

Banco de Dados

INTRODUÇÃO

```

1 declare @vtabela_id int,
2         @vtabela_nm varchar(256),
3         @vschema_nm varchar(256),
4         @vreferencia_id int,
5         @vreferencia_nm varchar(256),
6         @vParam varchar(500)
7
8 declare @tbDependencias table (
9         tabela_id int,
10        tabela_nm varchar(256),
11        schema_nm varchar(256),
12        referencia_id int,
13        referencia_nm varchar(256)
14    )
15
16 declare cr_tabelas cursor local for
17 select t.object_id, t.name as 'tabela', s.name as 'schema'
18 from sys.tables as t
19 inner join sys.schemas as s on t.schema_id = s.schema_id
20 where t.type = 'U'
21 order by s.name, t.name
22
23 open cr_tabelas
24 fetch next from cr_tabelas
25 into @vtabela_id, @vtabela_nm, @vschema_nm
26
27 while @@FETCH_STATUS = 0
28 begin
29     set @vParam = @vschema_nm + '.' + @vtabela_nm
30
31     insert into @tbDependencias (tabela_id, tabela_nm, schema_nm, referencia_id, referencia_nm)
32     select @vtabela_id, @vtabela_nm, @vschema_nm, referencing_id, referencing_nm
33     from sys.dm_sql_referencing_entities(@vParam, 'OBJECT')
34 end
35 fetch next from cr_tabelas
36 into @vtabela_id, @vtabela_nm, @vschema_nm
37 end
38 close cr_tabelas
39 deallocate cr_tabelas
40
41 Select * from @tbDependencias order by schema_nm, tabela_nm, referencia_nm
42

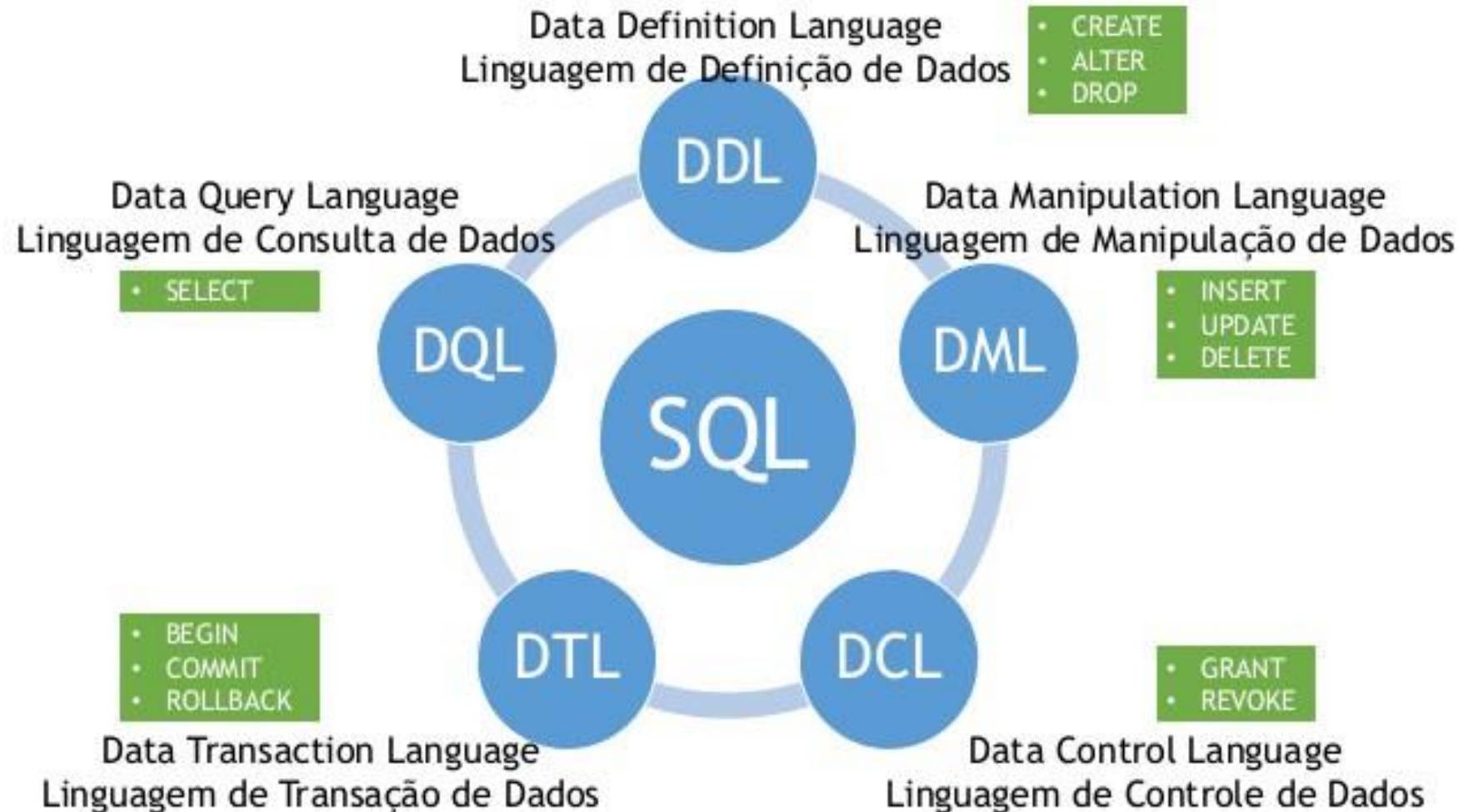
```

