

Escalabilidade Horizontal com Postgres-XL

William Ivanski

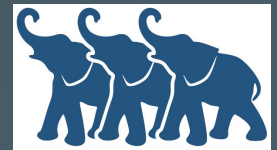
2ndQuadrant

PgConf Brasil 2018

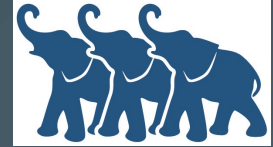
Sumário

- **Parte 1: Conceitos**
- **Parte 2: Desempenho**
- **Parte 3: Instalação**
- **Parte 4: Utilização**
- **Parte 5: Escalabilidade**

Parte 1: Conceitos

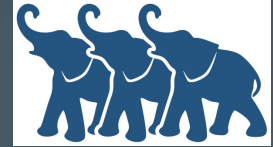


O que é Postgres-XL?

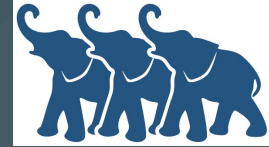


- É uma versão do PostgreSQL
 - **Melhor desempenho**
 - **Melhor escalabilidade**
- Open Source
- Mesma licença que o PostgreSQL
- Desenvolvimento iniciou há muitos anos, muitas pessoas e empresas envolvidas

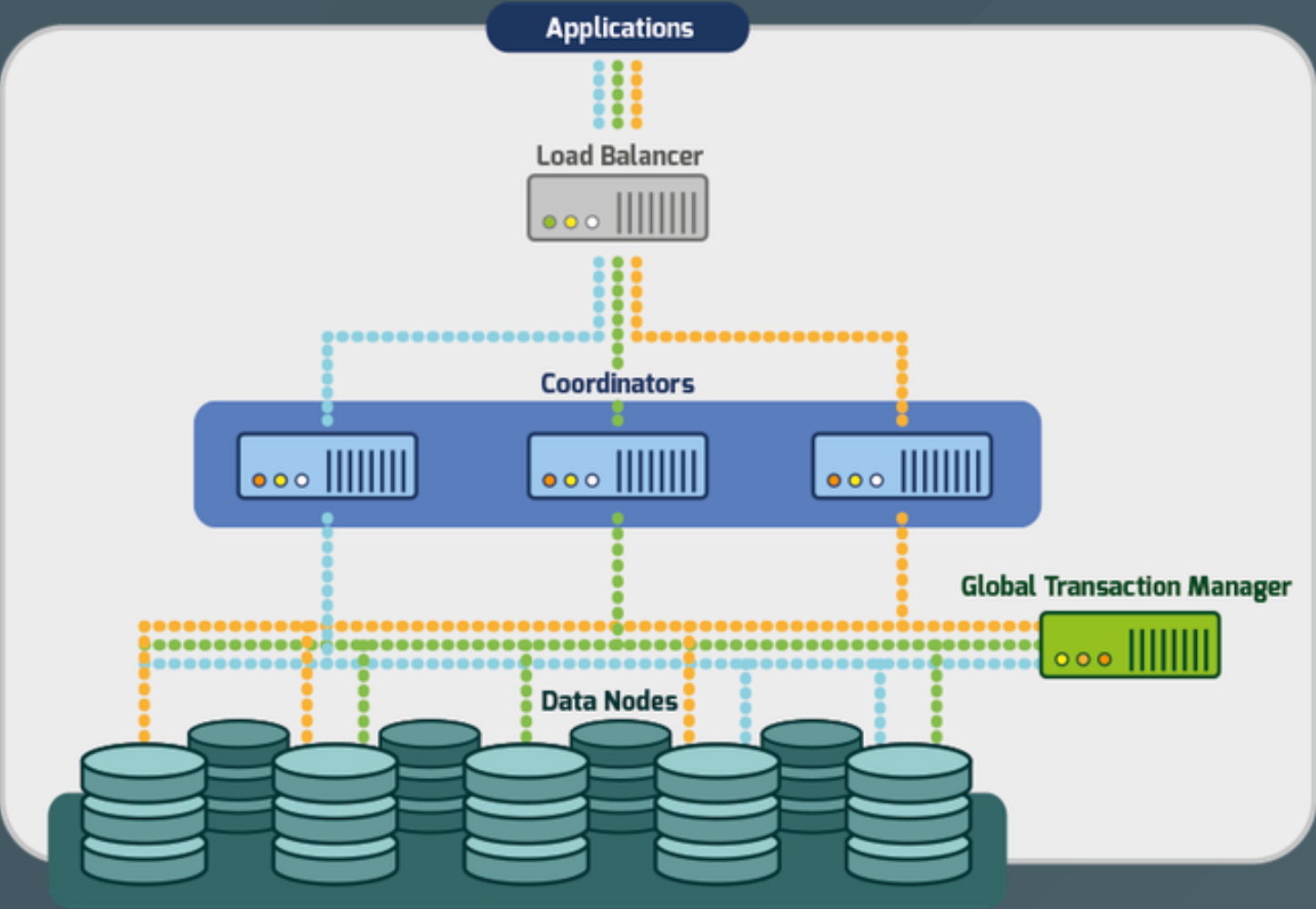
O que é Postgres-XL?



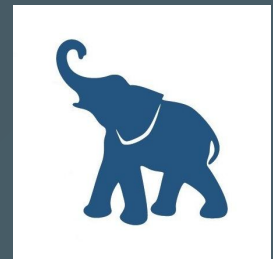
- **Scale-Out**
- Escalabilidade de leitura e escrita
 - Queries maiores divididas igualmente
 - Queries de apenas 1 chave tocam apenas 1 nodo
- **MPP**: Massive Parallel Processing
- Cluster-wide **ACID**
- **Big Data**



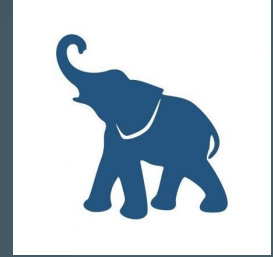
Arquitetura



Coordinator

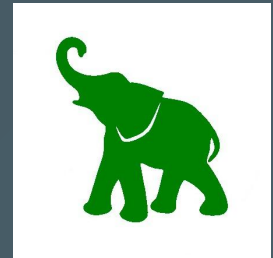


Coordinator

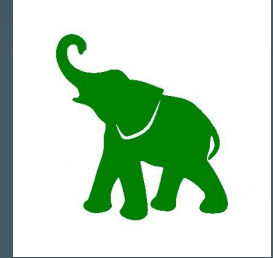


- **Parsing and planning** para queries
- Sabe como as tabelas estão distribuídas e usa essa informação para o planejamento
- Aplicações se conectam a ele
- É possível ter vários no mesmo cluster
- Não existe coordinator "mestre"
- Não é um componente crítico para o cluster

Datanode

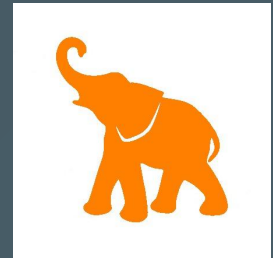


Datanode

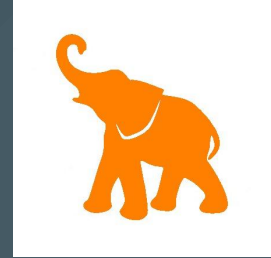


- Onde os dados são armazenados
- Execução local de consultas
- **Aplicações não se conectam a ele!!**
- Componente crítico para o cluster
 - Cada datanode pode ter vários **standbys**
- Falha de um nodo pode causar **erro** em DML/DDL (se a tabela não for replicada, erro em DQL também)

GTM



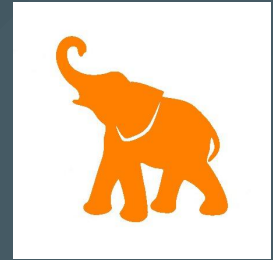
GTM



- **Global Transaction Manager**
- Componente central e único no cluster
- **GXID**: identificadores de transação globalmente visíveis e consistentes
- **Snapshots** para transações sendo executadas no cluster inteiro
- Graças a ele o cluster é totalmente **ACID**

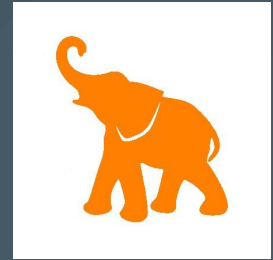
GTM

- SPOF??

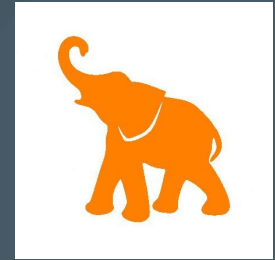


GTM

- SPOF??
 - Sim.

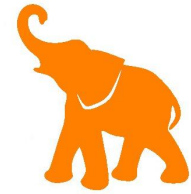


GTM



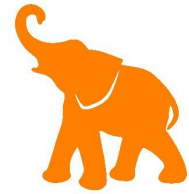
- **SPOF??**
 - Sim.
 - Solução: **GTM Standby**

GTM



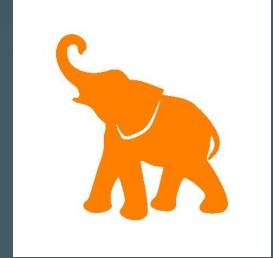
- **Performance bottleneck??**

GTM



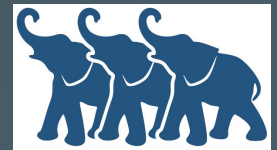
- **Performance bottleneck??**
 - Sim.

