

**ATENTO**




## DOCUMENTAÇÃO TÉCNICA DE RPA

**COE-DOC-0001**


# Padrões para Banco de Dados



	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

## 1 Índice

2	Sobre o Documento.....	3
2.1	Informações Técnicas .....	3
2.2	Objetivo .....	3
2.3	Tecnologias .....	3
2.4	Informações do cliente .....	3
2.5	Necessidades .....	3
2.6	Versionamento .....	3
2.7	Destino.....	3
3	Padronização de banco de dados.....	4
3.1	Por que padronizar? .....	4
4	Banco de dados.....	5
4.1	Caso prático .....	5
5	Usuário.....	6
5.1	Caso prático .....	6
6	Tabela .....	6
6.1	Regras importantes .....	6
6.2	Campos básicos .....	6
6.3	Caso Prático .....	7
6.4	Campos da Tabela.....	8
6.4.1	Regras importantes .....	8
6.4.2	Ordenação .....	9
7	Indices.....	<b>Erro! Indicador não definido.</b>
8	Chaves Primárias.....	10
9	Chaves Estrangeiras .....	10
10	View .....	10
11	Stored Procedure.....	11
12	Trigger.....	12
13	Váriavei .....	<b>Erro! Indicador não definido.</b>

	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

## 2 Sobre o Documento

### 2.1 Informações Técnicas

### 2.2 Objetivo

Padronizar a nomenclatura na criação de aplicações no Banco de Dados

### 2.3 Tecnologias

SQL Server, TSQL

### 2.4 Informações do cliente

O Documento foi escrito par a área de COE RPA com foco nos desenvolvedores de Automação

### 2.5 Necessidades

O Foco desse documento é para Microsoft SQL SERVER


### 2.6 Versionamento

Versão	Data	Criador	Cargo	Descrição
1	01/10/2020	Rafson Rafael	Responsável Técnico	Criação do documento
2	06/10/2020	Marcos Palladino	Programador de Automação	Complemento
3	14/12/2020	Rafson Rafael	Responsável Técnico	Sistema de Carga
4	28/12/2021	Rafson Rafael	Responsável Técnico	Ajustes e adições sobre UniqueIdentifier
5	11/01/2022	Rafson Rafael	Responsável Técnico	Correções e adição do Schema

### 2.7 Destino

Esse documento é de propriedade da Atento e seu conteúdo é confidencial e voltado apenas para projetos internos da Atento, Interfile e RBrasil. Este documento não pode ser modificado, repassado ou reproduzido em nenhuma parte sem a expressa autorização e conhecimento dos responsáveis do setor.

Considerações

	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

## 3 Padronização de banco de dados

### 3.1 Por que padronizar?

Quando você olha para um banco de dados e sua estrutura no SQL Management Studio você tem a dimensão exata do que é uma tabela, uma procedure, uma função ou qualquer outra coisa.

Quando você está olhando o código de uma aplicação e uma chamada feita através do banco de dados é feita, ela pode ser qualquer coisa como uma Procedure, uma View, Uma Função, ou mesmo uma tabela.

O intuito da padronização é fazer qualquer pessoa que conheça o padrão entender a estrutura e do que se trata aquele objeto à primeira vista, sem análises demoradas.

Existem algumas premissas básicas antes de entendermos a padronização.

1. Não se deve usar espaços em branco em nenhum objeto do banco de dados
2. Não se deve utilizar hífen nem caracteres especiais como exclamação, interrogação, acentos, Ç. Devemos utilizar apenas caracteres do alfabeto comum, underline “\_”
3. Não utilizar, mesmo que de forma camuflada as palavras reservadas do SQL SERVER como INSERT, DELETE, SELECT, PROCEDURE, VIEW e etc
4. Respeitar o limite máximo de 30 caracteres
5. Não utilizar verbos como Fazer, Encontrar, Procurar e etc.
6. Não utilizar plural, apenas singular
7. Não usar preposições
8. Não usar números
9. Não usar nomes próprios
10. Separação de nomes deve ser feita com underline “\_”
11. Crie nomes objetivos sem floreios
12. O nome do objeto deve ser direto e não ter duas interpretações
13. Todos objetos devem ter prefixo.