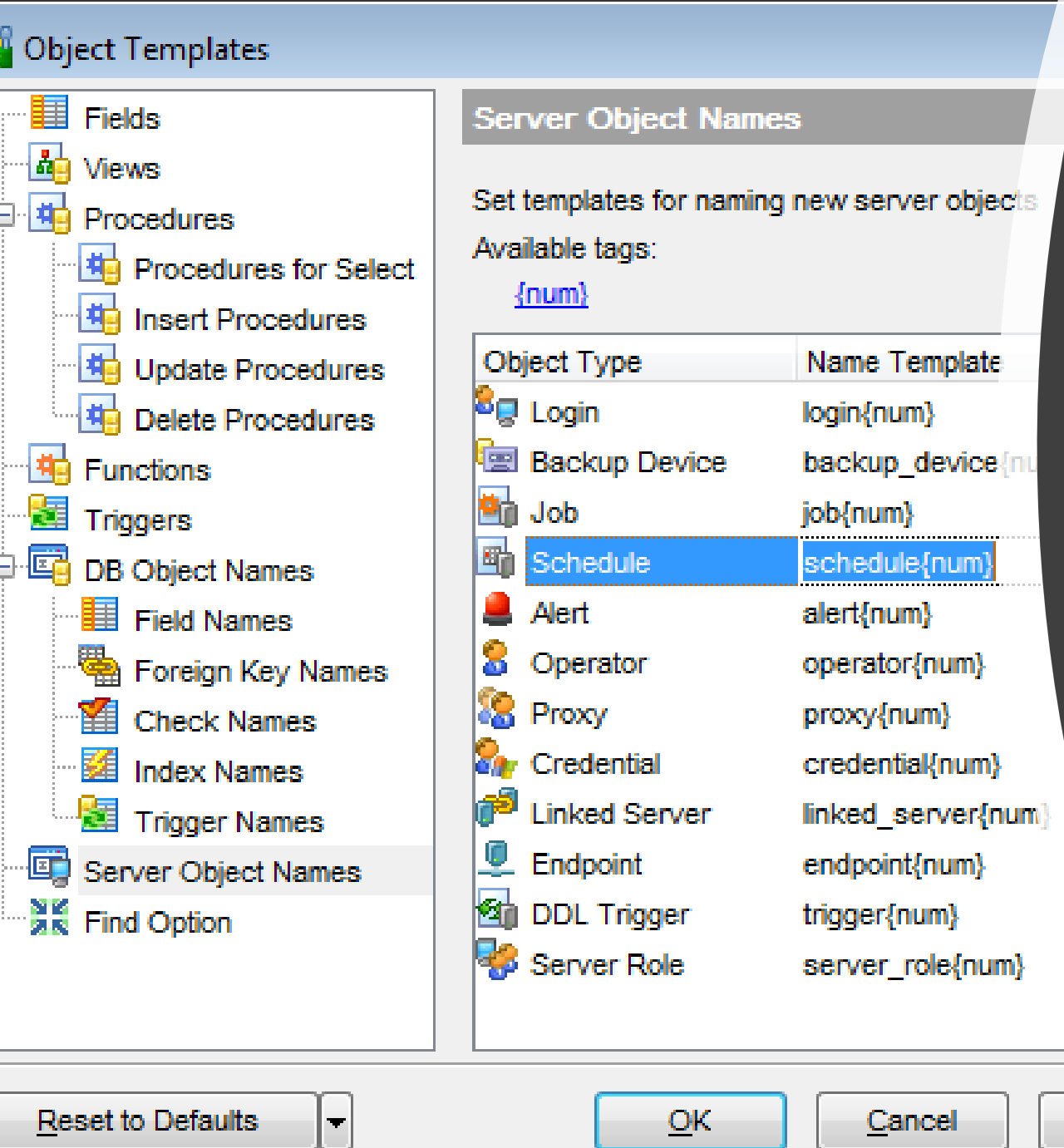
# TRANSACT SQL (T-SQL)

