UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

CAMPUS LUIZ MENEGHEL

ÍTALO RUY FERNANDES

ESTUDO COMPARATIVO ENTRE DOIS PARADIGMAS DE BANCOS DE DADOS PARA JOGOS DO TIPO MMORPG

Bandeirantes

2013

ÍTALO RUY FERNANDES

ESTUDO COMPARATIVO ENTRE DOIS PARADGIMAS DE BANCOS DE DADOS PARA JOGOS DO TIPO MMORPG

Trabalho de conclusão de curso apresentado à Universidade Estadual do Norte do Paraná – Campus Luiz Meneghel – como requisito parcial para a obtenção de grau de Bacharelado/licenciatura em SISTEMAS DE INFORMAÇÃO.

Orientadora: Prof. Esp. Fabio de Sordi Junior

Bandeirantes

2013

ÍTALO RUY FERNANDES

ESTUDO COMPARATIVO ENTRE DOIS PARADGIMAS DE BANCOS DE DADOS PARA JOGOS DO TIPO MMORPG

Trabalho de conclusão de curso apresentado a Universidade Estadual do Norte do Paraná – Campus Luiz Meneghel – como requisito parcial para a obtenção de grau de Bacharelado/licenciatura em SISTEMAS DE INFORMAÇÃO com nota final igual a _____, conferida pela Banca Avaliadora formada pelos professores.

Prof. Esp. Fabio de Sordi Junior
Universidade Estadual do Norte do Paraná – UENP

Prof. Me. Rafaella Aline Lopes da Silva
Universidade Estadual do Norte do Paraná – UENP

Prof. Me. Rodrigo Tomaz Pagma
Universidade Estadual do Norte do Paraná – UENP

Bandeirantes, ___ de_____ de 2013.

Dedico este trabalho a todos os meus amigos e companheiros de trabalho, sem o qual eu não teria tido forças de realizá-lo.

AGRADECIMENTOS

Primeiramente agradeço a Deus que colocou as pessoas certas, nos momentos certos, para que esse trabalho pudesse ser realizado, pela força, coragem, enfim por mais essa benção em minha vida.

A minha família, pela confiança, força e motivação, principalmente minha mãe Luciane e meu pai José Afonso pelas noites acordadas que eles passaram para que eu pudesse dormir mais tranquilo.

A meus amigos os quais são mais chegados que irmãos.

Aos meus colegas de trabalho, pela força e compreensão.

Aos meus professores, pela paciência e dedicação nesta etapa tão importante de minha vida.

A banca de Defesa pela contribuição para o aprimoramento deste estudo.

A todos que, com boa intenção, colaboraram para a realização e finalização deste trabalho.

"Combati o bom combate, acabei a carreira, guardei a fé." (2º Timóteo 4-7).

RESUMO

Com o avanço da tecnologia e o crescimento da quantidade de informações, gerou-se a necessidade de bancos de dados que suportem uma quantidade massiva de dados e consigam suprir a demanda de certas aplicações que trabalham de forma distribuída. Nesse contexto, foram criados os bancos de dados não relacionais, pois os bancos de dados relacionais apresentam algumas restrições ao lidar com uma enorme quantidade de dados, devido a sua dificuldade para conseguir alcançar a escalabilidade necessária para suprir certas necessidades dessas aplicações. Os jogos do tipo MMORPGs são aplicações com milhões de usuários e que estão em constante expansão de mercado, além de atualmente serem alvos de diversos estudos. Porém, quando se diz respeito a qual banco de dados tem o melhor desempenho neste tipo de aplicação, não se encontra estudos com esse objetivo. A fim de suprir esta lacuna, este trabalho tem como objetivo identificar qual banco de dados tem o melhor desempenho em jogos do tipo MMORPG. Para isso foi realizado um estudo comparativo entre os bancos de dados relacionais e os não relacionais, identificando suas propriedades e características afim de apontar qual atende de forma mais eficiente as necessidades dos jogos já citados.

Palavras-chave: Banco de dados Relacional. Banco de dados não Relacional. MMORPGs. NOSQL.

ABSTRACT

With the advancement of technology and the growth of the amount of information, spawned the need to have databases that support a massive amount of data and achieve meet the demand for same applications for ex: Facebook and Twiter working with a large amount of information.These conditions generated a need for solutions, that demand data appearing then databases NoSQL, in other words, Non-relational databases, because relational databases has shown flaws when dealing with an large amount of data, not reach scalability to meet the needs of these applications. In turn, the gaming market has experienced significant growth in recent years, especially games like MMORPGs, which generate millions in revenue per year, and some of these games reach millions of users, with new databases springing up in the market, there is a need to check which of these best would supply the need for this type of game. To do this, identify the characteristics of these games, as the issue of availability and scalability.

Key-words: Relational Database.NOSQL.MMORPGs**.**Non-relational databases.

Lista de Siglas

MMORPGs	Massive Online Role-Playing Games
NoSQL	Not Only Structured Query Language
SQL	Structured Query Language
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
BASE	Basically Available, Soft state, Eventual consistency
WOW	Word Of Warcraft
CAP	Consistency, Availability e Partition Tolerance
RPGs	Role-Playing Game
MUDs	Multi-User ID
CA	Consistency, Availability
CP	Consistency e PartitionTolerance
AP	Availability e PartitionTolerance
DDL	Data Definition Language
DML	Data Manipulation Language
DCL	Digital Command Language
LDAP	Lightweight Directory Access Protocol

Lista de Figuras

SUMÁRIO

1 INTRODUÇÃO

O avanço da tecnologia da internet não só enriqueceu o conteúdo e o formato de vários jogos online, como também alimentou o crescimento da indústria de jogos. De acordo com o relatório da empresa D.F.C (INTELLIGENCE, 2007), foi previsto um crescimento de US $ 4,5 bilhões no ano de 2006 para mais de US $ 13,1 bilhões no ano de 2012.

Um dos tipos de jogos que tem tido um crescimento significativo são os *Massively Multiplayer online Role-Playing game* conhecidos pela sigla (MMORPG), onde milhões de pessoas ao redor do mundo interagem entre si dentro destes ambientes virtuais oferecidos por estes jogos. A popularidade destes jogos aumentou nos últimos anos, isso fez com que surgissem estudos de comportamento dos jogadores bem como estudo de suas e características dos jogadores, entre outros (DUCHENEAUT et al, 2010; PITTMAN e DICKEY, 2010; SMAHEL, BLINKA, e LEDABY, 2011).

Com o crescimento significativo de jogadores adeptos a esses tipos de jogos, surgem algumas necessidades que devem ser supridas, dentre elas, suportar a elevada quantidade de usuários, escolher um banco de dados que atenda uma aplicação com essas características e a disponibilidade para atender todos os jogadores (AGRAWAL, e RAKESHETAL, 2008).

A pouco tempo surgiu um conceito descrito pelo termo NoSQL ou seja *Not Only Structured Query Language.* Um sistema cuja base de dados é distribuída, livre de esquemas e tabelas fixas, que normalmente evita operações de junção, que utiliza tipicamente escalas horizontais, não expõe uma interface SQL e pode ser *open source* (BRITO, 2010).

Porém com o crescimento das aplicações e a grande quantidade de dados, alguns requisitos oferecidos pelos bancos relacionais se tornaram cada vez mais críticos para lidar com estas características desses jogos, a flexibilidade, o desempenho e a escalabilidade tem deixado a desejar, apesar das inovações, grandes problemas aparecem rapidamente (LEAVITT, 2010).

Embora os sistemas de bancos de dados relacionais continuem sendo os mais utilizados no mercado. A natureza semi-estruturada e muitas vezes hierárquica da organização das informações na *Web*, bem como a demanda por escalabilidade

horizontal, esbarram em limitações práticas deste modelo. Estes são uns dos motivos que levaram o aumento da utilização de sistemas não relacionais nos últimos anos. (EURE, 2009)

Devido a essas limitações citadas anteriormente dos modelos relacionais, este trabalho visa apresentar um comparativo teórico entre as duas estruturas que compõem os modelos relacionais e não relacionais, as quais são representadas pelos acrônimos ACID (*Atomicity*, *Consistency*, *Isolation*, *Durability)* representando as propriedades dos bancos relacionais e BASE *(Basically Available*, *Softstate*, *Eventual consistency)* representando as propriedades dos bancos não relacionais.

O banco de dados não relacional surgiu como uma alternativa para o relacional, então o trabalho busca identificar qual dos mesmos se comportariam melhor em jogos do tipo MMORPGs. Posteriormente apontar quais as vantagens e desvantagens que esses bancos oferecem para uma aplicação que atende a milhares de pessoas e sustenta uma quantidade de milhões de personagens virtuais utilizados nesses jogos.

1.1 JUSTIFICATIVA

Os jogos do tipo, MMORPG são ambientes virtuais que permitem a interação entre os usuários, estes, normalmente selecionam um *avatar* virtual e colaboram entre si para resolver quebra-cabeças ou completar *quests* (missões).Estes jogos são populares e bem-sucedidos, podemos citar como exemplo, o jogo *World of Warcraft (WOW)* da empresa Blizzard, que tem milhões de assinantes e já geraram bilhões de dólares em receita (CARLES, 2007).

Os jogadores podem deixar o jogo a qualquer momento, e espera-se que suas realizações estejam gravadas quando voltarem, o jogo é composto por vários itens, armas, armaduras, roupas e também outros tipos de informação, como *level* e *status* dos avatares.

Todas essas características são significativas para os jogadores, e espera-se do jogo que todos esses itens estejam armazenados corretamente, pois alguns destes itens são comprados com dinheiro real.

No site da *Blizzard* tem uma breve descrição do jogo *World of Warcraft, que* mostra as características de um MMORPG:

> “Mergulhe no universo de World of Warcraft e junte-se a milhares de poderosos heróis em um mundo virtual repleto de fantasia, magia e aventuras sem limites. Explore sinuosos picos de neve, fortalezas enormes e desfiladeiros hostis. Veja zepelins cruzando campos de batalha, participe de combates épicos e vivencie as experiências lendárias que esperam por você. Entre no mundo de Warcraft (Blizzard,2012)."

