Fundação Educacional do Município de Assis
IMESA - Instituto Municipal de Ensino Superior de Assis

UM ESTUDO EXPLORATÓRIO ACERCA DE BANCO DE DADOS IN-MEMORY COMPARADO AOS BANCOS DE DADOS CONVENCIONAIS

Jonathan dos Santos MARTINS[1]; Alex Sandro Romeo de Souza POLETTO[2]
jonathan.santosmartins@gmail.com[1]; apoletto@femanet.com.br[2]

RESUMO:O presente trabalho tem por objetivo traçar um comparativo entre Sistemas de Bancos de Dados In-Memory e os Sistemas de Bancos de Dados Convencionais. Através desse comparativo, pretende-se demonstrar soluções desenvolvidas em Bancos de Dados In-Memory para melhor atender a desafios gerados pelo corrente uso de um volume cada vez maior de dados. Dentre esses desafios estão, maior rapidez no acesso aos dados; diminuição do tempo de processamento de aplicações complexas, aumentando assim, seu desempenho; acesso a grandes bases de dados para BI (Business Intelligence); e criação de conhecimentos mais precisos e completos para trabalho com informação em tempo real, necessidades que os Modelos Convencionais, até certo ponto, deixam a desejar.

PALAVRAS-CHAVE: In-Memory; Oracle; Comparação

ABSTRACT: The purpose of this paper is to draw a comparison between In-Memory Database Systems and Conventional Databases Systems. Through this comparison, we intend to demonstrate solutions developed in In-Memory Databases to better meet the challenges generated by the current use of an increasing volume of data. Among these challenges are: faster access to data; Reducing the processing time of complex applications, thus increasing its performance; Access to large databases for BI (Business Intelligence); And creation of more accurate and complete knowledge to work with real-time information, necessities that the Conventional Models, to a certain extent, leave to be desired.

KEY-WORDS: In-Memory; Oracle; Comparison

1. Introdução

Ao longo dos últimos 50 anos, os avanços em Tecnologia da Informação (TI) têm tido um impacto significativo no sucesso das empresas em todos os setores da indústria. As fundamentações para este sucesso são as interdependências entre negócios e TI, no sentido de que elas não somente tratem e aliviem o processamento de tarefas repetitivas, mas que sejam as facilitadoras para a criação de conhecimentos mais precisos e completos em uma empresa. Este aspecto tem sido muitas vezes descrito e associado ao termo "tempo real" que indica que cada mudança que ocorre dentro de uma empresa seja quase imediatamente visível por meio da TI [Plattner e Zeier 2011].

Atualmente, a maioria dos dados dentro de uma empresa ainda é distribuída através de uma vasta gama de aplicações e armazenada em vários servidores. Criar uma visão unificada sobre esses dados é um procedimento complicado e demorado. Além disso, relatórios analíticos normalmente não são obtidos diretamente de dados operacionais, mas a partir de dados agregados de um Data Warehouse (DW) (Depósito de Dados). Os dados operacionais são transferidos parao DW, permitindo a geração de relatórios *ad-hoc*, porém provocando atrasos.

Como conseqüência, os líderes de empresas têm de tomar decisões com base em informações insuficientes, ao contrário do que sugere o termo "tempo real". Prevê-se que isto esteja prestes a mudar devido as arquiteturas de hardware, que demonstraram grande evolução na última década. Múltiplos processadores e a disponibilidade de grandes quantidades de memória principal de baixo custo deverão proporcionar novos avanços na indústria de software.

Essa evolução deverá tornar possível armazenar conjuntos de dados de empresas inteiras totalmente na "memória principal", oferecendo um melhor desempenho do que o "sistema de discos tradicional". Os discos rígidos se tornarão obsoletos e em breve, só serão utilizados para fazer *backup* de dados. Com a computação dos dados na memória, o processamento transacional e analítico será unificado e o alto desempenho da memória irá mudar a forma como as empresas funcionam e, finalmente, cumprir a promessa de computação em tempo real [Plattner e Zeier 2011].

Os Sistemas Gerenciadores de Banco de Dados Relacionais, meio mais utilizado para o armazenamento físico de dados, vêm cada vez mais oferecendo mecanismos que possibilitam acesso mais rápido aos dados. Para isso, são constantemente realizados ajustes e aprimoramentos nos mecanismos de recuperação de informação, no sentido de otimizar cada vez mais a busca aos dados, e a tecnologia de *Tuning* é uma das mais utilizadas atualmente por esses sistemas, na otimização de consultas.

Voltando um pouco na história, os sistemas de processamento de arquivos representaram o primeiro sistema comercial de informações utilizado para armazenar grupos de fichas em arquivos separados. Embora esses sistemas tenham sido um grande melhoramento do sistema manual de arquivamento de registros, eles possuíam e ainda possuem algumas limitações, tais como: os dados são separados e isolados; os dados são frequentemente duplicados; os programas aplicativos são dependentes dos formatos dos arquivos; os arquivos são, com frequência, incompatíveis entre si; é difícil representar os dados na perspectiva dos usuários, etc. [Kroenke 1999].

