



LABORATÓRIO DE BANCO DE DADOS

IBD100

IBD100 **TRIGGERS E** **STORED-PROCEDURES**

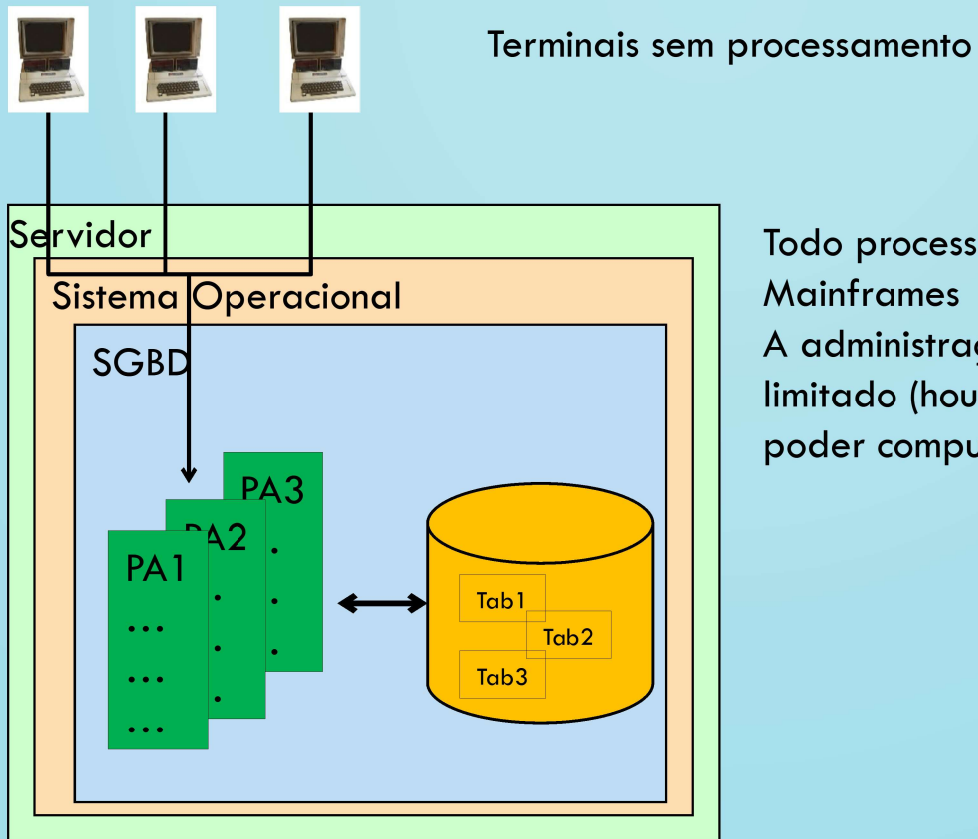
João Maurício Hypólito
prof.oao.hypolito@gmail.com

Programa

- Histórico dos SGBD Distribuídos
 - Origem (motivo)
 - Soluções interessantes
 - Mas tinham alguns problemas
 - Uma Solução (para alguns problemas) que veio para
- Definindo Gatilhos
 - Formas de implementação
 - O PostgreSQL implementa como?
 - Um pouco da PLpgSQL
 - Um exemplo de escrita de uma função de gatilho
- Os gatilhos que devem ser feitos
 - Como resolver? Método!

IBD100 – GATILHOS (TRIGGERS E STORED-PROCEDURES)

- A evolução dos SGBDs em 40 anos (de TI para TIC)
 - As máquinas tinham a arquitetura centralizada (40-80).

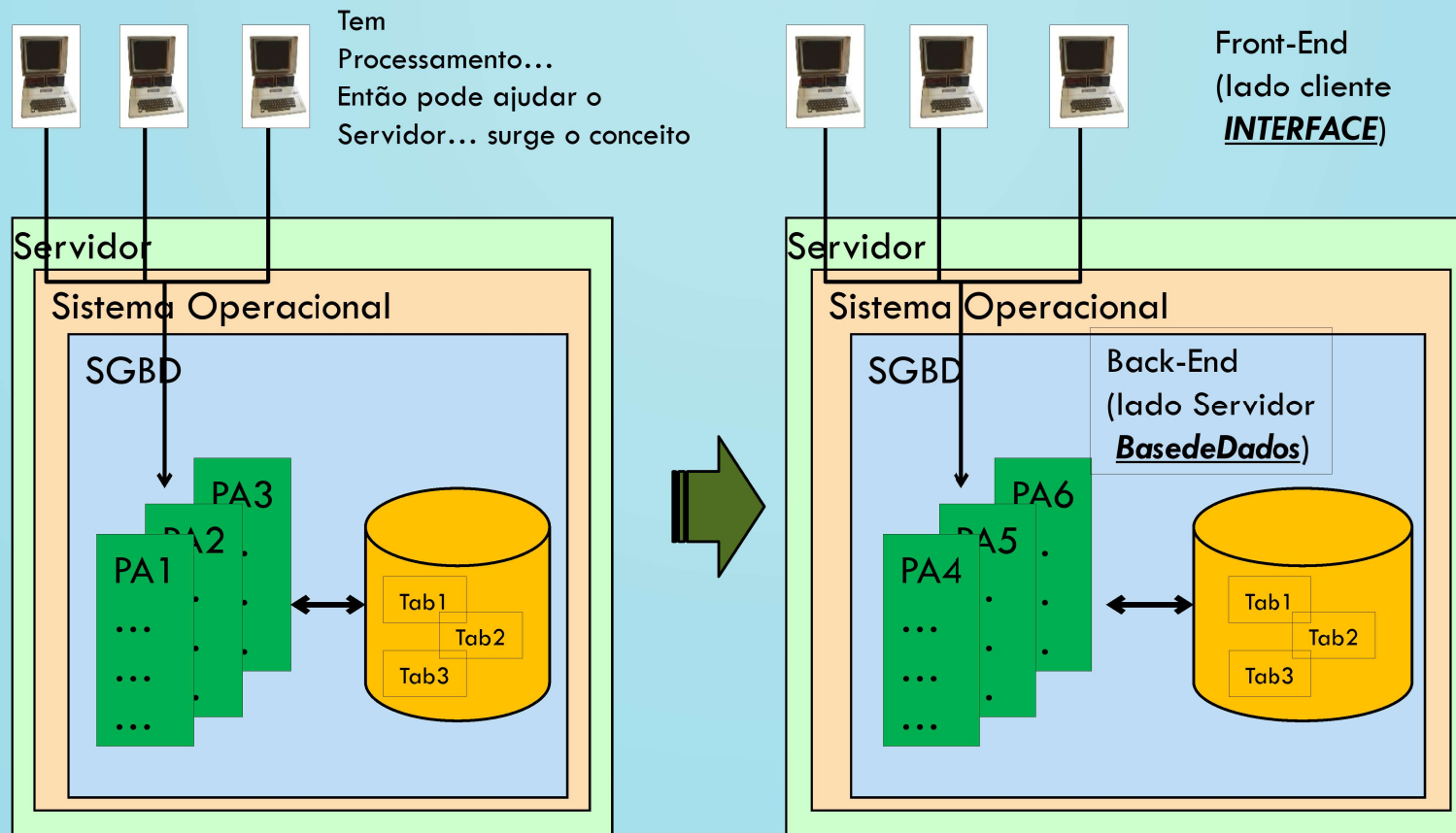


Todo processamento é oferecido em uma única máquina:
Mainframes

A administração é facilitada mas o recurso se tornou limitado (houve um aumento rápido da procura por poder computacional).