GTM

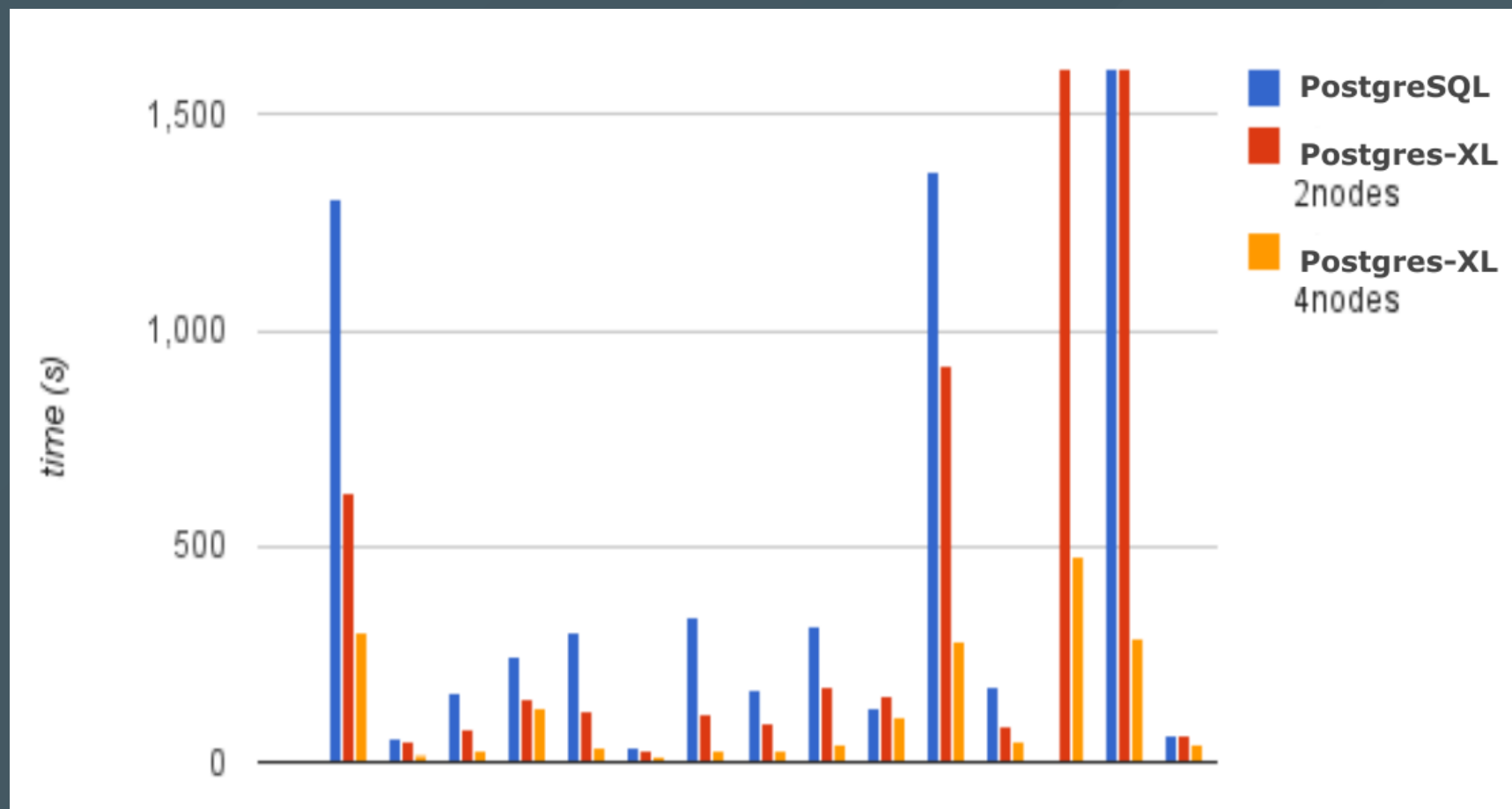
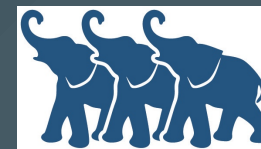


- **Performance bottleneck??**
 - Sim.
 - Solução: **GTM Proxy**
 - Roda no mesmo servidor físico que o Datanode ou Coordinator
 - Agrupa requisições e manda uma só vez
 - Recebe uma range de GXID e snapshots do GTM

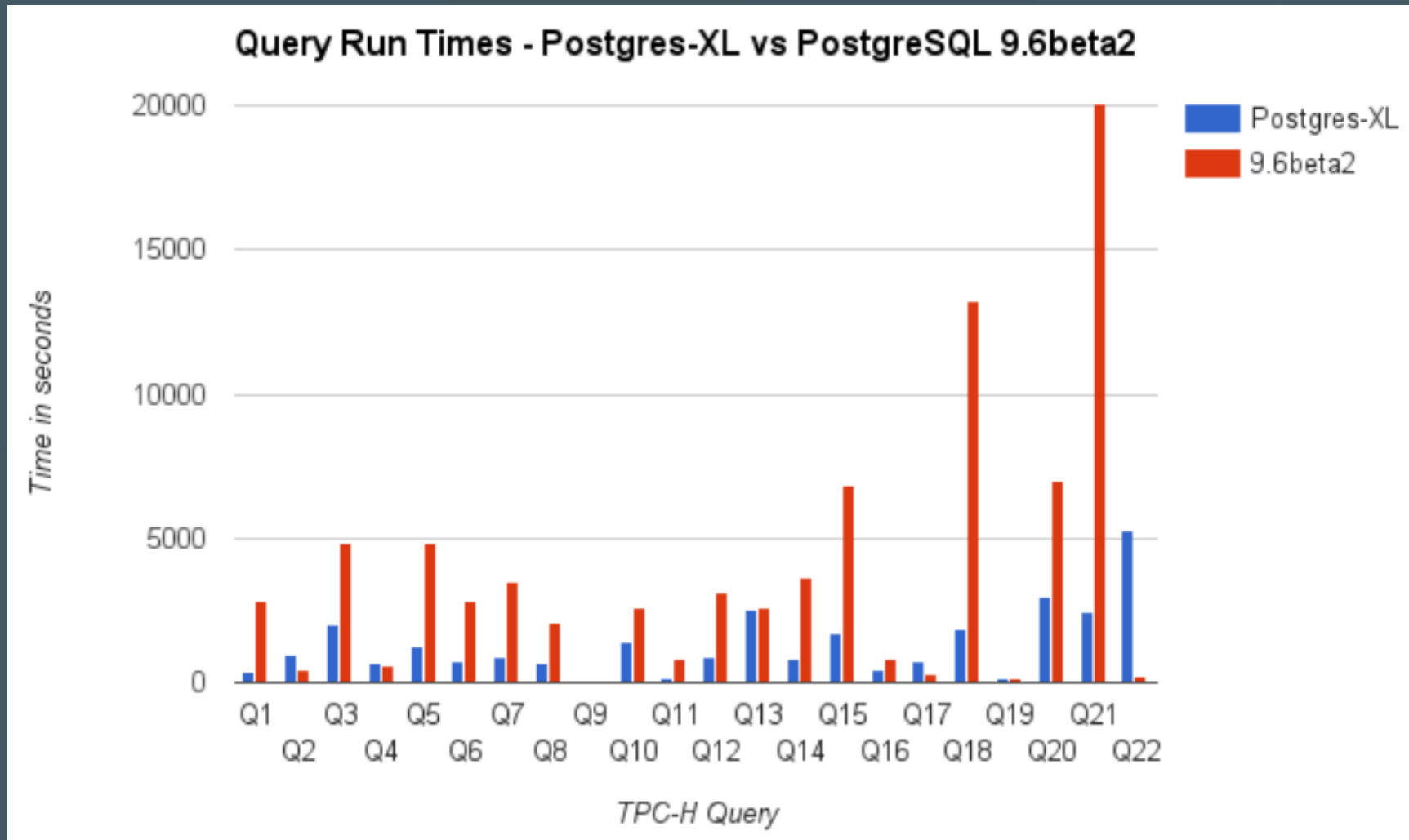
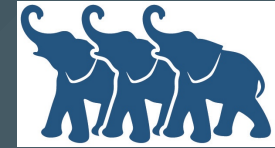
Parte 2: Desempenho



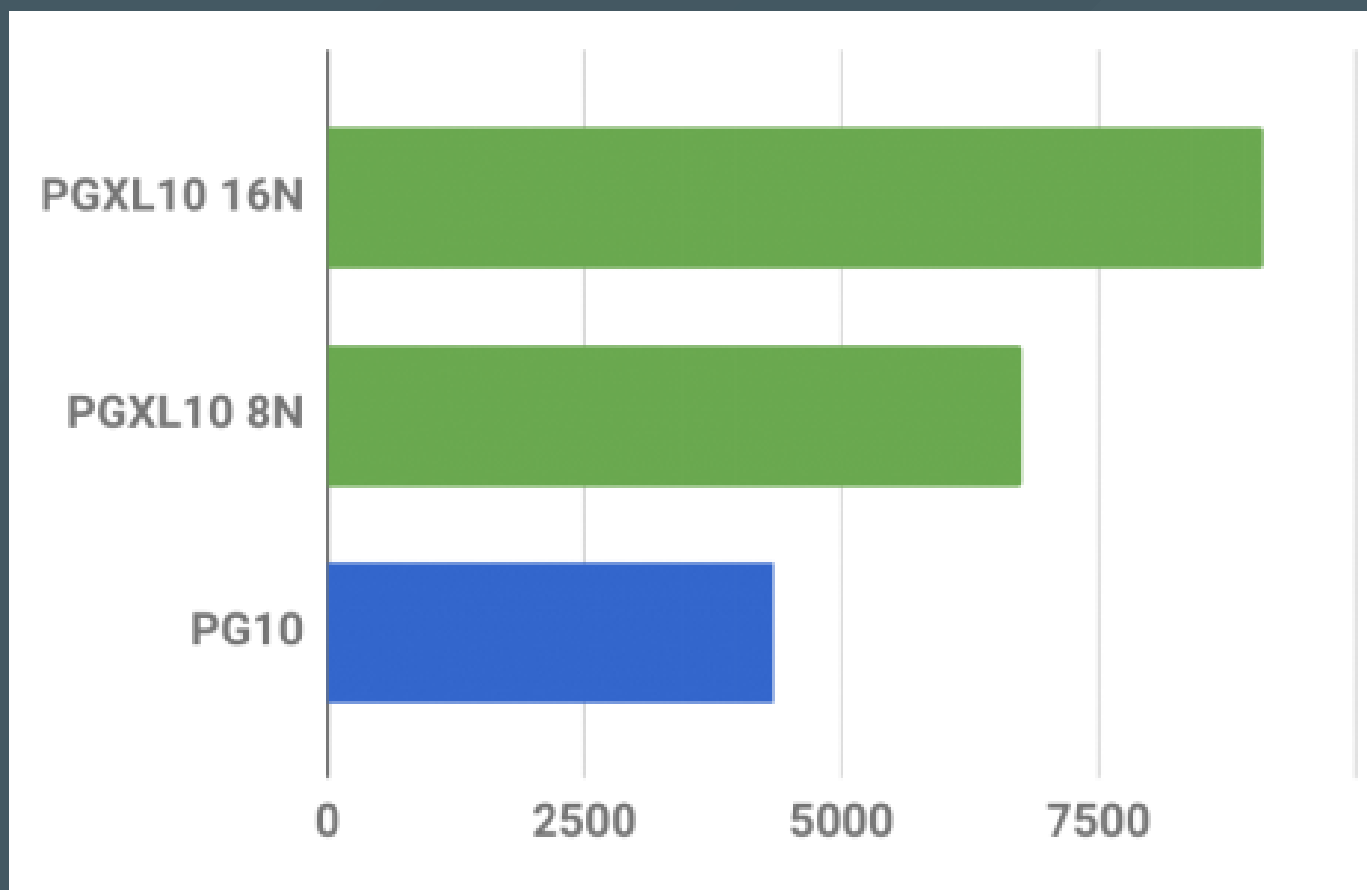
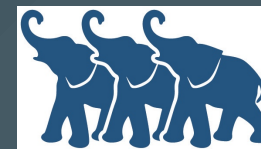
Desempenho PGXL 9.5