Por muito tempo, esses sistemas foram o meio mais utilizado para o armazenamento de dados, mas não havia garantia da consistência e da integridade dos mesmos, durante quedas de energia, falhas de rede e outros problemas técnicos. Clipper, Cobol e Dataflex eram as linguagens mais utilizadas no desenvolvimento desses sistemas em que tudo era definido, avalizado e dependente do código fonte.

Posteriormente, ocorreu o advento dos Sistemas Gerenciadores de Banco de Dados (SGBD), tecnologia desenvolvida para superar as limitações dos sistemas de processamento de arquivos, tornando a tarefa de programação da aplicação mais simples, já que os programadores de aplicações não precisam se preocupar com a maneira como os dados são fisicamente armazenados, visto que os programas de processamento de arquivos acessam diretamente os arquivos de dados. Já os programas de processamento de bancos de dados acionam o SGBD para acessar os dados armazenados, garantindo assim, a consistência e a integridade dos mesmos [Kroenke 1999]. O Modelo Relacional se tornou o mais utilizado, tendo como ambiente de desenvolvimento de software as ferramentas Visual Basic, Delphi, Centura, dentre outras.

Depois, vieram os SGBD Orientados a Objetos (SGBDOO) que não ganharam muito mercado. Logo surgiram, para desenvolvimento na Web, os SGBD Objeto-Relacionais (SGBDOR). Porém, os SGBD Relacionais continuaram e ainda continuam sendo o meio mais utilizado, mesmo com o advento das linguagens orientadas a objetos como Java e C#, no entanto, é necessário um mapeamento objeto-relacional para o processamento dos dados. No caso do Java esse mapeamento é realizado pela ferramenta Hibernate e no caso do C# pela ferramenta Microsoft Entity Framework (ORM – ObjectRelational Management).

2. SGBD sconvencionais

Um Sistema de Bancos de Dados (SBD) é composto por um conjunto de dados relacionados entre si e programas desenvolvidos para acessá-los. Devido ao teor desses dados os SGBDs devem

garantir meios de armazenar e recuperar as informações de forma eficiente. A gestão desses dados depende não apenas das estruturas definidas para acomodá-los, mas também dos mecanismos criados para sua manipulação. Tais sistemas de gerenciamento devem garantir a segurança das informações contra acessos não autorizados e falhas no sistema.

SGBDs em geral tem como finalidade solucionar problemas que ocorrem em outros métodos de gestão de informação, como por exemplo, um sistema de processamento de arquivos [Silberschatz 2006]. Tais como: Redundância e inconsistência dos dados; Dificuldade no acesso aos dados; Isolamento de dados; Problemas de integridade; Problemas de atomicidade; Anomalias de acesso concorrente e Problemas de segurança.

Para lidar com estes problemas um Sistema de Banco de Dados oferece a seus usuários ferramentas que os possibilita ter uma visão abstrata dos dados para assim facilitar sua manipulação e modelagem.

Ao ocultar certos detalhes do sistema, pontos mais complexos de sua implementação e desenvolvimento passam despercebidos para o usuário final simplificando sua interação com o mesmo. Não é necessário ao usuário de um caixa eletrônico saber qual modelo de dados foi aplicado ou mesmo como os dados de sua conta estão armazenados em nível físico, no entanto, é pertinente aos outros tipos de usuários possuírem uma visão mais ampla do banco de dados e suas especificações.

Ainda pertinente a abstração dos dados encontra-se os Modelos de Dados, coleções de ferramentas conceituais para descrever os dados, suas relações e restrições de consistência oferecendo uma maneira de descrever o projeto de um banco de dados em todos os níveis. O modelo mais utilizado é o Modelo Relacional.

Devido ao escopo do presente trabalho, será focado apenas no que tange ao armazenamento e recuperação de informações pelos SGBDs convencionais. Para lidar com estas tarefas os componentes funcionais do SGBD podem ser divididos em gerenciador de armazenamento e processador de consulta.

Com o massivo volume de dados corporativos produzidos o espaço necessário para armazená-los, normalmente é muito grande e como a memória principal dos computadores não é suficiente para acomodá-los, estes são armazenados em discos. Portanto, os dados devem ser movidos do disco para a memória principal conforme o necessário, e devido ao fato de esta tarefa ser custosa em tempo é fundamental que a forma em que os dados estejam estruturados pelo sistema de banco de dados

minimize a movimentação entre disco e memória principal. Por isso é necessário a existência de um modulo de programa que forneça uma interface entre os dados de baixo nível contidos no banco e os programas aplicativos e consultas feitas ao sistema. Tal módulo é o gerenciador de armazenamento.

O gerenciador de armazenamento promove a interação entre o gerenciador de arquivos que armazena os dados brutos em disco. O sistema de arquivos é geralmente fornecido pelo sistema operacional. O gerenciador de armazenamento é responsável por armazenar, recuperar e atualizar as informações no banco de dados, para tal ele traduz comandos DML (Data ManipulationLanguage) em comandos do sistema de baixo nível.