IBD100 – GATILHOS (TRIGGERS E STORED-PROCEDURES)

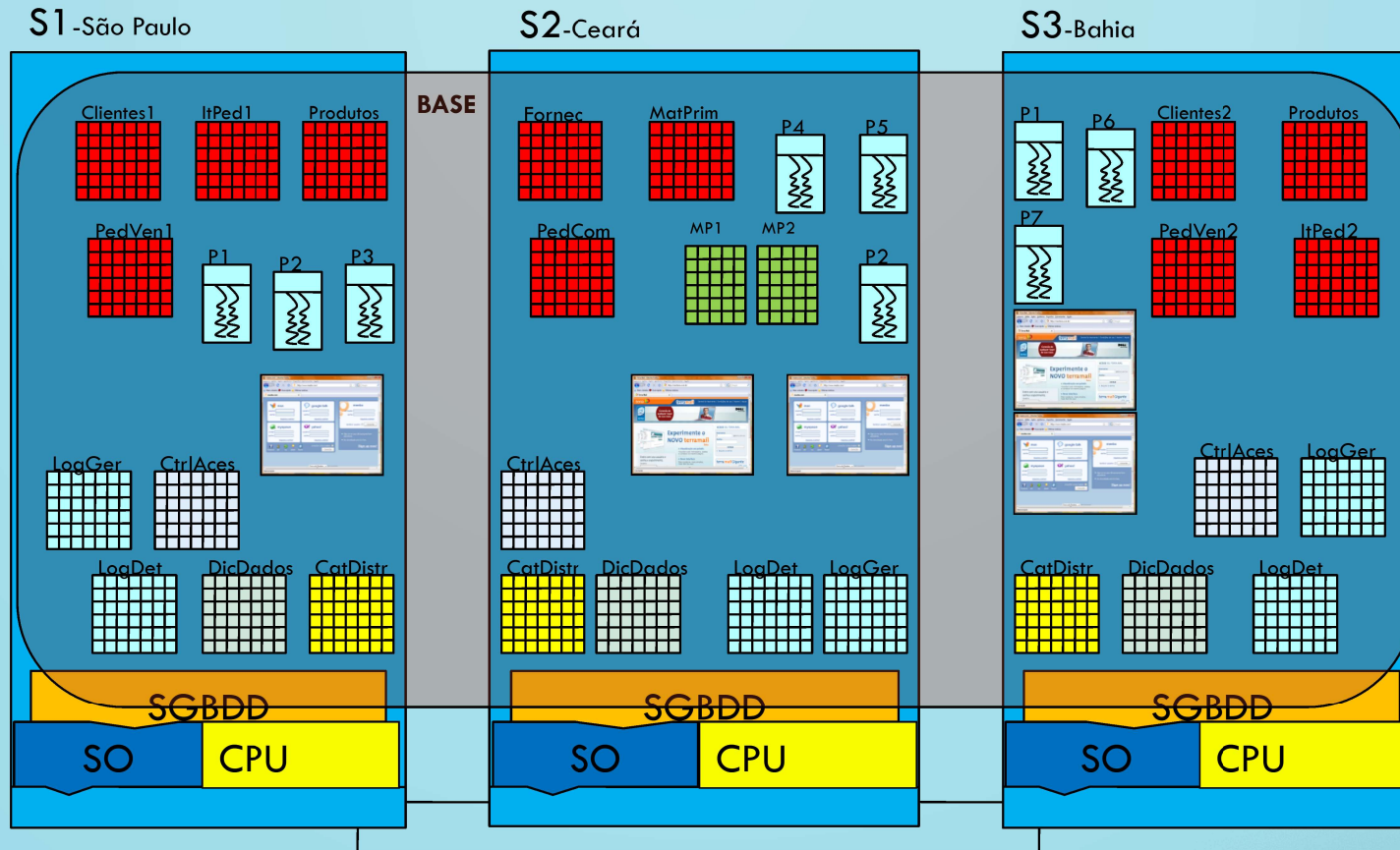
- A evolução dos SGBDs em 40 anos (de TI para TIC)
- Mas vieram as Redes de Computadores (80-90)



IBD100 – GATILHOS (TRIGGERS E STORED-PROCEDURES)

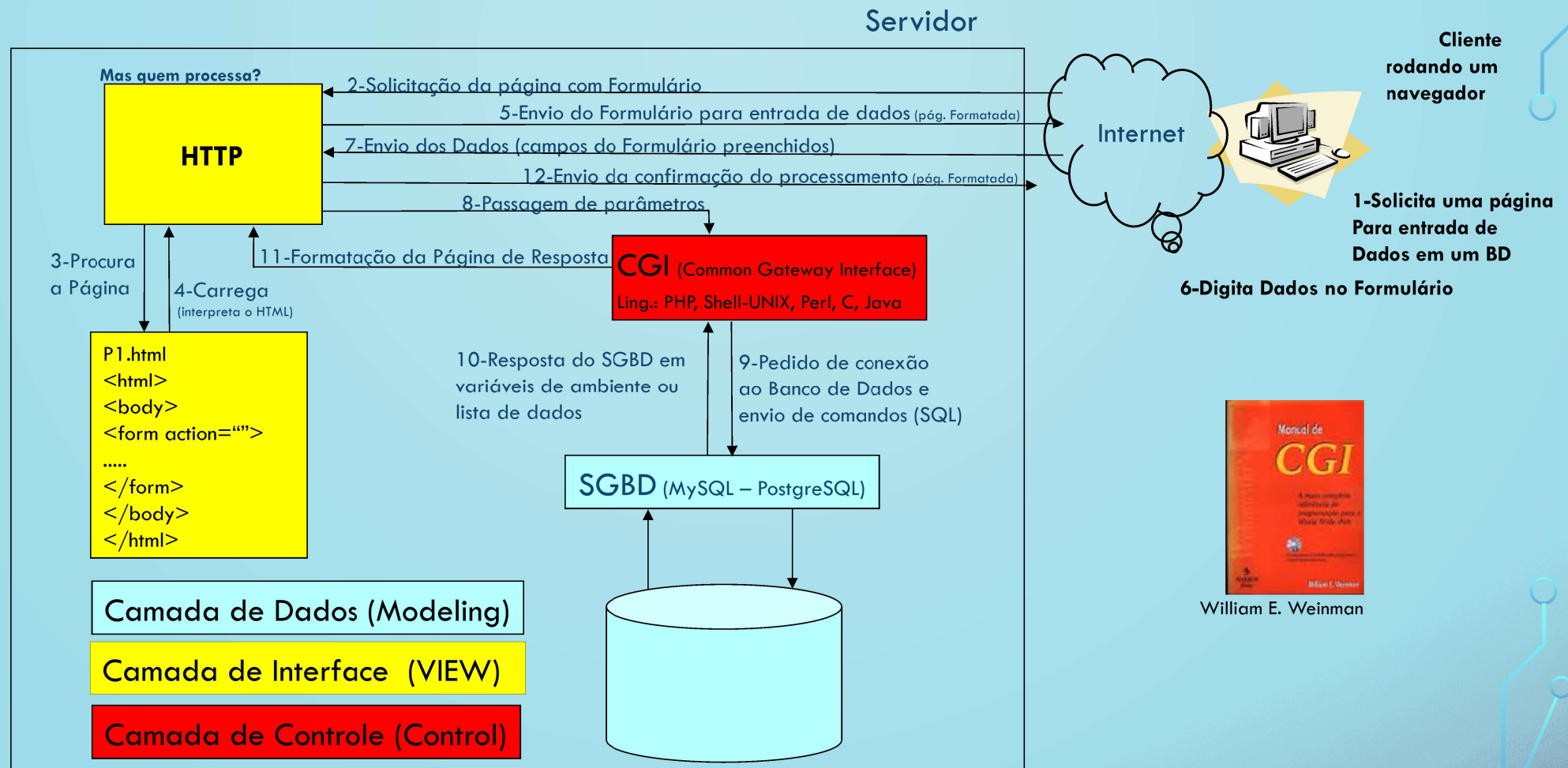
- Os Bancos de Dados Distribuídos

- O Conceito Básico: O usuário final, analistas e programadores NÃO sabem onde se hospedam os componentes da base de dados.



