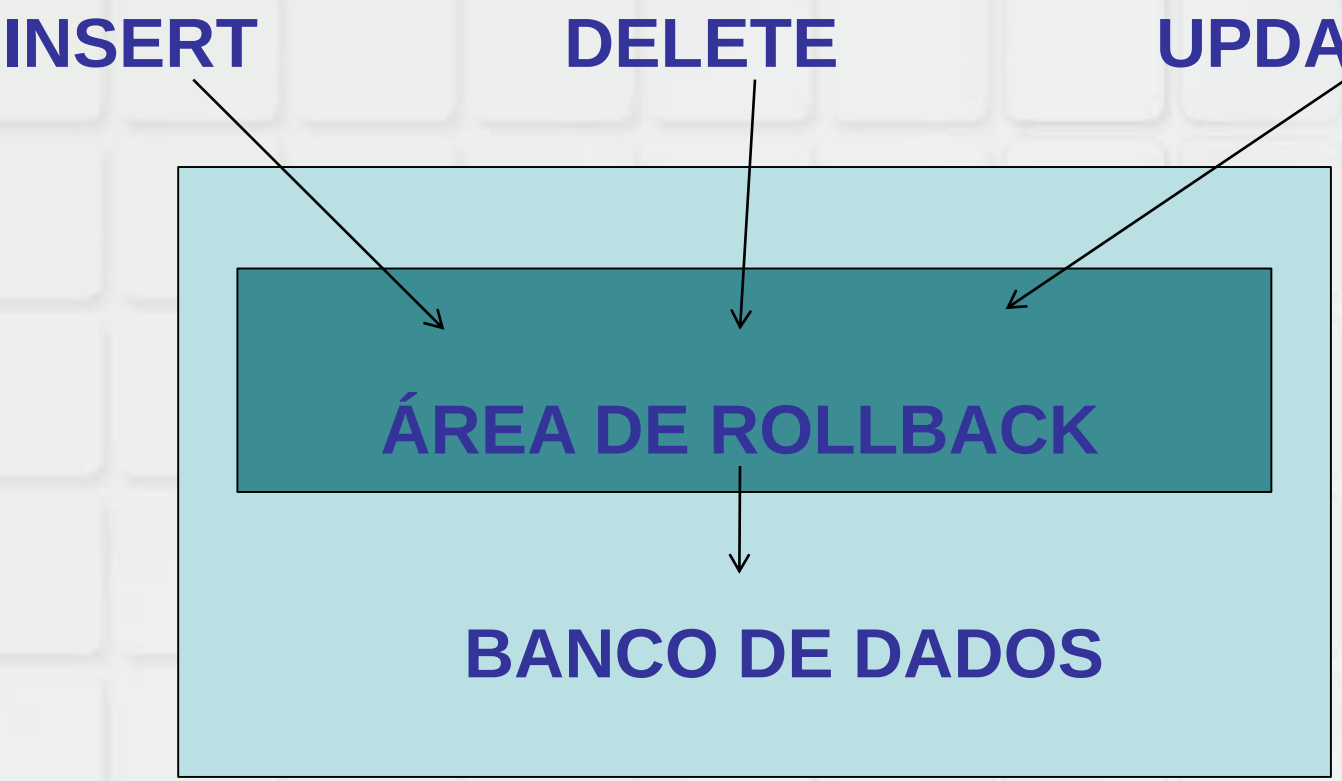
- DECLARE
  - .
- BEGIN
- BEGIN TRANSACTION
  - .
  - .
  - .
- COMMIT / ROLLBACK
- END

Exemplo DTL



Fonte: autoria própria.

Exemplo DTL



Fonte: autoria própria.

Exemplo DTL

- O comando COMMIT confirma / efetiva uma operação que causa alteração nos dados de uma tabela.
- O comando ROLLBACK desfaz a alteração.
- Enquanto nenhuma das instruções for processada, as alterações ficam esperando em um espaço chamado "área de *rollback*".
- Os comandos DTL são normalmente usados dentro de Stored Procedures.

DQL

- **É o tipo de linguagem que cuida da parte de *queries* (consultas) com os dados.**
- **Apesar de interagir com os dados, como a DML, ela não provoca alterações nos dados, serve apenas para consulta.**

São comandos DQL:

- **SELECT**

Observação: em algumas bibliografias, o comando SQL está dentro de DML.

Exemplo DQL

- SELECT *
- FROM ALUNO

  =
- SELECT RA, NOME
- FROM ALUNO
- Selecione
- * = Todas as colunas
- Tabela ALUNO

Exemplo DQL

- SELECT *
- FROM ALUNO
- WHERE RA = ‘1234567’

Observação: assim como o UPDATE e o DELETE, o comando SELECT também pode utilizar a cláusula WHERE, sendo opcional, sem trazer nenhum prejuízo além de um tempo maior de espera em caso de não utilização.