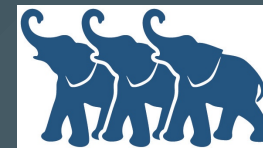
Desempenho PGXL 9.6



Desempenho PGXL 10

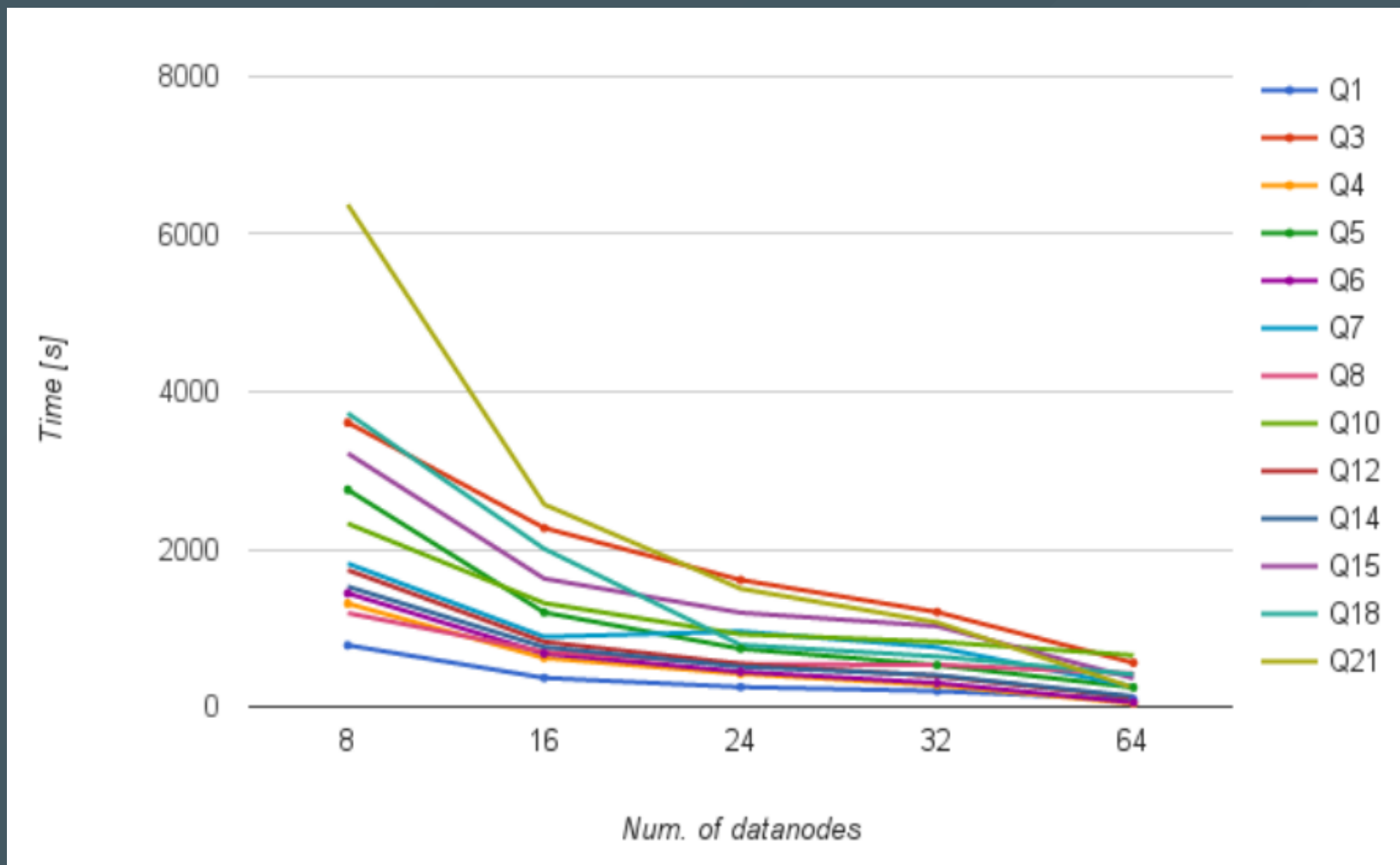
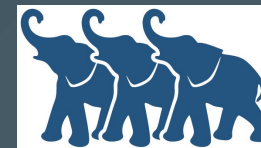


Desempenho Escalável

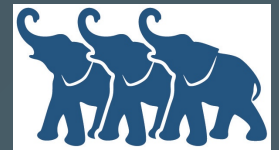


- Custo do Paralelismo: **~30% overhead** (TPC-W workload)
- Cluster com 1 GTM, 1 Coordinator, 1 Datanode:
 - **70%** do desempenho do "vanilla" PostgreSQL
- **10 nodos** deveria ter **desempenho 7x maior** que o do PostgreSQL
- Resultado atual: **6x - 6.4x**
- Mais nodos, mais transações abertas, maiores snapshots e maior verificação de visibilidade

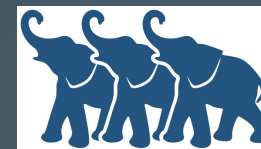
Desempenho Escalável



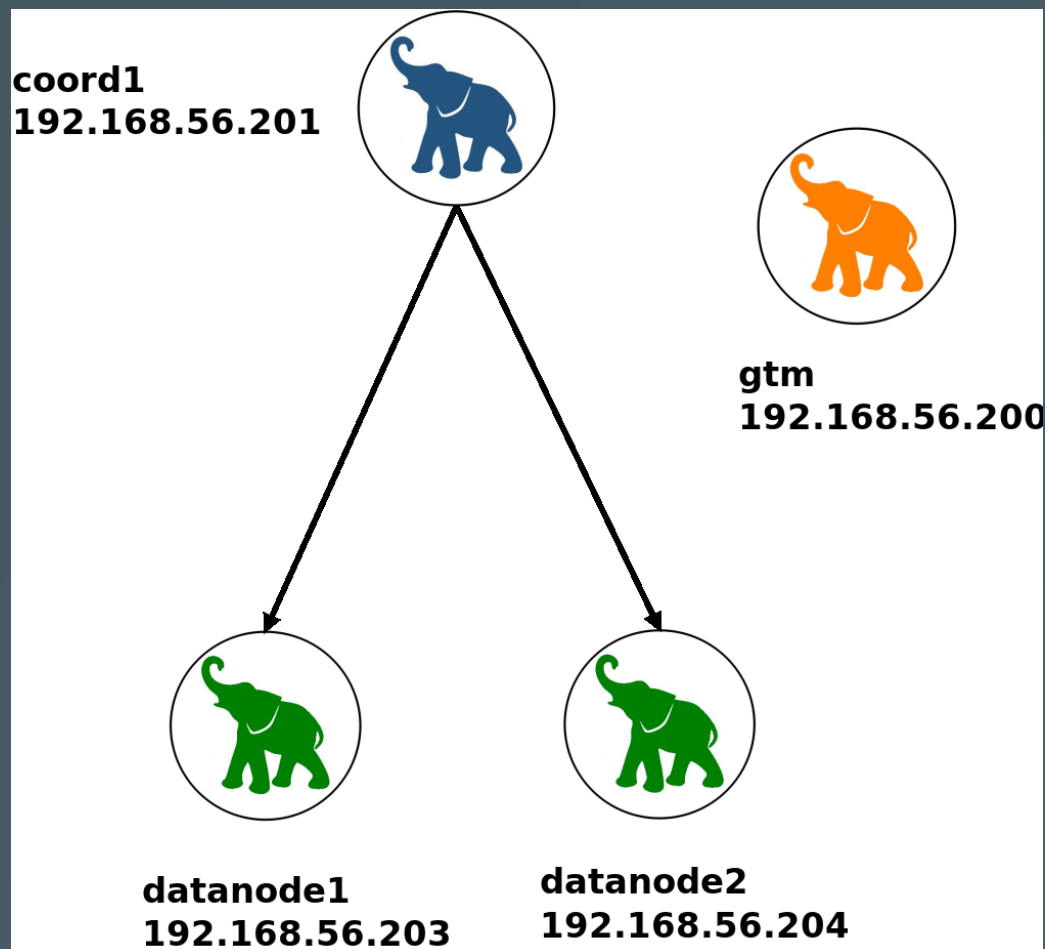
Parte 3: Instalação



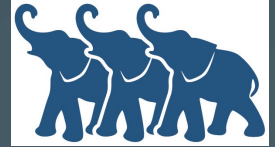
Instalação



- Cluster:
 - 1 GTM
 - 1 coordinator
 - 2 datanodes



Em todos os nodos



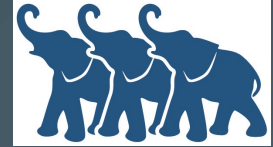
- Instalar dependências:

```
apt-get update
apt-get -y upgrade
apt-get -y install build-essential git bison flex \
    libreadline-dev zlib1g-dev
```

- Clonar repositório:

```
git clone git://git.postgresql.org/git/postgres-xl.git
cd postgres-xl
git checkout XL9_5_STABLE
```

Em todos os nodos



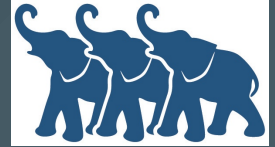
- Compilar:

```
./configure  
make -j4  
make install  
cd contrib  
make -j4  
make install
```

- Adicionar `/usr/local/pgsql/bin` ao `$PATH`:

```
echo "PATH=/usr/local/pgsql/bin:$PATH" > /etc/environment
```

Em todos os nodos



- Criar usuário `postgres`:

```
useradd -m -s /bin/bash -U postgres
```

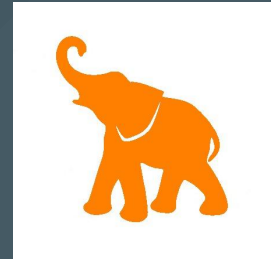
- Criar data folder:

```
mkdir /usr/local/pgsql/data  
chown postgres /usr/local/pgsql/data
```

- Logar como `postgres`:

```
su postgres
```

GTM



- Inicializar diretório de dados:

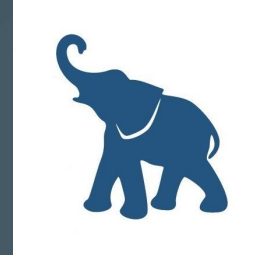
```
initgtm -D /usr/local/pgsql/data -Z gtm
```

Porta padrão é `6666`. É possível mudar a porta e outras opções no arquivo `gtm.conf`.