Vamos então aos padrões específicos


### 3.2 Uniqueidentifier

O nosso padrão irá utilizar o tipo de dado UniqueIdentifier como chave primária das tabelas, é um formato gerado automaticamente pelo SQL Server, é uma chave com 32 dígitos. Nesse manual vamos identificar o campo UNIQUEIDENTIFIER também como GUID (**G**lobal **U**nique **I**dentificator)

Parece um contrassenso a substituição do formato INT por UNIQUEIDENTIFIER, mas existem vantagens práticas

#### 3.2.1 Vantagens

- É um identificador exclusivo dentro do banco de dados inteiro
- Cada registro do banco pode ser identificado separadamente
- Permite a distribuição de dados em servidores distribuídos
- Permite a mescla de dados distribuídos de maneira simplificada

	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

- O ID pode ser gerado tanto pelo banco, como pela aplicação sem a necessidade de uma checagem
- Como não é um sequencial, dificulta invasões onde o invasor tente uma sequência de códigos
- Aumenta o limite de tamanho dos registros de uma tabela que no INT é de -2.147.483.648 até 2.147.483.647 que dá em torno de 4 trilhões de registros. Já no GUID 5.316.911.983.139.663.491.615.228.241.121.400.000 ou seja cinco undecilhões, daria para dar um GUID para cada grão de areia na terra e ainda sobraria
- Torna mais fácil a busca de dados entre as entidades
- Os códigos no mesmo padrão já são utilizados por APIs e outros processos padronizados para passagem de dados, o que torna isso mais seguro.

### 3.2.2 Desvantagens

- É quatro vezes maior que o INT
- Pode Aumenta o tamanho do banco
- O Debug das aplicações pode ser um pouco mais complexo vide que visualmente “where userid= '{BAE7DF4-DDF-3RG-5TY3E3RF456AS10}’” não é tão agradável quanto “where userid=122”

Precisam ser gerados de forma sequencial para melhorar a performance do banco, se não os dados podem acarretar lentidão

## 4 Banco de dados

O prefixo para criação de banco de dados é o “DB\_” e deve ser utilizado sempre após o nome do banco.

O Nome do banco deve estar em Pascal Case


Sintaxe:

`DB_{[A..Z][a..z]}` -> DB\_Xxxxx, onde: Xxxxx – indica o nome da aplicação que o banco de dados irá atender.

### 4.1 Caso prático

O Nome do banco de dados é a base do banco. O Comum é que o nome identifique o negócio ou o produto e depende muito do contexto

Por exemplo:

	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

Temos uma empresa chamada Bananinha e estamos fazendo um sistema chamado “Sistema de Pagamentos”.

O nosso banco de dados é compartilhado entre diversos projetos de diversas empresas, entre elas a Maçã SA, Mandioca Industries e Capoeira Sistemas.

Nesse caso devemos identificar o cliente e o objetivo da seguinte maneira:

DB\_BananinhaPagamento

## 5 Usuário

O nome do usuário do banco de dados segue um padrão simples

USR\_{[A..Z][{a.z}]} USR\_Xxxxx, onde: Xxxxx – indica o nome do usuário referenciando o banco de dados ou o programa

### 5.1 Caso prático

O nosso “Sistema de Pagamentos” citado anteriormente precisa de um acesso então o nome deverá ser algo como

USR\_SistemaPagamento

Esse usuário pode ser uma espécie de administrador do banco, mas se você quiser algo mais específico, um usuário que tem apenas a permissão de leitura pode acrescentar um sufixo explicativo como

USR\_SistemaPagamet\_Consulta

## 6 Tabela

O Nome da tabela deverá ser o mais claro possível mostrando de cara qual é o tipo de informação que ela armazena.

