



**Instituto Infnet – MIT em Big Data**

**Projeto de Bloco – Bloco C – Armazenamento Heterogêneo de Dados**

**Alunos:**

*Maria Fátima Soares de Souza*

*Lucas Pereira de Sousa*

**Professor:**

*Eduardo Morelli*

# O que é?



- ➡ Banco de dados relacional (RDBMS = *Relational Database Management System*)
- ➡ Banco de dados transacional totalmente ACID (Atomicidade, Consistência, Isolamento e Durabilidade)
- ➡ Sua arquitetura permite adicionar facilmente processadores no cluster para atender ao crescimento do volume de dados e de transações
- ➡ utiliza o armazenamento em memória
- ➡ Fundado por Michael Stonebraker, porém também é desenhado por Sam Madden e Daniel
- ➡ Possui uma versão Open source, mas também possui uma versão paga

# Quem deve usar?

É destinado a um segmento específico de computação empresarial, voltado especificamente para ***dados rápidos***, ou seja, aplicativos que devem processar grandes fluxos de dados rapidamente.

- Aplicações financeiras
- Aplicações de mídia social
- Crescente campo da Internet das Coisas

Os principais requisitos para estas aplicações são:

- Escalabilidade
- Confiabilidade
- Alta disponibilidade
- Rendimento excepcional

# Como funciona?

Cada banco de dados VoltDB é otimizado para uma aplicação específica dividindo as tabelas e as *stored procedures* que acessam essas tabelas através de múltiplos "sites" ou partições em uma ou mais máquinas host para criar o banco de dados distribuído, e com isso se pode realizar consultas em paralelo.

Cada transação pode ser executada e concluída sem a interferência de bloqueio individuais de registros que consome a maior parte do tempo de processamento nas bases de dados tradicionais.

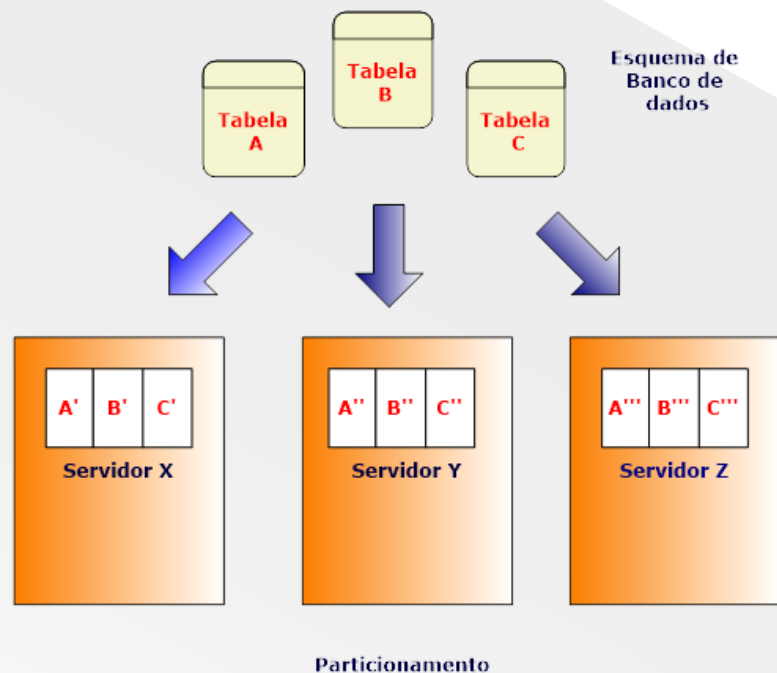
# Particionamento de Tabelas

O particionamento organiza o conteúdo de uma tabela de banco de dados em unidades autónomas separadas, semelhante a sharding. O particionamento no VoltDB é único porque:

➤ O VoltDB particiona automaticamente as tabelas de banco de dados com base em uma coluna de particionamento especificada. Não se tem que gerir manualmente as partições.