- Iniciar GTM:

```
gtm_ctl -Z gtm -D /usr/local/pgsql/data -l logfile start
```

Coordinator 1



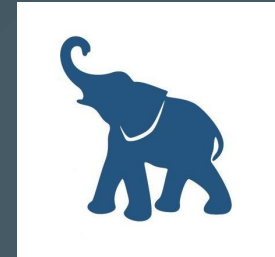
- Inicializar diretório de dados:

```
initdb -D /usr/local/pgsql/data --nodename coord1
```

- Alterar `postgresql.conf`:

```
listen_addresses = '*'  
pooler_port = 40100 # único para cada node  
gtm_host = '192.168.56.200'  
gtm_port = 6666
```

Coordinator 1



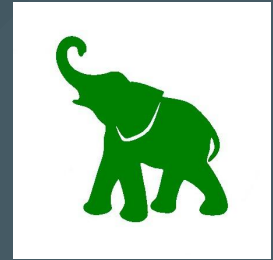
- Alterar `pg_hba.conf`:

```
local all postgres trust
host all postgres 127.0.0.1/32 trust
host all postgres ::1/128 trust
host all postgres 192.168.56.203/32 trust # Datanode 1
host all postgres 192.168.56.204/32 trust # Datanode 2
host all all 192.168.56.1/32 md5
```

- Iniciar coordinator 1:

```
pg_ctl -D /usr/local/pgsql/data -Z coordinator \
-l logfile start
```


Datanode 1



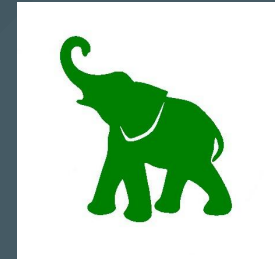
- Inicializar diretório de dados

```
initdb -D /usr/local/pgsql/data --nodename datanode1
```

- Alterar `postgresql.conf`:

```
listen_addresses = '*'  
pooler_port = 40102 # único para cada node  
gtm_host = '192.168.56.200'  
gtm_port = 6666
```

Datanode 1



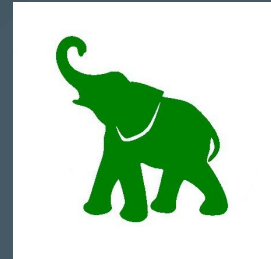
- Alterar `pg_hba.conf`:

```
local all postgres                                trust
host   all postgres 127.0.0.1/32                  trust
host   all postgres ::1/128                      trust
host   all postgres 192.168.56.201/32            trust # Coord 1
host   all postgres 192.168.56.204/32            trust # Datanode 2
host   all all       192.168.56.1/32              md5
```

- Iniciar datanode 1:

```
pg_ctl -D /usr/local/pgsql/data -Z datanode \  
-l logfile start
```

Datanode 2



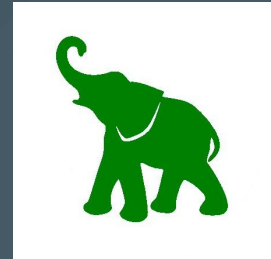
- Inicializar diretório de dados

```
initdb -D /usr/local/pgsql/data --nodename datanode2
```

- Alterar `postgresql.conf`:

```
listen_addresses = '*'  
pooler_port = 40103 # único para cada node  
gtm_host = '192.168.56.200'  
gtm_port = 6666
```

Datanode 2



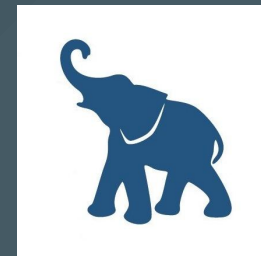
- Alterar `pg_hba.conf`:

```
local all postgres                                trust
host   all postgres 127.0.0.1/32                  trust
host   all postgres ::1/128                      trust
host   all postgres 192.168.56.201/32            trust # Coord 1
host   all postgres 192.168.56.203/32            trust # Datanode 1
host   all all       192.168.56.1/32             md5
```

- Iniciar datanode 2:

```
pg_ctl -D /usr/local/postgresql/data -Z datanode \  
-l logfile start
```

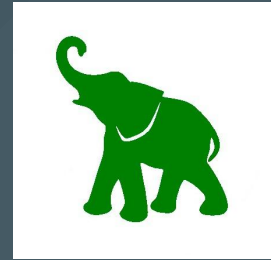
Coordinator 1



- Criar todos os nodes:

```
ALTER NODE coord1 WITH (TYPE = 'coordinator',  
                        HOST = 'localhost', PORT = 5432);  
  
CREATE NODE datanode1 WITH (TYPE = 'datanode',  
                           HOST = '192.168.56.203', PORT = 5432);  
  
CREATE NODE datanode2 WITH (TYPE = 'datanode',  
                           HOST = '192.168.56.204', PORT = 5432);  
  
SELECT pgxc_pool_reload();
```

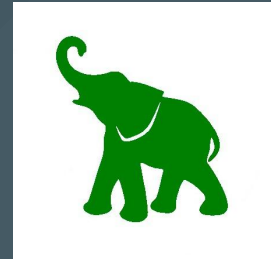
Datanode 1



- Criar todos os nodes:

```
ALTER NODE datanode1 WITH (TYPE = 'datanode',  
    HOST = 'localhost', PORT = 5432);  
  
CREATE NODE coord1 WITH (TYPE = 'coordinator',  
    HOST = '192.168.56.201', PORT = 5432);  
  
CREATE NODE datanode2 WITH (TYPE = 'datanode',  
    HOST = '192.168.56.204', PORT = 5432);  
  
SELECT pgxc_pool_reload();
```

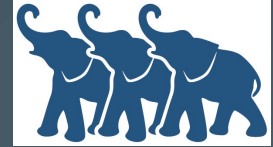
Datanode 2



- Criar todos os nodes:

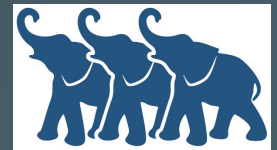
```
ALTER NODE datanode2 WITH (TYPE = 'datanode',  
    HOST = 'localhost', PORT = 5432);  
  
CREATE NODE coord1 WITH (TYPE = 'coordinator',  
    HOST = '192.168.56.201', PORT = 5432);  
  
CREATE NODE datanode1 WITH (TYPE = 'datanode',  
    HOST = '192.168.56.203', PORT = 5432);  
  
SELECT pgxc_pool_reload();
```

Pronto!

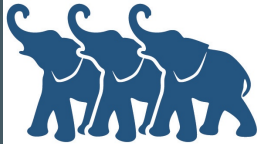


- Agora seu cluster XL está pronto para uso
- Teria sido mais fácil usar `pgxc_ctl` (recomendado também para clusters muito grandes)
- Agora você pode:
 - Mudar a senha do usuário `postgres`
 - Criar tabelas e começar a alimentar o cluster
 - Usar `pg_dump`/`pg_restore` para migrar de uma instância PostgreSQL existente

Parte 4: Utilização



Conectando ao cluster



OMNiDB Connections

Snippets 2.10.0 Postgres-XL Coord1

(Postgres-XL Coord1) postgres@postgres
192.168.56.201:5432

Active database: postgres

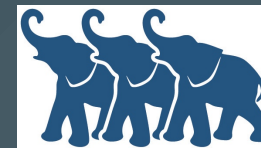
PostgreSQL 9.5.12 (Postgres-XL 9.5r1.6)

- Databases (1)
 - postgres
 - Schemas
 - Extensions
 - Foreign Data Wrappers
- Tablespaces
- Roles
- Replication Slots
- Postgres-XL
 - Nodes (3)
 - coord1
 - Type: coordinator
 - Host: localhost
 - Port: 5432
 - Primary: false
 - Preferred: false
 - datanode1
 - datanode2
 - Groups

Refresh Number of records: 2

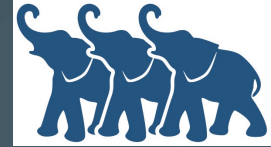
	Actions	datid	datname	pid	usesysid	username	application_name	client_addr
1	✖	12387	postgres	17305	10	postgres	psql	
2	✖	12387	postgres	17325	10	postgres	OmniDB	192.168.56.1