A linguagem Transact-SQL é uma extensão ao padrão SQL-92, sendo a linguagem utilizada por desenvolvedores na construção de aplicações que manipulam dados mantidos no SQL Server. Seus comandos podem ser classificados em quatro grupos, de acordo com sua função:

- DDL (Linguagem de Definição de Dados)
- DML (Linguagem de Manipulação de Dados)
- DCL (Linguagem de Controle de Dados)
- DTL (Linguagem de Transação de Dados)
- DQL – Linguagem de Consulta de Dados

# SQL e suas variações





# TIPOS DE OBJETOS NO MS SQL SERVER

- Banco de Dados
- Tabelas
- Índices
- Views
- Procedures
- Functions
- Triggers
- Logins
- Users
- Roles

# DDL – Linguagem de Definição de Dados

Esse subconjunto apoia a criação de **objetos** no banco de dados, alterar a estrutura da base de dados ou deletar o banco de dados. Seus principais comandos são:

- CREATE
- ALTER
- DROP



# DDL – Sintaxe

*CREATE Tipo\_de\_Objeto Nome\_do\_Objeto*

*CREATE Tipo\_de\_Objeto Nome\_do\_Objeto*

*DROP Tipo\_de\_Objeto Nome\_do\_Objeto*

*Exemplo:*

*CREATE DATABASE MeuBanco*

*ALTER VIEW MinhaView*

*DROP PROCEDURE MinhaProcedure*

## TIPOS DE OBJETOS:

- Banco de Dados
- Tabelas
- Índices
- Views
- Procedures
- Functions
- Triggers
- Logins
- Users
- Roles

# Banco de Dados

Variáveis, Tratamento de Erro, Condicionais e Laço

# Banco de Dados

Variáveis, Tratamento de Erro, Condicionais e Laço

# Variáveis

- Em algumas situações pode ocorrer a necessidade de armazenar valores de forma temporária para posteriormente ser utilizado.
- Assim como acontece nas linguagens de programação, a solução para essa necessidade é com a utilização de variáveis.
- No Microsoft SQL Server a utilização das variáveis (declaração, atribuição, e exibição) acontece sempre dentro de um mesmo escopo de execução, em outras palavras, não é permitido declarar uma variável executar o comando em um escopo e em seguida quer acessar a variável em outro escopo.
- A instrução DECLARE inicializa uma variável.
- Toda variável deve começar com o caractere @ seguido do tipo de dado.