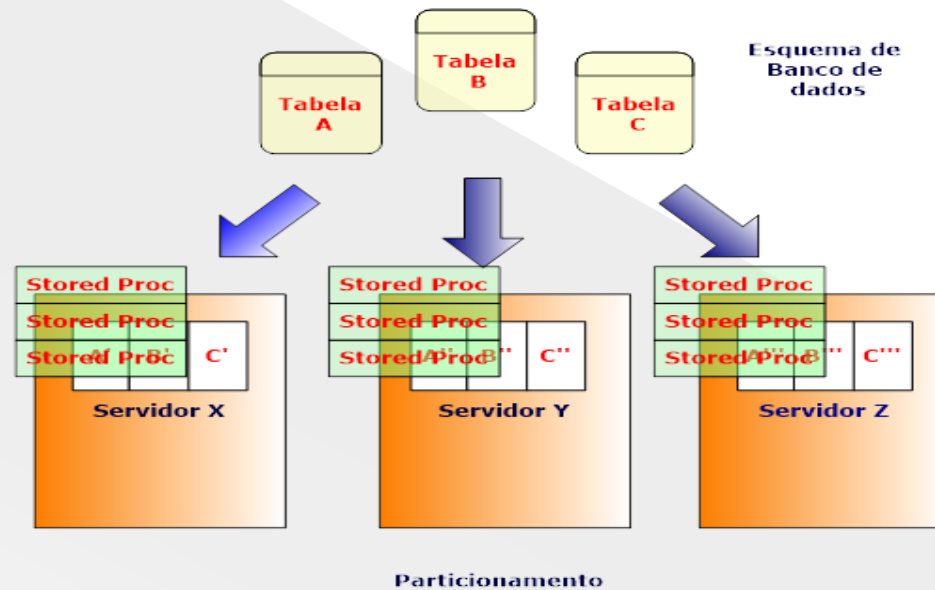
➤ Pode-se ter várias partições ou sites, em um único servidor. O particionamento não é apenas para dimensionamento do volume de dados, mas para o desempenho também.

➤ O VoltDb particiona os dados e a stored procedure e assim aproveita o paralelismo e obtém ganhos de performance.



# Processo Serializado (single-threaded)

Usando o processamento serializado consegue garantir a consistência transacional sem a sobrecarga de bloqueio, travamento e logs de transações.

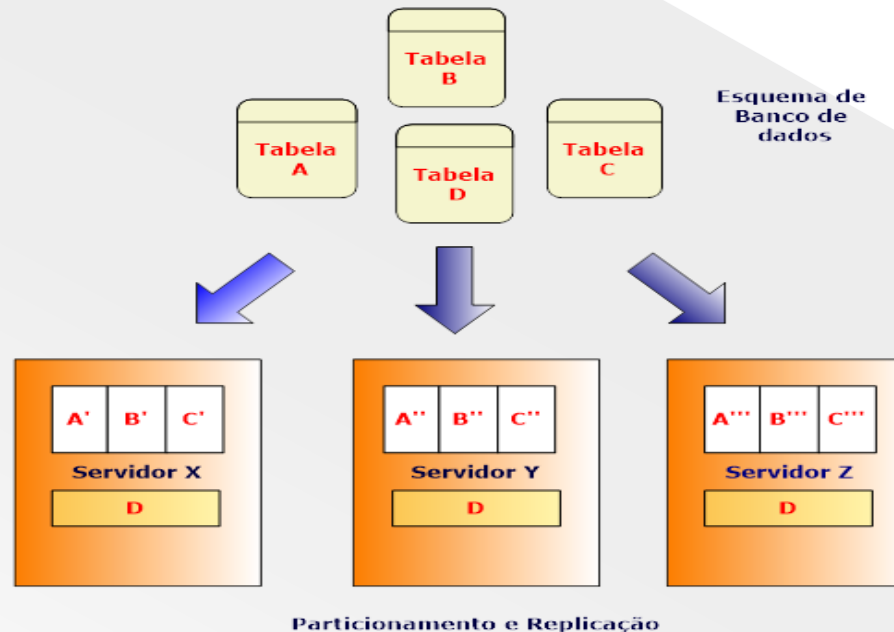


A arquitectura VoltDB é otimizada para minimizar a latência

# Replicação de Tabelas

Para otimizar ainda mais o desempenho, podemos replicar certas tabelas do banco de dados para todas as partições do cluster.

O benefício de ter os dados de uma tabela replicados em todas as partições é que pode ser lido a partir de qualquer partição individual, porém se houver atualizações ou inserções a uma tabela replicada deve ser executada em todas as partições e é por isso que não se deve replicar tabelas que são atualizadas com frequência.