Figura 1 - interagindo em um ambiente virtual no jogo WOW

(WOW, 2013)

Os jogos do tipo *Massively Multiplayer Online Games,* exemplo figura 1, emergiram na ultima década como uma aplicação de grande escala distribuída, propondo simulações em tempo real acontecendo em um mundo virtual, propondo entretenimento a jogadores espalhados por todo o mundo. Diante disso vários desafios como escalabilidade, confiança e a consistência de dados foram identificadas pela comunidade de sistemas distribuídos e dos bancos de dados (DEMERS, 2009).

A dificuldade de se escolher um banco de dados dentre os muitos que se encontram no mercado, é escolher um que resolva os problemas que podem surgir nesse tipo de aplicação (BRITO, 2010).

Este trabalho apresentará comparativos teóricos entre as estruturas relacionais e não relacionais, analisando qual suprirá melhor as necessidades dos jogos do tipo MMORPG, apontando as vantagens e desvantagens de utilizar a estrutura relacional e a não relacional.

1.2 OBJETIVOS

O objetivo principal desse trabalho é analisar as estruturas de dois paradigmas de bancos de dados, o relacional e o não relacional, com a finalidade de identificar qual paradigma é mais adequado a jogos do tipo MMORPGs.

Os objetivos específicos:

1) Pesquisar conceitos das estruturas dos bancos de dados relacionais; e não relacionais.
2) Levantar os requisitos necessários para o funcionamento dos jogos em relação ao banco de dados.
3) Comparar as estruturas dos dois paradigmas de banco de dados;
4) Apontar as vantagens e desvantagens das estruturas que compõem os dois paradigmas para que possam suportar as exigências apresentadas pelos jogos do tipo MMORPGs.

1.3 METODOLOGIA

A presente pesquisa pode ser classificada quanto a sua natureza como uma pesquisa aplicada, pois visa gerar conhecimentos para aplicação prática dirigida à solução de problemas específicos. Quanto sua abordagem, pode ser classificada como uma pesquisa qualitativa, em que segundo Gil (1991), não requer os uso de métodos e técnicas estatísticas e o pesquisador é o instrumento chave.

Quanto aos seus objetivos, esta pesquisa pode ser classificada como exploratória, já que pretende proporcionar maior familiaridade com a com o problema e apontar possíveis soluções. Já quanto aos procedimentos técnicos, pode-se

classifica-los como uma pesquisa bibliográfica, pois é elaborada a partir de material previamente publicado.

Esta pesquisa aborda os conceitos de dois paradigmas de banco de dados, os relacionais e os não relacionais, apontando as características de suas estruturas para serem utilizadas em jogos do tipo MMORPGs.

Para tal fim, primeiramente foi feito um levantamento bibliográfico dos jogos do tipo MMORPGs para entender sua estrutura e sua significância na sociedade. Posterior a isso, foi realizado uma análise sobre o teorema de CAP, para compreender as propriedades dos paradigmas de bancos de dados, tanto relacionais quanto não relacionais.

Finalmente, foi realizado um levantamento bibliográfico sobre as características dos bancos relacionais e não relacionais, para compreender as estruturas ACID e BASE e possibilitar a criação de um comparativo para indicar qual banco de dados tem melhor desempenho de acordo com a característica do jogo em questão.

2 Fundamentação Teórica

Esta seção está dividida da seguinte forma: na seção 4.1 é realizado um estudo sobre os jogos de computadores, desde seu surgimento até os MMORPGs. A seção 4.2 contém uma breve explicação sobre sistemas distribuídos e teorema de CAP, já na seção 4.3 será abordado os principais conceitos dos paradigmas de banco de dados.

2.1 Jogos de Computadores

Jogos de computadores surgiram na segunda metade do século 20, começou com Noughts e cruzes, criado em 1952 por AS Douglas da Universidade de Cambridge, já o primeiro jogo doméstico foi lançado em 1972 pela empresa Magnavox com o nome de Odyssey (REVOLUTION, 2007). Em 1975, o primeiro *role-playing games* (RPGs) foi implementado em um computador, inicialmente utilizando apenas textos e, mais tarde com interfaces gráficas para usuários, juntamente com Domínios *Multi-User* (MUDs), que permitiu que múltiplos jogadores interagissem (BARTON, 2007).

2.1.1 MASSIVE ONLINE ROLE-PLAYING GAMES

Os domínios *Multi-User* pavimentaram o caminho para o surgimento dos tipos de jogos *Massive Online Role-Playing Games,* jogos onde um grande número de jogadores interage, simultaneamente dentro de um ambiente virtual. Este tipo de jogo de computador entrou em evidência no final de 1990 com o Ultima Online e EverQuest, Lineage e outros, mas recentemente passou por um enorme crescimento, em particular no caso do *World of Warcraft*, que possui mais de 10 milhões de jogadores adquiridos em um pouco mais de três anos após seu lançamento (BLIZZARD, 2008).

O jogo simula um mundo virtual, onde o jogador tem uma vida, uma missão. No decorrer de um MMORPG, o jogador poderá participar de por missões individuais ou em grupo, o jogo individual é a experiência que um jogador desempenha tudo por si mesmo, explorando a terra virtual, matando monstros, progredindo em direção a *levels* mais altos, sem colaborar com outros jogadores. A opção de jogar

coletivamente que geralmente assume a forma de *group-play* ou *guilds* em que um jogador colabora com os demais jogadores para cumprir missões do jogo (MURPHY, 2007).

Devido ao crescimento significativo no número de usuários de jogos online, a receita da indústria dos jogos online tem aumentado significativamente, como consequência, diferentes gêneros de jogos têm sido desenvolvidos, pois os jogos online são uma forma de lazer aceita, e tendem a estabelecer relações interpessoais ou nas comunidades virtuais (KOREA GAME, 2010).

Os jogos MMORPS por usar uma plataforma multi-servidor e considerado um sistema distribuído, para melhor compreender o próximo capitulo será abordado um pouco sobre o que é um Sistema Distribuído.

2.2 Sistemas Distribuídos

Gerenciar enormes quantidades de dados, a procura pela escalabilidade do sistema, a heterogeneidade das redes, de aplicações, sistemas operacionais, hardware, e um tratamento de falhas, todas essas necessidades motivaram para a construção de um sistema distribuído.

(COULOURIS, 2007) traz em seu livro: “Um sistema distribuído, é aquele no qual os componentes localizados em diferentes computadores, interconectados através da rede, se comunicam e coordenam suas ações através do envio de mensagens possibilitando o compartilhamento de recursos”.

Os sistemas distribuídos costumam ser fortemente consistente, pois como há o compartilhamento de recursos e informações, assegurar a integridade dos dados é valioso para o sucesso deste sistema (COULOURIS, 2007).

Para uma melhor compreensão sobre o que é um sistema distribuído será analisado o teorema de CAP, que serve de base para compreender o funcionamento de um sistema distribuído e quais são as suas propriedades.

2.2.1 O Teorema de CAP

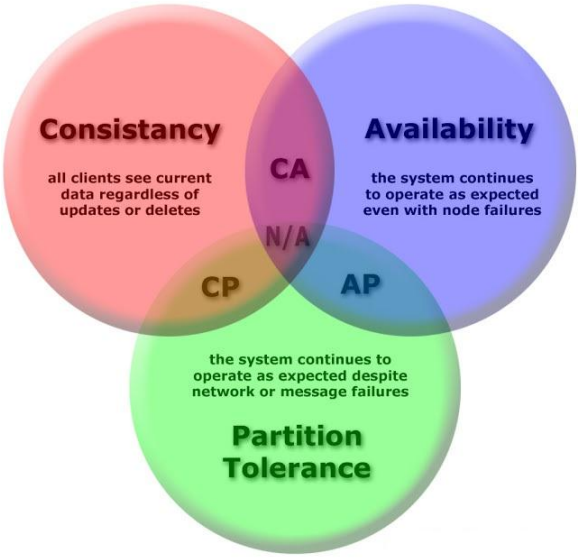
O professor Eric Brewer apresentou o teorema de C.A.P, no ano de 2000, este teorema representa um sistema distribuído que possui três propriedades sendo elas consistência, disponibilidade e tolerância de particionamento de rede, o professor afirmou que o sistema pode garantir somente duas de suas propriedades simultaneamente (HAN, 2011).

A consistência do sistema é a propriedade em que toda a transação enxerga uma instância íntegra do banco de dados, cada execução deve ser garantida. Exemplos de sistemas fortemente consistentes são os que implementam as propriedades ACID (WEI, 2009).

A disponibilidade que, segundo Gleidson (2010), deve garantir que um sistema esteja sempre fornecendo acesso e funcionalidades a seus usuários, é uma propriedade indispensável para sistemas de empresas que precisam disponibilizar seus serviços continuamente, por exemplo, empresas como o Google ou a Microsoft.

A tolerância ao particionamento está relacionada à habilidade de um sistema continuar operando após ocorrerem particionamentos na rede. Assim, se um sistema disponibilizar essa propriedade, ele ainda será capaz de realizar operações como leitura e escrita mesmo após serem realizados diversos particionamentos na rede (GLEIDSON, 2010).

Figura 2 - Propriedades do teorema de CAP e seus possíveis relacionamentos



(LEITE, 2010)

Sistemas que trabalham com as propriedades *consistancy* e *availability* (CA), (na figura 2) não toleram particionamento na rede, ocasionando uma limitação na sua escalabilidade tornando o sistema tolerante somente a escalabilidade vertical. Utilizado normalmente pelos bancos de dados tradicionais relacionais, ex: Microsoft SQL Server, DB2, Postgresql (HAN, 2011).