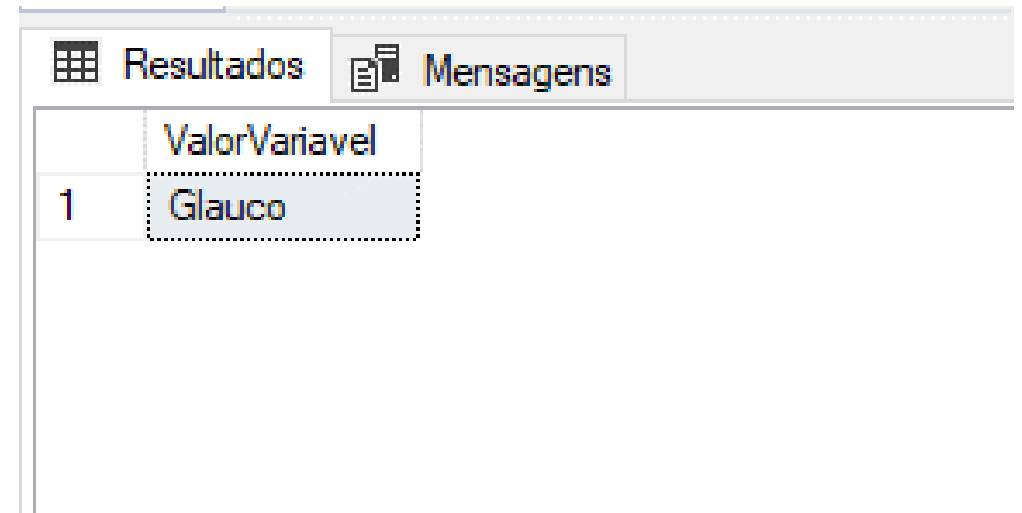
# Variáveis - Sintaxe

```
-- declarando uma variável.  
declare @Nome varchar(100)  
  
-- atribuindo valor à variável.  
set @Nome = 'Glauco'  
  
-- exibindo o conteúdo de uma  
variável.  
select @Nome as ValorVariavel
```

Resultados		Mensagens	
	ValorVariavel		
1	Glauco		

# Variáveis – Sintaxe 2

```
-- declarando uma variável.  
declare @Nome varchar(100) = 'Glauco'  
  
-- exibindo o conteúdo de uma variável.  
select @Nome as ValorVariavel
```



The screenshot shows a SQL query result window with two tabs: 'Resultados' (Results) and 'Mensagens' (Messages). The 'Resultados' tab is active, displaying a single row of data. The column is labeled 'ValorVariavel' and the value is 'Glauco'.

	ValorVariavel
1	Glauco

# Variáveis – Sintaxe 3

```
-- declarando mais de uma variável.  
declare @Nome varchar(100), @Idade tinyint  
  
-- atribuindo valor as variáveis.  
set @Nome = 'Glauco'  
set @Idade = 34  
  
-- exibindo o conteúdo das variáveis.  
select @Nome as VariavelNome,  
       @Idade as VariavelIdade
```

Resultados		Mensagens
	VariavelNome	VariavelIdade
1	Glauc <u>o</u>	34

# Variáveis – Sintaxe 4

- -- Atribuindo para uma variável o conteúdo existente na tabela.
- `declare @NomeDisciplina varchar(50)`
- `set @NomeDisciplina = (select Nome from Disciplina whereCodigoDisciplina = 1)`
- `select @NomeDisciplina`

Resultados		Mensagens	
		(Nenhum nome de coluna)	
1		Banco de Dados	

# Variáveis – GLOBAIS

- Retornam uma informação do Servidor.
- Não podem ser criadas pelo usuário. (ready-only)
- Elas possuem como prefixo @@
- Podemos obter o conteúdo destas variáveis através do comando.
- `SELECT @@Nome_Variavel`
- `PRINT @@Nome_Variavel`





# Variáveis Globais: Exemplos

SELECT

`@@CONNECTIONS` AS 'Retorna o número de conexões desde que o SQL iniciou',

`@@LANGUAGE` AS 'Idioma',

`@@SERVERNAME` AS 'Nome do Servidor',

`@@SPID` AS 'Número do Processo atual',

`@@IDENTITY` AS 'Retorna o ultimo valor Identity inserido',

`@@ROWCOUNT` AS 'Retorna o número de linhas do último comando executado. ',

`@@ERROR` AS 'Retorna o código do ultimo erro ocorrido',

`@@VERSION` AS 'Versão do SGBD'