TB\_{[A..Z][{a.z}]}->TB\_XxxxXxxx


### 6.1 Regras importantes

1. Utilizar PascalCase ex: TB\_FeriadoMunicipal
2. Limite máximo de 30 caracteres
3. Nome da tabela deve ser sempre no singular
4. Evite abreviações, se necessário use as mais conhecidas para que fique claro
5. Sem acentos, espaços ou caracteres especiais

### 6.2 Campos básicos

TODA TABELA deve conter os seguintes campos

1. Campo “Id” do tipo UniqueIdentifier que é incrementado automaticamente com **NEWID()** que cria um ID Automático.

	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

- 1.1. O campo do tipo UniqueIdentifier utiliza uma cadeia de caracteres conhecida com “Guid” e é tem esse formato: “CB893E23-B3EC-4940-ABBB-46F6035005F8”
- 1.2. A Ideia do Guid é que o registro tenha um código único entre todos os registros do banco de dados.
- 1.3. **IMPORTANTE:** Ao se criar uma tabela deve-se colocar a função no **NewSequentialID()** default do campo para que os registros sejam criados de forma sequencial e possam ser indexados, por exemplo

```
CREATE TABLE #TABELA
(
    ID UNIQUEIDENTIFIER default NEWSEQUENTIALID(),
    VALOR VARCHAR(100)
)
```

Com esse essa informação os Ids serão inseridos automaticamente e serão sequenciais, como no exemplo:

	ID	VALOR
1	98E0861D-EC67-EC11-95A9-F8A2D6F2F329	A
2	99E0861D-EC67-EC11-95A9-F8A2D6F2F329	B
3	9AE0861D-EC67-EC11-95A9-F8A2D6F2F329	C
4	9BE0861D-EC67-EC11-95A9-F8A2D6F2F329	D

Essa sequencia mudará para cada entidade.

2. Campo de “CreatedOn” que deve ter o valor padrão de **GetDate()** que implementa a “Data da Entrada” do registro automaticamente.

Por exemplo:

```
CreatedOn DateTime Default GetDate()
```

## 6.3 Caso Prático

Por exemplo, se o departamento da empresa e o desenvolvimento achou necessário fazer uma tabela para os feriados do ano, a tabela deverá usar a seguinte nomenclatura:

TB\_Feriado

Tabela de Pessoas


TB\_Pessoa

Tabela específica de Funcionários da empresa Abacaxi que foi adquirida pela empresa Bananinha

TB\_FuncionarioAbacaxi

Tabela de Grupo de produtos

TB\_GrupoProduto

	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

## 6.4 Campos da Tabela

Seguindo um padrão similar ao nome da tabela mas sem prefixo, o nome do campo deve ser claro e transmitir imediatamente a ideia do seu conteúdo

Sintaxe:

{ [A..Z][a.z]} -> XxxxxXxxxxXxxxx, onde

Xxxxx – indica o nome do campo

### 6.4.1 Regras importantes

1. Utilizar PascalCase, por exemplo um campo de “Nome Completo” fica “NomeCompleto”
2. Não Utilizar preposições, ex: o Campo “Endereço do Logradouro” fica “EnderecoLogradouro”
3. Siglas todas em maiúsculo Exemplo “Código do FGTS” fica “CodigoFGTS”
4. Utilizar nomes distintos para dados distintos, ou seja não use “EnderecoA”, “EnderecoB”, “EnderecoC” a não ser que o nome seja esse mesmo, utilize algo mais palpável como “EnderecoPrincipal”, “EnderecoComercial”, “EnderecoCobranca” e etc.
5. Nomes sempre no singular
6. Em caso de abreviações utilize as mais conhecidas, de domínio público e limite-se a 4 letras. Exemplo “Nome do Funcionário Responsável do Financeiro” fica “NomeFuncRespFinanc” para que fique claro de se visualizar

### 6.4.2 Relacionamentos

1. O nome do campo que indica o relacionamento deve referenciar diretamente o ID da outra tabela.
  - 1.1. Temos a tabela de Funcionário e a tabela de Setor, que armazena o setor do funcionário, o campo na tabela de funcionário que referencia o setor dele é a “IdSetor” que referencia o código do Setor.