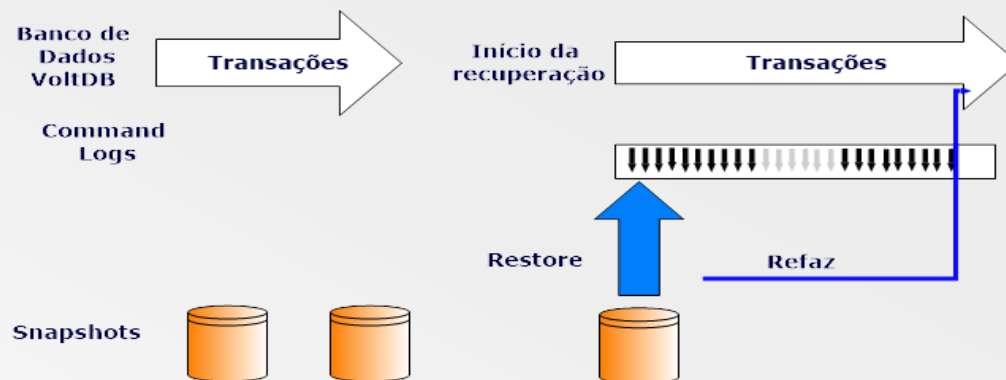
# Durabilidade

Controle da persistência do dado garantindo que após o “commit” é necessário que o dado esteja 100% íntegro e disponível mesmo em caso de falha. No que se refere a durabilidade, o VoltDB possui as seguintes funções:

**K-Safety:** Replicação de dados. O K-1 significa que cada partição é replicada em duas máquinas. K-2, significa que é replicada em 3 máquinas e assim por diante.


**Snapshot:** Cópia das informações num determinado momento. Backup, que pode ser executado de forma discreta ou contínua (por exemplo, a cada 250 ms)

**Command Logging:** De tempos em tempos as informações que estão na memória são armazenadas no disco. Se houver uma falha e o banco tiver que ser restaurado, a base é restaurada e são aplicados os logs do ponto do último backup até o ponto da falha.





# Informações Técnicas

<b>Website</b>	Voltdb.com
<b>Desenvolvedor</b>	Voltdb Inc.
<b>Logotipo</b>	
<b>Versão inicial</b>	2010
<b>Versão atual</b>	6.0 de janeiro de 2016
<b>Licença</b>	Open source
<b>Linguagem de programação suportadas</b>	C#, C++, Java, PHP, Python
<b>Método de Particionamento</b>	Sharding
<b>Método de Replicação</b>	Master-slave replicação
<b>Sistema Operacional</b>	<p>Requer Linux 64 bits .</p> <p>Kits são construídos e qualificados para as seguintes plataformas:</p> <ul style="list-style-type: none"> <li>-CentOS versão 6.6 ou posterior maior, incluindo a 7.0;</li> <li>- Red Hat (RHEL) versão 6.6 ou posterior, incluindo a 7.0;</li> <li>- Ubuntu versões 12.04 e 14.04</li> <li>- Também disponível para Macintosh OS X10.9 ou posterior.</li> </ul>
<b>CPU</b>	<p>Dual core (b) x86_64 processadores</p> <p>64 bits</p> <p>1.6 GHz</p>
<b>Memomry</b>	4 Gbytes (c)
<b>Java (d)</b>	<p>Servidor VoltDb: Java 8</p> <p>Java and JDBC Client: Java 7 ou 8</p>
<b>Required software</b>	NTP (e)
<b>Software recomendado</b>	<p>Python 2.6 ou versão posterior de 2.x</p> <p>Eclipse 3.x (ou outro Java IDE)</p>

<b>(a)</b>	CentOS 6.6, CentOS 7.0, RHEL 6.6, RHEL 7.0 e Ubuntu 12.04 e 14.04 são os únicos oficialmente sistemas operacionais suportados para VoltDB. No entanto, VoltDB é testado em vários outros compatível com POSIX e Linux baseado em sistemas operacionais de 64 bits, incluindo Macintosh OS X 10.9.
<b>(b)</b>	Processadores dual core são um requisito mínimo. Quatro ou oito núcleos físicos são recomendados para desempenho ideal
<b>(c)</b>	Os requisitos de memória são muito específicos para as necessidades de armazenamento do aplicativo e o número de nós no cluster. No entanto, 4 gigabytes deve ser considerada uma configuração mínima
<b>(d)</b>	O VoltDb suporta JDKs de OpenJDK ou Oracle/Sun.
<b>(e)</b>	NTP minimiza diferenças de tempo entre nós em um cluster de banco de dados, que é crítico para VoltDB. Todos os nós do cluster deve ser configurado para sincronizar contra o mesmo servidor NTP. Usando um único servidor NTP local é recomendado, mas não é obrigatório

# Recursos

- Segurança (usuário e senha)
- Backup
- Restore
- Importação de dados
- Exportação de dados para outras bases

# Exportação

O VoltDb foi feito para trabalhar com outros bancos, para tanto, possui funções exportação de dados.