Diferentemente dos sistemas CA, existem aplicações que não podem ficar indisponíveis, portando necessitam de alta disponibilidade. Para obter *availability* e *partition tolerance*, é preciso seguir o modelo de consistência eventual. As bases de dados NOSQL, por exemplo, Cassandra, Amazon Dynamo (HAN, 2011).

Os sistemas CP asseveram *consistancy* e *partition tolerance*, diferente mente dos sistemas CA e AP. Esse sistema garante forte consistência e tolerância a falhas de particionamento, porém, não garante que estará sempre disponível. Exemplo de sistemas CP são BigTable e MongoDB (HAN, 2011).

2.3 Paradigmas de Bancos de Dados

Segundo Kroenke, (1999) um banco de dados é um conjunto de dados relacionados que atende as necessidades de um conjunto de usuários, ele contém um grau de coerência lógica em seus dados os quais podem ser armazenados sobre alguma forma física.

Silberschatz (2006) traz em seu livro: “Um sistema de gerenciamento de banco de dados é uma coleção de dados inter-relacionados e um conjunto de programas para acessar esses dados.”. A finalidade de um SGBD é prover um ambiente eficiente para armazenar e recuperar informações do banco de dados, facilitando quanto à manutenção de dados.

Segundo Ricarte (2008) enfatiza que as principais contribuições dos SGBD foram fornecer a possibilidade de separar dados armazenados, e a descrição de sua estrutura, através de esquemas, onde restrições sobre tipos de dados e esquemas, que constituem o modelo de dados.

Um banco de dados é utilizado para armazenar os dados, que utiliza de ferramentas para trabalhar os mesmo existem vários tipos de banco de dados denominados paradigmas de banco de dados dentre eles existem o relacional e o não relacional, os quais são descritos nas próximas sub-seções.

2.3.1 Paradigmas de Bancos de dados Relacionais

O modelo de dados relacional foi considerado um marco na evolução dos modelos de dados, oferece uma visão do conjunto de dados composto por tabelas (ou relacionamentos), onde, as tabelas contêm múltiplas linhas (ou tuplas) e colunas (ou atributos), pode-se disponibilizar consultas e manipulações da base de dados utilizando linguagens independentes de sistema como o *Structured Query Language* (SQL) (SILBERSCHARTZ, 1999).

A linguagem SQL desenvolvida pela empresa IBM primeiramente chamada de Sequel e posteriormente passou a ser chamada pelo acrônimo SQL, estabelece-se como linguagem padrão dos modelos relacionais. Esta linguagem possui diferentes partes, a linguagem de definição de dados (DDL) é utilizada para definir, modificar esquemas e excluir relacionamentos; a linguagem de manipulação de dados (DML) possibilita a inserção, modificação e exclusão de dados; a linguagem

de controle de dados (DCL) habilita a busca das informações contidas nos bancos de dados (DONALD, 2012).

Os SGBDs Relacionais possuem diversos recursos que os colocam em uma situação de destaque com relação aos diversos tipos de ambientes computacionais, eles, vêm facilitando a vida de desenvolvedores de aplicações de forma que eles passaram a se preocupar mais com suas aplicações deixando a preocupação com outros aspectos como verificação e garantias de integridade dos dados, controle de concorrência, recuperação de falhas e segurança para os SGBDs (SILBERSCHARTZ, 1999).

Como sucessor dos modelos hierárquico e de rede, o modelo relacional tem sido utilizado pela grande maioria dos SGBD, como por exemplo: o SQL Server, o ORACLE e o PostgreSQL dentre outros (BRITO,2010).

As operações como a leitura e a escrita em um banco de dados relacional, são vistas pelo os usuários como transações. Em banco de dados relacionais, estas transações devem garantir quatro propriedades: atomicidade, consistência, isolamento e durabilidade. (RAMAKRISHNAN, 2008).

O relacionamento entre tabelas é feito por meio de restrições, esses elementos são responsáveis por garantir a integridade dos dados. As restrições mais comuns são denominadas chaves, mais particularmente, as chaves primárias e estrangeiras. As chaves primárias têm por finalidade asseverar a unicidade de cada tupla de cada tabela, por meio de identificadores. As chaves estrangeiras é a maneira mais explícita de analisar o relacionamento entre entidades, permite uma relação de dependência entre atributos de tabelas distintas (RAMAKRISHNAN, 2008).

Projetar tabelas e suas relações de forma adequada não é uma tarefa simples, para auxiliar esse processo foram criadas regras para normalizá-las, outra importante característica do modelo relacional (SILBERCHATZ, 2006).

Segundo Silberchatz, (2006) “Uma transação é uma unidade de execução do programa que acessa e possivelmente atualiza vários itens de dados”. O termo se refere a uma sequência de operações que forma uma única unidade de trabalho lógica, que são vistas pelos SGBDs como uma série de operações de leitura e escrita.

A atomicidade determina que, uma transação deve ser executada por completo ou não ser executada, desfazendo transações incompletas. O isolamento é

responsável por assegurar que, mesmo que as transações sejam de forma intercalada, os resultados devem ser idênticos a executar cada transação uma após a outra de forma serial. A consistência garante que toda a transação obtenha uma instância íntegra do banco de dados, onde cada execução deve ser garantida (RAMAKRISHNAN, 2008).

O problema, segundo Leavitt, (2010) é que a dificuldade de se trabalhar com junções *(join)* de maneiras distribuídas em paradigmas de bancos de dados relacionais, já que suas estruturas não foram desenvolvidas para trabalhar com particionamento de dados.

Brito (2010) alega que, de certa forma pode-se alcançar a escalabilidade em bancos de dados relacional, entretanto, essa tarefa pode ter um custo elevado e se tornar complexa. Quando necessário o aumento de infra estrutura para um banco de dados é normal usar como primeiro recurso distribuição vertical de servidores (*scale up*), ou seja, quanto mais dados, maior o poder de processamento como memória, processador e disco. Essa pode ser uma solução excelente se tratando de curto prazo. Contudo, no futuro o problema da escalabilidade pode voltar a aparecer, uma vez que o hardware possui limitações físicas e tecnológicas.

Outra solução é adotar a distribuição horizontal (*scale out*), isto é, quanto mais dados, mais servidores, não necessariamente de alta capacidade de processamento. Entretanto, isso pode acarretar em outro problema, pois não é uma tarefa fácil adicionar mais um servidor em um sistema distribuído, uma vez que o processo normalmente é de alta complexidade, demorado e de custo elevado (BRITO, 2010).

Em resposta ao crescimento das limitações encontradas no modelo relacional, tem se focado cada vez mais em soluções não relacionais, como o uso de bancos de dados NOSQL (BRITO, 2010).

2.3.2 PARADIGMA DE BANCO DE DADOS NÃO RELACIONAIS

Os bancos de dados não relacionais apresentam características interessantes como alta performance, escalabilidade, replicação, suporte à dados estruturados e sub colunas. Caracterizam-se também por facilitar o trabalho da equipe de desenvolvimento de software, já que não impõem a estrutura de dados rígida, imposta pelos bancos relacionais. Mais especificamente essa classe surgiu da

necessidade de uma melhor performance e fácil escalabilidade em sistemas web, que precisam manter grandes volumes de dados e um escalonamento frequente (IMASTER, 2010).

Na visão de Leavitt, (2010) a tecnologia não relacional não tem como propósito a substituição total do modelo de dados relacional, mas de atender determinadas necessidades que se fazem mais adequadas. Uma prova disto é a possibilidade de acessar um banco de dados não relacional e um banco de dados relacional em uma mesma aplicação. A partir deste ponto do trabalho será usado o termo NoSQL como sinônimo de um banco não relacional.

Como descrito por Brito, (2010), NoSQL é um termo genérico para uma classe de banco de dados não-relacional que apresenta uma alternativa aos bancos de dados relacionais. Ao invés de oferecer a propriedade ACID (*Atomicity*, *Consistency*, *Isolation*, *Durability*), oferece a propriedade BASE (*Basically Available*, *Soft state*, *Eventual consistency*).

> [...] Apesar de possuírem algumas características em comum como, por exemplo, uma maior escalabilidade em relação a bancos de dados relacionais, alta disponibilidade e serem livres de esquema, os bancos de dados NOSQL possuem algumas singularidades como a distribuição de dados [...] BRITO, (2010).

A pesar da maioria das soluções NoSQL serem distribuídas, como é o caso de alguns dos exemplos citados anteriormente, existem certos sistemas, como o MongoDb ou CouchDB, que promovem o particionamento e a replicação dos dados. Enquanto outros, segundo Brito (2010), podem deixar essa tarefa para o cliente, como é o caso da Amazon qual é livre de esquemas e distribuído.

Sistemas NoSQL, oferecem soluções mais simples em relação a arquitetura de armazenamento dos dados, esses sistemas foram construídos com a idéia de que, simplificando a forma como opera um banco de dados sobre os dados, um arquiteto pode prever melhor o desempenho de uma consulta (HECHT e JABLONSK, 2011).

Em muitos sistemas NoSQL, a lógica complexa de consulta é deixada á aplicação, o que resulta em um armazenamento de dados com o desempenho da

consulta mais previsível, por causa da falta de variabilidade em consultas(HECHT e JABLONSK, 2011).

O modelo de dados de um banco de dados especifica como os dados são organizados logicamente. Seu modelo de consulta determina como os dados podem ser recuperados e atualizados. Sistemas NoSQL combinam dados e modelos diferentes de consulta, resultando em diferentes considerações de arquitetura (STONEBRAKER , 2010).

Com relação aos modelos de bancos de dados não relacional às categorias básicas são:

- Sistemas baseados em armazenamento chave-valor que, segundo Levaitt (2010), são sistemas que armazenam valores indexados por chaves para posterior recuperação.
- Sistemas orientados a documentos, que armazenam e organizam os dados como coleções de documentos ao invés de tabelas estruturadas como é feito no Modelo Relacional (LEAVITT, 2010).
- Sistemas baseados em grafos, cujos dados são armazenados em nós de um grafo e as associações entre os dados são representas através de suas arestas (CHANG, DEAN, e GHEMAWAT 2008).