Interatividade

Assinale a alternativa correta.

a) **O comando SELECT necessita obrigatoriamente da cláusula WHERE.**

b) **ROLLBACK é um comando DTL que desfaz uma ação dentro de uma Stored Procedure.**

c) **Todos os comandos necessitam do comando COMMIT para serem efetivados.**

d) **Ao utilizarmos uma transação, automaticamente o Banco de Dados cria uma área de ROLLOUT.**

e) **Todas as alternativas estão corretas.**

UNIP
Interativa

Resposta

Assinale a alternativa correta.

a) O comando SELECT necessita obrigatoriamente da cláusula WHERE.

b) ROLLBACK é um comando DTL que desfaz uma ação dentro de uma Stored Procedure.

c) Todos os comandos necessitam do comando COMMIT para serem efetivados.

d) Ao utilizarmos uma transação, automaticamente o Banco de Dados cria uma área de ROLLOUT.

e) Todas as alternativas estão corretas.

O comando SELECT

- O comando SELECT é o comando mais utilizado dentro de um Banco de Dados.
- Ele serve para retornar dados de uma ou mais tabelas.
- Para retornar dados de mais de uma tabela, utilizamos o comando JOIN.

Sintaxe do SELECT

Com uma tabela

- **SELECT CAMPO1, CAMPO2**
- **FROM TABELA1**

Sintaxe do SELECT

Com mais de uma tabela

- SELECT TABELA1.CAMPO1
- , TABELA2.CAMPO1
- FROM TABELA1
- INNER JOIN TABELA2
- ON TABELA1.CAMPO1 = TABELA2.CAMPO1

Joins

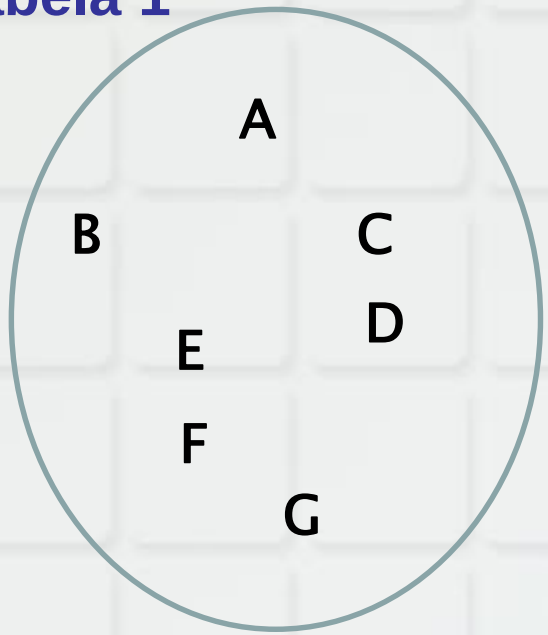
- **Joins ou junções são a forma que o SQL usa para retornar dados de duas ou mais tabelas que possuam algum relacionamento entre si.**

Entendendo Joins por meio da teoria dos conjuntos

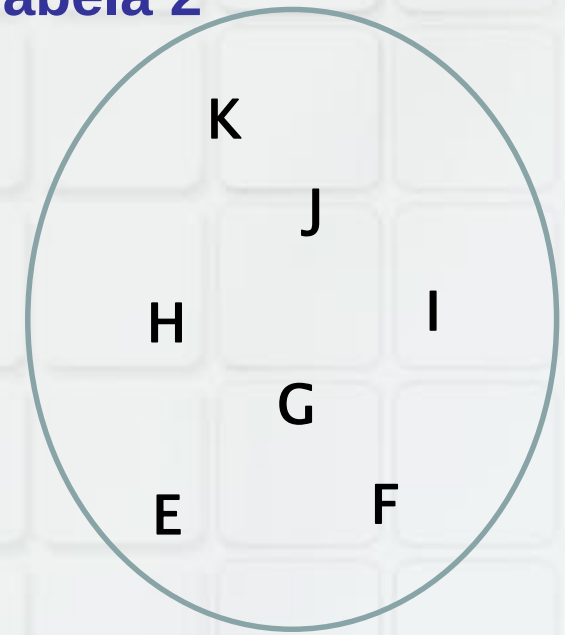
- Banco de Dados em si utiliza muito da teoria dos conjuntos.
- Cada tabela é / armazena um conjunto de dados sobre um determinado assunto.