IBD100 – GATILHOS (TRIGGERS E STORED-PROCEDURES)

- Finalmente, tudo converge para o separado e especializado (cada um é muito bom no que sabe fazer).



William E. Weinman

- Durante o desenvolvimento da tecnologia de Bancos de Dados Distribuídos uma série de problemas não apresentavam solução satisfatória:
 - Como controlar o Desenvolvimento de Aplicações sobre bancos de dados em um contexto que deveria atender a diferentes áreas funcionais de uma organização?
 - Como poder garantir que PA feitos em uma AF respeitassem as regras do negócio da organização?
 - Como evitar que as estações de trabalho fossem sobrecarregadas com o processamento de dados?
- Estes problemas aumentaram com o estudo e desenvolvimento de serviços e produtos na estrutura MVC.
- Os desenvolvedores de cada camada passaram a trabalhar em suas próprias soluções acelerando o desenvolvimento.
- Na área de ferramentas para bancos de dados...

IBD100 – GATILHOS - SOLUÇÃO NA ÁREA DE BD

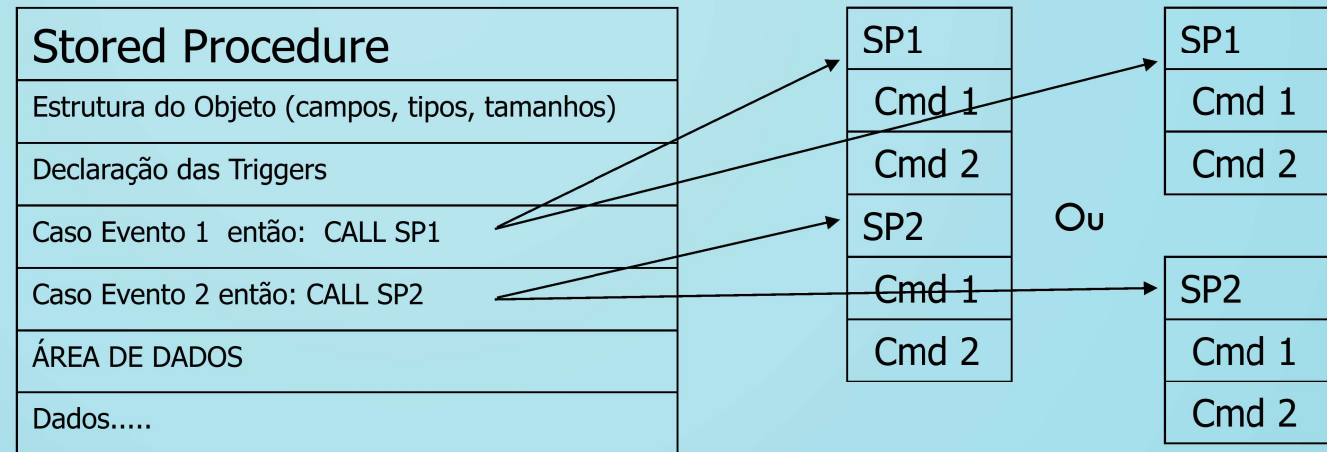
- Outra solução foi o desenvolvimento de SGBDs que permitiam que segmentos de códigos fossem anexados à estrutura das tabelas de modo que, quando realizados eventos (IEA ou IDU) eram disparados a execução dos segmentos de código.
- Trechos de **programas** associados aos arquivos de dados e **disparados** quando ocorrem eventos (Inc,Exc,Alt).
 - Então pensamos em 'programar' o banco de dados, ou ainda em 'grudar' as regras nos dados: "É método de Classe?"... NÃO!
- Podem ser denominados de *triggers*, *stored-procedures*, *constraints* ou *code rule* (aí depende do SGBD).
 - Nessa tecnologia os SGBDs andaram na frente dos Padrões (de SQL). Por isso, existem muitas formas de implementar o mesmo conceito.
- As tabelas passam a **conter** as regras de negócios
 - Conceito muito próximo de Objetos (Dados+Processos)...
 - Mas cuidado. Tabela não É Classe. Classe existe ↔ existem Atributos & Métodos. A Tabela existe sem métodos.
- Quem implementa: DBA.
 - DBA?? Quebra as atividades dos perfis profissionais envolvidos no projeto de Sis.
 - O DBA passa a 'programar' as regras de negócio 'grudados' aos dados das tabelas
 - Vantagem:
 - Concentra as atividades e diminui a margem de erros (sobrecarrega os DBAs, mas quem se importa?)
 - Desvantagem:
 - Concentra as atividades e desloca o foco das atividades do DBA para o negócio da empresa.

- Trigger - A regra de negócio está no mesmo arquivo que acomoda o dado.
- Vantagem:
 - Rapidez na execução (em geral tudo vai para a RAM)
 - as regras disponíveis estão em um só arquivo
- Desvantagem:
 - Dificuldade de Manutenção:
Para *redefinir* regras é preciso bloquear o acesso ao BD
(pode ser 'resolvido' com um computador para desenvolvimento)

Triggers
Estrutura do Objeto (campos, tipos, tamanhos)
Declaração das Triggers
Caso EVENTO 1
Comando 1
Comando 2
Caso Evento 2
Comando 3
Comando 4
ÁREA DE DADOS
Dados.....
Dados ...

IBD100 – GATILHOS - TIPOS BÁSICOS

- Store Procedure: No arquivo de dados existe uma chamada para um arquivo de programa onde a regra pode ser implementada.