# Tratamento de Erro – Try Catch

- São comandos utilizados para realizar o tratamento de erro.
- Similar ao encontrado em outras linguagens de programação como Java e C#
- Utilize dentro do bloco Try o comando passível de erro, e coloque dentro do bloco Catch o tratamento para ser realizado em caso de um erro acontecer.

```
BEGIN TRY
    { sql_statement }
END TRY
BEGIN CATCH
    { sql_statement }
END CATCH
```

# Tratamento de Erro – Funções que detalham o erro

- No escopo de um bloco CATCH, as funções de sistema abaixo podem ser usadas para obter informações sobre o erro que causou a execução do bloco CATCH.
- **ERROR\_NUMBER()** retorna o número do erro.
- **ERROR\_SEVERITY()** retorna a severidade.
- **ERROR\_STATE()** retorna o número do estado do erro.
- **ERROR\_PROCEDURE()** retorna o nome do procedimento armazenado ou do gatilho no qual ocorreu o erro.
- **ERROR\_LINE()** retorna o número de linha dentro da rotina que causou o erro.
- **ERROR\_MESSAGE()** retorna o texto completo da mensagem de erro. O texto inclui os valores fornecidos para qualquer parâmetro substituível, como cumprimentos, nomes de objeto ou horas.

# Tratamento de Erro – THROW

- Utiliza-se o THROW para disparar um erro, ele recebe 3 argumentos:
- **error\_number** = É uma constante ou uma variável que representa a exceção. error\_number é int e precisa ser maior ou igual a 50000 e menor ou igual a 2147483647.
- **Message** = É uma cadeia de caracteres ou variável que descreve a exceção. message é nvarchar(2048) .
- **State** = É uma constante ou variável entre 0 e 255 que indica o estado a ser associado à mensagem. state é tinyint.
- Exemplo:
- `THROW 51000, 'The record does not exist.', 1;`

# Tratamento de Erro – Exemplo

```
declare @Numerador decimal(5,2),
        @Denominador decimal(5,2),
        @Total decimal(5,2)

set @Numerador = 10
set @Denominador = 0

begin try
    set @Total = @Numerador / @Denominador
end try
begin catch
    THROW 51000, 'Não é possível fazer o calculo', 1;
end catch
```

## Resultados

Mensagem 51000, Nível 16, Estado 1, Linha 11  
Não é possível fazer o calculo

Horário de conclusão: 2021-01-31T15:27:31.5953595-03:00



# Condicionais

- É possível utilizar condicionais no SQL, assim como ocorre em uma linguagem de programação, o funcionamento é o mesmo, o que irá variar é a sintaxe.
- No SQL é possível utilizar o comando IF ELSE conforme exemplos a seguir:

# Comando IF ELSE (Sintaxe)

## Sintaxe:

```
IF condicao BEGIN  
    comandoIF
```

```
END
```

```
ELSE IF condicao BEGIN  
    ComandoElseIF
```

```
END
```

```
ELSE BEGIN  
    ComandoElse
```

```
END
```

# Comando IF ELSE (Exemplo)

```
declare @Salario money
set @Salario = 5000

-- Se o salário for menor ou igual 1000 desconta 3% de imposto.
-- Se o salário for maior que 1000 e menor ou igual 5000 desconta
10% de imposto.
-- Se o salário for maior que 5000 desconta 30% de imposto.
IF @Salario <= 1000 BEGIN
    set @Salario = @Salario - (@Salario * 0.03)
    print 'Desconto de 3%'
END
ELSE IF @Salario > 1000 and @Salario <= 5000 BEGIN
    set @Salario = @Salario - (@Salario * 0.1)
    print 'Desconto de 10%'
END
ELSE BEGIN
    set @Salario = @Salario - (@Salario * 0.3)
    print 'Desconto de 30%'
END

select @Salario as SalarioDescontadoImposto
```

Resultados		Mensagens
SalarioDescontadoImposto		
1	4500,00	

# Comando WHILE

- Embora não seja comum, é possível executar laços no SQL.
- Dependendo da quantidade de iterações o comando pode-se tornar “pesado” ao SGBD.
- Cuidado com o loop infinito, incremente a variável de controle!