Script e explicações de funcionamento


Vamos então usar um script básico para criar a tabela, a sintaxe é a seguinte

```
CREATE TABLE <<NOME DA TABELA>>(
  Id uniqueidentifier default NewSequentialID() not null,
  <<CAMPOS>>
  CreatedOn Datetime Default Getdate()
)
```

Baseado nessa regra, temos o exemplo:

```
CREATE TABLE TB_TabelaBasica(
```



	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

```
Id uniqueidentifier default NewSequentialID() not null,
Nome varchar(100) Not Null,
Endereco varchar(200) Not Null,
EmailComercial varchar(400) Not Null,
CreatedOn Datetime Default Getdate()
)
```

Lembre-se que os campos Id e DataCriação são campos Padrão e devem estar em **TODAS AS TABELAS**.

Ao fazer um INSERT em nossa tabela TB\_TabelaBasica não precisamos incluir o Id e nem a DataCriação, por exemplo:

```
INSERT INTO
    TB_TabelaBasica(
        Nome,
        Endereco,
        EmailComercial)
    Values(
        'Antonio José do Santos',
        'Rua das Flores, 883 Vila Laranjeiras',
        'antoniojosesantos80@gmail.com'
    )
```

O resultado é algo nesse formato

Results		Messages			
	Id	Nome	Endereco	EmailComercial	CreatedOn
1	52D0A5E9-E39F-4A24-8138-4B5C6701A4A2	Antonio José do Santos	Rua das Flores, 883 Vila Laranjeiras	antoniojosesantos80@gmail.com	2020-10-06 10:33:55.393

Note que os campos de Id e CreateOn ganharam dados e formatos automáticos.

### 6.4.3 Ordenação

Para se fazer uma busca na tabela deve-se sempre fazer a ordenação pelo campo CreatedOn pois ele dita a ordenação da entrada de dados

## 7 Índices


Deve-se utilizar a mesma semântica utilizada para as tabelas.

Sintaxe:

*IX\_{[A..Z][a.z]}{[1..99] | [identificador da chave]} -> IX\_XXXXXXXXXXXX*, ou seja,

IX\_<nome\_da\_tabela><identificador\_da\_chave>nn, onde:

IX\_ - prefixo para identificar que se trata de um índice

	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

Xxxxxxx - nome da tabela a qual o índice está associado

nn – número sequencial da tabela

Xxxxxxx - número sequencial dos índices que se referem a tabela ou identificador da chave

## 8 Chaves Primárias

Para nomear as chaves primárias utilizar:

Sintaxe:

*PK\_{[A..Z][a..z]} -> PK\_Xxxxxx*, onde

PK\_ - indica que é uma chave primária

Xxxxx - indica o nome da tabela a qual o índice está associado

## 9 Chaves Estrangeiras

Para nomear as chaves estrangeiras utilizar:

Sintaxe:

*FK\_{[A..Z][a..z]}\_{[A..Z][a..z]} -> FK\_Xxxxxx\_Xxxxxx*, ou seja,

FK\_<nome da tabela origem(pai)>\_<nome da tabela destino(filho)>, onde:

FK\_ - indica que se trata de uma chave estrangeira


Xxxxx - indica o nome da tabela origem(pai)

Xxxxx - indica o nome da tabela destino(filho)

## 10 View

As Views nada mais são do que amarrações “pré fabricadas” entre tabelas, muitas vezes com dados apresentados de uma forma mais inteligente ou de algum jeito específico, tem o objetivo de mostrar algo direto no sistema ou mesmo deixar com o usuário final mais avançado garantido o formato de dados entregue.

O formato dela deve ser o mesmo da Tabela mas com um prefixo de VW no lugar do TB

	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

VW\_{[A..Z][{a.z}]}->VW\_XxxxXxxx

As regras de nomenclatura de campos devem ser as mesmas das Tabelas para as Views

## 11 Stored Procedure

Procedures são blocos de programação onde podemos trabalhar a inteligência do sistema de forma mais prática, é aqui que podemos construir parte do processamento pesado no banco de dados.