Os bancos de dados não relacionais e seus modelos trabalham com algumas características que diferem do modelo de dados relacional segue algumas dessas características:

Mapreduce: é dividido em duas fases Map e Reduce, na qual a primeira fase, o nó principal recebe os dados, divide em partes menores e envia aos outros nós para processarem as informações. Ao término do processo os nós enviam o resultado ao nó principal. Na fase Reduce, o nó principal combina as respostas obtidas pelos outros nós gerando o resultado final do processamento (HAN e SONG, 2011).

Replicação: A replicação escalar por duplicação de informações é a cópia das informações em mais de um banco para aumentar a capacidade de recuperar as informações (WEI-PING, MING-XIN, e HUAN 2011)

Schema-Free :Um dos fatores que contribuem para um banco de dados NoSQL escalar, por causa da ausência de um esquema (*schema*). Todos os bancos desse modelo possuem em comum *key-value stores*, que salvam um conjunto de

entradas formadas por uma chave associada a um valor, sendo esse valor de qualquer tipo (LEAVITT, 2010).

Clusterização: Basicamente compreende um banco de dados armazenado e gerenciado por mais de um servidor, provê uma alta disponibilidade e um alto desempenho do sistema, a organização se beneficiam diminuindo o tempo de inoperabilidade do banco (STONEBRAKER, 2010).

Sharding: Consiste em dividir os dados horizontalmente, ou seja, quebrar as tabelas, diminuindo o seu número de linhas e separando-as em ambientes diferentes. Neste ponto todos os dados de uma partição não devem conter referências aos dados de outras partições, sendo que os dados em comum deverão ser replicados entre as bases (WEI-PING et al., 2011).

2.3.2.1 Modelos de dados não relacional

Para compreender melhor os modelos de dados está subseção traz uma breve descrição dos modelos citados anteriormente.

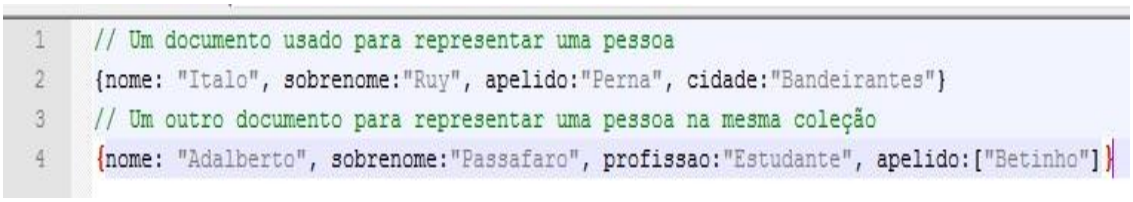
Modelo Documental

Este tipo de banco NoSQL provê o armazenamento de documentos mais complexos como árvores em XML (*Extensible Markup Language*). Eles podem ser buscados através de sua chave ou por qualquer um dos valores contidos dentro de seu documento. Pode-se citar o Mongo ou o Couch, como bancos que usam documentos para armazenar seus dados.

Umas das categorias mais conhecidas dentre os modelos dos bancos de dados NoSQL se caracteriza pelo fato de não possuir tabelas com um número fixo de campos bem definidos, mas sim coleções. Cada coleção pode armazenar um ou mais documentos (no lugar de registros em uma tabela). Um documento é apenas um agregado de atributos que não possui uma regra rígida que defina quais os tipos de cada atributo e qual tipo cada um deve possuir (são sem esquema, *schemaless*) (STONEBRAKER, 2010).

Normalmente os documentos são salvos num formato similar ao JSON, tal como pode ser visto no exemplo na figura 3:

Figura 3 - Exemplo de documento modelo de dados documental.



(Própria autoria)

Nos documentos pode-se ter registros bem diferentes um do outro dentro da mesma coleção. Diferentemente do modelo relacional que é preso a esquemas e relacionamentos, onde geralmente tem-se uma quantidade de tabelas esparsas.

Uma tabela esparsa é aquela na qual existem colunas das quais apenas algumas são sempre preenchidas, e as demais apenas raramente. Para representar essa situação segue um exemplo na tabela 1:

Tabela 1- Representação de uma tabela esparsa

Código	Nome	Sobrenome	Cidade	Estado	Sexo	Idade
1	Larissa		Londrina	PR	F	
2	Vinicius	Silva				24
3	Paulo		Bandeirantes	PR	M	
4	Fabio	Sordi		PR		28

(Própria autoria)

Na tabela 1, representa um mau uso do modelo relacional, as colunas que raramente são preenchidas, ao incluir colunas raramente usadas em tabelas do modelo relacional, estão ocupando um espaço maior de armazenamento e comprometendo o desempenho do sistema como um todo.

Os bancos documentais não são presos a regras de armazenamento de dados. Geralmente o usuário, armazena objetos semelhantes na base de dados, porém caso aparece um novo atributo, pode-se incluí-lo apenas nos documentos onde é mesmo necessário (STONEBRAKER, 2010).

Por ser livre de regras, uma desvantagem é que o usuário deve tomar cuidado para que a organização da base de dados não fique totalmente desorganizada, para que não cause dificuldade na hora de analisar o banco caso seja necessário.

As vantagens que esse modelo apresenta é a facilidade de representar os objetos do mundo real, diferentemente do modelo relacional que trabalha com uma estrutura bidimensional, também o fato de poder adicionar os atributos conforme a necessidade da aplicação.

Modelo Orientado a Grafos

O modelo de dados orientados a grafos é dirigido para os relacionamentos entre os objetos. A metáfora neste caso é o grafo, o modelo de grafos não trabalha mais com coleções de dados, apenas vértices representando os objetivos do sistema e arestas indicando o relacionamento entre os mesmo, normalmente cada vértice representa um agregado de atributos, similar ao modelo documental (BRITO, 2010).

As pesquisas podem ser feitas tanto por algum dos atributos dos vértices quanto pelos relacionamentos. Um exemplo interessante poderia ser a representação de alguma estrutura do tipo causa/efeito, como uma fábrica, em que cada etapa de produção gera a entrada para uma ou mais etapas que precisem ser monitoradas (STONEBRAKER, 2010).

O banco de dados orientado a grafos permite uma aplicação de um modelo navegacional entre os elementos persistidos, o usuário pode, caso ele queira, caminhar no grafo ao implementar um algoritmo de busca em profundidade ou busca de menor caminho (BRITO, 2010).

Um exemplo de uma aplicação que seria possível utilizar o banco de dados orientado a grafos suponha que aplicação necessite de relacionamentos complexos, ou seja, um sistema de viajem que mantem onde eles moraram e para qual lugar viajaram em um modelo relacional geraria as seguintes tabelas ilustradas na figura 4.

Figura 4 - Tabelas modelo relacional

Pessoas

id_pessoa	nome
1	João
2	Ricardo
3	Carolina
4	Maria
5	Fernando
6	Fábio

Cidades

id_cidade	nome
1	Toronto
2	Roma
3	Berlim
4	Bruxelas
5	Paris
6	Nova Iorque
7	Tóquio

Residencias

id_pessoa	id_cidade	data_inicio	data_fim
2	1	-	-
4	3	-	-
5	3	-	-
2	5	-	-
1	6	-	-

Viagens

id_pessoa	id_cidade	data
1	5	-
2	2	-
2	4	-
3	2	-
4	6	-
4	7	-
5	4	-
5	5	-
5	7	-
6	6	-

(ALMEIDA, 2013)

Com está estrutura de tabelas pode-se realizar tipos de consultas em um banco relacional, utilizando a linguagem SQL, como por exemplo, na tabela 2 pessoas que viajaram para o mesmo lugar que o (Ricardo) uma possibilidade de SQL seria.

Tabela 2 - código SQL

```
select distinct(v.id_pessoa),  p.nome from viagens v inner join pessoas p on (
     v.id_pessoa = p.id
  )
where
  v.id_cidade in select  viagens_sub.id_cidade from viagens viagens_sub
where viagens_sub.id_pessoa = 2
  ) and v.id_pessoa <> 2;
```

(própria autoria)

No caso como, não devemos retornar o próprio “Ricardo”, o nosso ponto de partida, por isso, usa-se o filtro id <> 2.

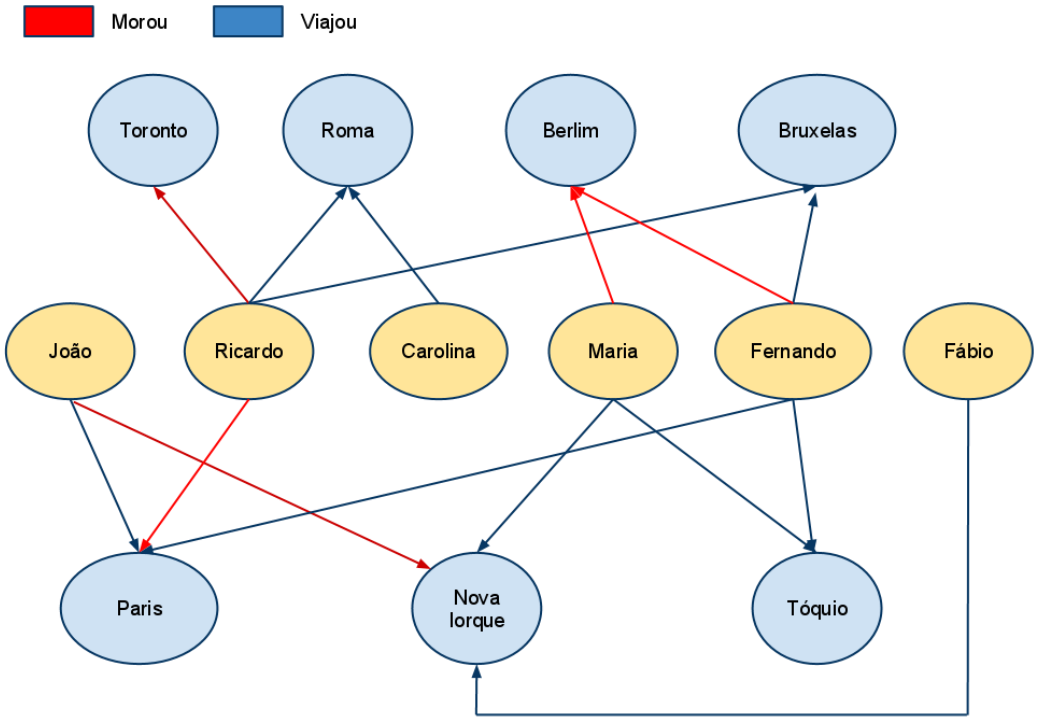
Mesmo sendo uma consulta realizada em dois passos, por causa do sub-selects, a pesquisa em si não é difícil de ser realizada, por causa da subdivisão da *query* primeiro busca as cidades que ele passou e em seguida busca-se quem foi pra essas cidades.

