

**INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DO
MARANHÃO
CAMPUS SÃO LUÍS - MONTE CASTELO
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

JHONATHAN DA ROCHA DA CRUZ

KATERINY BISPO DE DEUS

SAULO FERRO MACIEL

Contribuições para o PostgreSQL

SÃO LUÍS – MA

2025

SUMÁRIO

1. Contribuições Técnicas.....	3
1.1.1. Contexto da Contribuição.....	3
1.1.2. Ambiente de Desenvolvimento.....	3
1.1.3. Implementação do Teste.....	4
1.1.4. Execução e Validação.....	7
1.1.5. Relevância da Contribuição.....	7
1.1.6. Forma de Submissão à Comunidade.....	7
2. Contribuições de Documentação/Suporte.....	8
Nova seção: Exemplo prático completo de Replicação Lógica.....	9
Comportamento do Autovacuum.....	9
3. Contribuições Criativas/Divulgação.....	10

1. Contribuições Técnicas

1.1. Teste

Esta contribuição consiste na elaboração de um **teste automatizado de regressão** para o PostgreSQL, utilizando a suíte oficial de testes do projeto, baseada na ferramenta `pg_regress`.

O objetivo do teste é **validar o comportamento de funções de agregação SQL na presença de valores NULL**, garantindo que operações como **COUNT** e **AVG** sigam o padrão esperado e não apresentem regressões em versões futuras do sistema.

1.1.1. Contexto da Contribuição

O PostgreSQL é um sistema de gerenciamento de banco de dados relacional de código aberto amplamente utilizado, cujo desenvolvimento é sustentado por uma comunidade ativa. Para assegurar a estabilidade e a confiabilidade do sistema, o projeto mantém uma ampla suíte de testes automatizados.

Dentro desse contexto, a contribuição realizada insere-se na categoria de **Testes Automatizados**, conforme previsto no menu de contribuições da disciplina.

1.1.2. Ambiente de Desenvolvimento

Para a realização da contribuição, foi utilizado o **Windows Subsystem for Linux (WSL)**, com distribuição Ubuntu, por oferecer um ambiente Linux compatível com o fluxo oficial de desenvolvimento do PostgreSQL.

As principais etapas de configuração do ambiente envolveram:

- Instalação das dependências de compilação (GCC, Make, bibliotecas de desenvolvimento);
 - Clonagem do repositório oficial do PostgreSQL;

```
Arquivos\ de\ Programas\ MSOCache\ Program\ Files\ (x86)\ Windows\ ktb@eijoade:/mnt/c$ git clone https://github.com/postgres/postgres.git
Cloning into 'postgres'...
remote: Enumerating objects: 1091280, done.
remote: Counting objects: 100% (504/504), done.
remote: Compressing objects: 100% (205/205), done.
remote: Total 1091280 (delta 333), reused 313 (delta 298), pack-reused 1009776 (from 3)
Receiving objects: 100% (1091280/1091280), 686.68 MiB | 3.93 MiB/s, done.
Resolving deltas: 100% (906488/906488), done.
Updating files: 100% (7396/7396), done.
```

- Execução do script `configure` para verificação do ambiente:

```
[root@ejioada ~]# cd postgres  
[root@ejioada ~]# /etc/init.d/postgresql-9.1 start  
[root@ejioada ~]# psql -U postgres  
psql (9.1.4) Type "help" for help.  
postgres=1> \c ejioada  
You are now connected to database "ejioada" as user "postgres".  
ejioada=1> \d  
           List of relations  
 Schema |   Name    | Type  |  Owner  
-----+-----+-----+-----+  
 public | ejioada | table | postgres  
 ejioada=1> \q
```

- Compilação do sistema por meio do comando `make`.

```
:~/t@Feijoada:/mnt/c/postgres$ make  
make -C ./src/backend generated-headers  
make[1]: Entering directory '/mnt/c/postgres/src/backend'  
make -C ../include/catalog generated-headers  
make[2]: Entering directory '/mnt/c/postgres/src/include/catalog'  
'/usr/bin/perl' ../../src/backend/catalog/genbki.pl --include-path=../../.../  
    -c -o /.../src/include/catalog/pg_bki.h /.../src/include/catalog/pg_bki.h
```

1.1.3. Implementação do Teste

O teste foi implementado seguindo o padrão adotado pela comunidade do PostgreSQL, que utiliza arquivos SQL para definição dos testes e arquivos de saída esperada para validação dos resultados.