Deve ser prefixada com a sigla “STP\_” seguida do nome da tabela para a stored procedure. Para stored procedures especifica como inclusão (I), exclusão (D), alteração (U) e consulta (S) utilizar a notação específica, apresentada na sintaxe.

Sintaxe:

**STP\_{[ D | I | S | U ]}\_{[A..Z][{a..z}]} -> STP\_X\_Xxxxxx**, ou seja,

STP\_<identificador\_da\_ação\_principal><nome\_da\_stored\_procedure>, onde:

STP – indica que se trata de uma stored procedure

X - Identificador-da-Ação Principal - D – DELETE, I – INSERT, S – SELECT e U – UPDATE, quando houver operações compostas utilizar o identificador em ordem alfabética.

Xxxxxx – nome da stored procedure

Exemplos:

**STP\_I\_[nome-da-sp] - Stored procedure de inclusão.**


Ex: STP\_I\_Funcionario

**STP\_D\_[nome-da-sp] - Stored procedure de exclusão.**

Ex: STP\_D\_Campanha

**STP\_U\_[nome-da-sp] - Stored procedure de alteração.**

Ex: STP\_U\_Pessoa

	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

---

*STP\_S\_[nome-da-sp] - Stored procedure de consulta.*

Ex: STP\_S\_Campanha

## 12 Trigger

As Triggers são gatilhos utilizado para ações específicas quando determinadas coisas são feitas nas tabelas.

Deve respeitar a mesma nomenclatura das Tabelas mas com o prefixo TRG e alguns adendos como inclusão (I), exclusão (D), alteração (U) e consulta (S) utilizar a notação específica, apresentada na sintaxe.

Exemplos:

*TRG\_I\_[nome-da-trg] - Trigger de inclusão.*


Ex: TRG\_I\_ClienteLog

*TRG\_D\_[nome-da-trg] - Trigger de exclusão.*

Ex: TRG\_D\_ClienteCorrecao

*TRG\_U\_[nome-da-trg] - Trigger de alteração.*

Ex: TRG\_U\_ClienteRefazer


	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

## 13 Variáveis

- Os identificadores de variáveis para tipos de dados devem consistir em duas partes:  
A base, que descreve o conteúdo da variável
- Prefixo, que descreve o tipo de dados da variável

Os prefixos corretos para cada tipo de dados são mostrados abaixo:

Tipo de dado	Prefixo	Exemplo
Char	chr	@chrPrimeiroNome
Varchar	chv	@chvAtividade
Nchar	chn	@chnSegundoNome
Nvarchar	chvn	@chvnUltimoNome
Text	txt	@txtAnotacao
Ntext	txtn	@txtnComentario
Datetime	dtm	@dtmDataFim
Smalldatetime	dts	@dtsDataFim
Tinyint	iny	@inyAtividadeID
Smallint	ins	@insAtividadeID
Integer	int	@intAtividadeID
Bigint	inb	@inbAtividadeID
Numeric ou Decimal	dec	@decMedida
Real	rea	@reaVelocidade
Float	flt	@fltTamanho
Smallmoney	mns	@mnsCustoReal
Money	mny	@mnyPreco
Binary	bin	@binPath
Varbinary	biv	@bivContract
Image	img	@imgLogo
Bit	bit	@bitAtivo
Timestamp	tsp	@tspOrderID
Uniqueidentifier	guid	@guidAtividadeID
sql_variant	var	@varInventario
Cursor	cur	@curInventario
Table	tbl	@tblPessoa

	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

## 14 Sistema de Carga

O Sistema e carga é um processo que garante que todas as automações já nasçam com a característica “Multi Thread” para que possa ser executada em várias máquinas ao mesmo tempo bastando duplicá-la.