O gerenciador de armazenamento é composto por: Gerenciador de autorização e integridade; Gerenciador de transação; Gerenciador de arquivos e Gerenciador de buffer.

O gerenciador de autorização e integridade faz testes para verificar se as restrições de integridade foram atendidas e se os usuários possuem autoridade para acessar determinados dados.

O gerenciador de transações garante a consistência do banco de dados apesar da ocorrência de falhas no sistema e também que a execução de transação concorrente ocorra com êxito.

O controle da alocação de espaço para armazenamento no disco e as estruturas utilizadas para representar as informações nele contido é realizado pelo controle de arquivos.

Por fim, o gerenciador de buffer busca os dados armazenados no disco, coloca em memória principal e, além disso, decide quais dados colocar em cache da memória principal. Este gerenciador permite a manipulação de dados muito maiores do que o tamanho da memória principal, por isso é de extrema importância.

Além destes componentes o gerenciador de armazenamento implementa estruturas de dados que visam otimizar seu trabalho: Arquivos de dados armazenam o banco de dados em si; O Dicionário de dados armazena metadados sobre a estrutura de dados do banco, principalmente no que diz respeito ao esquema do bando de dados; e por fim, os Índices nos possibilitam acesso rápido aos dados funcionando como índices de livros e revistas, estes nos fornecem ponteiros para itens que contenham valores específicos.

Já o Processador de Consultas possui como componentes um interpretador DDL (Data DefinitionLanguage) responsável por interpretar instruções DDL e registrar as definições no dicionário de dados; Um compilador DML para traduzir instruções em linguagem de consulta para

uma linguagem de baixo nível para que o mecanismo de avaliação possa entendê-lo. Por fim, mas não menos importante o Mecanismo de Avaliação de Consulta se encarrega de executar as instruções de baixo nível gerado pelo compilador DML.

2.1Armazenamento e estrutura de arquivo

O SGBD pode então recuperar, processar e atualizar os dados que estão armazenados conforme for necessário. O local em que estes estão fisicamente dispostos forma uma hierarquia de armazenamento que, segundo Silberschatz, pode ser dividida em duas categorias principais: Armazenamento primário e armazenamento secundário ou terciário.

Armazenamento primário inclui as mídias que operam diretamente sobre o controle da CPU como a memória principal e as memórias cache. São mais rápidas, porém limitadas e mais caras em comparação com as do tipo secundário e terciário.

As mídias de armazenamento secundário incluem discos rígidos. Já as mídias removíveis como os discos ópticos são consideradas de armazenamento terciário. Tais mídias são mais baratas e possuem maior capacidade em relação às de armazenamento primário, no entanto, são mais lentas, pois não podem ser diretamente processadas pelo CPU.

Nos sistemas de computador atuais os dados são transportados através desta hierarquia de meios de armazenamento, a memória mais rápida e cara está presente em menor quantidade enquanto a mais lenta e barata ocupa a maior parte do sistema. No extremo mais caro da hierarquia está a memória cache, um tipo de RAM estática usada pela CPU para agilizar a execução de instruções de programas usando técnicas como pipeling e pré-busca. Em seguida vem o DRAM (Dynamic RAM) que é comumente chamado de memória principal. É ele que mantém as instruções de programa e dados para a CPU. No nível de armazenamento secundário e terciário da hierarquia estão os discos magnéticos e no extremo mais barato da corrente as fitas magnéticas.

Programas em geral são executados e residem no DRAM, por isso, em SGBDs convencionais o armazenamento dos bancos de dados permanentes é feito em armazenamento secundário e partes do banco são lidas e escritas de buffers de memória principal de acordo com a necessidade das aplicações. Além disso, as mídias de armazenamento secundário estão imunes a circunstâncias que podem causar perdas permanentes de dados em meios de armazenamento primário, por este motivo os meios de armazenamento secundário são também conhecidos como armazenamento não volátil.

Os discos magnéticos são o meio de armazenamento mais utilizado para guardar as informações de grandes bancos de dados e são conhecidos como dispositivos *on-line*, ou seja, dispositivos que podem ser acessados diretamente a qualquer momento. Entretanto, não é incomum o uso de fitas magnéticas para fazer o *backup* dos dados já que estas são um meio barato, porém custoso em termo de tempo de processamento. Um empecilho ao uso das fitas é que as mesmas necessitam da intervenção de um operador ou um dispositivo de carga automática para tornar as informações disponíveis. As fitas são dispositivos *off-line*.

Devido à ampla utilização de discos foram desenvolvidas várias técnicas para armazenar grande quantidade de dados em meio físico.