Foram criados os seguintes arquivos:

- Um arquivo `.sql`, contendo os comandos SQL a serem testados;

```
SQL
-- Teste de comportamento de agregações com valores NULL

CREATE TEMP TABLE t (x INT);

INSERT INTO t VALUES (1), (NULL), (2);

-- COUNT ignora NULL
SELECT COUNT(x) FROM t;

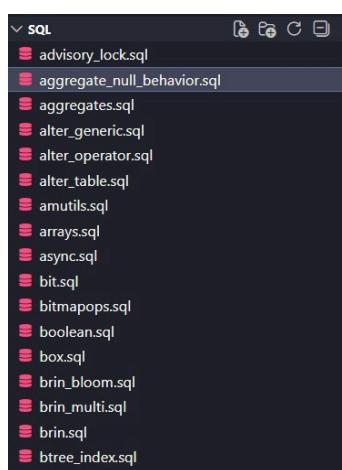
-- COUNT(*) conta todas as linhas
SELECT COUNT(*) FROM t;

-- AVG ignora NULL
SELECT AVG(x) FROM t;

-- Teste com tabela vazia
DELETE FROM t;

SELECT AVG(x) FROM t;
```

A imagem a seguir exibe o arquivo de contribuição de teste inserido no diretório clonado da comunidade:



Um arquivo `.out`, contendo o resultado esperado da execução desses comandos;

resultado esperado da execução

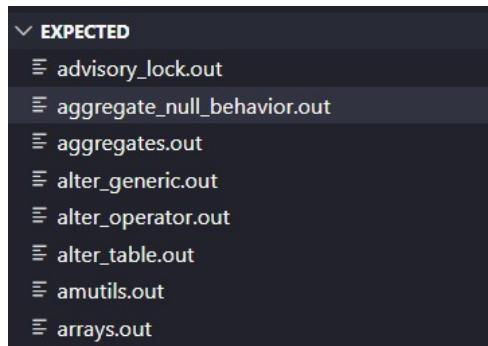
```
SQL
count
-----
  2
(1 row)

count
-----
  3
(1 row)

avg
-----
1.500000000000000
(1 row)

avg
-----
(1 row)
```

A imagem a seguir exibe o arquivo de contribuição de teste inserido no diretório clonado da comunidade:



Registro do teste no arquivo parallel_schedule, responsável por organizar a execução paralela dos testes.

SQL

```
# -----
# Another group of parallel tests
# aggregates depends on create_aggregate
# join depends on create_misc
# -----
test: select_into select_distinct select_distinct_on select_implicit select_having subselect
union case join aggregates transactions random portals arrays btree_index hash_index update
delete namespace prepared_xacts
test: select_into select_distinct select_distinct_on select_implicit select_having subselect
union case join aggregates transactions random portals arrays btree_index hash_index update
delete namespace prepared_xacts aggregate_null_behavior
```

```
# -----
# Another group of parallel tests
# aggregates depends on create_aggregate
# join depends on create_misc
# -----
test: select_into select_distinct select_distinct_on select_implicit select_having subselect u
test: select_into select_distinct select_distinct_on select_implicit select_having subselect u
|
# -----
# Another group of parallel tests
```

O teste verifica especificamente:

- O comportamento da função **COUNT** ao ignorar valores NULL;
- A diferença entre **COUNT(coluna)** e **COUNT(*)**;
- O comportamento da função **AVG** em conjuntos de dados contendo valores NULL e em tabelas vazias.

1.1.4. Execução e Validação

Após a compilação do PostgreSQL, o teste foi executado de forma isolada utilizando o comando:

```
kt@Feijoada:/mnt/c/postgres$ make check TESTS=aggregate_null_behavior
make -C ./src/backend generated-headers
make[1]: Entering directory '/mnt/c/postgres/src/backend'
make -C ../include/catalog generated-headers
make[2]: Entering directory '/mnt/c/postgres/src/include/catalog'
make[2]: Nothing to be done for 'generated-headers'.
make[2]: Leaving directory '/mnt/c/postgres/src/include/catalog'
make -C nodes generated-header-symlinks
make[2]: Entering directory '/mnt/c/postgres/src/backend/nodes'
make[2]: Nothing to be done for 'generated-header-symlinks'.
make[2]: Leaving directory '/mnt/c/postgres/src/backend/nodes'
make -C utils generated-header-symlinks
```

A execução do teste compara automaticamente a saída gerada pelo sistema com o arquivo de resultado esperado, validando a consistência do comportamento observado.