No entanto, caso a consulta tivesse que retornar, por qual cidades passaram, (seja viajando ou morando), retornando as pessoas que passaram por Roma, a consulta neste caso já não é mais trivial de ser desenvolvida, a *query* de consulta

seria de uma complexidade maior, poderia exigir vários *joins* e *subqueries* fazendo com que a consulta perca sua *performance*.

No banco de dados orientado a grafos, pode-se ser modelada com grafos como na figura 5.

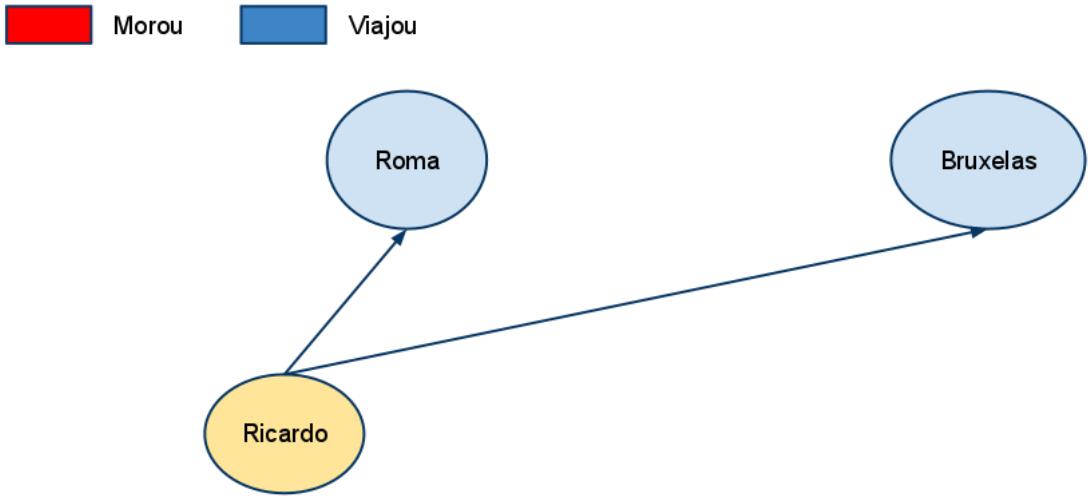
Figura 5 - modelagem grafos



(ALMEIDA, 2013)

Neste caso, o grafo apresenta uma melhor visão dos relacionamentos ocorridos, uma visão da primeira *query* apresentada anteriormente, para saber a onde o Ricardo viajou como mostra na figura 6.

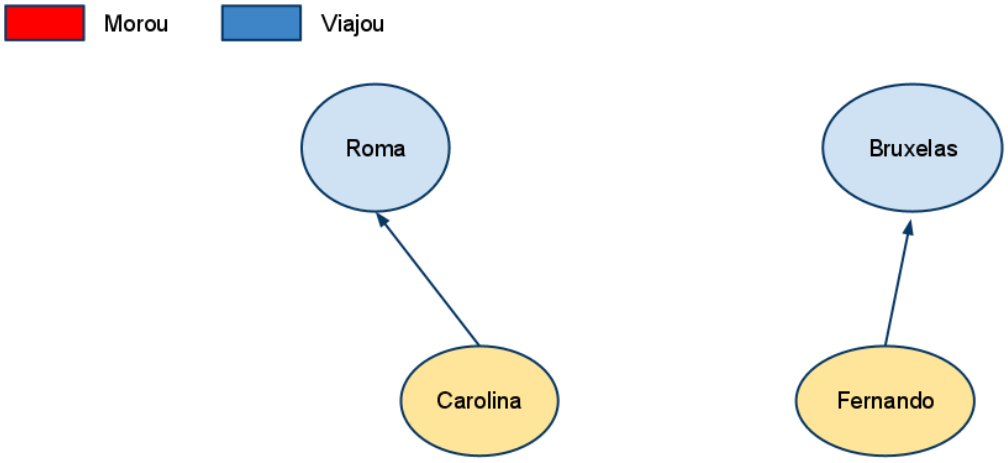
Figura 6 - Cidades onde Ricardo viajou e morou



(ALMEIDA, 2013)

Em seguida, a partir de cada cidade, é preciso dos *relacionamentos de entrada*, que também sejam do tipo VIAJOU e, com isso, têm-se as pessoas que viajaram para os mesmos lugares que o Ricardo como mostrado na figura 7.

Figura 7 - Pessoas que viajaram para as mesmas cidades que Ricardo.



(ALMEIDA, 2013)

O banco de dados orientado a grafos permite uma melhor visão e organização dos relacionamentos, facilitando para o administrador do banco de dados realizar consultas desde tipo.

O código abaixo representa a consulta complexa colocada para os bancos relacionais, podemos listar os relacionamentos de ambos as arresta tanto, para quem viajou ou morou, as entradas e saídas neste caso.

Tabela 3 Exemplo código de busca banco de dados orientado a grafos

```
Node lugarDeOrigem = db.getNodeById( id );
 Iterable<Node> nos = Traversal.description()
.relationships(Relacionamentos.MOROU_EM, Direction.BOTH)
relationships(Relacionamentos.VIAJOU_PARA, Direction.BOTH)
 evaluator(Evaluators.atDepth(2)) traverse(lugarDeOrigem).nodes();
```

(Própria Autoria)

O código indica qual é o Nó de origem e retorna as entradas e saídas dos mesmos.

Modelo Chave Valor

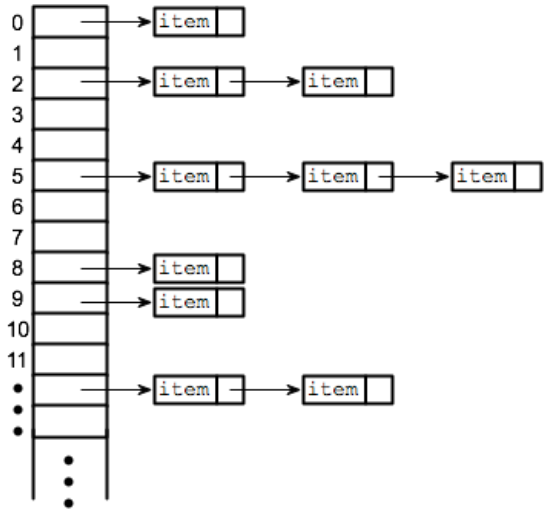
Segundo Strauch, (2012), o modelo chave-valor, apresenta uma alta escalabilidade, porem apresenta baixa consistência, mas, no entanto a maioria deles omite recursos ricos para consultas e ferramentas analíticas ex: operações de *joins* e agregação.

Segundo Toth (2011), o modelo chave-valor na maioria dos casos apresenta uma interface composta principalmente pelos seguintes métodos:

- Put (chave, valor) – representa o método que insere um registro, garante maior velocidade a operação;
- Get (chave) – responsável por recuperar um registro, não oferece opções para buscas mais complexas.

Uma categoria mais simples dos modelos NoSQL, tem o maior ganho de performance quando bem aplicado. A consulta e realizada, de uma chave, que pode ou não ter algum valor (de qualquer tipo) relacionado. Utilizado geralmente em sistemas que utilizam de caches ou para informações que precisam ser alteradas em tempo real. O método de pesquisa mais simples dentre os outros modelos baseado em uma tabela hash da chave representado na figura 8.

Figura 8 - Exemplo de tabela hash



(TOTH, 2011)

3 Comparativo Banco de dados Relacional e não Relacional

Esta seção está dividida da seguinte forma: na seção 5 traz um comparativo entre os bancos de dados relacionais e não relacionais. Na seção 5.1 está contida, uma breve explicação e comparações entre as propriedades dos bancos de dados. Na seção 5.2 será abordada a arquitetura dos jogos do tipo MMORPGs e suas características.

Uma das principais diferenças entre os dois modelos de banco de dados é a consistência e a escalabilidade, esta ultima, é considerada um dos principais motivos para o surgimento dos bancos de dados não relacional. A seguir na tabela 3, será exposto uma breve comparação entre os bancos relacionais e os NoSQL.