- Vantagem: Facilidade de Manutenção
- Desvantagem: Mais chamadas ao Sistema de arquivos do SO
(muitos arquivos acessados limita o desempenho)

IBD100 – GATILHOS - O PADRÃO SQL E AS SOLUÇÕES DOS SGBDS

- Um pouco de história:

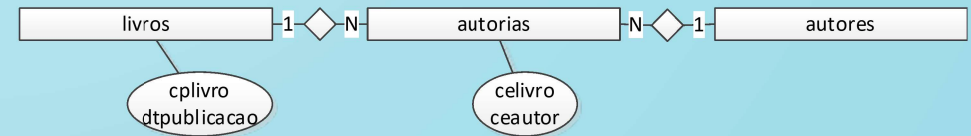
- O Modelo relacional foi apresentado em 1970 (com tudo no 'combo' – incluindo AR)
- O 'Rascunho' do SQL começou a ser estudado em 73 e em 74 já havia o SEQUEL, depois o SQUARE e a seguir o SEQUEL2 → SystemR da IBM.
- O SQL 'aparece' em 1980 e torna-se um 'padrão de mercado' com muitos SGBDs seguindo este 'modelo de linguagem'.
- 1986 o ANSI apresenta um padrão de SQL e é homologado pela ISO.
- MAS o SQL já estava começando a 'descolar' da AR pois implementava novos comandos (junção aberta e processamento de funções mais complexas), daí o ANSI teve que acelerar as revisões do Padrão 'seguindo' as tendências de mercado. Vieram os padrões ANSI:92.
- O ANSI:92 foi adotado por muitos SGBDs como o SQL referenciado, PORÉM o mundo de Banco de Dados já começava a lidar com BD DISTRIBUÍDOS. Daí o padrão novamente começou a 'ficar para trás'. Novamente o ANSI teve que estudar a forma do SQL tratar todos os avançados tecnológicos e (só em 1999) apresenta o ANSI:99 com a implementação *proposta* para a implementação de gatilhos.
- Mas, os SGBDs já tinham implementado os seus meios para os conceitos de gatilhos. Daí que vem o 'descolamento' dos SGBDs e do Padrão ANSI.
- Hoje, temos uma diversidade de formas de implementar gatilhos.
- No SGBD PostgreSQL os termos de definição parecem indicar um gatilho (trigger) mas as estruturas envolvidas sugerem que seja uma Stored-Procedure.
- De qualquer forma o conceito é o mesmo... No PostgreSQL se implementa o conceito de segmento de código executado quando se ATUALIZA uma tabela da base de dados.
- No PostgreSQL escrevemos a codificação do gatilho no componente "TRIGGER FUNCTIONS"
- Depois de codificados, as Trigger Functions devem ser VINCULADAS as tabelas. Isso se faz criando NA TABELA um TRIGGER que 'dispara' uma 'Trigger Function'.

prof.joaquim@unifor.br

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- Mas... Como construir os gatilhos?
- Um gatilho deve ser sempre associado a UMA tabela (só uma, embora possa consultar dados de outras tabelas).
- Um gatilho deve ser sempre associado a um evento sobre os dados da tabela. Evento={Inclusão, Exclusão ou Alteração} (IUD).
- Um gatilho deve ser sempre associado a um momento quando deve ser executado. Em geral ANTES | DEPOIS de uma ação (Before | After).
- Exemplo:
 - Um autor não pode ter uma autoria incluída se o último livro escrito tiver sido escrito a menos de 2 anos. Lembrando que...
 - Então...
 - Tabela referenciada = **autorias**.
 - Evento = **I**nclusão.
 - Momento = Antes da ação (**B**efore)

Isso PODE indicar o
nome do gatilho:
autoriasBI



IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- O que deve ser feito no desenvolvimento deste gatilho?
- Pegar em uma variável (vdias) a diferença (em dias) da data de hoje menos a data da última publicação de livros do mesmo autor do registro que se está tentando incluir em autorias.
- SE o valor de vdias for menor do que 731 dias (dois anos),
ENTÃO a autoria não pode ser incluída,
SENÃO a autoria pode ser incluída.
- Hum...
Teremos que declarar uma variável, ler o dado da tabela livros (dtpublicacao), fazer a comparação e retornar Ok! Ou NOk!
- Mas em qual linguagem?
 - O PG aceita que programas de gatilhos possam ser escritos em diversas linguagens.
 - Usaremos a Procedural Language do PostgreSQL (PL/pgSQL).
- Codificando...

- Codificando...

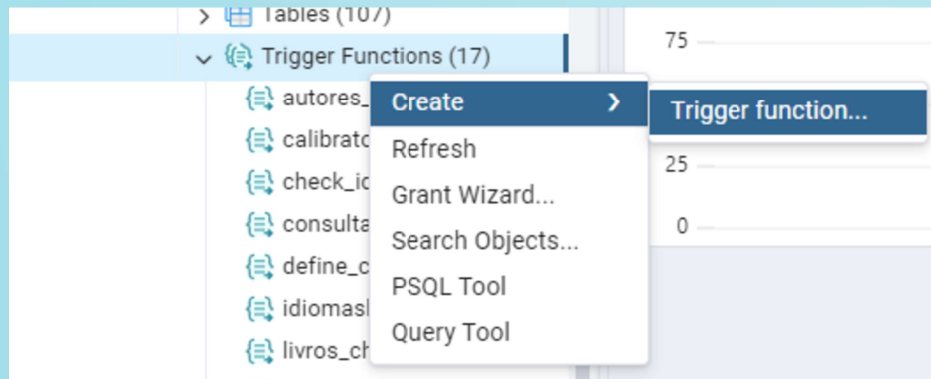
```
DECLARE
    vdias smallint;
BEGIN
    SELECT current_date-MAX(dtpublicacao) INTO vdias
        FROM livros INNER JOIN autorias ON livros.pklivro=autorias.fklivro
        WHERE autorias.fkautor=NEW.fkautor
        GROUP BY fkautor
        ORDER BY fkautor;

    IF ( vdias<731 )
    THEN
        RAISE EXCEPTION 'O Autor (%) publicou a menos de 2 anos',NEW.fkautor;
        RETURN NULL;
    ELSE
        RETURN NEW;
    END IF;
END
```

- Este código é identificado com nome: **autoriasBI** e é escrito como uma *FUNÇÃO* de GATILHO (*trigger function*) no PG (esse SGBD tem esta estratégia). Uma Trigger Function pode (até) ser 'colada' em mais de tabela (meio difícil disso acontecer, mas no PG é possível).

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- Construindo no pgAdmin4.
- Acessando a lista de Triggers Function



- Repare que a lista de Funções de Gatilhos está abaixo da lista de Tabelas da Base de Dados
- À seguir vê-se uma janela flutuante 'padrão' do pgadmin 4 onde se informam os dados de definição da função de gatilho.
- Considere a seguinte regra: Um Autor NÃO pode publicar dois livros em um intervalo de tempo menor que DOIS ANOS.
Isso Determina a Tabela alvo da função da gatilho: AUTORIAS, MAS datas de publicação é um campo da tabela autorias.
Então para um novo registro em autorias tem-se que ver qual a máxima data de publicação do autor (dado gravado em NEW.ceautor) e ver se existe um intervalo de dias maior que 731 dias (anos).

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- Construindo no pgAdmin4.
- Editando a janela de entrada de dados para definir a Triggers Function

Create - Trigger function

General Definition Code Options Parameters Security SQL

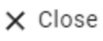
Name 

Owner  postgres

Schema  public

Comment

 'Name' cannot be empty. 

os dados nas abas Definition e Code são obrigatórios na aba SQL sempre é exibido o comando que será executado para definir a função de gatilho.

É obrigatório
Informar o
Nome da Trigger-Function
Sugestão:
NomeDaTab+Momento+Evento
Exemplo:
autoriasBIU

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- Construindo no pgAdmin4.

The screenshot shows the 'Create - Trigger function' dialog box in pgAdmin4. The 'Definition' tab is selected. The 'Return type' is set to 'trigger' and the 'Language' is set to 'plpgsql'. A red error message at the bottom states: 'Name' cannot be empty. The dialog has buttons for Close, Reset, and Save.