1.1.5. Relevância da Contribuição

A inclusão de novos testes automatizados contribui diretamente para:

- Aumento da cobertura de testes do PostgreSQL;
- Prevenção de regressões em funcionalidades existentes;
- Fortalecimento da confiabilidade do sistema ao longo de sua evolução.

Esse tipo de contribuição é valorizado pela comunidade, pois auxilia na manutenção da qualidade do software livre sem a necessidade de alterações diretas no código-fonte principal.

1.1.6. Forma de Submissão à Comunidade

A contribuição foi estruturada de acordo com o fluxo oficial do PostgreSQL, por meio da geração de um arquivo `.patch`, que pode ser submetido à lista de discussão da comunidade ([pgsql-hackers](#)) para revisão e possível integração ao projeto.

- Email de submissão do Patch de testes

[PATCH] Add regression test for aggregate NULL behavior



kateriny bispo <katerinybispo06@gmail.com>
para pgsql-hackers ▾

Hi,

This patch adds a new regression test covering the behavior of aggregate functions when handling NULL values, including COUNT, COUNT(*) and AVG on non-empty and empty relations.

The test was implemented using the pg_regress framework and follows the existing regression testing conventions.

Feedback is welcome.

Best regards,
Kateriny

1 anexo • Verificados pelo Gmail ⓘ Adicionar ao Google Drive



“Olá,

Este patch adiciona um novo teste de regressão que abrange o comportamento de funções de agregação ao lidar com valores NULL, incluindo COUNT, COUNT(*) e AVG em relações vazias e não vazias.

O teste foi implementado usando o framework pg_regress e segue as convenções de teste de regressão existentes.

Sugestões são bem-vindas.

Atenciosamente.”

- Resposta equipe moderadora do repositório:

A screenshot of an email message in Portuguese. The subject is "Message to postgresql-hackers held for moderation". The message body starts with "Parece que esta mensagem está em inglês" (It seems that this message is in English) and includes a link to translate it to Portuguese. It then states: "Your message to postgresql-hackers with subject '[PATCH] Add regression test for aggregate NULL behavior' has been held for moderation. It will be delivered to the list recipients as soon as it has been approved by a moderator. If you wish to cancel the message without delivery, please click this link: <https://lists.postgresql.org/manage/unqueue/69980f2d223045aba40a949249114358827673752d76d82a742cc433d8969bb/>".

“Sua mensagem para postgresql-hackers com o assunto “[PATCH] Adicionar teste de regressão para comportamento de agregação NULL” foi liberada da moderação e será entregue a todos os membros da lista em breve.”

2. Contribuições de Documentação/Suporte

2.1. Replicação Lógica — logical-replication.sgml

A documentação:

- Explicava conceitos (publisher, subscriber, slot, publication)
- Tinha poucos exemplos completos
- Não deixava claro o fluxo real de configuração
- Não abordava erros comuns

Com a melhoria:

Nova seção: Exemplo prático completo de Replicação Lógica

Replicação lógica permite replicar dados de forma seletiva usando um modelo de publicação e assinatura.

Este exemplo demonstra um fluxo completo de configuração entre dois servidores PostgreSQL.

2.2. Autovacuum — maintenance.sgml

A documentação:

- Autovacuum descrito de forma teórica
- Pouca clareza sobre quando ele não roda
- Pouca explicação de parâmetros críticos

Com a melhoria

Comportamento do Autovacuum

O autovacuum é um processo automático responsável por:

- Remover tuplas mortas
- Atualizar estatísticas
- Prevenir o *wraparound* de IDs de transação

No entanto, o autovacuum pode não executar imediatamente mesmo quando o limite de tuplas mortas é atingido, dependendo de:

- Carga do sistema
- Número de workers disponíveis
- Prioridade do processo

2.3. Freeze e Wraparound — mvcc.sgml

A documentação:

- Conceito de Freeze explicado
- Pouca ligação prática com Autovacuum
- Pouca orientação operacional

Com a melhoria

- Relação com Autovacuum
- Alerta crítico adicionado

- E-mail com o .path com as novas sessões adicionadas.

[PATCH] Docs: logical replication examples and autovacuum/XID freeze clar...   