Tabela 4 - Comparativo entre Banco de Dados Relacional e Banco de Dados NoSQL

Banco de Dados Relacional	**Banco de Dados NoSQL**
O que é: Baseia-se no conceito de entidade e relacionamento, no qual, todos os dados são guardados em tabelas. Os dados são separados de forma única tentando diminuir ao máximo a redundância. Pois a informação é criada pelo conjunto dos dados, onde são as relações entre as tabelas que fazem esse serviço.	O que é: Uma solução alternativa para os bancos de dados relacionais. Possuem uma alta escalabilidade, disponibilidade, desempenho e menor tempo de resposta a consultas.
Características: tabelas, esquema definido, hierarquia, redundância mínima, entidade e relacionamento, formas normais, transações ACID (Atomicidade, Consistência, Isolamento, Durabilidade).	Características: registros, schema-free, tolerância à falha, escalabilidade, clusterização, mapreduce, sharding, paradigma BASE (*Basically, Available, Soft state, Eventual Consistency)*.
Necessidades: Sistemas locais, sistemas financeiros, sistemas corporativos, segurança da informação, consistência dos dados.	Necessidades: Sistemas em nuvem, análises sociais, alta escalabilidade, performace na consulta/escrita, replicação.
SGBDs: *DB2, Firebird, InterBase, Microsoft SQL Server, MySQL, Oracle, PostgreSQL.*	SGBDs: Cassandra, BigTable, MongoDB, CouchDB, Dynamo.
Algumas Empresas: SAP, OpenERP, Previdência Social, Caixa, Itaú, Salesforce, Vale.	Algumas Empresas:Twitter, Facebook, Digg, Amazon, LinkedIN, Google, Yahoo, The New York Times, Bit.ly, eBay.

.(SOUSA; ROCHA, 2010)

Os pontos fortes dos bancos de dados relacionais são: a segurança das informações; o controle de concorrência; a recuperação de falhas; a otimização de

consultas, e alguns processos de validação, verificação e garantias. Estes bancos garantem também a integridade dos dados, pois utilizam a estrutura das entidades e relacionamentos, e por estarem a mais tempo no mercado, possui uma confiabilidade de seus usuários.

A seguir, será apresentado um quadro comparativo sobre as propriedades escalabilidade, consistência e disponibilidade nos dois paradigmas.

Tabela 5 - Pontos fundamentais: Bancos de Dados Relacionais x Não Relacionais

Propriedade	Bancos Relacionais	Bancos Não Relacionais
Escalabilidade	Devido à estruturação do modelo, é possível, mas complexo.	É sem dúvida a principal vantagem do NoSQL. Possui mais flexibilidade na inclusão de dados.
Consistência	Ponto forte do Modelo Relacional. A regra de consistência presente garante um maior rigor à consistência das informações.	Não garante a consistência da informação, caso nenhuma informação seja atualizada, retornará a todos os pedintes o mesmo valor.
Disponibilidade	Este modelo não suporta eficientemente grande demanda, dado a dificuldade e distribuição de dados.	Outro ponto forte do BD NoSQL. Possui um alto grau de distribuição de dados, e garante um maior número de solicitações.

(SOUSA; ROCHA, 2010)

3.1 Escalabilidade

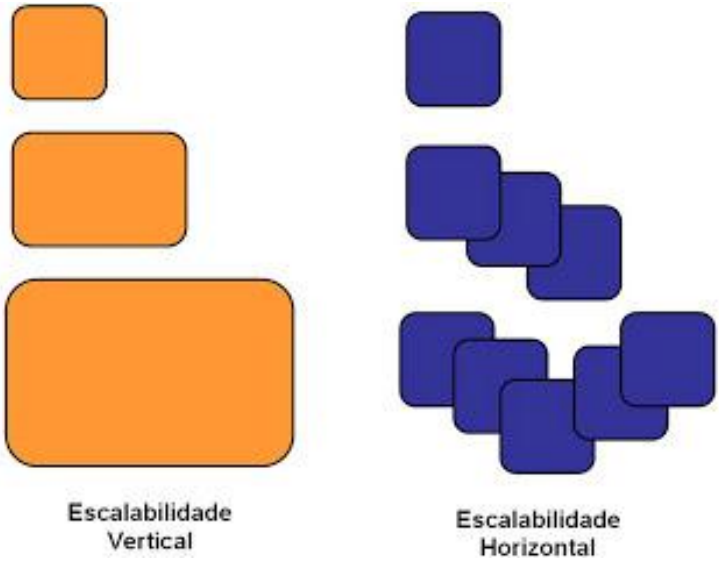
Uma característica fundamental dos sistemas NoSQL é "*shared nothing*" , ou seja, nada compartilhado na escala horizontal (na figura 3), na replicação e no particionamento de dados ao longo de muitos servidores. Isto lhes permitem a suportar um grande número de operações de leitura/escrita estas cargas de simples operações é chamado de OLTP (processamento de transação online), mas também são comuns em aplicações web modernas.

O termo "escalabilidade horizontal”, é a capacidade de distribuir os dados e as cargas dessas simples operações ao longo de muitos servidores, sem RAM ou disco compartilhado entre os servidores, escala horizontal difere de escala "vertical", em que um sistema de base de dados utiliza muitos núcleos, e o uso eficaz de múltiplos

núcleos é importante, porém o número de núcleos que pode ser compartilhado na memória é limitada.

A escala horizontal geralmente, se revela menos custoso em termos financeiros, pois não exige servidores de alto desempenho. Note-se que o particionamento de dados não está relacionado com a escalabilidade, pois o particionamento é uma característica de um banco de dados, ao contrário da escalabilidade que esta relacionada com o paradigma.

Figura 9: representação de escalabilidade horizontal e vertical



(MANOEL VERAS 2009)

3.1.1 ESCALABILIDADE EM SISTEMAS RELACIONAIS E NÃO RELACIONAIS

A escalabilidade em sistemas de banco de dados relacionais, devido a sua estrutura padronizada através das propriedades ACID, limita a sua escalabilidade. É possível escalar um banco relacional horizontalmente, onde é necessário um conhecimento avançado da linguagem de SQL, linguagem utilizada nos bancos de dados relacionais e Segundo Brito (2010) um banco relacional que trabalha em uma escalabilidade horizontal não consegue garantir as suas propriedades.

Figura 10: Propriedades Segundo o teorema de CAP para bancos de dados relacionais

(ELEMARJR 2011)

Portanto se optarmos por disponibilidade e consistência (na figura 4) dos dados descarta-se a probabilidade de se trabalha com a escalabilidade horizontal O que ocorre na maioria dos casos, para aumentar o desempenho do servidor de banco de dados investe-se em escala vertical, ou seja, aumentar o poder de processamento da maquina.

3.1.2 Consistência em sistemas relacionais e Não Relacionais

Consistência é definida por sistemas que estão integralmente funcionando, ou não estão funcionando, ou seja, caso o banco de dados estiver realizando alguma operação em cima do dado o mesmo fica inacessível até que a operação esteja completa. A consistência garante a informação, espera-se que em cada translação os dados estejam consistentes.

Figura 11: Propriedades Do teorema de CAP representando a consistência em bancos de dados relacionais

(ELEMARJR 2011)

Portanto se optar por consistência e tolerância ao particionamento (na figura 5), então a disponibilidade fica comprometida, toda a vez que ocorrer uma transação, o banco precisará replicá-la para os “espelhos” deixando a informação indisponível até que a replicação esteja completa.

Figura 12: Consistência nos bancos de dados não relacionais

(ELEMARJR 2011)

Caso optar pelas propriedades disponibilidade e tolerância ao particionamento (na figura 6), então o sistema oferece uma consistência eventual.

A consistência nos bancos de dados não relacional é denominada consistência eventual, ela disponibiliza o acesso ao dado mesmo que este dado ainda não tenha sido replicado, ex: caso haja alguma alteração em algum tipo de dado até que o dado tenha sido atualizado em todos os nodos ela disponibiliza o acesso ao dado anterior diferente da forte consistência dos bancos de dados relacionais.

A consistência eventual permite que os sistemas de banco de dados não relacional, sejam muito mais escaláveis, porém, ocorrem alguns conflitos na atualização dos dados até que a atualização alcance todos os nodos, existe a dificuldade de fazer com que as réplicas tenham o mesmo valor em todos os nods. O banco de dados Dynamo, da Amazon, é um exemplo onde ocorreu que os registros eliminados em alguns nods acabam reaparecendo (DECANDIA, 2007).

Sendo assim, a consistência eventual permite que vários nós possam escrever em um mesmo registro, porém acaba por aumentar a possibilidade de conflito entre versões de um determinado dado. Como forma de tratamento deste problema cada banco de dados NoSQL possui suas técnicas, como no caso do Projeto Cassandra que utiliza de valores “*time stamps*” para determinar a ultima versão de determinado dado (ZYP, 2010).

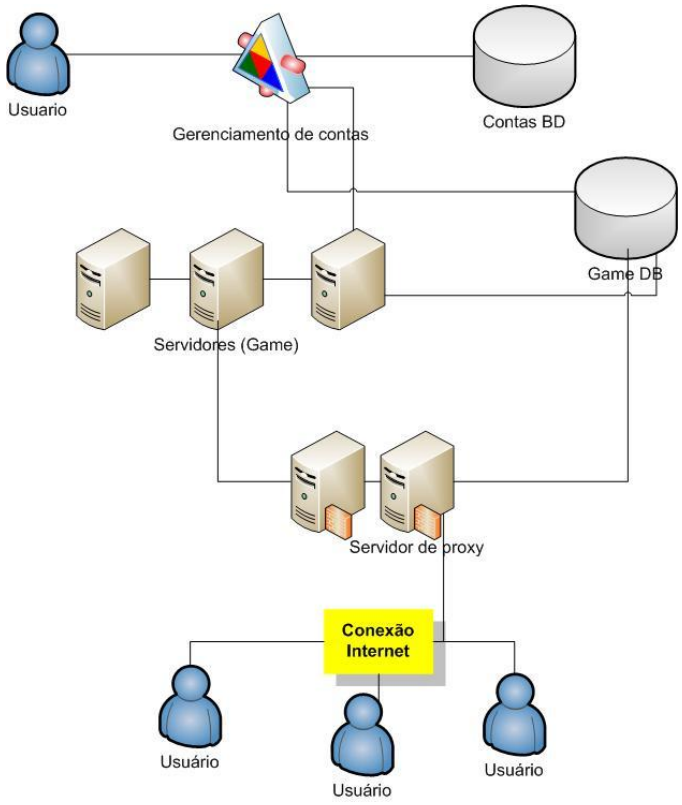
3.2 Arquitetura MMORPGs

A maioria dos MMORPGs atuais adota a arquitetura cliente servidor (C/S), para servir uma quantidade massiva de jogadores, o modelo multi-servidor é muitas das vezes adotada, uma visão simplificada da arquitetura MMORPG, consiste em quatro partes: o cliente, a conexão com a internet, o servidor do jogo e gerenciamento de contas de usuários.