# Comando WHILE

- Sintaxe:

```
WHILE condicao BEGIN
```

```
    comando
```

```
    incrementa variável de controle
```

```
END
```



# Comando WHILE (Exemplo)

```
declare @i tinyint,  
        @total tinyint,  
        @valor tinyint  
  
set @i = 1  
set @total = 5  
  
while @i <= @total begin  
    set @valor = @total + @i  
    select @valor  
    set @i = @i + 1 --incrementando @i  
end
```

Resultados		Mensagens
	(Nenhum nome de coluna)	
1	6	
	(Nenhum nome de coluna)	
1	7	
	(Nenhum nome de coluna)	
1	8	
	(Nenhum nome de coluna)	
1	9	
	(Nenhum nome de coluna)	
1	10	



VAMOS POR A MÃO  
NA MASSA!!!





# VAMOS POR A MÃO NA MASSA!!!

- Siga as orientações do professor na aula.

# Banco de Dados

DTL - Data Transaction Language - Linguagem de Transação de Dados.



# O que são Transações SQL?

- Unidade de trabalho executada em um banco de dados.
- São realizadas em uma ordem lógica.
- Todas as transações possuem início e fim.
- Uma transação pode ser salva ou desfeita.
- Em resumo uma transação é uma execução de uma ou mais operações no Banco de Dados.



# Transações

- Todo Banco de Dados possui seu log de transação.
- O T-Log é um componente que registra todas as transações e modificações efetuadas em um Banco de Dados.
- Se houver alguma falha no sistema, o log de transação pode ser utilizado para retornar o sistema para um estado consistente.

# Princípios de uma transação

- São requisitos que sempre deverão ser seguidos pelo SGBD em uma transação.
- Atomicidade
- Consistência
- Isolamento
- Durabilidade ou Persistência

# Princípios de uma transação

- **Atomicidade:** As transações são atômicas, isto é, ou tudo ou nada. Todas as operações devem estar completas ou nenhuma operação será realizada;
- **Consistência:** As transações preservam a consistência do banco de dados. Antes de iniciar a transação o banco de dados deve estar em um estado consistente e permanecendo assim após a execução da transação;

# Princípios de uma transação

- **Isolamento:** As transações são isoladas uma das outras. Existem várias transações ocorrendo simultaneamente no banco de dados, porém os dados que elas estão atualizando devem estar isolados um do outro, isto é, duas transações distintas não podem estar atualizando o mesmo item de dados em transações diferentes;
- **Durabilidade ou Persistência:** Depois de efetivada as transações, elas devem permanecer no banco de dados mesmo ocorrendo uma falha no sistema;

# Comandos Transacionais

- Os seguintes comandos são utilizados para gerenciar as transações
- BEGIN TRANSACTION
- ROLLBACK TRANSACTION
- COMMIT TRANSACTION

# BEGIN TRANSACTION

- Comando Transacional utilizado para **iniciar** uma Transação.
- Sintaxe:
  - BEGIN TRANSACTIONou também podem ser:
  - BEGIN TRAN