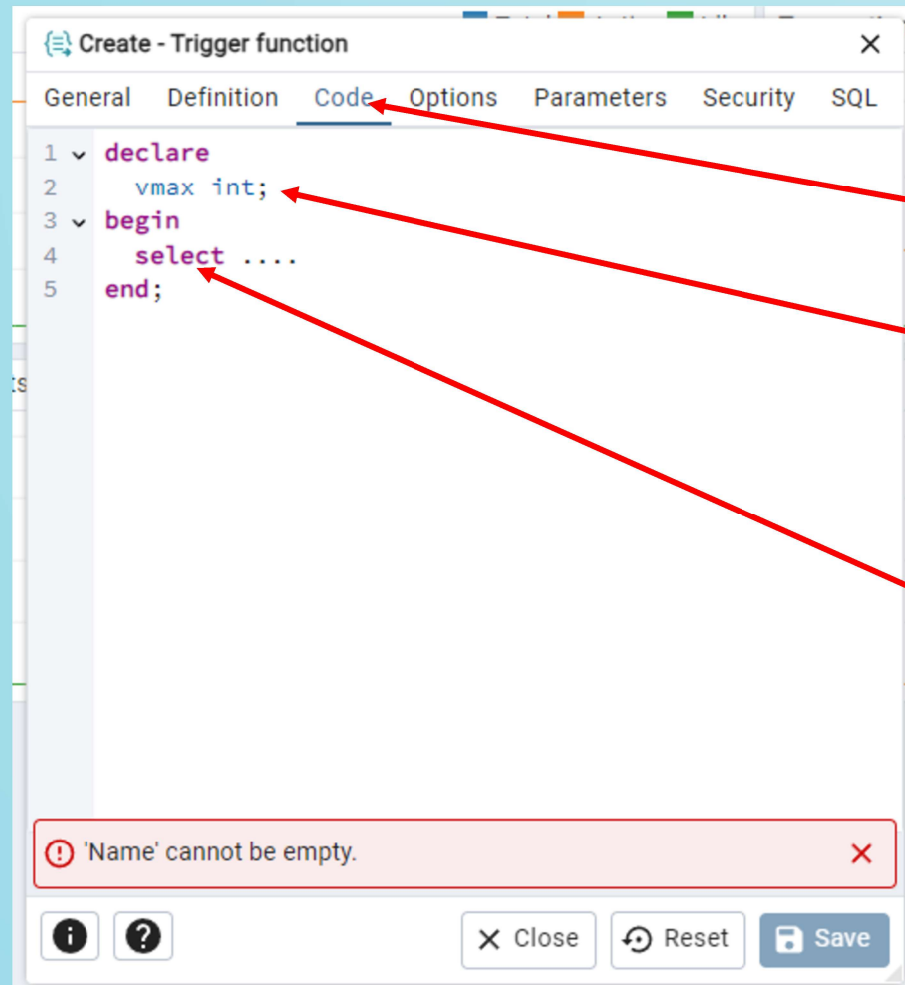
Escolha do Tipo de Retorno da Função. No PostgreSQL se implementa o conceito de função como em 'C' (podendo a função conter SQL).

Aqui se escolhe a linguagem de implementação da função de gatilho. Lembre: A linguagem base de desenvolvimento do PostgreSQL é 'C'. Por este motivo 'C' é linguagem disponível. A PLPGSQL [Procedural Language do PG-SQL) é a outra linguagem 'nativa' para programação de gatilhos.

Durante a montagem deste exemplo eu não preenchi o nome da Função de Gatilho. Por isso esta mensagem 'persiste' em todas as abas desta janela.

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- Construindo no pgAdmin4.



A codificação vai nesta aba.

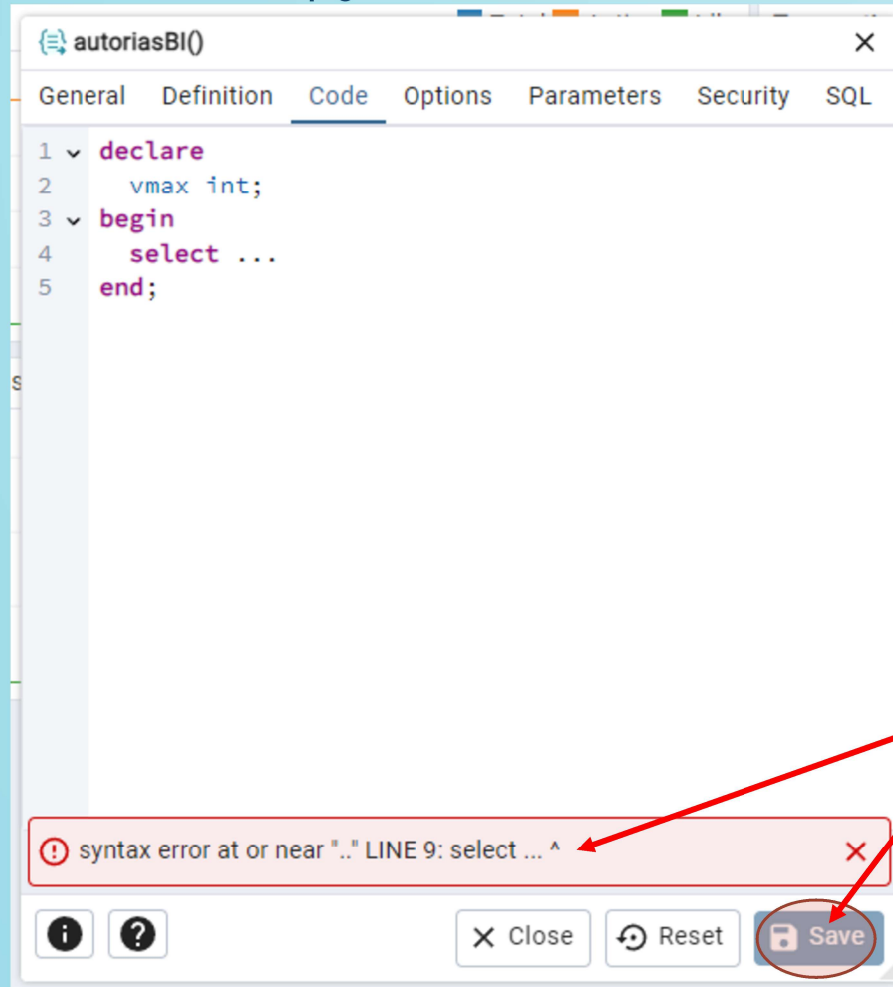
SE existirem variáveis na função, então deve-se apresentá-las com o 'bloco' DECLARE, indicando nome e tipo de dado de cada variável.
Cada Declaração deve terminar com ';'.

O 'bloco' de programação deve SEMPRE começar com 'begin' e terminar com 'end;' [note o ';' no final do comando].

O 'bloco' quase sempre contém um processamento de dado (SQL) e uma tomada de decisão. Muitas vezes se usa o comando RAISE para enviar mensagens para o usuário ou aplicação que recebe o resultado do processamento da função.

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- Construindo no pgAdmin4.



