

Integração Assíncrona de Instâncias de Banco de Dados em Esquemas Homogêneos

*Marcelo Camacho de Souza – João Eduardo Ferreira
{mcsouza, jef}@ime.usp.br*

*Universidade de São Paulo
Instituto de Matemática e Estatística (IME)*

1. Introdução

Ultimamente o uso de computadores móveis está ganhando muita popularidade. O número de usuários com tais computadores está crescendo e a tendência é que isto continue no futuro, quando o número de clientes móveis deverá superar em muito o número de clientes tradicionais em ambientes corporativos. Aplicações executadas em clientes móveis requisitam informações conectando-se aleatoriamente aos repositórios de dados. Clientes móveis constituem uma nova carga de trabalho e exibem padrões de acesso diferentes daqueles vistos nos tradicionais sistemas cliente-servidor de banco de dados.

Os aplicativos disponíveis para computação móvel têm se apoiado cada vez mais em SGBD, pois necessitam do armazenamento persistente das informações e do controle da integridade relacional dos dados. Uma das possíveis abordagens encontrada nos trabalhos de Badrinath [5],[6],[7],[8],[9] considera a habilidade de tratar a desconexão de transações que são submetidas ao SGBD. Tal abordagem considera a situação onde os clientes móveis estão inaptos a se comunicarem com o conjunto de dados armazenados no SGBD. Nesta situação, o cliente móvel não tem acesso aos dados que estão localizados no SGBD central. O problema da desconexão e o controle de integridade das réplicas têm sido solucionados através de esquemas de replicação otimista, que permitem aos clientes móveis realizarem atualizações locais nos dados replicados, e posteriormente repassarem tais atualizações para o SGBD central. O problema fundamental desses esquemas é a reconciliação, isto é, a serialização das atualizações potencialmente conflitantes nas réplicas dos clientes desconectados.

O projeto SIDAM (Sistemas de Informação Distribuídos para Agentes Móveis), desenvolvido no Instituto de Matemática e Estatística da Universidade de São Paulo (USP), conta com a participação de vários pesquisadores, pertencentes as mais variadas áreas de pesquisa acadêmica e tecnológica, e tem como principal objetivo estudar os principais problemas relacionados a implementação de sistemas de informação distribuídos em ambiente de computação móvel, satisfazendo de forma eficiente as necessidades de seus usuários.

Propomos a criação de uma unidade computacional capaz de flexibilizar o controle da replicação de dados e a reconciliação das atualizações realizadas remotamente. Essa solução envolverá além dos métodos de replicação, a formatação e composição dos chamados *Objetos Réplica*, descritos na seção 4, que representam os dados replicados e suas respectivas formas de reconciliação. As réplicas podem ser atualizadas a todo momento, estando ou não o cliente conectado ao servidor central. A reconciliação dessas

atualizações, no servidor de dados central, é feita no momento da reconexão do cliente móvel, garantindo-se assim a propagação das atualizações dos dados, sem no entanto desrespeitar as regras de consistência e integridade exigidas pelos SGBDs, conforme apresentado por Ferreira e Finger [15].

Este documento está organizado da seguinte forma. Na seção 2, é descrita a motivação desse trabalho, bem como as necessidades tecnológicas existentes e os benefícios a serem alcançados. A seção 3 descreve os principais trabalhos relacionados a solução proposta. Na seção 4 é apresentada a solução proposta, descrevendo sua estrutura, *Estratégias de Replicação* e *Mecanismos de Reconciliação* a serem implementadas. Finalmente, na seção 5, descreve algumas conclusões e perspectivas sobre o trabalho.

2. Motivação

Foram realizados vários estudos utilizando-se algumas ferramentas de banco de dados móveis existentes no mercado, tais como Cloudscape [10,13] e JDataStore [11,13], desenvolvidas respectivamente pela Informix e Inprise Corporation. O intuito de tal estudo foi proporcionar um maior conhecimento das necessidades tecnológicas e seus limites, de modo a auxiliar a definição das estruturas, *Estratégias de Replicação* e *Mecanismos de Reconciliação* que compõem a solução híbrida de replicação de dados otimista proposta.

