**Curso DES11**

**Administração de Banco de Dados**

**Leitura Complementar – Capítulo 7**

**Instalando o pgBouncer através do código-fonte:**

$ sudo yum install libevent-devel openssl-devel

$ cd /usr/local/src/

$ sudo wget <https://www.pgbouncer.org/downloads/files/1.15.0/pgbouncer-1.15.0.tar.gz>

$ tar -xvf pgbouncer-1.15.0.tar.gz

$ cd pgbouncer-1.15.0/

$ ./configure --prefix=/usr/local

$ make

$ sudo make install

**Unlogged Tables**

Unlogged Tables são tabelas que não usam o mecanismo de WAL, assim dados escritos para ela não são escritos nas logs de transação. Por esta característica Unlogged Tables fornecem vantagens de performance, porém não fornecem segurança em caso de crash e não são replicadas para servidores standby.

Elas diferem de uma tabela temporária pelo fato de serem visíveis a qualquer usuário e não são apagadas no final de uma sessão.

Mas em caso de um crash ou um shutdown forçado seus dados serão removidos.

Algumas aplicações interessantes para Unlogged Tables poderiam ser:

* Parâmetros de sessão em aplicações Web
* Cache de dados para aplicações
* Controle de sessões de usuários
* Qualquer situação onde se deseja os dados globais e com boa performance de gravação, porém que em caso de perda por crash não seja um problema.

Para usar uma Unlogged Table basta usar o atributo UNLOGGED durante a criação da tabela:

postgres# CREATE UNLOGGED TABLE teste(id int, nome varchar(50));

Mais informações:

Unlogged table performance in postgresql 9.1

<http://paquier.xyz/postgresql-2/unlogged-table-performance-in-postgresql-9-1/>

WAITING FOR 9.1 – UNLOGGED TABLES

<http://www.depesz.com/2011/01/03/waiting-for-9-1-unlogged-tables/>

**Materialized Views**

Materialized Views, ou visões materializadas, são criadas como uma view comum a partir de uma query, porém os dados são armazenados como uma tabela.

Numa view comum, quando ela é acessada a query interna dela é executada na hora sob demanda, os dados da view são produzidos na hora.

Na view materializada os dados da query interna já foram calculados e estão armazenados. Além disto, é possível criar índices e constraints, melhorando ainda mais o acesso.

Os dados da view materializada são gerados no momento que ela é criada, com CREATE MATERIALIZED VIEW e não são atualizados automaticamente.

Para atualizar os dados é necessário executar o comando REFRESH MATERIALIZED VIEW, disponível desde a versão 9.5.

**Criação**

postgres# CREATE MATERIALIZED VIEW v\_contas\_matview AS

SELECT h.\*

FROM pgbench\_history h

JOIN pgbench\_accounts a ON h.aid=a.aid

JOIN pgbench\_branches b ON h.bid=b.bid

**Atualização**

postgres# REFRESH MATERIALIZED VIEW v\_contas\_matview;

**Exemplo:**

1. Criando uma view comum, para comparar com a materializada

postgres# CREATE VIEW v\_contas AS

SELECT h.\*

FROM pgbench\_history h

JOIN pgbench\_accounts a ON h.aid=a.aid

JOIN pgbench\_branches b ON h.bid=b.bid;

2. Executar o EXPLAIN nas duas views para comparar o custo

postgres# EXPLAIN (ANALYZE)

SELECT \* FROM **v\_contas** WHERE aid > 800000;

"Hash Join (cost=14.92..**2442.11** rows=293 width=116)

(actual time=0.298..98.123 rows=478 loops=1)"

postgres# EXPLAIN (ANALYZE)

SELECT \* FROM **v\_contas\_matview** WHERE aid > 800000;

"Seq Scan on v\_contas\_matview (cost=0.00..**43.25** rows=478 width=116)

(actual time=0.030..4.295 rows=478 loops=1)"

3. Ainda é possível criar um índice na view materializada para melhorar o desempenho

postgres# CREATE INDEX idx\_matview ON v\_contas\_matview(aid);

postgres# EXPLAIN (ANALYZE,BUFFERS)

SELECT \* FROM v\_contas\_matview WHERE aid > 800000;

"Bitmap Heap Scan on v\_contas\_matview

(cost=11.98..**32.96** rows=478 width=116)

(actual time=1.504..4.966 rows=478 loops=1)"

As materialized views podem ser uma ótima ferramenta para ajudar no desempenho de seu banco, é necessário analisar os casos que é possível utilizá-la, pois os dados podem estar defasados em relação as tabelas originais até que um REFRESH possa ser feito.

Pode-se agendar os REFRESH periodicamente, ou chamá-lo em uma trigger nas tabelas envolvidas na construção da view.

Mais Informações:

<https://www.postgresql.org/docs/current/sql-creatematerializedview.html>