# ROLLBACK TRANSACTION

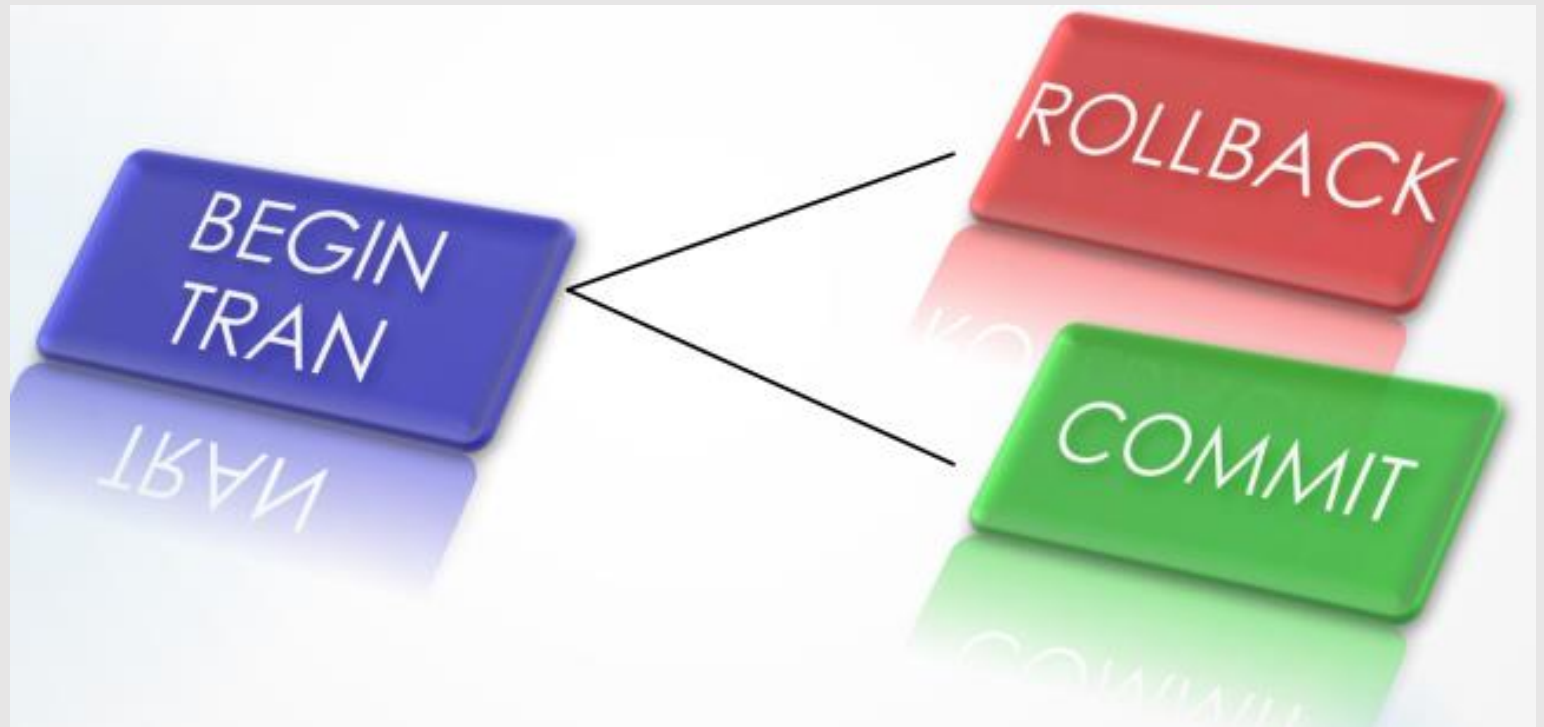
- Comando Transacional utilizado para desfazer os comandos executados após o início da Transação.
- Sintaxe:
  - ROLLBACK TRANSACTIONou também podem ser:
  - ROLLBACK

# COMMIT TRANSACTION

- Comando Transacional utilizado para efetivar os comandos executados após o início da Transação.
- Sintaxe:
  - COMMIT TRANSACTIONou também podem ser:
  - COMMIT

# Comandos Transacionais

Os comandos **ROLLBACK** e **COMMIT** finalizam a Transação, e somente deverá ser executado ou um ou outro.



# FICA A DICA

- Mantenha as transações curtas.
- Lembre-se que transações restringem o acesso aos dados até a sua finalização, isso pode gerar tempo de espera aos usuários.





VAMOS POR A MÃO  
NA MASSA!!!





# VAMOS POR A MÃO NA MASSA!!!

- Siga as orientações do professor na aula.