O uso de interfaces web para o gerenciamento de contas já tornou-se padrão, usando um aplicativo web cuja a função é controlar as contas de usuários, tal que o mesmo não esta relacionado ao jogo.

Utiliza-se um Servidor de proxy que se comunica com todos os jogadores, normalmente também é usado um algoritmo de balanceamento de carga, como o *round Robin,* que é usado para selecionar o próximo jogador que quer se juntar ao jogo, ou seja o servidor de Proxy atua como um distribuidor entre clientes e servidores dos jogos, como pode ser analisado na figura 5.

Figura 13: Arquitetura MMORPG



FONTE: (FEIJÓ 2003)

Nota-se que a arquitetura utilizada nos jogos MMORPGs, utiliza-se de duas bases de dados sendo um para o servidor de gerenciamento de contas e outro para o game World, para compreender qual banco de dados escolher, precisamos entender o funcionamento do servidor que vai controlá-lo.

3.2.1 SERVIDOR WEB

Sabe-se que os usuários dos jogos, passam primeiramente por um sistema de autenticação web separado da base de dados do jogo, poderia se utilizar um banco de dados que permita uma resposta rápida para autenticação.

Dentre os modelos de banco de dados não relacionais, podemos citar o modelo chave-valor, como um dos modelos mais simples que utiliza de uma tabela hash. Segundo Nvrsdqef (2010), este banco de dados é composto por um conjunto de chaves, as quais estão associadas ao um único valor, que pode ser uma string ou um binário.

Ao Utilizar um servidor Lightweight Directory Access Protocol (LDAP), ou seja, um protocolo para acesso a banco de dados, especializados chamados “diretórios”. Geralmente o banco mais utilizado em um sistema LDAP é o banco de dados da Oracle o Berkeley um banco de dados não relacional, que utiliza uma estrutura chave-valor.

3.2.2 BANCOS DE DADOS DO JOGO

O banco de dados do jogo funciona embarcado, ou seja, os jogos MMORPGs são divididos em duas partes a parte que é denomina Cliente, que é a parte que o usuário faz o download e instala em seu computador, e a parte do Servidor que onde o jogo fica armazenado (SUZNJEVIC, STUPAR, e MATIJASEVIC, 2011).

Para coordenar o fluxo de dados de um MMORPG é necessário um *game* cliente e um servidor, o *game* Cliente é o aplicativo que roda no computador do usuário, o Servidor é o computador onde o usuário se conecta quando estiver pronto para começar a jogar (ZHONG, 2011).

No *game* Cliente está armazenado quase tudo sobre o jogo as telas, mapas, paisagens os *mobs* (são os monstros nos jogos), estes dados está localizado em arquivos na base de dados do computador, o computador usa um conjunto de regras para traduzir esses arquivos, o próprio computador processa o som e os gráficos do jogo (FENG, BRANDT, e ONLINE 2007).

O game cliente recebe informações do banco de dados do servidor, enquanto o jogo esta em processo o banco de dados do servidor informa ao banco de dados do cliente, o lugar a onde o seu personagem está e quais os mobs e jogadores que estão próximos ao usuário. Calcula a distancia de ataque caso o usuário queira atacar algum *mobs* ou outro usuário no jogo, notifica se o usuário está sendo atacado ou não, instrui ao computador do usuário a animação, caso ocorrer à morte de um *mob* ou de algum outro usuário (HOWSTUFFWORKS, 2013).

O desempenho dos MMORGs não depende somente do banco de dados, os jogos em si requerem certo processamento da máquina do usuário, e da velocidade de conexão com a internet do jogador, caso haja algum empecilho no caminho o jogador deve passar por alguma latência (HOWSTUFFWORKS, 2013).

4 Resultados Obtidos

Como visto na subseção 3.2.2, o banco de dados dos jogos MMORPGs, trabalham com operações simples, por trabalhar como um banco embarcado, apenas fornece a localização, e armazena itens e *levels* do usuário, e normalmente não chega a trabalhar com uma quantidade massiva de dados.

Teoricamente tanto os bancos relacionais como os bancos não relacionais, conseguiriam suprir essas necessidades, salvo o banco de dados não relacional que pertence à categoria chave/valor. Não é possível apontar qual o melhor para esse tipo de aplicação, por que a velocidade do jogo também depende de outras variáveis como o computador do usuário, a potência do servidor e da latência da rede, e somente depois seria influenciado pelo desempenho do banco de dados.

Uma opção seria a realização de testes de desempenho, porém esta opção se mostrou inviável por depender de todas as questões de estrutura. Outros fatores que torna inviável os testes é a necessidade de popular o jogo, para a realização dos mesmos.

Tanto o banco de dados relacional no caso o MySQL, como o não relacional como o MongoDB, conseguem trabalhar embarcados, porém o não relacional, por ser um banco de dados que surgiu a pouco tempo, pode mostrar algumas dificuldades em termos de configuração para trabalhar dessa maneira, devido a falta de documentação relacionada a este método de trabalho.

No entanto devido á pesquisa realizada, pode-se indicar algumas estruturas de bancos de dados para melhor desempenho de acordo com a necessidade da aplicação.

Caso sua aplicação necessite de uma integridade absoluta dos dados, optaria pelo banco de dados relacional o qual a sua estrutura, garante essa tal integridade, exemplo de alguns sistemas que devem garantir uma integridade seria uma aplicação de um sistema bancário no caso o qual a integridade deve ser mantida.

Portabilidade, se a aplicação depende de uma portabilidade maior poderia opinar por um banco de dados relacional, pois o banco de dados relacional é mais antigo no mercado, por isso alcança uma portabilidade maior que as dos bancos de dados não relacionais.

O Sistema possui atributos baseados em domínio, ou possuem hierarquia de derivadas, nesse caso os bancos de dados não relacional se comportaria melhor devido a sua estrutura livre de esquema.

O modelo de persistência possui muitas tabelas esparsas, neste caso o modelo não relacional teria um melhor desempenho devido a sua estrutura.

Os relacionamentos entre as entidades utilizam de muita complexidade, então neste caso o melhor modelo seria o banco de dados orientado a grafos já que o foco desse modelo de dados é o relacionamento entre objetos. Exemplo de um sistema de relacionamentos complexo seria uma rede social.

A persistência nos dados deve sempre ser garantida neste caso os dois paradigmas supririam essa necessidade, no caso do não relacional, optaria por um modelo de dados orientado a documentos.

Se o espaço em disco é de extrema importância, neste caso o modelo relacional seria o ideal, pois ele não utiliza de redundância de dados.

Caso a memoria principal seja importante o desempenho tanto do modelo relacional e o não relacional (orientado a documentos), seria de alto desempenho.

Caso a aplicação dependa de pesquisas por identificares, neste caso é melhor utilizar um banco de dados relacional, por ser um modelo estruturado é utilizar de fatores como chave primaria e chaves estrangeiras.

Se a intenção é somente cachear dados, neste caso o banco com melhor desempenho seria um modelo chave-valor devido a sua estrutura.

A tabela 5 foi feita para uma melhor organização destes requisitos:

Tabela 5 Relação dos bancos de dados conforme a necessidade da aplicação (continuação)

Requisitos	**Relacional**	**Não Relacional**		
		Orientado a Documento	**Orientado a Grafos**	**Chave- Valor**
Necessidade de integridade absoluta dos dados	Alta	Não garante	Não garante	Não garante
Necessidade de portabilidade do banco de dados	Alta	Médio	Médio	Médio

Tabela 5 – Relação dos bancos de dados conforme a necessidade da aplicação (conclusão)

Requisitos	Relacional	Não Relacional		
		Orientado a Documentos	Orientado a Grafos	Chave-Valor
O Sistema possui atributos baseados em domínio, ou possuem hierarquia de derivadas.	Não Garante	Alta	Alta	Não se Aplica
Banco de dados com muitas tabelas esparsas.	Alta	Médio	Médio	Não se Aplica
Os relacionamentos entre as entidades do banco de dados utilizam de muita complexidade	Não Garante	Não se Aplica	Alta	Não se Aplica
Necessidade de garantir a persistência dos dados.	Alta	Alta	Não Garante	Alta
O espaço em disco é de extrema importância	Alta	Médio	Médio	Alta
Memória primaria é de extrema importância	Alta	Alta	Não se Aplica	Não se Aplica
O Sistema utiliza de pesquisas por identificadores como as mais importantes	Alta	Baixa	Não garante	Baixa
Utiliza o banco de dados apenas para cachear os dados	Médio	Médio	Médio	Alta

Nos requisitos que estão marcados “não se aplica”, a estrutura do banco de dados não permite trabalhar com esses requisitos, salvo em casos da utilização de outras aplicações para tratar esses requisitos e fazer com que o banco de dados trabalhe com tais requisitos, No caso dos marcados como “não garante” o banco de dados até suportaria tais requisitos, porém não garantiria um melhor desempenho.

Nos requisitos, que estão marcados Alta, significa que o banco teria um melhor desempenho nessa função devido sua estrutura, o Médio, e Baixa representa

que os bancos de dados conseguiriam realizar a determinada tarefa, porém ocorreria uma perca de desempenho.

5 Conclusão

Neste trabalho foi abordado o assunto sobre banco de dados o qual está dividido em dois paradigmas, relacional e não relacional, para tentar encontra uma melhor solução para auxiliar na escolha de um banco de dados para os jogos do tipo MMORPGs.