Como as aplicações normalmente precisam apenas de uma parte dos dados o SGBD busca estes dados no disco, os processa na memória principal um de cada vez, e os reescreve no disco apenas em caso de alteração na forma de arquivos de registros. Os registros são coleções de valores que podem ser interpretados como fatos sobre entidades, seus atributos e relacionamentos. A forma como os registros são armazenados no disco é algo muito importante, pois confere eficiência na localização dos mesmos garantindo ganhos em questão de desempenho.

Há varias formas de se organizar arquivos e cada uma determina como os registros serão armazenados fisicamente em disco e, por conseguinte como serão acessados. Um arquivo desordenado ou de heap coloca os arquivos sem qualquer ordem acrescentando novos registros ao final do arquivo. Um arquivo seqüencial ou classificado coloca os valores ordenados por um campo chamado de campo de classificação. Além destas estruturas existem as estruturas de organização como as estruturas em árvore.

Como dito acima os discos magnéticos são utilizados para armazenar grandes quantidades de dados. A unidade de dados mais básica do disco é o bit e ao se magnetizar as áreas do disco é possível fazer com que elas representem valores de 0 e 1. Para codificar a informação os bits são agrupados em bytes ou caracteres e seu tamanho pode variar de 4 a 8 bits dependendo do computador ou dispositivo. A capacidade de um disco é medida pela quantidade de bytes que este pode armazenar.

Independente da capacidade de armazenamento todos os discos são feitos de um material magnético modelado como um disco fino protegido por uma camada de acrílico ou plástico. Discos podem ser de face simples, quando armazenam informação apenas em uma de suas superfícies, ou de face dupla quando usam suas duas superfícies para aumentar sua capacidade de armazenamento. Para

ter um aumento ainda mais significativo os discos são montados em *disk packs* que podem incluir vários discos. As informações são gravadas na superfície do disco em círculos concêntricos de diâmetros diferentes chamados de trilhas. Em *disk packs* as trilhas de mesmo diâmetro em suas superfícies são chamados de cilindros devido à forma que teriam se estivessem conectadas no espaço, tal conceito é importante, pois os dados armazenados em cilindro podem ser recuperados muito mais rápido do que se estivessem distribuídos em vários cilindros.

O numero de trilhas contidas em um disco pode variar entre centenas a milhares e devido à quantidade de informação que carregam são dividas em setores e blocos. A divisão da superfície em setores após ser fixada não pode ser alterada. É possível organizar as trilhas de varias formas como, por exemplo, fazer com que os setores se estendam por ângulos menores no centro à medida que se move para fora, mantendo assim uniformidade na densidade de gravação. Ou então chamar uma parte especifica da trilha que se estende por um ângulo fixo no centro de um setor.

A divisão de uma trilha em blocos de disco ou páginas é feita pelo sistema operacional durante a formatação ou inicialização do disco e seu tamanho não pode ser alterado de forma dinâmica podendo variar entre 512 a 8192 bytes. Cada bloco é separado por lacunas de tamanho fixo que contem informações codificadas gravadas durante a inicialização do disco. Tal informação serve para determinar qual bloco da trilha segue cada lacuna de bloco.

Discos são dispositivos endereçáveis por acesso aleatório. A transferência de dados entre o disco e a memória principal se da entre unidades de bloco de disco. O endereço de hardware de um bloco de disco, ou seja, uma combinação do numero do cilindro, número da trilha e número do bloco é dado ao hardware de E/S. Em seguida, o endereço de um *buffer*, uma área reservada próxima ao armazenamento principal, também é fornecido. Desta forma, para que um comando de leitura ocorra, o bloco de disco é copiado para o *buffer*, já um comando de gravação requer que *buffer* seja copiado para bloco de disco.

No entanto, o mecanismo que faz a gravação/leitura real no disco é chamado de cabeça de leitura/gravação e faz parte do sistema chamado de unidade de disco. Discos e *disk packs* são montados na unidade de disco que possui um motor para realizar a rotação dos mesmos, cabeças de leitura/gravação conectadas a braços mecânicos para as superfícies dos discos que são movidos por acionador que os posiciona sobre as trilhas especificadas em um endereço de bloco.

Os discos são girados a velocidades que podem variar de 5.400 a 15.000 RPM. Quando a cabeça de leitura/gravação se posiciona na trilha correta e o bloco especificado no endereço de bloco

se move sob ela, o componente eletrônico da cabeça de leitura/gravação é ativado para fazer a transferência dos dados. Existem unidades de disco que possuem cabeça de leitura/gravação fixas que corresponde em numero as trilhas do disco e são chamadas de discos de cabeça fixa enquanto as unidades que usam acionador são conhecidas como discos de cabeça móvel. Os discos de cabeça fixa atuam de maneira mais ágil que dos que os móveis, mas em contrapartida o custo das cabeças de leitura/gravação adicionais é muito alto.