[pgsql-docs@postgresql.org](#), [pgsql-hackers@postgresql.org](#)

[PATCH] Docs: logical replication examples and autovacuum/XID freeze clarifications

Hello,

This patch adds complementary documentation sections with practical examples for logical replication, and clarifies autovacuum behavior and transaction ID freezing (anti-wraparound).

No existing documentation was removed or modified.

Files changed:

- doc/src/sgml/logical-replication.sgml
- doc/src/sgml/maintenance.sgml
- doc/src/sgml/mvcc.sgml

Tests performed:

- make html

Patch attached.

Regards,
Jhonathan Cruz

[0001-docs-add-logical-replication-example... \(9 K\)](#)  

- Resposta do Repositório:

Message to pgsql-hackers held for moderation

 External

 Caixa de Entrada

pgsql-hackers Owner <pgsql-hackers-notice+M111886-dc8d3c6181f87dd7afdbe439ecef506c6c54f9ba@pg...>
para mim ▾

Your message to pgsql-hackers with subject
"[PATCH] Docs: logical replication examples and autovacuum/XID freeze clarifications"
has been held for moderation.

It will be delivered to the list recipients as soon as it has been
approved by a moderator.

2.3. DISTINCT ON + ORDER BY e JIT with aggregation

3. Contribuições Criativas/Divulgação

A equipe propôs criar alternativas para melhorar a divulgação aos novos usuários. Sendo assim, foi analisado o funil de marketing e clipping em redes sociais sobre o usufruto do PostgreSQL para poder ter uma noção dos problemas do projeto com sua base de usuários estruturais.

Foi constatado via comentários em fóruns que a maior problemática é a falta de uma unificação acessível para a introdução ao PostgreSQL.

Com isso, o integrante Saulo aproveitou sua game engine própria, na qual batizaram de Game Slide, para fazer um videojogo que unificasse os principais passos para aprendizado do SGBD.

edvsagun 2 d

Ótimo o projeto, eu particularmente achei incrível e pode se tornar até um negócio futuramente kkkk

Isso de "gameficar" foi genial, faz com que o usuário não só realize a instalação e tudo, mas que interaja também.

Uma única coisa que pensei, é caso eles precisem voltar para alguma área, os slides voltam a aparecer?

1 ❤️ ⚡ ... ↗ Responder

TabNews Relevantes Recentes Pesquisar

TabStore: Agora você pode vender seus TabCash! (ads.tabstore.com.br) ↗ Contribuído com alexolidev

1. STL Studio: O Editor STL Online para Impressão 3D e Modelagem Rápida 11 tabcoins · 0 comentário · gmasson · 1 dia atrás
2. Tive uma ideia inusitada para um trabalho da faculdade 7 tabcoins · 0 comentário · sauloFerroMaciel · 22 horas atrás

Obrigado por escolher o PostgreSQL!

Você acabou de tomar uma das decisões mais inteligentes para 2026.

O PostgreSQL é o SGBD open source mais avançado do mundo.

Apple, Netflix, Instagram, Uber, Spotify, Reddit e milhares de outras gigantes já fizeram a mesma escolha.

Saulo Maciel faz um dia

Tive uma ideia inusitada para um trabalho da faculdade

#HTML #JavaScript #Banco de dados relacional

Depois de 3 dias de desenho de elementos (cenário e personagens), o jogo de plataforma foi divulgado em fóruns e no youtube como alternativa de aprendizado que usa o viés de “**gamificação infantil**”; princípio de simplificar processos para que todas as faixas etárias consigam absorver partes do conteúdo com pouco esforço.

Como trata-se de um protótipo, o jogo foi disponibilizado via Github Pages com licença MIT. Pode usufruí-lo pelo link: rebrand.ly/qsdw7w

O projeto foi bem abraçado pelas comunidades. Alguns sugeriram que

transformassem o Game Slide em um serviço educacional, como pode-se observar nas imagens retiradas do Diolinux (link: rebrand.ly/eeqlqr4).

Podemos observar que o projeto obteve aceitação por vários usuários que forneceram feedbacks coesos, sensatos e humorados, o que evidencia êxito da equipe em inovar na comunicação mais assertiva e divertida.

Em outros fóruns, o projeto teve bastante visualizações. Chegando a ficar nos tópicos mais relevantes como foi no caso do TabNews (link: rebrand.ly/3kqenmm) onde ficou em 2º lugar e no Dio (link: rebrand.ly/xjdaa7i) que passou de 100 visualizações e impulsionamentos.