A impossibilidade de se apontar um banco de dados de melhor desempenho para os jogos do tipo MMORPGs, se deu pelo fato que sua estrutura está relacionada a outros fatores que estão ligados ao desempenho, o processamento do computador do usuário e a potência do servidor, a estabilidade e velocidade da rede. Somente após estes fatores, se chegaria ao desempenho do banco de dados.

A única maneira de descobrir qual banco de dados teria melhor desempenho, seria através de teste de desempenho no próprio jogo, o que se mostrou inviável, pois teria que montar toda a estrutura do MMORPGs e popular com uma quantidade significativa de usuários, e somente depois, testar o desempenho dos bancos de dados relacional e não relacional.

As dificuldades encontradas no decorrer da pesquisa foram motivadas por se tratar de um assunto que a informação é escassa, pois os bancos não relacionais surgiram há pouco tempo, e ainda, os jogos MMORPGs geralmente são softwares comerciais e não se encontra informações sobre a sua estrutura, as empresas privadas não disponibilizam a sua estrutura, ou até mesmo como eles trabalham.

Porém através das pesquisas realizadas consegue-se apresentar algumas indicações para aplicação de qual banco de dados se comportaria melhor, como está descrito na tabela 3.

Para trabalhos futuros, sugere-se uma comparação dos paradigmas utilizados neste trabalho para comparação mediante outros tipos de aplicações. Outra possibilidade seria a utilização desta pesquisa para a implementação de um jogo do tipo MMORPG para realizar comparações efetivas entre as possibilidades de bancos de dados.

REFERÊNCIAS BIBLIOGRÁFICAS

Adriano Almeida, “**Trabalhando com Relacionamentos: bancos de dados baseados em grafos e o Neo4j**”, http://blog.caelum.com.br/trabalhando-com-relacionamentos-bancos-de-dados-baseados-em-grafos-e-o-neo4j, Acesso em 01 julho. 2013.

A.J Demers, “**An evaluation of checkpoint recovery for massively multiplayer online games**”, 2009.

ANDERSON, J. Chirs; LEBNARDT, Jan; Slater Noab.**CouchDB: The DefinitiveGuide**.
1.ed. California, O’Reilly Media, Inc., 2010.exatas). Universidade Federal do Paraná, Curitiba.

Agrawal,Rakeshetal., "**The Claremont report on database research**", http://doi.acm.org/10.1145/1462571.1462573, SIGMOD Record (ACM) 37 (3): 9–19. ISSN 0163-5808, 2008

BROWNE Julian. **Brewer's CAP Theorem**, 2009. Disponível em: <http://www.julianbrowne.com/article/viewer/brewers-cap-theorem>. Acesso em: 1 novembro.2012

Barton, M , “**The history of computer RPGs: Part 2: the golden age** (1985–1993)”.
Gamasutra.,<http://www.gamasutra.com/features/20070223a/barton_02.shtml-2007> Acesso em: 18 Outubro. 2012.

Blizzard, “ **World of Warcraft reaches 10 million players**”
<http://www.worldofwarcraft.com.br/index.xml-2008>Acesso em 17 Outubro de 2012.

Blizzard, <http://us.blizzard.com/pt-br/games/wow/-2012> Acesso em: 15 Novembro 2012.

BHAT, Uma; SHRADDHA, Jadhav. **MovingTowards Non-Relational Databases**.*In*: 2010.
International Journal of Computer Applications (0975 – 8887) Volume 1 – No. 13, 2010.

BREWER, E. A. **Towards robust distributed systems**. Principles of Distributed Computing (PODC), Portland, Oregon, Julho 2000.

BRITO, Ricardo W.. **Bancos de Dados NoSQL x SGBDs Relacionais**: Análise Comparativa. Faculdade Farias Brito e Universidade de Fortaleza, 2010. Disponível em :<http://www.infobrasil.inf.br/pagina/anais-2010>. Acesso em: 16 outubro. 2012.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim; **Sistemas distribuídos**:
conceitos e projeto. Tradução por: João Tortello. 4 ed. Porto Alegre, Bookman, 2007.

Chang, Fay, Jeffrey Dean, and Sanjay Ghemawat. 2008. "**Bigtable: A distributed storage system for structured data.**" *ACM Transactions on ….* Acessado em 15,de setembro 2012 (http://dl.acm.org/citation.cfm?id=1365816).

Donald D. Chamberlin. ``**Early History of SQL**´´,2012.

Dynamo: **Amazon's Highly Available Key-Value Store. *In*: ACM SIGOPS Operating**
Systems Review, Volume 41, Edição 6, Dezembro 2007, SOSP '07, p.205–220.

Ducheneaut, Yee, Nickell, Pittman , Dickey; Smahel, Blinka, &Ledabyl"**Self-discrepancy and MMORPGs**: **testing the moderating effects of avatar identification and pathological gaming in world of warcraft**", 2010

ELOY, Leonardo Abranques de Oliveira. **O Estado da Arte em Bancos de Dados Orientados a Documentos**. 2009. 78f. Trabalho de conclusão de curso (Graduação).

Universidade de Fortaleza, Ceará.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados, 6ª edição**. Editora Pearson do Brasil, 2011.

RAMAKRISHNAN, R.; GEHRKE, J. **Sistemas de Gerenciamentos de Bancos de Dados**. 3a ed., McGraw Hill Brasil, 2008.

GIL, Antônio Carlos. **Como elaborar projetos de pesquisa**. 3º ed. São Paulo: Atlas, 1996.

HOWSTUFFWORKS. **Como funcionam os MMORPGs.** Disponível em: <http://eletronicos.hsw.uol.com.br/mmorpg.htm>. Acesso em: 10 jun. 2013.

Han, Jing, Meina Song, and Junde Song. 2011. "**A Novel Solution of Distributed Memory NoSQL Database for Cloud Computing.**" *2011 10th IEEE/ACIS International Conference on Computer and Information Science* 351–355. Disponivel em (http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6086493).

Hecht, Robin, and Stefan Jablonski. 2011. "**NoSQL evaluation: A use case oriented survey.**" *2011 International Conference on Cloud and Service Computing* 336–341. Disponivel em (http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6138544).

IMASTER. **3 razões para usar MongoDB.** Disponível em: <http://imasters.com.br/artigo/18334/mongodb/3-razoes-para-usar-mongodb/>. Acesso em: 10 abr. 2013.

J. Kim, J. Choi, D. Chang, T. Kwon, Y. Choi, eE. Yuk, "**Traffic Characteristics of a Massively Multiplayer Online Role Playing Game and Its Implications**," em Net Games, October 2009.

J. Han, H. H E and G. Le, **"Survey on NoSQL Database**," Piscataway, NJ, USA: IEEE, 2011, pp.363-3.

Korea Game Development & Promotion Institute, "**The Rise of Korean Games**", 2010.

KROENKE, David M.. **Banco de dados**: fundamentos, projeto e implementação. 6.ed.

LAKSMAN A.; MALIK P..**Cassandra - A Decentralized Structured Storage System**.
LADIS, 2009. Disponível em <http://www.cs.cornell.edu/projects/ladis2009/papers/lakshman-ladis2009.pdf >. Acesso em: 21 Setembro. 2012.

LEAVITT, Neal; **Will NoSQL Databases Live Up to Their Promise?**,Computer, vol. 43,
no. 2, p. 12-14, Fev. 2010.

Leavitt, Neal. 2010. "**Will NoSQL Databases Live Up to Their Promise?**" *Computer* 43(2):12–14. Retirado de (http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5410700).

MMORPG.com, "**Total game list**," <http://www.mmorpg.com/gamelist.cfm-2010>Acesso em: 16 outubro. 2012.

RICARTE, I. L. M., **Sistemas de Bancos de Dados Orientados a Objetos**, Faculdade de Engenharia Elétrica e de Computação – UNICAMP, Campinas, 2008.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de banco de dados**, 2008.

S. Carless. **The Activision/Blizzard Merger**: Five Key Points, 2007 Chang, Fay, Jeffrey Dean, and Sanjay Ghemawat. 2008. "Bigtable: A distributed storage system for structured data." *ACM Transactions on* …. Acessado em 4 julho, 2013 (http://dl.acm.org/citation.cfm?id=1365816).

Feijó, Bruno. 2003. "**Arquiteturas para Jogos**."

Feng, Wu-chang, David Brandt, and E V E Online. 2007. “**A Long-Term Study of a Popular MMORPG**.” 1(March 2004):19–24.

Han, Jing, Meina Song, and Junde Song. 2011. “**A Novel Solution of Distributed Memory NoSQL Database for Cloud Computing**.” *2011 10th IEEE/ACIS International Conference on Computer and Information Science* 351–355. Disponivel em (http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6086493).

Hecht, Robin, and Stefan Jablonski. 2011. “**NoSQL evaluation: A use case oriented survey**.” *2011 International Conference on Cloud and Service Computing* 336–341. (http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6138544).

Leavitt, Neal. 2010. “**Will NoSQL Databases Live Up to Their Promise?**” *Computer* 43(2):12–14.(http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5410700).

Stonebraker, Michael. 2010. “**SQL databases v. NoSQL databases**.” *Communications of the ACM* 53(4):10.(http://portal.acm.org/citation.cfm?doid=1721654.1721659).

Suznjevic, Mirko, Ivana Stupar, and Maja Matijasevic. 2011. “**MMORPG player behavior model based on player action categories.**” *2011 10th Annual Workshop on Network and Systems Support for Games* 1–6.(http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6080982).

Wei-ping, Zhu, Li Ming-xin, and Chen Huan. 2011. **“Using MongoDB to implement textbook management system instead of MySQL.”** *2011 IEEE 3rd International Conference on Communication Software and Networks* 303–305. (http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6013720).

Zhong, Zhi-Jin. 2011. “**The effects of collective MMORPG (Massively Multiplayer Online Role-Playing Games**) play on gamers’ online and offline social capital.” *Computers in Human Behavior* 27(6):2352–2363. Disponivel em (http://linkinghub.elsevier.com/retrieve/pii/S074756321100152X).