A unidade de disco normalmente também possui um controlador de disco que a controla e interliga ao sistema de computação. O controlador aceita instruções de E/S de alto nível e as utiliza para posicionar mecanicamente o braço de leitura/gravação sobre a trilha correta para executar a transferência de blocos de disco. O tempo exigido para a execução deste processo leva algo na ordem de 5 a 10 milissegundos em desktops e de 3 a 8 milissegundos em servidores, tal período de tempo é chamado de tempo de busca. Além do tempo de busca há o tempo de latência ou atraso rotacional que ocorre enquanto o inicio do bloco desejado gira até a posição sob a cabeça de leitura/gravação. Este período de atraso é determinado pelo das rotações por minuto (RPM) do disco. Tal cálculo é feito da seguinte maneira: a 15.000 RPM, o tempo médio de rotação é de 4 milissegundos e o atraso rotacional é o tempo por meia rotação, ou seja, 2 milissegundos. Além de todo o tempo gasto para localizar as informações no disco ainda é necessário considerar o tempo gasto para a transferência dos próprios dados, o que chamamos de tempo de transferência de bloco. Como pode-se notar, localizar e copiar blocos de dados é uma tarefa que consome tempo considerável, e como o tempo de transferência de bloco costuma ser mais rápido do que o tempo de busca e o atraso rotacional, é comum realizar a transferência de blocos consecutivos de uma mesma trilha ou cilindro o que resulta numa economia de tempo pois se faz a busca uma única vez.

Todo este processo de localizar e transferir blocos de disco pode levar de 09 a 60 milissegundos, o que é considerado muito tempo se for levado em conta o tempo de processamento das CPUs, tornando a localização dos dados no disco um dos principais gargalos nas aplicações de banco de dados.

3. Bancos de dados In-Memory

Devido ao surgimento da computação em nuvem, sua base maciça de usuários e grande exigência de transações e altas taxas de transferência, as empresas encontraram-se na tarefa de desenvolver formas de escalar os serviços de forma rápida a um baixo custo. Como acontece já há algum tempo os caches e todo e qualquer tipo de dado subprocessado ganha cada vez mais relevância

para as empresas que possuem um montante de dados cada vez maior para explorar, e o processamento de tal volume de dados requer uma plataforma rápida e escalável.

A tecnologia IMDB (In-MemoryDataBase) se apresenta como solução para este problema. Os IMDB possuem a capacidade de carregar os dados direto na memória removendo assim uma quantidade considerável de entrada/saída (E/S) relacionada a problemas de desempenho do sistema de banco de dados. Entretanto, os IMDBs também possuem alguns riscos. Tais riscos compreendem a durabilidade dos dados, a flexibilidade dos controles de segurança em comparação a bancos de dados convencionais e algumas preocupações quanta a migração de um sistema a outro. Ao se considerar a utilização de um IMDB os riscos acima citados devem ser levados em conta.

Antes de adentrar na questão da segurança dos dados em IMDBs, torna-se importante tecer algumas considerações a respeito da Escalabilidade de Aplicações, ou seja, a capacidade de acrescentar componentes como nós de computação ou periféricos sem que haja interrupções na disponibilidade dos serviços ativos. Existem duas formas de escalar aplicações: Horizontal e Vertical.

Na forma horizontal pode-se utilizar aplicações que precisam apenas de novos nós de computação à medida que sua demanda por capacidade for aumentando. Aplicações que exigem a realização de um grande número de operações exclusivas e/ou excessivas são adequadas para a paralelização horizontal, permitindo que cada transação atômica, ou seja, não dependente de outra transação concorrente, possa ser dirigida a um nó de computação para ser processada em separado. Exemplos de aplicações que utilizamos diariamente e se aproveitam de plataformas de escala horizontal são aplicações web como Facebook e Twitter. Entretanto, existe certa dificuldade em se transportar certas aplicações para plataformas de escala horizontal devido a problemas de compatilibilidade já que em geral as aplicações não são pensadas com a simultaneidade em mente.

Já escalar verticalmente diz respeito a aumentar a capacidade interna de um sistema para que ele possa lidar com mais transações, este método é mais rápido, pois não promove alterações consideráveis no ambiente de operação do sistema ou em sua arquitetura, porém geralmente é mais caro. Embora não se limite apenas a adição de hardware, o aumento de memória ou de armazenamento em disco de um sistema de computação para processar um maior numero de transações é um exemplo de escala vertical.

A tecnologia de computação In-Memory é uma forma de aumentar a escalabilidade de sistemas verticalmente. Ao se escalar um sistema é necessário identificar gargalos nas transações, uma vez encontrados estes pontos de lentidão torna-se possível a otimização dos mesmos sem a aquisição de mais hardware. Um gargalo ocorre quando uma aplicação demanda grande interação

entre os dados e, em seguida, acesso ao disco. Devemos sempre ter em mente que diferentes tipos de aplicações terão diferentes gargalos dependendo dos níveis de recurso que utilizam e em aplicações baseadas em dados o gargalo mais provável geralmente está ligado a E/S ou armazenamento em disco.