Esse sistema é composto por três tabelas Carga, Controle de Fila e Etapa

As tabelas estão padronizadas

### 14.1 Tabela de Carga

A estrutura da tabela de carga é a seguinte:

Campo	Tipo	Descrição
Id	UniqueIdentifier	Código sequencial
NumeroContrato	varchar(50)	Número Contrato <<variável>>
CamposAdicionais	-	Campos adicionais conforme a necessidade


A tabela de carga é o início do processo, nela deve-se conter a informação principal a ser trabalhada. Note que há a coluna “CamposAdicionais” que não vai constar no script base, pois aqui você vai colocar algum campo que você necessite para tratativa dos dados

O Campo “NumeroContrato” também é um campo variável, ele pode mudar de acordo com a necessidade “NumeroContrato”, “CodigoUsuario”, “CPF”, “CodigoCartao”, “ADE” e etc.

### 14.2 Tabela de Controle de Fila

A estrutura da tabela de Controle de Fila é a seguinte:

Campo	Tipo	Descrição
Id	UniqueIdentifier	Código sequencial
IdCarga	UniqueIdentifier	Id relacionado da tabela TB_Carga
IdEtapa	UniqueIdentifier	Id relacionado da tabela TB_Etapa
DataInsercao	DateTime	Data e hora quando foi inserido o registro nessa tabela
DataInicio	DateTime	Data e hora quando o registro começou a ser processado
DataFim	DateTime	Data e hora quando o registro terminou de ser processado
Status	varchar(100)	Status do registro como "Concluído", "Erro", "Cancelado"
Observacao	varchar(100)	Alguma observação pertinente ao registro
Usuario	varchar(50)	Usuario que esta executando a automação
Maquina	varchar(50)	Máquina que está executando a operação

	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

Aqui é onde os dados serão trabalhados, essa tabela que vai controlar o multiprocessamento, pois cada vez que o Robô pegar um registro ele vai marcar a “DataInicio” e isso vai evitar que outra instância do Robô pegue a mesma instância, possibilitando assim o processamento múltiplo.

### 14.3 Tabela de Etapa

A estrutura da tabela de Etapa é a seguinte:

Campo	Tipo	Descrição
Id	UniqueIdentifier	Código sequencial
Descricao	varchar(100)	Descrição da Etapa

A tabela de Etapa é “apenas” um descritivo para saber onde está o registro, por exemplo:

Quando se pega o registro podemos utilizar “Download de Arquivo” e a automação vai fazer todo esse processo até que ela evolua para gravar no banco de dados e a etapa pode mudar para “Gravando em banco de dados” na sequência pode mudar para “Gerando arquivos”, “Upload” e depois “Finalizado”.

#### 14.3.1 Por que usar o sistema Etapa?

É importante a utilização desse sistema para o caso de ocorrer algum crash descontrolado como Queda na Rede, Desligamento da máquina, Quebra da aplicação.


Nesses exemplos, caso a automação não consiga retomar a automação podemos saber exatamente em que etapa ficou paralisado.

### 14.4 Script

```
CREATE TABLE [dbo].[TB_Carga](
    [Id] [UniqueIdentifier] default NewSequentialID() NOT NULL ,
    [NumeroContrato] [varchar](50) NULL
    /*Campos Adicionais*/
    CONSTRAINT [PK_TB_Carga] PRIMARY KEY CLUSTERED
    (
        [Id] ASC
    )
    WITH (
        PAD_INDEX = OFF,
        STATISTICS_NORECOMPUTE = OFF,
        IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON,
        ALLOW_PAGE_LOCKS = ON,
        OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
)
```

GO

```
CREATE TABLE [dbo].[TB_ControlaFila](
    [Id] [UniqueIdentifier] default NewSequentialID() NOT NULL ,
    [IdCarga] [int] NULL,
```

	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

```

[IdEtapa] [int] NULL,
[CreatedOn] [datetime] NULL default GetDate(), /*Antigo campo de DataInsercao*/
[DataInicio] [datetime] NULL,
[DataFim] [datetime] NULL,
[Status] [varchar](100) NULL,
[Observacao] [varchar](max) NULL,
[Usuario] [varchar](50) NULL,
[Maquina] [varchar](50) NULL
CONSTRAINT [PK_TB_ControleFila] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)
WITH (
    PAD_INDEX = OFF,
    STATISTICS_NORECOMPUTE = OFF,
    IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON,
    ALLOW_PAGE_LOCKS = ON,
    OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
)