Tabelas Replicadas

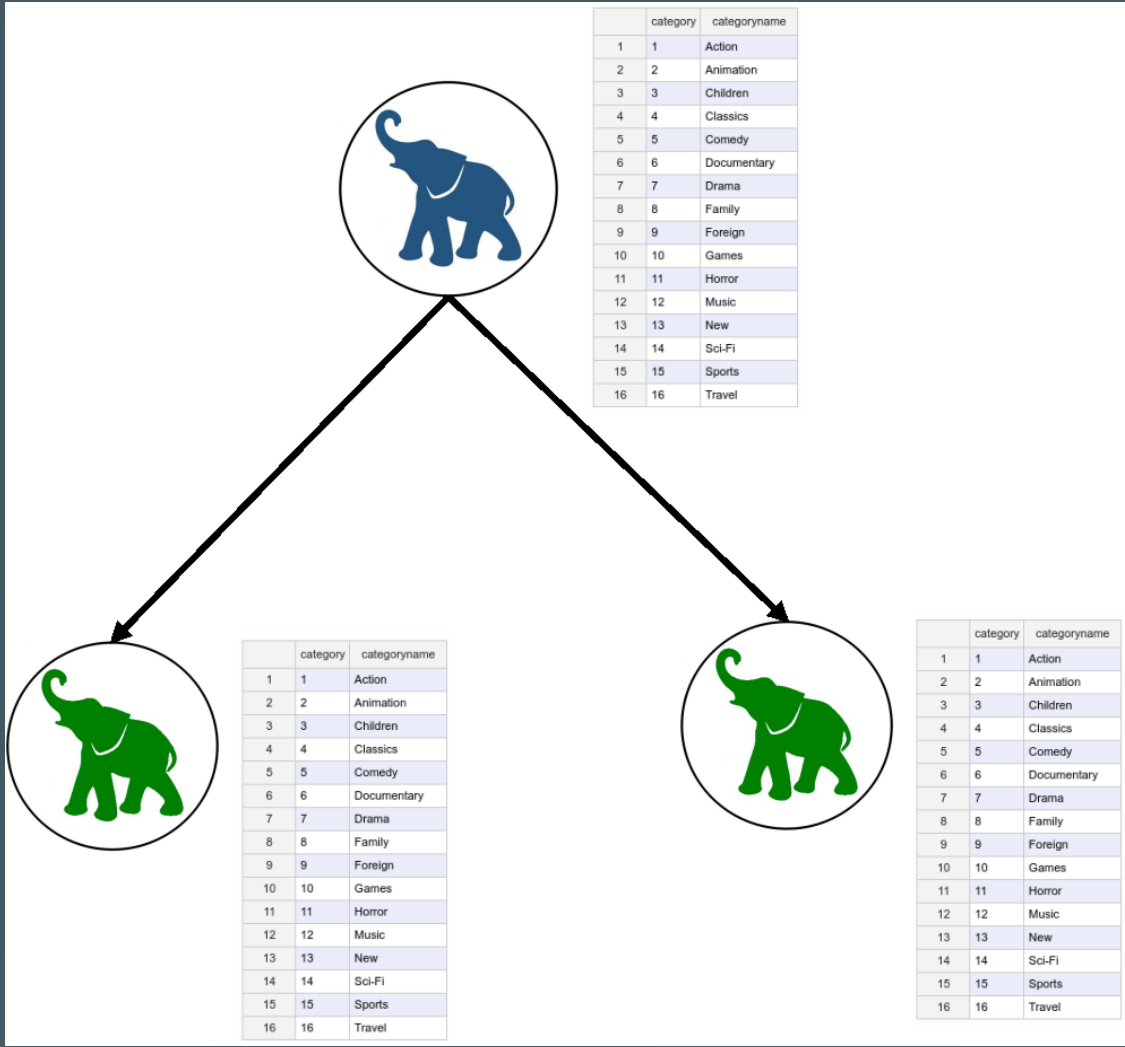


- Cada registro existe em **todos os datanodes**
- Bom para tabelas **read-only** e **read-mainly**

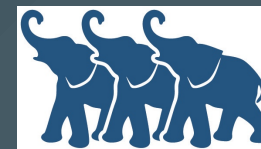
```
CREATE TABLE categories
(
    category integer NOT NULL,
    categoryname character varying(50) NOT NULL,
    CONSTRAINT categories_pkey PRIMARY KEY (category)
)
DISTRIBUTE BY REPLICATION;
```



Tabelas Replicadas



Tabelas Replicadas



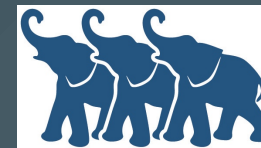
- Datanodes podem ser especificados como **PRIMARY** ou **PREFERRED**
- **PRIMARY** significa que será escrito primeiro

```
ALTER NODE datanode1 WITH (PRIMARY = true);
```

- **PREFERRED** significa que o datanode é preferido em operações de leitura

```
ALTER NODE datanode1 WITH (PREFERRED = true);
```

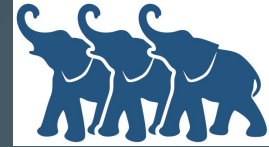
Tabelas Distribuídas



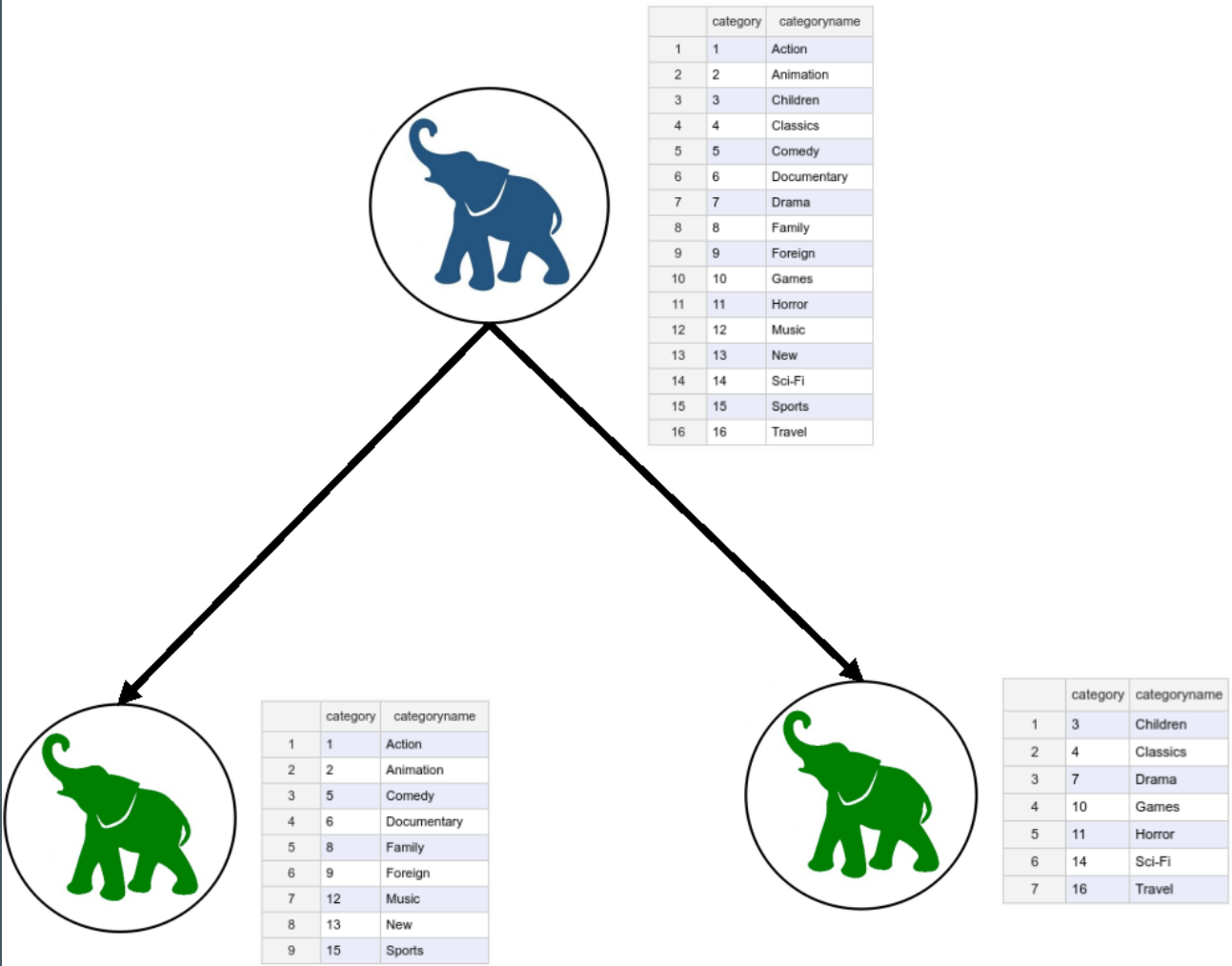
- **Fragmentação**: um registro qualquer existe em apenas **1 datanode**
- Bom para tabelas **write-only** e **write-mainly**

```
CREATE TABLE categories
(
    category integer NOT NULL,
    categoryname character varying(50) NOT NULL,
    CONSTRAINT categories_pkey PRIMARY KEY (category)
)
DISTRIBUTE BY HASH(category);
```

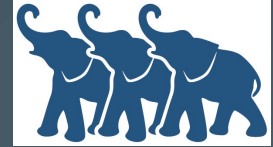
- `HASH(col)`, `MODULO(col)`, `ROUNDROBIN`



Tabelas Distribuídas



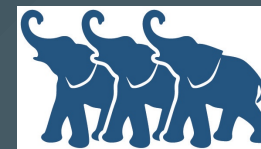
EXECUTE DIRECT



- Permite executar consultas diretamente em um datanode específico
- Deve ser rodado em um coordinator
- Somente superusuários podem rodar
- Transações, DDL e DML são proibidos!

```
EXECUTE DIRECT ON (datanode2) $$  
SELECT *  
FROM categories  
ORDER BY category  
$$
```


Dados da Distribuição



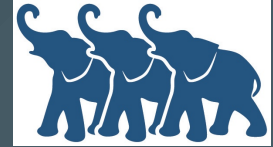
The screenshot shows the pgAdmin interface with a PostgreSQL database connection. The left pane displays the database structure, including the 'public' schema and the 'categories' table. The right pane shows a SQL query and its results.

```
1 SELECT t.category
2       , t.categoryname
3 FROM public.categories t
4 ORDER BY t.category
```

Number of records: 16
Start time: 08/02/2018 19:23:14

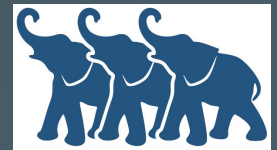
	category	categoryname
1	1	Action
2	2	Animation
3	3	Children
4	4	Classics
5	5	Comedy
6	6	Documentary
7	7	Drama
8	8	Family
9	9	Foreign
10	10	Games
11	11	Horror
12	12	Music
13	13	New
14	14	Sci-Fi
15	15	Sports
16	16	Travel

Limitações

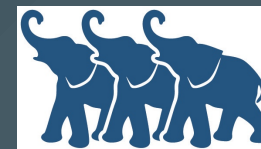


- Só é possível distribuir por apenas 1 coluna
 - PK e/ou FK precisam incluir essa coluna
- Não suportados no Postgres-XL atualmente:
 - Triggers
 - `CREATE INDEX CONCURRENTLY`
 - Foreign Data Wrappers
 - Mais detalhes: <https://www.postgres-xl.org/documentation/release-xl-9-5r1.html>

Parte 5: Escalabilidade

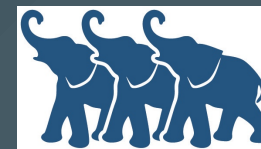


Escalabilidade



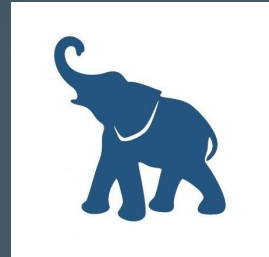
- **Escalabilidade horizontal** significa poder adicionar novos servidores ao cluster
- Por que adicionar mais nodos?
 - Dados aumentaram, limitação de hardware
 - Aumentar o desempenho do cluster atual
 - Se os dados aumentaram, manter o desempenho atual