Entendendo Joins por meio da teoria dos conjuntos

Conjunto / Tabela 1



Conjunto / Tabela 2

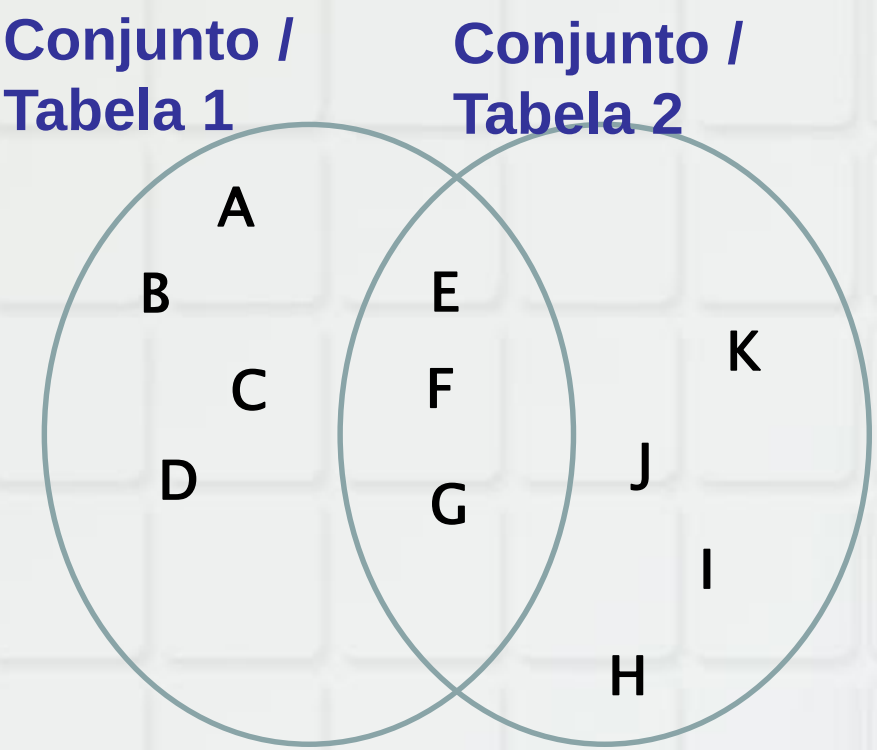


Fonte: autoria própria.

Inner Join

- É o tipo de Join mais usado.
- Retorna apenas os dados que existem em comum nas duas tabelas.

Inner Join



Fonte: autoria própria.

Sintaxe SQL para Joins – INNER JOIN

- SELECT *
- FROM TB_1
- INNER JOIN TB_2
- ON TB_1.COD = TB_2.COD

Resultado – INNER JOIN

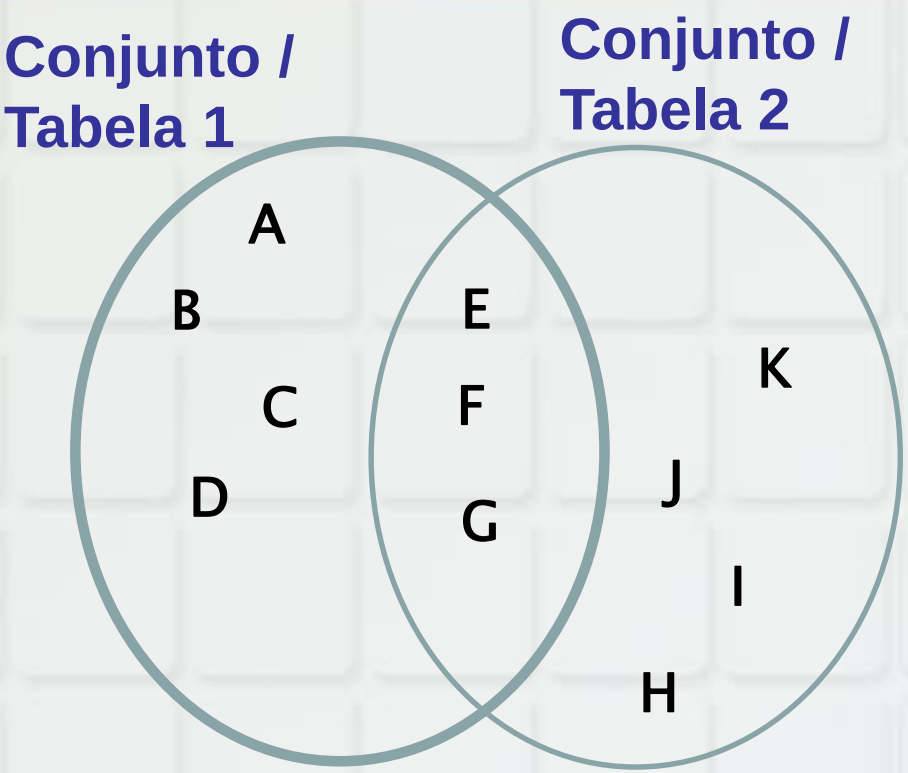
TB_1	TB_2
E	E
F	F
G	G

Fonte: autoria própria.