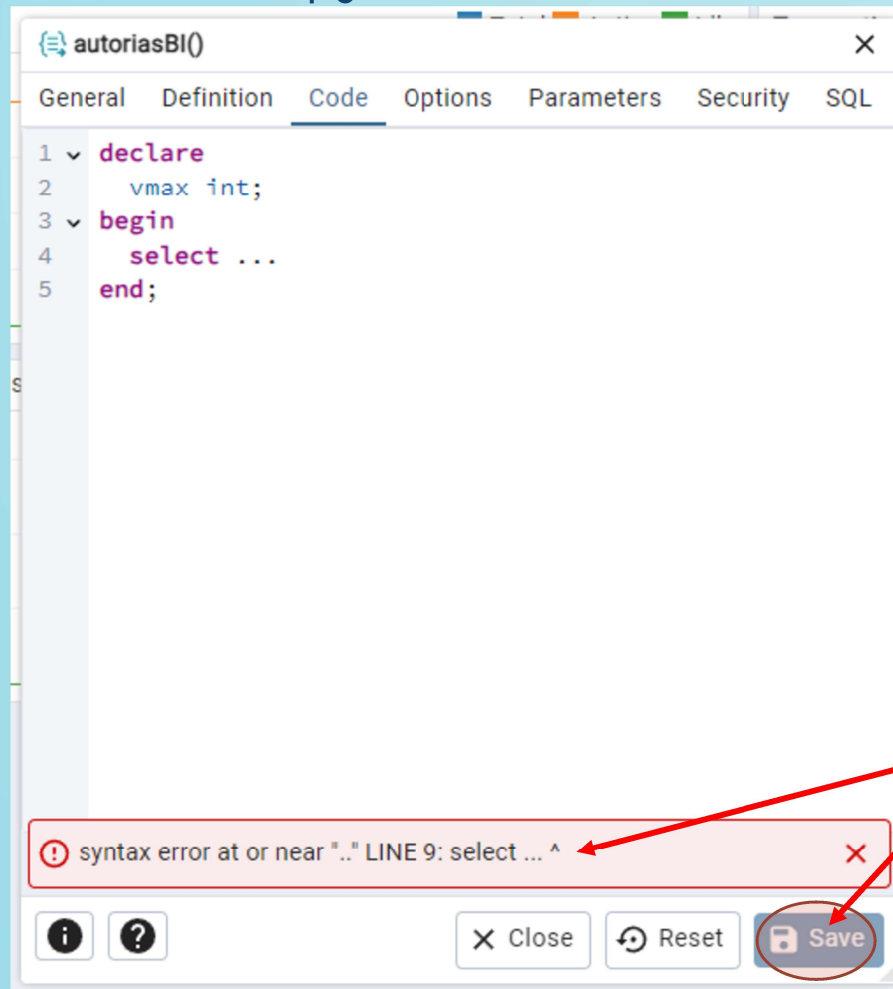
Nesse exemplo de código não terminado, quando se tenta gravar é visto uma mensagem de erro relativa ao 'code' mal escrito. O PostgreSQL informa o número da linha considerando o comando escrito na aba SQL.

A 'tentativa de GRAVAR a função pode ser feita a qualquer momento DESDE QUE a estrutura mínima exista (as palavras BEGIN e END;).

À qualquer momento a edição pode continuar acessando 'Properties' sobre a Função de Gatilho.

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- Construindo no pgAdmin4.



Nesse exemplo de código não terminado, quando se tenta gravar é visto uma mensagem de erro relativa ao 'code' com erro.

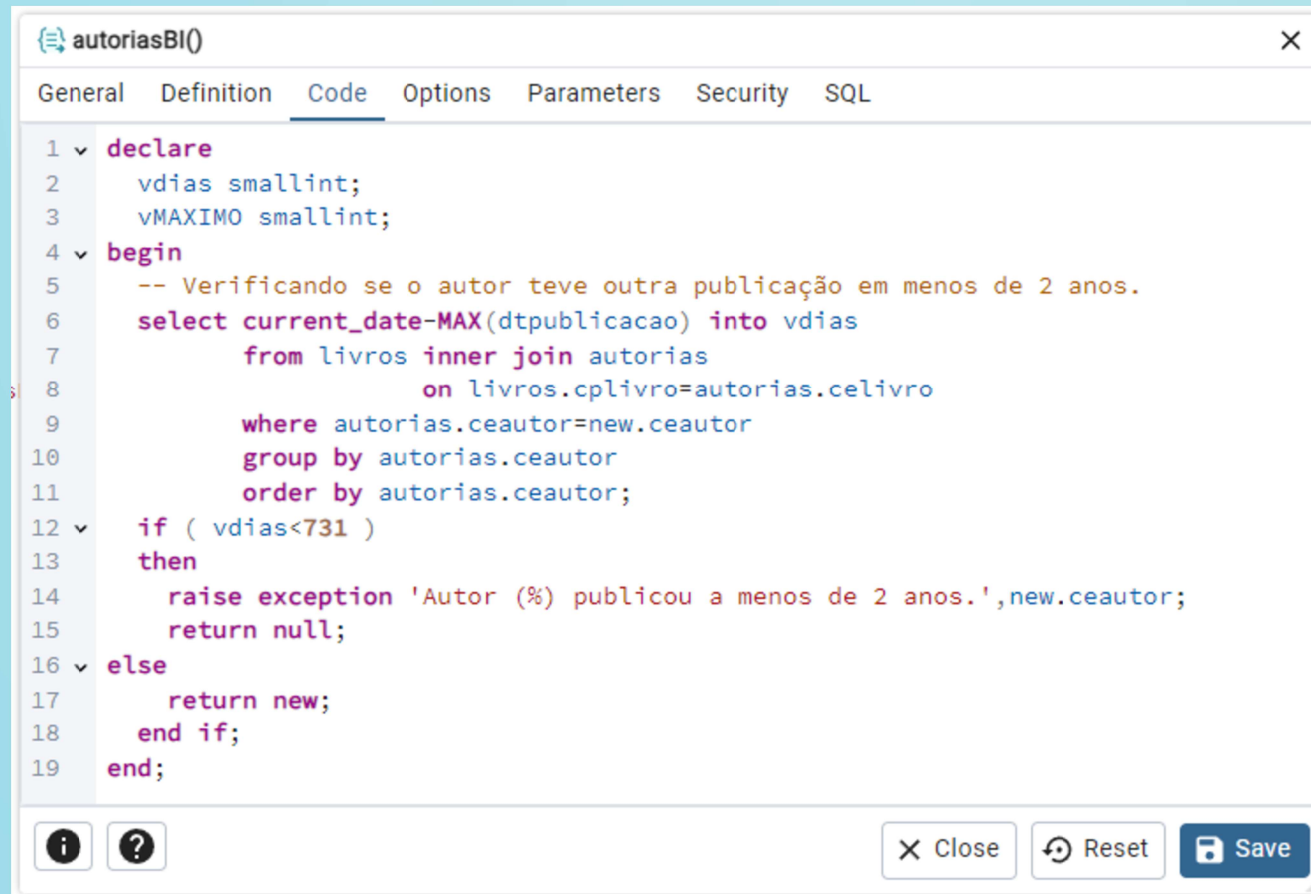
O PostgreSQL informa o número da linha considerando o comando escrito na aba SQL. Clique na aba SQL para ver o código completo.

A 'tentativa de GRAVAR a função pode ser feita a qualquer momento DESDE QUE a estrutura mínima exista (as palavras BEGIN e END;).

À qualquer momento a edição pode continuar acessando 'Properties' sobre a Função de Gatilho.