Escalabilidade



- Vamos expandir o cluster para ter:
 - Novo coordinator `coord2`
 - Dois novos datanodes `datanode3` e `datanode4`
- Obrigatório que o cluster seja colocado em **modo de backup**
- Adicionar um novo nodo de cada vez

Coordinator 1

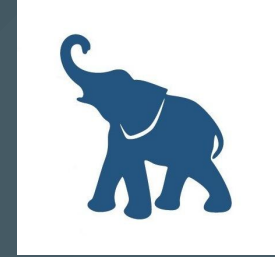


- Alterar `pg_hba.conf`:

```
local all postgres                                trust
host   all postgres 127.0.0.1/32                  trust
host   all postgres ::1/128                      trust
host   all postgres 192.168.56.202/32            trust # Coord2
host   all postgres 192.168.56.203/32            trust # Datanode 1
host   all postgres 192.168.56.204/32            trust # Datanode 2
host   all postgres 192.168.56.205/32            trust # Datanode 3
host   all postgres 192.168.56.206/32            trust # Datanode 4
host   all all      192.168.56.1/32              md5
```

- Recargar: `select pg_reload_conf();`

Coordinator 2



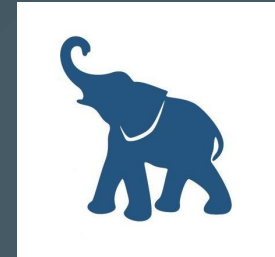
- Inicializar diretório de dados:

```
initdb -D /usr/local/pgsql/data --nodename coord2
```

- Alterar `postgresql.conf`:

```
listen_addresses = '*'  
pooler_port = 40101 # único para cada node  
gtm_host = '192.168.56.200'  
gtm_port = 6666
```

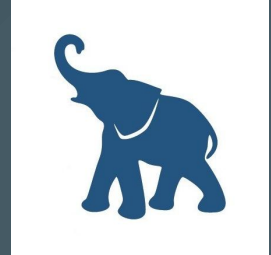
Coordinator 2



- Alterar `pg_hba.conf`:

```
local all postgres trust
host all postgres 127.0.0.1/32 trust
host all postgres ::1/128 trust
host all postgres 192.168.56.201/32 trust # Coord1
host all postgres 192.168.56.203/32 trust # Datanode 1
host all postgres 192.168.56.204/32 trust # Datanode 2
host all postgres 192.168.56.205/32 trust # Datanode 3
host all postgres 192.168.56.206/32 trust # Datanode 4
host all all 192.168.56.1/32 md5
```


Coordinator 2



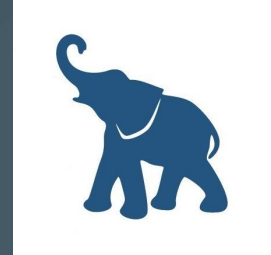
- No coordinator 1, bloquear o cluster inteiro (manter sessão aberta):

```
select pgxc_lock_for_backup();
```

- Criar dump do coordinator1:

```
pg_dumpall -h 192.168.56.201 -s --include-nodes \  
--dump-nodes --file=coord1.sql
```

Coordinator 2



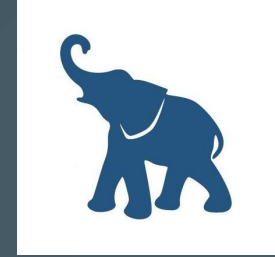
- Iniciar coordinator2 em modo de recuperação

```
pg_ctl -D /usr/local/pgsql/data -Z restoremode \  
-l logfile start
```

- Restaurar dump do coordinator1 no coordinator2:

```
psql -f coord1.sql
```

Coordinator 2



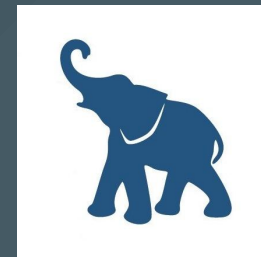
- Reiniciar coordinator2

```
pg_ctl -D /usr/local/pgsql/data -Z restoremode \  
-l logfile stop  
pg_ctl -D /usr/local/pgsql/data -Z coordinator \  
-l logfile start
```

- Ajustar endereços:

```
ALTER NODE coord2 WITH (TYPE = 'coordinator',  
                        HOST = 'localhost', PORT = 5432);  
ALTER NODE coord1 WITH (TYPE = 'coordinator',  
                        HOST = '192.168.56.201', PORT = 5432);  
SELECT pgxc_pool_reload();
```

Coordinator 2

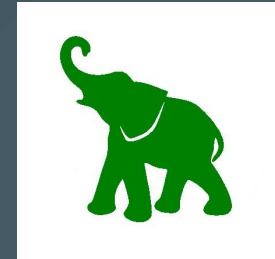


- Em todos os demais nodos:

```
CREATE NODE coord2 WITH (TYPE = 'coordinator',  
                        HOST = '192.168.56.202', PORT = 5432);  
SELECT pgxc_pool_reload();
```

- Matar sessão do coordinator1 que estava bloqueando o cluster para backup

Datanode 1

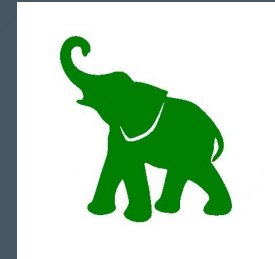


- Alterar `pg_hba.conf`:

```
local all postgres                                trust
host   all postgres 127.0.0.1/32                   trust
host   all postgres ::1/128                       trust
host   all postgres 192.168.56.201/32             trust # Coord 1
host   all postgres 192.168.56.202/32             trust # Coord 2
host   all postgres 192.168.56.204/32             trust # Datanode 2
host   all postgres 192.168.56.205/32             trust # Datanode 3
host   all postgres 192.168.56.206/32             trust # Datanode 4
host   all all       192.168.56.1/32               md5
```

- Recargar: `select pg_reload_conf();`

Datanode 2

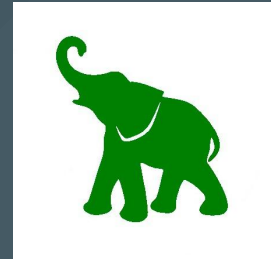


- Alterar `pg_hba.conf`:

```
local all postgres                                trust
host   all postgres 127.0.0.1/32                  trust
host   all postgres ::1/128                      trust
host   all postgres 192.168.56.201/32            trust # Coord 1
host   all postgres 192.168.56.202/32            trust # Coord 2
host   all postgres 192.168.56.203/32            trust # Datanode 1
host   all postgres 192.168.56.205/32            trust # Datanode 3
host   all postgres 192.168.56.206/32            trust # Datanode 4
host   all all       192.168.56.1/32             md5
```

- Recargar: `select pg_reload_conf();`

Datanode 3



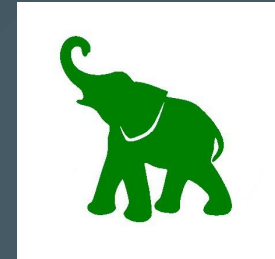
- Inicializar diretório de dados

```
initdb -D /usr/local/pgsql/data --nodename datanode3
```

- Alterar `postgresql.conf`:

```
listen_addresses = '*'  
pooler_port = 40104 # único para cada node  
gtm_host = '192.168.56.200'  
gtm_port = 6666
```

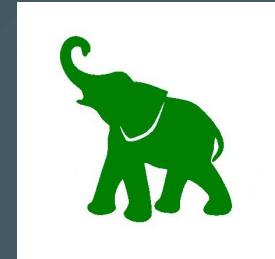
Datanode 3



- Alterar `pg_hba.conf`:

```
local all postgres                                trust
host   all postgres 127.0.0.1/32                    trust
host   all postgres ::1/128                       trust
host   all postgres 192.168.56.201/32             trust # Coord 1
host   all postgres 192.168.56.202/32             trust # Coord 2
host   all postgres 192.168.56.203/32             trust # Datanode 1
host   all postgres 192.168.56.204/32             trust # Datanode 2
host   all postgres 192.168.56.206/32             trust # Datanode 4
host   all all       192.168.56.1/32              md5
```


Datanode 3



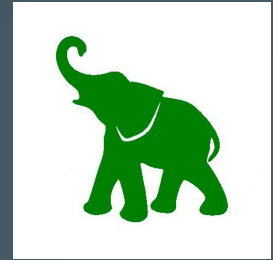
- In coordinator 1, bloquear o cluster inteiro (manter sessão aberta):

```
select pgxc_lock_for_backup();
```

- Criar dump do datanode1:

```
pg_dumpall -h 192.168.56.203 -s --include-nodes \  
--dump-nodes --file=datanode1.sql
```

Datanode 3



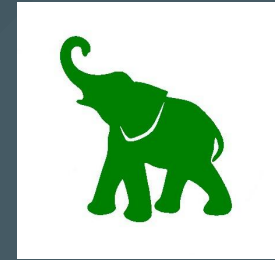
- Iniciar datanode3 em modo de recuperação

```
pg_ctl -D /usr/local/postgresql/data -Z restoremode \  
-l logfile start
```

- Restaurar dump do datanode1 no datanode3:

```
psql -f datanode1.sql
```

Datanode 3



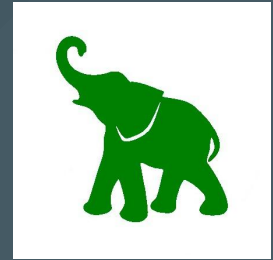
- Reiniciar datanode3

```
pg_ctl -D /usr/local/pgsql/data -Z restoremode \  
-l logfile stop  
pg_ctl -D /usr/local/pgsql/data -Z datanode \  
-l logfile start
```

- Ajustar endereços:

```
ALTER NODE datanode3 WITH (TYPE = 'datanode',  
HOST = 'localhost', PORT = 5432);  
ALTER NODE datanode1 WITH (TYPE = 'datanode',  
HOST = '192.168.56.203', PORT = 5432);  
SELECT pgxc_pool_reload();
```

Datanode 3

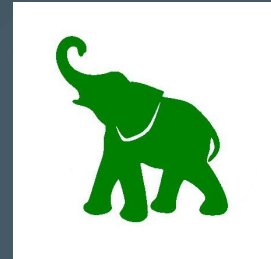


- Em todos os demais nodos:

```
CREATE NODE datanode3 WITH (TYPE = 'datanode',  
                            HOST = '192.168.56.205', PORT = 5432);  
SELECT pgxc_pool_reload();
```