Aplicações ligadas a bancos de dados, em sua maioria, são associadas a E/S e como visto na seção 2.1. o acesso a informações contidas em disco é trabalhosa sendo medida na escala dos milissegundos enquanto o acesso à memória normalmente é medida em microssegundos, ou seja, uma solução para gargalos ligados a E/S em aplicações complexas seria o uso da tecnologia In-Memory já que o acesso a memória é muito mais ágil. Neste caso todos os dados são carregados na memória e todas as transações executadas na mesma. Os IMDBs são o maior expoente desta tecnologia oferecendo ganhos significativos de desempenho ao armazenar os dados na memória principal ao invés de discos e executando operações de E/S inteiramente na memória principal.

Para contornar o fato de que os dados na memória são perdidos quando a energia é cortada e garantir a durabilidade dos mesmos à maioria das soluções In-Memory possui mecanismos para assegurar a preservação dos dados.

A maneira mais comum de se realizar tal tarefa é gravar novamente os dados em armazenamento persistente, no entanto, isso exige a dependência de discos. A maioria das soluções presentes no mercado é capaz de fazer uso de gravação em cache e na memória principal "preguiçosa" ou "imprecisa". As transações são armazenadas na forma de um buffer de log que também está na memória principal e posteriormente são transferidas para os discos. Claro, ainda existe a possibilidade de que ocorra perda de dados caso o buffer de log não consiga completar a gravação em disco quando houver algum tipo de interrupção no fornecimento de energia, porém a maior parte do banco continuará intacta.

Alguns IMDBs oferecem a possibilidade de controlarmos a preguiça da memória na gravação que desejamos utilizar dependendo da relevância da transação realizada. Dessa maneira as transações menos relevantes são atrasadas diminuindo o fluxo de entradas e saídas e as mais relevantes são gravadas de forma síncrona para maior persistência. Essa flexibilidade permite aos usuários adaptar a preguiça às demandas de sua aplicação.

Outro recurso disponibilizado por IMDBs para driblar o uso excessivo de discos é a replicação. Tal recurso se aproveita de que a taxa de transferência da rede é geralmente mais rápida do que a do disco permitindo que várias instâncias de um IMDB sincronizem os dados contidos em um sistema. Em sua configuração mais comum há um único banco de dados ativo, replicado como

um banco de dados em modo de espera ou somente leitura. Isto diminui drasticamente as chances de falha já que as chances de que todos os sistemas se encerrem de forma inesperada é ínfima.

Existem ainda IMDBs que se utilizam de uma tecnologia de replicação não compartilhada. As informações desses bancos são distribuídas por nós de computação para balanceamento de carga e alta disponibilidade. Como é possível escalar a carga para vários nós de computação, a replicação não compartilhada é um exemplo de plataforma escalar vertical e horizontal.

No que diz respeito à linguagem de consulta estruturada (SQL) nem todas as soluções In-Memory possuem um conjunto completo de funcionalidades para gerenciamento de sistemas de bancos de dados relacionais. Isso pode causar o enfraquecimento de algumas restrições trazendo vulnerabilidades ao sistema.

Alguns IMDBs não oferecem o mesmo nível de gerenciamento de usuários e direitos que é comum encontrar em sistemas baseados em disco. Em certos casos o acesso a uma instância do banco de dados permite o acesso a todos os dados desta instância, por isso os administradores são obrigados a criar instâncias distintas para aplicações diferentes.

Outro fator que deve ser levado em conta ao se utilizar um IMDB é o seu principal recurso, a memória. Deve-se ter em conta que bancos de dados muito grandes podem não se enquadrar nas quantidades de memória RAM comercialmente disponíveis. Por isso é necessário o gerenciamento da memória e um recurso disponível em algumas ferramentas é a compressão dos dados. No entanto, o período de compressão dos dados é feito à custa de ciclos da CPU para balancear o tamanho absoluto dos dados a serem processados.

3.1 Exemplos de IMDBs

À medida que a geração de dados aumenta devido a interação de usuários na web, se prolifera também o desenvolvimento de soluções In-Memory, comerciais e gratuitas, para a exploração de tais informações para os mais diversos fins, desde usos científicos a propósitos comerciais. Alguns exemplos são: SAP Hana, Oracle TimesTen, Oracle Database 12c, VoltDB, MySQL Cluster e vários outros. O presente trabalho não pretende discorrer sobre todos os exemplos citados, porém procurará elaborar um pouco mais alguns deles.

3.1.1 Oracle Database 12c: In-Memory

Publicado em julho de 2014 o Oracle Database 12 Release 1 (12.1.0.2) passou a contar com características de tecnologia In-Memory. SGBDs convencionais normalmente utilizam dados analíticos (Analytics) de sistemas Data Warehousing (DW) guardados em sistemas OLTP (Online TransationalProcessing). Analytcs são *queries* complexas que trabalham em grandes tabelas de DW. Entretanto, DW não são processados em tempo real como os sistemas OLTP. O Banco de Dados Oracle In-Memory é capaz de executar consultas e análises complexas tanto em OLTP quanto em Data Warehouse, suportando assim bancos de dados com carga mista.