Left Join

- Usado quando é necessário retornar todos os dados de uma tabela, mesmo que não existam na outra tabela
- O Left retorna os dados da tabela da esquerda.

Left Join



Fonte: autoria própria.

Sintaxe SQL para Joins – LEFT JOIN

- SELECT *
- FROM TB_1
- LEFT JOIN TB_2
- ON TB_1.COD = TB_2.COD

Resultado – LEFT JOIN

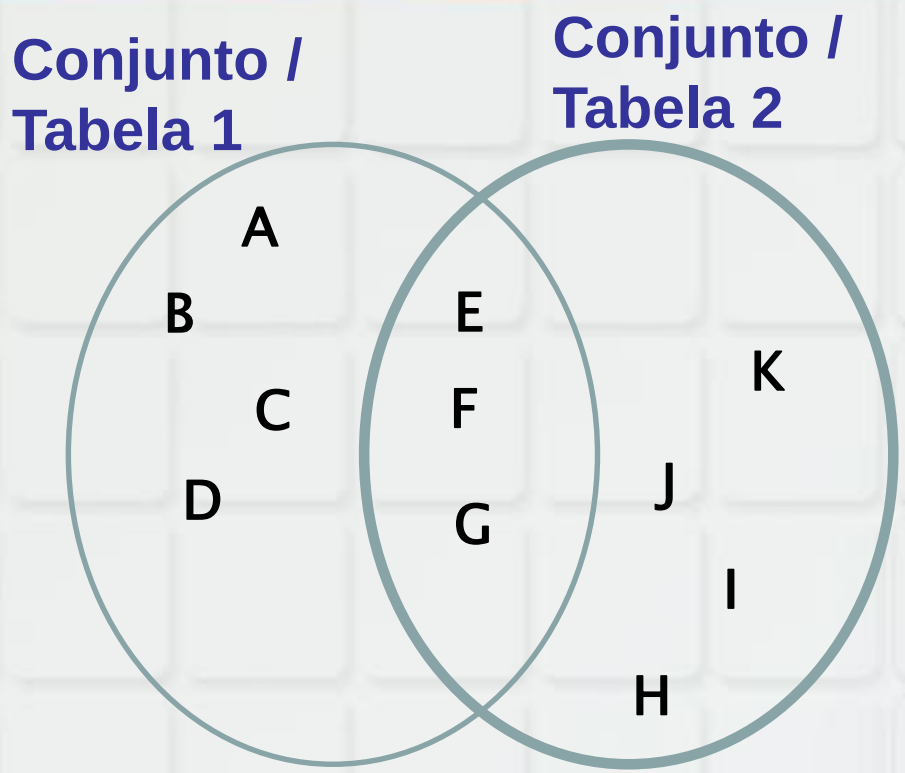
TB_1	TB_2
A	Null
B	Null
C	Null
D	Null
E	E
F	F
G	G

Fonte: autoria própria.

Right Join

- Assim como o Left, é usado quando é necessário retornar todos os dados de uma tabela, mesmo que não existam na outra tabela.
- O Right retorna os dados da tabela da direita.

Right Join



Fonte: autoria própria.

Sintaxe SQL para Joins – RIGHT JOIN

- SELECT *
- FROM TB_1
- RIGHT JOIN TB_2
- ON TB_1.COD = TB_2.COD

Resultado – LEFT JOIN

TB_1	TB_2
E	E
F	F
G	G
Null	I
Null	J
Null	K
Null	L

Fonte: autoria própria.

Entendendo esquerda e direita

LADO ESQUERDO

LADO DIREITO

```
SELECT *
FROM
TB_ESQUERDA
INNER JOIN TB_DIREITA
```

O LEFT JOIN retorna os dados da TB_ESQUERDA

O RIGHT JOIN retorna os dados da TB_DIREITA

Interatividade

Ao realizar uma consulta em duas tabelas utilizando um LEFT JOIN, o resultado será:

a) **todos os registros das duas tabelas;**

b) **apenas os registros da primeira tabela;**

c) **todos os registros da primeira tabela mais os registros da segunda tabela que tiverem referência na primeira.**

d) **apenas registros que forem comuns em ambas;**

e) **todos os registros da segunda tabela mais os registros da primeira tabela que tiverem referência na segunda.**

Resposta

Ao realizar uma consulta em duas tabelas utilizando um LEFT JOIN, o resultado será:

a) **todos os registros das duas tabelas;**

b) **apenas os registros da primeira tabela;**

c) **todos os registros da primeira tabela mais os registros da segunda tabela que tiverem referência na primeira.**

d) **apenas registros que forem comuns em ambas;**

e) **todos os registros da segunda tabela mais os registros da primeira tabela que tiverem referência na segunda.**

ATÉ A PRÓXIMA!