**LEIA COM ATENÇÃO** Exportação não é backup!

- ➡ São só exportados os dados selecionados
- ➡ A exportação é um processo contínuo, em vez de um evento one-time.
- ➡ O resultado da exportação de dados é a informação que é utilizada por outros processos de negócio, e não como uma cópia de backup para restaurar o banco de dados.

# O VoltDB utiliza a sintaxe SQL

Exemplos:

➡ INSERT

➡ SELECT

➡ UPDATE

➡ DELETE

# Alguns dos principais clientes...



# Conclusão

O VoltDB é um banco de dados relacional que vêm atingir um ramo onde os RDBMS estavam perdendo mercado para os NoSQL, que é o mercado de escalabilidade. Ele une os mundos do relacional com os de volume de dados, com as da necessidade de obtenção de dados de forma rápida e o da escalabilidade.

Através do particionamento de dados unificados com as instruções em *stored procedures* na mesma partição permite uma velocidade maior na obtenção das informações sem perder a consistência. O fato de utilizar as linguagens DDL e SQL permite uma mais rápida absorção pelos administradores de banco de dados e pelos desenvolvedores, até mesmo pelos usuários que já estão acostumados com a utilização de queries em SQL.

# Conclusão



## Melhores práticas:

- Particionar as tabelas. Particionamento permite redimensionar dinamicamente o tamanho do seu banco de dados. Ele também permite o processamento simultâneo de transações.
- Particionar as *stored procedures*. Para maximizar a produção, maximizar a frequência das operações partição única e minimizar multipartição das transações.
- Conectar-se a todos os nós do cluster. A criação de múltiplas conexões de cliente evita possíveis gargalos e permite VoltDB para aproveitar a afinidade do cliente (no cliente Java).
- Usar um número ímpar de nós. VoltDB protege contra a perda de dados devido a falha de rede ou do nó da replicação (chamado KSafety). Usando um número ímpar de nós no cluster maximiza a proteção e disponibilidade. Por exemplo, com dois nós de  $K = 1$  cluster, caso se verifique uma falha de nó, há uma possibilidade de 50/50 do outro nó parar a fim de evitar uma parada na rede. Com três nós  $K = 1$  cluster isso não acontecerá, a menos que dois nós falhem.

# Conclusão



## O que não se deve fazer:

- Não retornar quantidades excessivas de dados de *Store Procedures*. Conjuntos de resultados muito grandes podem aumentar a latência da rede e impactar a performance. VoltDB limita os resultados das procedures para 50MB. Manter os resultados em conjuntos pequenos para maximizar o rendimento e minimizar a latência.
- Não criar tabelas enormes. VoltDB funciona melhor com pequenas a médias tabelas. Criação de tabelas com centenas de colunas ou muitas colunas grandes podem impactar o desempenho da consulta. É melhor criar várias tabelas menores. VoltDB limita colunas para 1MB cada uma e todas as colunas em uma única tabela de 2MB.
- Não fazer processamento desnecessário em *stored procedures* ou rotinas de retorno de chamada. Manter as procedures otimizadas e bem definidas para evitar procedimentos de longa duração e bloquear outras transações. Além disso, rotinas de retorno de chamada são processadas uma de cada vez dentro do cliente, portanto manter os procedimentos de retorno de chamada eficientes para evitar parar o cliente após a conclusão da transação.
- Não executar operações não determinísticas em *stored procedures*. VoltDB exige resultados determinísticos para garantir a consistência entre as partições e durante a recuperação. Evitar quaisquer operações em *stored procedures* que dêem resultados variáveis, como a hora do sistema, números aleatórios, arquivo de I / O, etc. Em vez disso, gerar os dados variáveis no cliente e usá-lo como entrada para o procedimento.



# Referências Bibliográficas

<http://db-engines.com/en/ranking>

<https://451research.com/state-of-the-database-landscape>

<http://www.voltdb.com>

<https://docs.voltdb.com/UsingVoltDB>

Edward Ribeiro - <https://www.infoq.com/br/presentations/voltb-arquitetura-desenv>

**OBRIGADO**



**PELA ATENÇÃO**

GERADORMEMES.COM