O Oracle Database armazena os dados em formato de linha, fazendo com que um registro seja armazenado em várias colunas em blocos de dados no disco. Com a organização em linhas, cada nova transação é armazenada como uma nova linha na tabela, este formato mantém todos os dados para um registro juntos na memória e em disco o que facilita seu acesso, tornando-o muito eficiente para uso em banco de dados OLTP e processamento de instruções DML.

No Oracle Database a memória é dividida em duas áreas distintas, o SGA (System Global Area) e o PGA (Program Global Area). Para lidar com os pedidos de processamento dos usuários ligados à instância o Oracle cria processos servidores e uma das tarefas mais importantes dos processos servidores é a leitura de blocos de dados de objetos de *datafile* dentro de um *buffer* do banco de dados. No Oracle 12c foi adicionado uma área opcional no SGA chamada de IM: ColumnStore (In MemoryColumnStore). A IM: ColumnStore armazena cópias de tabelas, *partitions*, colunas, objetos especificados como “in-memory” usando DDL chamados de materializedviews em um formato de colunas para leituras rápidas.

3.1.2 Oracle TimesTen

O Oracle TimesTen é um banco de dados In-Memory com características de um banco de dados relacional convencional. Ele fornece um tempo de resposta baixíssimo e *throughput* muito alto para sistemas com desempenho crítico. Além disso, o TimesTen pode ser usado como um banco de dados de registro ou como um armazenamento em cache de um banco de dados Oracle.

O TimesTen possui métodos otimizados de acesso a memória que lhe permitem obter resultados como transações de leitura em menos de 5 microssegundos e atualizações em 15 microssegundos.

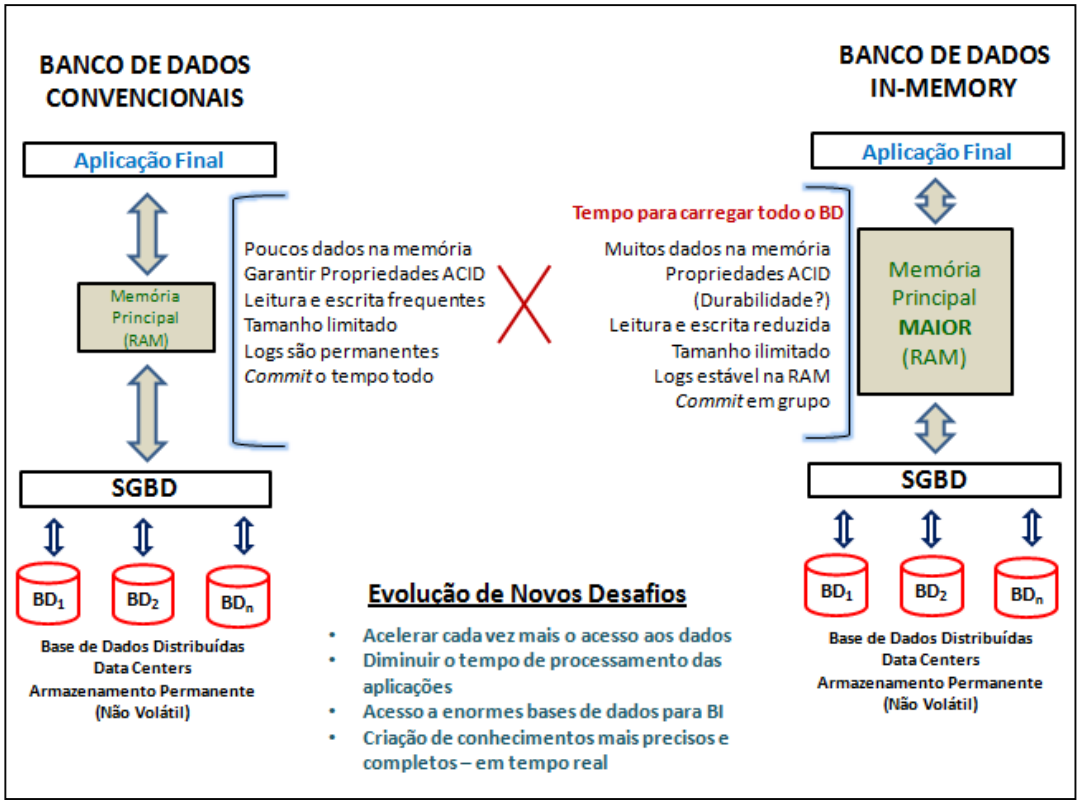
O armazenamento é feito em grupos de cache que são coleções de tabelas do banco de dados usadas com frequência relacionadas através de restrições de chaves externas. Esses grupos podem ser

definidos pelo usuário através de programação. O TimesTen permite a criação de índices nas tabelas In-Memory que podem ou não corresponder aos índices no banco de dados Oracle.

No quesito disponibilidade, quando o TimesTen é usado exclusivamente como banco de dados de registro ele garante alta disponibilidade de seus dados através de replicação, diversas operações online e inúmeros utilitários que suportam falhas, recuperação e atualizações on-line.