- Matar sessão do coordinator1 que estava bloqueando o cluster para backup

Datanode 4



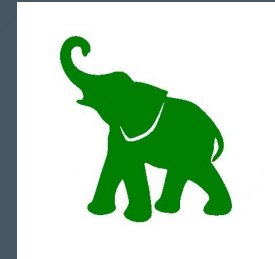
- Inicializar diretório de dados

```
initdb -D /usr/local/pgsql/data --nodename datanode4
```

- Alterar `postgresql.conf`:

```
listen_addresses = '*'  
pooler_port = 40105 # único para cada node  
gtm_host = '192.168.56.200'  
gtm_port = 6666
```

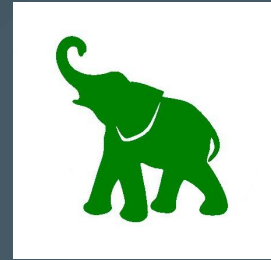
Datanode 4



- Alterar `pg_hba.conf`:

```
local all postgres                                trust
host   all postgres 127.0.0.1/32                    trust
host   all postgres ::1/128                       trust
host   all postgres 192.168.56.201/32             trust # Coord 1
host   all postgres 192.168.56.202/32             trust # Coord 2
host   all postgres 192.168.56.203/32             trust # Datanode 1
host   all postgres 192.168.56.204/32             trust # Datanode 2
host   all postgres 192.168.56.205/32             trust # Datanode 3
host   all all       192.168.56.1/32               md5
```

Datanode 4



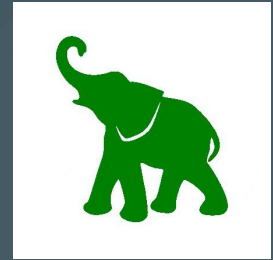
- In coordinator 1, bloquear o cluster inteiro (manter sessão aberta):

```
select pgxc_lock_for_backup();
```

- Criar dump do datanode1:

```
pg_dumpall -h 192.168.56.203 -s --include-nodes \  
--dump-nodes --file=datanode1.sql
```

Datanode 4



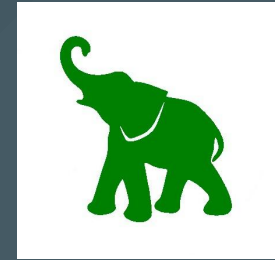
- Iniciar datanode4 em modo de recuperação

```
pg_ctl -D /usr/local/postgresql/data -Z restoremode \  
-l logfile start
```

- Restaurar dump do datanode1 no datanode4:

```
psql -f datanode1.sql
```


Datanode 4



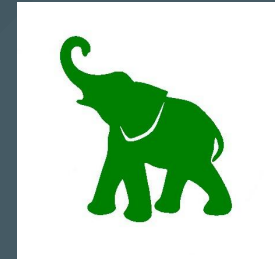
- Reiniciar datanode4

```
pg_ctl -D /usr/local/pgsql/data -Z restoremode \  
-l logfile stop  
pg_ctl -D /usr/local/pgsql/data -Z datanode \  
-l logfile start
```

- Ajustar endereços:

```
ALTER NODE datanode4 WITH (TYPE = 'datanode',  
HOST = 'localhost', PORT = 5432);  
ALTER NODE datanode1 WITH (TYPE = 'datanode',  
HOST = '192.168.56.203', PORT = 5432);  
SELECT pgxc_pool_reload();
```

Datanode 4

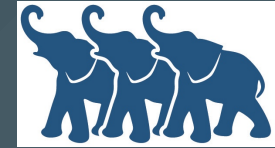


- Em todos os demais nodos:

```
CREATE NODE datanode4 WITH (TYPE = 'datanode',  
                            HOST = '192.168.56.206', PORT = 5432);  
SELECT pgxc_pool_reload();
```

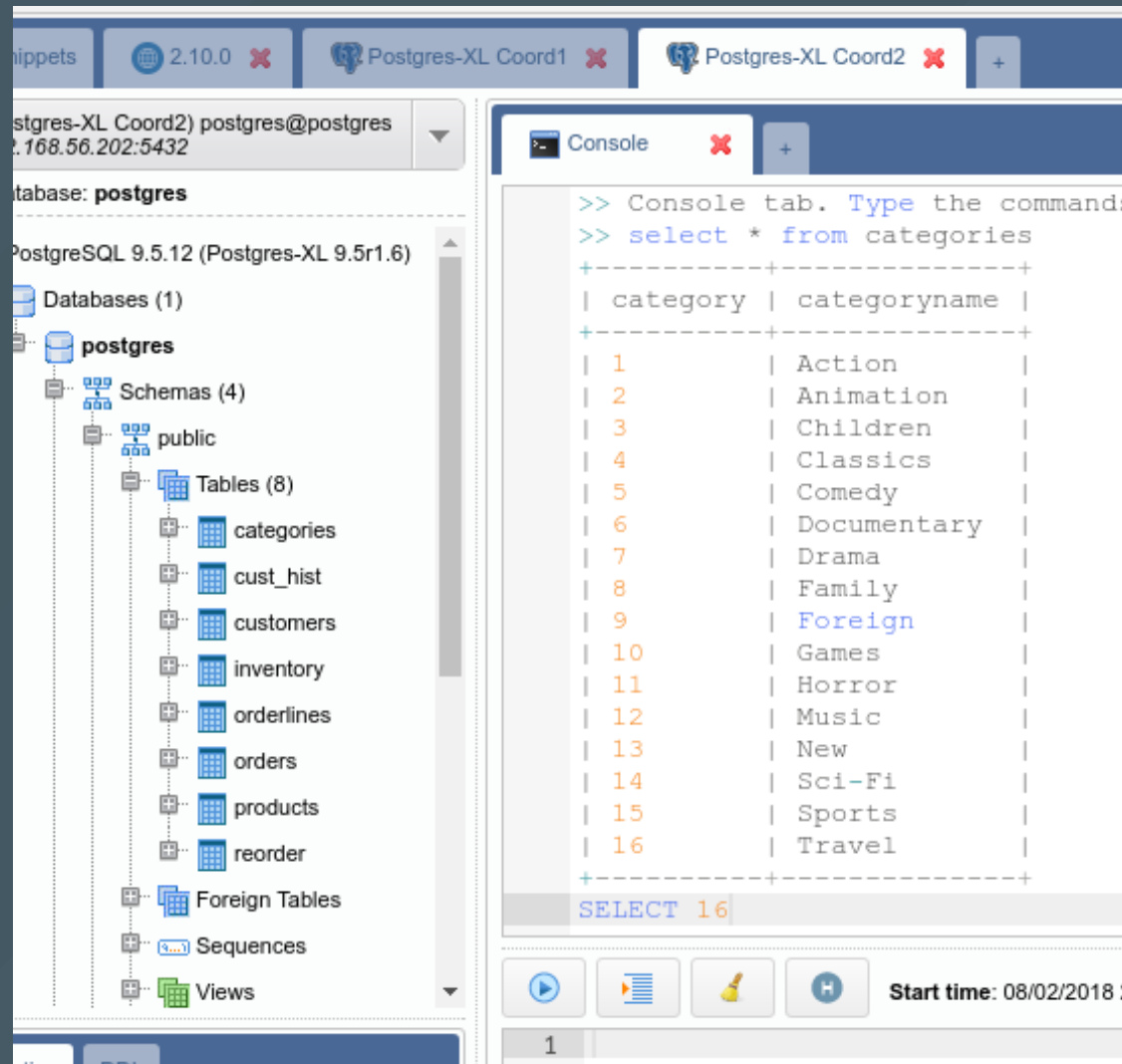
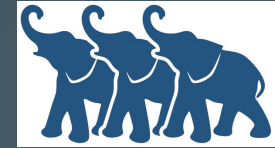
- Matar sessão do coordinator1 que estava bloqueando o cluster para backup

Novo Cluster



The screenshot displays a database management tool interface. At the top, there are tabs for 'Snippets', '2.10.0', and 'Postgres-XL Coord1'. Below the tabs, the connection details are shown: '(Postgres-XL Coord1) postgres@postgres' and '192.168.56.201:5432'. The active database is 'postgres'. The main area shows a tree view of the cluster configuration. Under 'Nodes', there is a 'Replication Slots' section and an 'XL Postgres-XL' section. The 'XL Postgres-XL' section contains a 'Nodes (6)' group, which lists six nodes: 'coord1', 'coord2', 'datanode1', 'datanode2', 'datanode3', and 'datanode4'. A detailed view of a node is shown below the list, with the following properties: Type: datanode, Host: 192.168.56.206, Port: 5432, Primary: false, and Preferred: false. At the bottom, there are tabs for 'Properties' and 'DDL'.

Novo Cluster



The screenshot displays a PostgreSQL management tool interface. The left sidebar shows a tree view of the database structure for a cluster named 'postgres'. The tree includes:

- Databases (1)
 - postgres
 - Schemas (4)
 - public
 - Tables (8)
 - categories
 - cust_hist
 - customers
 - inventory
 - orderlines
 - orders
 - products
 - reorder
 - Foreign Tables
 - Sequences
 - Views

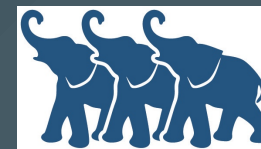
The right pane shows a 'Console' window with the following content:

```
>> Console tab. Type the commands
>> select * from categories
```

category	categoryname
1	Action
2	Animation
3	Children
4	Classics
5	Comedy
6	Documentary
7	Drama
8	Family
9	Foreign
10	Games
11	Horror
12	Music
13	New
14	Sci-Fi
15	Sports
16	Travel

At the bottom of the console, the text 'SELECT 16' is visible. The interface also shows browser tabs for '2.10.0', 'Postgres-XL Coord1', and 'Postgres-XL Coord2'. The start time is indicated as '08/02/2018 2'.

Tabelas no Novo Cluster



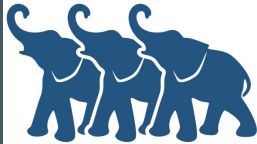
The screenshot displays a PostgreSQL management tool interface. On the left, a tree view shows the database structure for 'postgres'. Under 'Tables (8)', the 'categories' table is expanded, showing its columns, primary key, foreign keys, uniques, checks, excludes, indexes, rules, triggers, and inherited tables. Below this, the 'Postgres-XL' section is visible, indicating the table is distributed by replication and located in all nodes. A context menu is open over the 'Located in node' section, showing options for 'Refresh' and 'Add Node'.