```

GO

```

CREATE TABLE [dbo].[TB_Etapa]
(
    ID [UniqueIdentifier] default NewSequentialID() not Null ,
    Descricao varchar(100)
    CONSTRAINT [PK_TB_Etapa] PRIMARY KEY CLUSTERED
    (
        [Id] ASC
    )
    WITH (
        PAD_INDEX = OFF,
        STATISTICS_NORECOMPUTE = OFF,
        IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON,
        ALLOW_PAGE_LOCKS = ON,
        OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
)

```


**\*Nota nessa versão do documento não foi realizado o relacionamento das tabelas de fila**

## 15 Schema

Algumas soluções necessitam, por decisões específicas, que as aplicações utilizem um banco de dados compartilhados.

Nesses casos não devemos nomear tabelas com nomes específicos e sim utilizar esquemas.



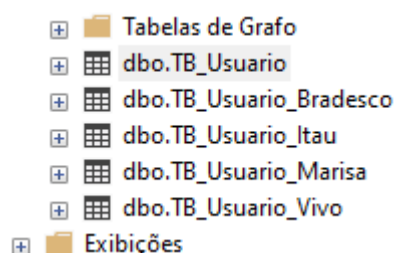
	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

Schemas devem ser utilizados para agrupar informações de banco de dados.

## 15.1 Cenário:

Um banco de dados foi criado e nele temos cinco automações rodando, a solução que o programador deu foi para solucionar o problema, não há erro na solução e sim falta de padronização.

A tabela é a de TB\_Usuario

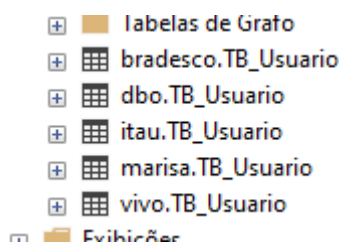


Note que a solução foi adicionar um sufixo com o nome do cliente, visualmente é fácil de visualizar, mas despadronizar totalmente o banco de dados que visa impessoalizar as tabelas.

A Solução ideal nesse ponto seria unificar todas as tabelas de TB\_Usuario e tipar os usuários com uma tabela de TB\_Cliente, mas vamos agir na correção visual, pois muitas dessas soluções podem ser desacopladas a qualquer momento, ou seja, movidas para outro banco, e a tipagem vai dificultar isso.

## 15.2 Solução

A solução para isso é utilizar o Schema (Esquema), que vai manter os nomes padronizados e separar as informações, então a mesma estrutura com esquema fica da seguinte maneira



### 15.2.1 Por que utilizar Schemas

Porque eles agrupam os objetos do banco de dados de acordo com a necessidade. Nesse exemplo usamos Clientes diferentes, mas podem ser usados para setores, áreas, frentes de atuação etc.


### 15.2.2 Como Criar e utilizar um Schema

A nomenclatura do Schema deve ser sempre em minúsculo e não necessita de um prefixo

[{a.z}]-> meuschema

Uma query com Schema

Em uma tabela comum temos o seguinte:

 <small>FULL SERVICE BPO</small> <small>EMPRESA DO GRUPO ATENTO</small> <b>ATENTO</b>	<b>Código:</b>	COE-DOC-0001
	<b>Projeto:</b>	Padrões para Banco de Dados
	<b>Cliente:</b>	COE RPA

---

```
SELECT * FROM TB_Usuario
```

Para a versão com Schema, apenas adicionamos o Schema

```
SELECT * FROM itau.TB_Usuario
```

O mesmo vale para Joins entre tabela:

```
SELECT * FROM banana.TB_Usuario as bananaUsuario
      INNER JOIN
      uva.TB_Usuario as uvaUsuario ON bananaUsuario.id = uvaUsuario.id
```

Obs: O Exemplo foi mostrado com tabela, mas a mesma regra e padrão é válido para Função, Procedure e View Também