4. Breve comparação

Figura 1 – Comparação Banco de Dados convencionais x IMDB



Fonte:

Como ilustrado na Figura 1, em SGBDs convencionais a quantidade de memória principal (RAM) é pequena e utilizada para carregar apenas os dados necessários para transações correntes enquanto a maior parte fica armazenada fisicamente nos Bancos de Dados para armazenamento permanente. Como demonstrado no Capitulo 1 esse processo contribuí para garantir a conformidade do banco com as propriedades ACID, porém devido a hierarquia de arquivos que constitui o sistema o tempo de transito dos dados dentro da hierarquia torna-se cada vez mais alto a medida que o próprio volume de dados produzidos a ser processado aumenta.

Em contrapartida, em um Banco de Dados In-Memory todos os dados ou uma grande parte são carregados em memória principal reduzindo a transferência dos mesmos pela hierarquia de armazenamento. As transações passam a ser executados em tempo de processamento, ou seja, na casa dos microssegundos.

Muito embora os IMDBs tenham como ponto forte a capacidade de processar os dados a altíssima velocidade, demonstra flexibilização de suas restrições de segurança quando se compara aos sistemas tradicionais. Ao se carregar todos os dados diretamente na memória principal há o risco da perda destes dados devido à volatilidade de tal memória, não garantindo totalmente a propriedade de Durabilidade. Os meios de armazenamento permanentes ainda fazem parte dos sistemas In-memory, no entanto são utilizados com menos frequência e tem como principal função a realização de *backups.*

Para contornar este problema e tentar a assegurar a durabilidade dos dados os serviços In-Memory dispõe de vários recursos como o controle do período para a realização dos *backups*, a replicação de instâncias do banco e sua distribuição para múltiplos nós de computação para equilibrar a carga de transações, dentre outras.

5. Conclusão

Como pode ser observado, ambas as abordagens em banco de dados possuem vantagens e desvantagens que se adéquam a situações especificas de seus usuários. No entanto, as soluções In-Memory vêm demonstrando ao longo dos anos que são uma alternativa viável na superação de problemas de desempenho apresentados pelos bancos de dados convencionais.

Ao carregar os dados direto na memória principal os IMDBs podem economizar tempo ao evitar acessos ao disco obtendo assim resultados na casa dos microssegundos como no caso do Oracle TimesTen contra os milissegundos necessários para a mesma tarefa em SGBDs baseados em discos magnéticos.

Embora considerações devam ser feitas a respeito da segurança e durabilidade dos dados, a tecnologia In-Memory apresenta constante aperfeiçoamento neste quesito. Oferecendo replicação do banco, controle de memória cache preguiçosa para a realização de *backup* em discos magnéticos e a distribuição da carga de informações para vários nós de computação para garantir a disponibilidade dos dados.

Portanto, pode-se concluir que os bancos de dados In-Memory podem ser utilizados em diversas ocasiões com razoável segurança e eficácia quando se traça um comparativo entre seus recursos com aqueles disponibilizados pelos bancos de dados baseados em discos magnéticos convencionais.

REFERÊNCIAS BIBLIOGRÁFICAS

GUIMARÃES, Carlos. **SAP HANA - Empresas e Negócios em Tempo Real**. Setor de Database& Technology, março de 2012.

MARQUES, Luís Manuel Oliveira.**Bases de Dados em Memória (IMDB): Sistema de Indexação**. Instituto Superior de Engenharia do Porto. Licenciatura de Engenharia em Informática. Setembro de 2002.

KROENKE, David M. **Banco de Dados: Fundamentos, Projeto e Implementação**. 6. ed. Rio de Janeiro: Editora LTC – Livros Técnicos e Científicos S.A, 1999.

KRUEGER, Jeans; HUEBNER, Florian; WUST, Johannes; BOISSIER, Martin; ZEIER, Alexandre; PLATTNER, Hasso.**Main Memory Databases for Enterprise Applications**.ISBN 978-1-61284-449-7.IEEE 2011. Institute for IT Systems Engineering. University of Potsdam, Germany. 2011.

PLATTNER, Hasso; ZEIER, Alexandre.**In-Memory Data Management: An Inflection Point for Enterprise Applications**. ISBN 978-3-642-19362-0 e-ISBN 978-3-642-19363-7 – DOI 10.1007/978-3-642-19363-7.Springer Heidelberg Dordrecht London New York.Berlin.Springer-Verlag, 2011.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S.**Sistemas de Bancos de Dados.** 4. ed. Rio de Janeiro: Editora Elsevier, 2006.

____, S. **Sistema de Banco de Dados**. 6. ed. Tradução de Marília Guimarães Pinheiro e Cláudio César Canhette. São Paulo: Makron Books, 2012.

SYBASE Inc. **Performance & Tuning for In-Memory Databases Adaptive Server Enterprise 15.5**.SYBASE, Inc. Worldwide Headquarters one Sybase Drive. Dublin. CA 94568-7902 - USA, 2010.