The main console window shows the following SQL commands:

```
1 ALTER TABLE public.categories ADD NODE (datanode3);
2
3 ALTER TABLE public.categories ADD NODE (datanode4);
```

Below the console, there are buttons for 'Data', 'Messages', and 'Explain'.

Tabelas no Novo Cluster



Checks

Excludes

Indexes

Rules

Triggers

Inherited Tables

XL Postgres-XL

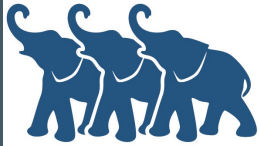
- Distributed by: replication
- Located in all nodes: True
- Located in nodes (4)
 - datanode1
 - datanode2
 - datanode3
 - datanode4

cust_hist

customers

Data Done.

Tabelas no Novo Cluster

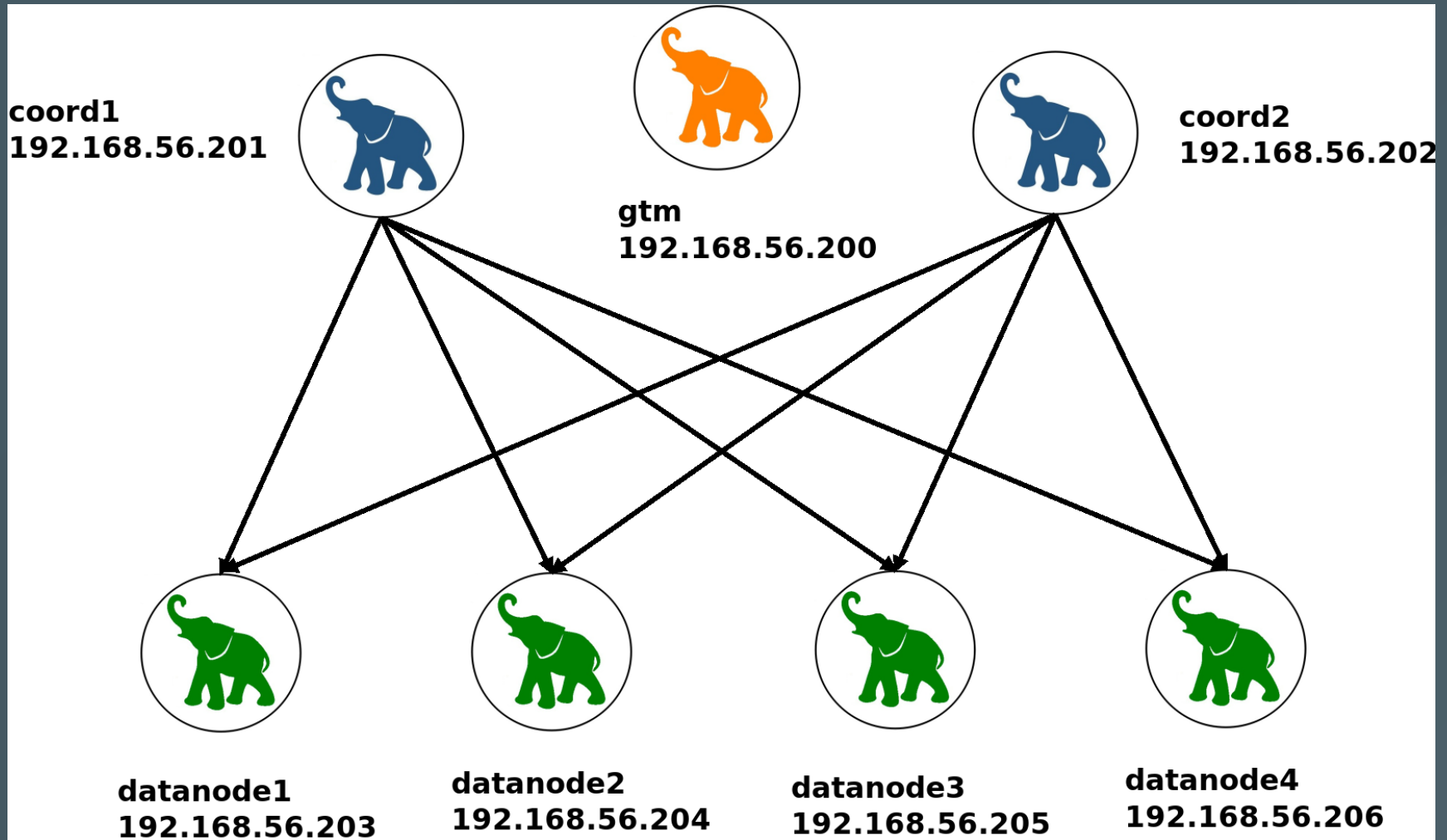
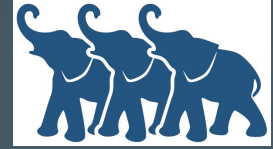


The screenshot displays a database management interface for a PostgreSQL-XL cluster. The left sidebar shows a tree view of the cluster's metadata, including 'Excludes', 'Indexes', 'Rules', 'Triggers', 'Inherited Tables', and 'Postgres-XL'. Under 'Postgres-XL', the following settings are visible:

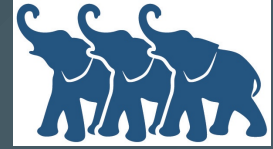
- Distributed by: replication
- Located in all nodes: True
- Located in nodes (4):
 - datanode1
 - datanode2
 - datanode3 (highlighted)
 - datanode4

A context menu is open over 'datanode3', offering 'Refresh' and 'Delete Node' options. Below the tree view, a table view shows the following tables: 'cust_hist', 'customers', 'inventory', and 'orderlines'. The right pane shows 'Data', 'Messages', and 'Explain' tabs, with 'Done.' displayed below.

Novo Cluster

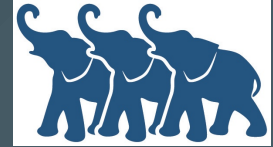


Manutenção



- **High Availability:**
 - Datanodes podem ser replicados via **SR**
 - Replicação Sync / Async
 - **Hot Standby** atualmente **não suportado**
 - **Failover** precisa ser feito manualmente
 - `pgxc_ctl` facilita bastante

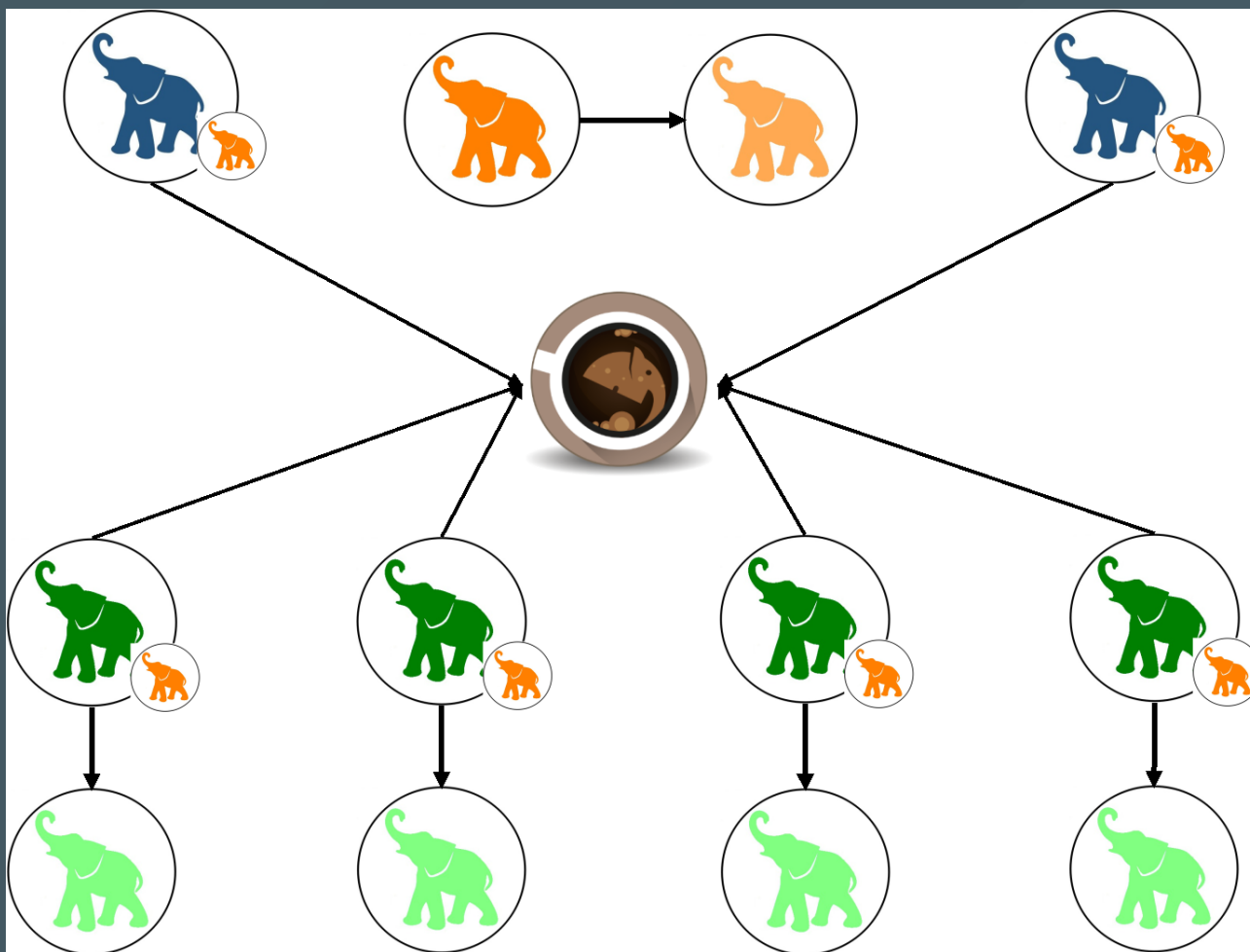
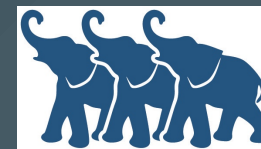
Manutenção



- **Disaster Recovery:**

- `pg_dump` / `pg_restore`
- `pg_basebackup` combinado com WAL archiving/restore
 - PITR, `barman`
- `CREATE BARRIER sunday_night`
 - `recovery_target_barrier` in **recovery.conf**

Novo Cluster + DR + HA



Perguntas?

Muito obrigado!!



william.ivanski@2ndquadrant.com