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- Construindo no pgAdmin4.
- Lembra-se da regra da função de gatilho? “Considere a seguinte regra: Um Autor NÃO pode publicar dois livros em um intervalo de tempo menor que DOIS ANOS”. Isso pode ser implementado na função:



The screenshot shows the pgAdmin4 interface with the 'Code' tab selected for the function 'autoriasBI()'. The SQL code is as follows:

```
1 declare
2     vdias smallint;
3     vMAXIMO smallint;
4 begin
5     -- Verificando se o autor teve outra publicação em menos de 2 anos.
6     select current_date-MAX(dtpublicacao) into vdias
7         from livros inner join autorias
8             on livros.cplivro=autorias.celivro
9         where autorias.ceautor=new.ceautor
10        group by autorias.ceautor
11        order by autorias.ceautor;
12 if ( vdias<731 )
13 then
14     raise exception 'Autor (%) publicou a menos de 2 anos.',new.ceautor;
15     return null;
16 else
17     return new;
18 end if;
19 end;
```

At the bottom of the window, there are icons for information and help, and buttons for 'Close', 'Reset', and 'Save'.

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- Construindo no pgAdmin4.
- Na aba SQL vê-se o código completo:

O tratador de erros referencia esta numeração de linhas nas mensagens que envia ao pgAdmin 4 enquanto a função de gatilho está sendo escrita.
O pgAdmin 4 'montou' este comando para ser 'submetido' ao servidor PostgreSQL.

Note:

Uso de comentários.

Uso de indentação.

Mensagens claras para o usuário no RAISE

Note:

O 'return null' 'anula a linha' e cancela a operação que se realizava na tabela.

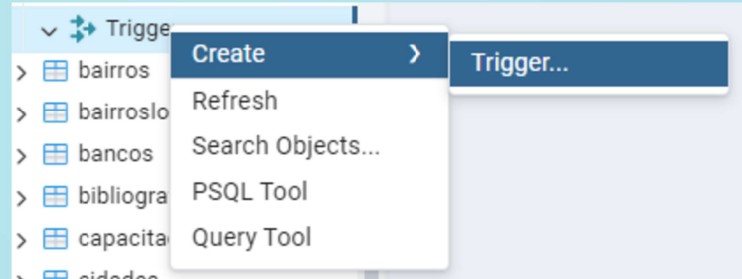
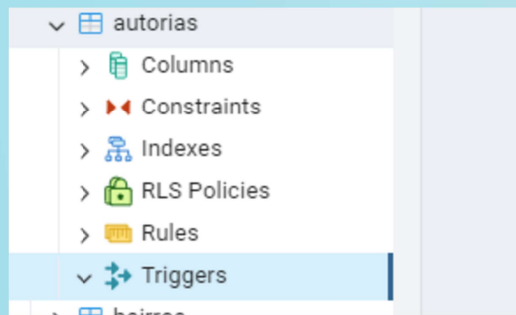
O 'return NEW' retorna a linha para seguir na operação que se fazia na tabela.

```
1 CREATE OR REPLACE FUNCTION public."autoriasBI"()  
2 RETURNS trigger  
3 LANGUAGE 'plpgsql'  
4 VOLATILE  
5 COST 100  
6 AS $BODY$  
7 declare  
8     vdias smallint;  
9     vMAXIMO smallint;  
10 begin  
11     -- Verificando se o autor teve outra publicação em menos de 2 anos.  
12     select current_date-MAX(dtpublicacao) into vdias  
13     from livros inner join autorias  
14     on livros.cplivro=autorias.celivro  
15     where autorias.ceautor=new.ceautor  
16     group by autorias.ceautor  
17     order by autorias.ceautor;  
18     if ( vdias<731 )  
19     then  
20         raise exception 'Autor (%) publicou a menos de 2 anos.',new.ceautor;  
21         return null;  
22     else  
23         return new;  
24     end if;  
25 end;  
26 $BODY$;
```

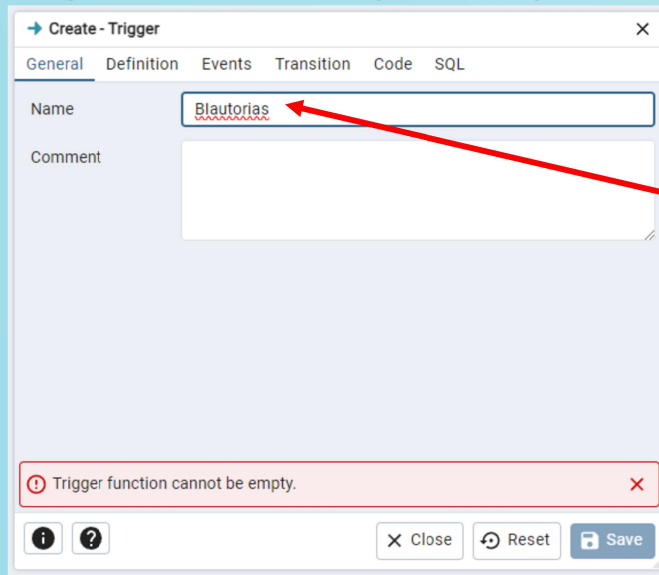
Agora pode-se SALVAR a Função de Gatilho.

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- Construindo no pgAdmin4.
- Agora temos que 'vincular' a função de gatilho na tabela 'autorias', para isso:



- Agora inclui-se o 'gatilho' que 'dispara' a função de gatilho, na janela flutuante



Nome do Gatilho: Momento+Ação+nome da tabela.
Não pode ser repetido dentro de uma BASE de Dados.

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- Construindo no pgAdmin4.
- Na aba 'Definition' deve-se informar:

Gatilho disparado ao atualizar: LINHA

Gatilho de tratamento de regras de restrição (de valores de chaves primária e estrangeira) pode ser disparado no Final de uma transação. Quando isso NÃO precisar ser feito, NÃO se marcam estes itens.

Escolha da Função de Gatilho que será executada quando o Gatilho for disparado.

The screenshot shows the 'Create - Trigger' dialog box in pgAdmin4, with the 'Definition' tab selected. The dialog has several tabs: General, Definition, Events, Transition, Code, and SQL. The 'Definition' tab contains the following fields and controls:

- Row trigger?**: A toggle switch that is currently turned on (blue).
- Constraint trigger?**: A toggle switch that is currently turned off (grey).
- Deferrable?**: A toggle switch that is currently turned off (grey).
- Deferred?**: A toggle switch that is currently turned off (grey).
- Trigger function**: A dropdown menu with a red arrow pointing to it from the text box. The dropdown is open, showing a list of functions: public.autores_biu, public."autoriasBI", public.calibratotalpedvenda, public.check_id_cidade_prof, public.consulta_insert, public.define_cd_cliente, public.idiomasbi, and public.livros_check_datas.
- Arguments**: A text area for defining the trigger's arguments.
- Trigger function c**: A red box with an exclamation mark icon and the text 'Trigger function c'.
- Info and Help icons**: Two small icons at the bottom left of the dialog.

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- Construindo no pgAdmin4.
- Na aba 'Definition' deve-se informar:

Momento de disparado: BEFORE ou AFTER
De se executar a ação.

As ações que o PostgreSQL
permite 'disparar' um gatilho.

O PostgreSQL permite até
escolher ações que disparam um
mesmo gatilho.

Verifique se o SGBD que você usa
permite isso.

→ Create - Trigger

General Definition **Events** Transition Code SQL

Fires BEFORE

Events

INSERT ☒

UPDATE ☒

DELETE ☐

TRUNCATE ☐

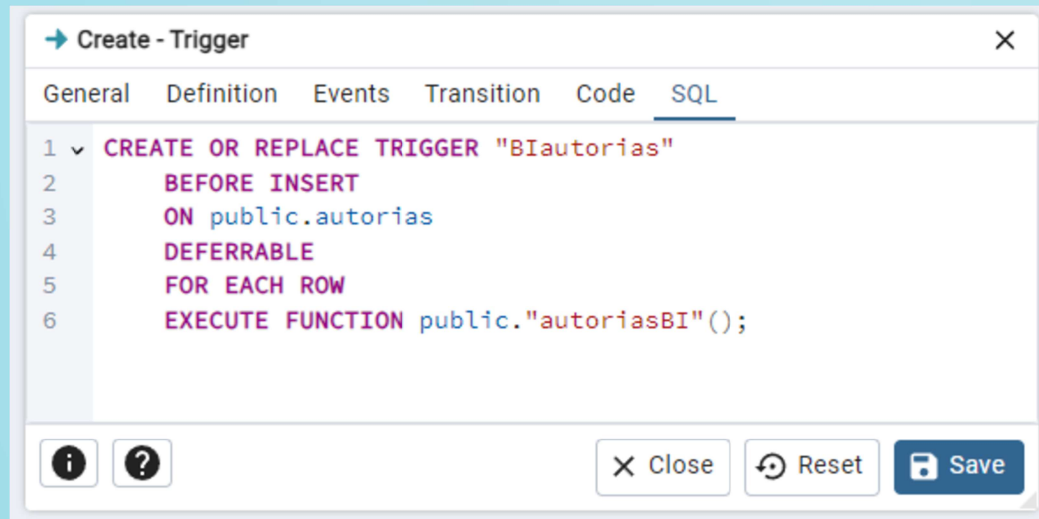
When 1

Columns Select an item...

Close Reset Save

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

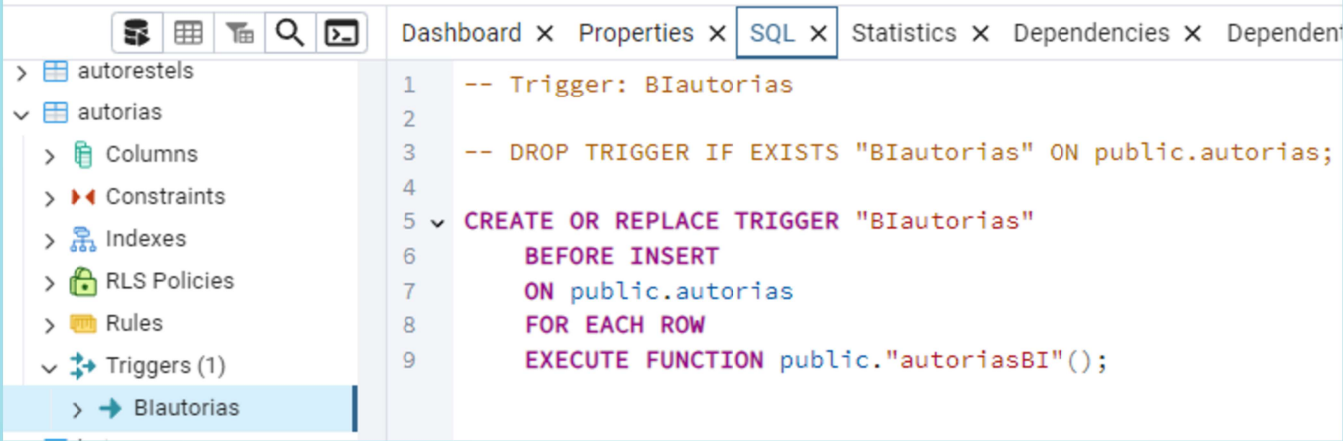
- Construindo no pgAdmin4.
- É interessante observar a aba SQL e ver o comando que ‘monta’ o gatilho.



- Note: na aba ‘Events’ o item UPDATE foi desmarcado. Por isso ‘BEFORE INSERT’ somente.
- Agora pode-se ‘SALVAR’ este gatilho.

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- Construindo no pgAdmin4.
- Veja como fica a definição do gatilho vinculado na tabela:



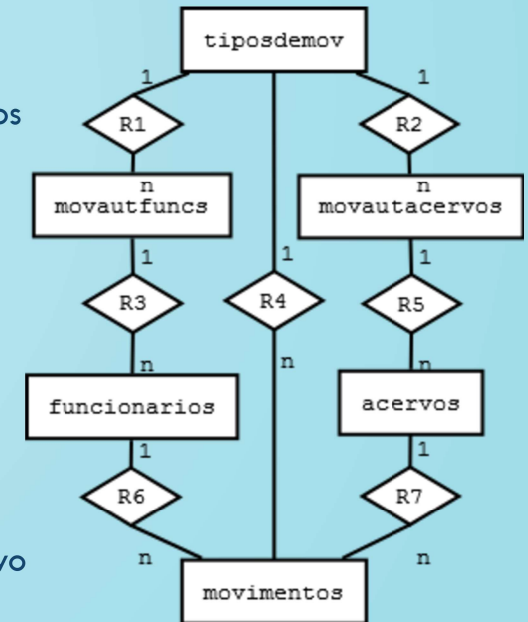
The screenshot shows the pgAdmin 4 interface with the SQL editor open. The left sidebar shows the database structure with 'Triggers (1)' expanded, showing 'BIautorias'. The SQL editor contains the following code:

```
1  -- Trigger: BIautorias
2
3  -- DROP TRIGGER IF EXISTS "BIautorias" ON public.autorias;
4
5  CREATE OR REPLACE TRIGGER "BIautorias"
6    BEFORE INSERT
7    ON public.autorias
8    FOR EACH ROW
9    EXECUTE FUNCTION public."autoriasBI"();
```

- Agora podemos baixar e iniciar a análise e escrita dos gatilhos do Projeto – 2.
- Mas antes... um exercício.

IBD100 – GATILHOS - O MODO DE IMPLEMENTAÇÃO DO POSTGRESQL

- Agora vamos para algo mais complicado...
- Quando um funcionário movimentar uma peça do acervo, deve-se verificar SE o tipo de movimento pretendido (no insert movimentos) está na lista de tipos de movimentos autorizados para o funcionário E para a peça do acervo.
- Note: ANTES de fazer um insert na tabela movimentos, devemos fazer duas seleções nas tabelas movautfuncs e movautacervos e verificar se o valor do código do tipo de movimento (cdtipomov) está nas duas tabelas.
- Podemos contar a quantidade de linhas e verificar se o valor é maior que 0 (zero) nas duas contagens.
- Escreva os comandos de definição da Trigger Function e o vínculo com a tabela em um arquivo TXT e anexe na tarefa proposta no Teams



- Agora escolha **dois** dos gatilhos propostos no exercício de Gatilhos&Relatórios.
- Determine seu nome, evento e momento associado.
- Use o pgAdmin4 para montar a função de gatilho e o gatilho.

Bom trabalho!



LABORATÓRIO DE BANCO DE DADOS

IBD100