As principais necessidades que as tecnologias disponíveis desejam atender são :

- Computação fora do *firewall* : Ultimamente as organizações clássicas estão buscando aumentar a colaboração comercial de sua empresa, através da interação com os sistemas comerciais de seus parceiros. Desejam conseguir ou aumentar a relação com seus clientes, por meio de ferramentas de informação de fácil uso. E finalmente, seus funcionários estão locomovendo-se cada vez mais, e precisam do sistema para permitir que mantenham sua produtividade sempre quando estiverem desconectados da empresa.
- Dilema do cliente leve ou pesado : As aplicações cliente/servidor provêm a rica funcionalidade e alta eficiência esperada para aplicações corporativas, mas são muito complexas de serem desenvolvidas fora do *firewall*, devido a restrições como : segurança, conectividade, etc. Em contraposição, os *browsers* conectam-se a um servidor central web que provê alta conectividade, mas com limitada funcionalidade e eficiência. Para muitas aplicações que envolvem somente acesso de leitura, simples interação ou somente uso ocasional, a aplicação *browser* web é adequada. Mas para aplicações de negócio que são utilizadas por parceiros colaborativos, para aumento da intimidade com clientes, e para suporte a trabalhadores desconectados, o desafio é conciliar o melhor dos dois mundos.
- Sincronização das aplicações : Permite às aplicações receberem dados e esquemas atualizados de um computador central, e também o envio de eventos de negócio dos clientes remotos ao servidor central para processamento corporativo. Tem como característica a alta disponibilidade e escalabilidade, o aumento de segurança, a habilidade para integrar dados locais aos demais, e a possibilidade de trabalho desconectados.

Benefícios alcançados com tais necessidades :

- Acesso : Operações nos bancos de dados móveis são mais rápidas porque o tamanho do banco é menor, devido a reduzida quantidade de registros se comparado aos existentes no banco de dados central. Além disso, sua execução não compete com recursos no servidor de dados compartilhado. O DBMS está na mesma máquina da aplicação, ou, no pior caso, na mesma rede LAN, na qual tem velocidade muito maior do que uma rede WAN para acesso ao banco de dados central na origem.
- Disponibilidade : Aplicações podem executar mesmo quando o banco de dados origem ou seu acesso estão indisponíveis. Essa flexibilidade resulta em maior disponibilidade da aplicação, porque uma falha em um simples ponto não consegue afetar todo o sistema.
- Segurança : Acesso restrito apenas a subconjuntos dos dados do banco de dados origem, segundo autorização do usuário. Restrições nas conexões com o banco de dados origem. Proteção dos dados destino através de encriptação dos dados em disco.
- Balanceamento de Carga : Distribuição de operações do banco de dados central para outros computadores, mesmo com processamento desconectado, sem adicionar custo de manutenção ao sistema.
- Escalabilidade : Aplicações distribuídas podem tomar vantagem do fato da soma do poder dos computadores remotos exceder ao poder dos servidores centrais.
- Portabilidade : Fácil adaptação desses bancos de dados as existentes aplicações desenvolvidas em java. Execução transparente a aplicação cliente (mesma *Java Virtual Machine*), não necessitando de nenhum recurso adicional.
- Mobilidade / Computação Desconectada : Aplicações continuam disponíveis mesmo na ausência de conexão ao sistema corporativo. A sincronização dos dados garante a propagação das alterações realizadas ao sistema.
- Persistência dos Dados : Armazenamento em disco dos dados no cliente.

Tais necessidades tecnológicas existentes nessas ferramentas de banco de dados móveis, bem como os benefícios a serem alcançados com tais necessidades, constituem algumas das principais motivações a serem contempladas na solução proposta.

3. Trabalhos Relacionados

Recentemente muitas pesquisas tem sido direcionadas para esquemas de replicação otimistas em banco de dados móveis com reconciliação assíncrona. O problema tem sido muito estudado no contexto de sistemas de arquivo.

Em Bayou [2], [3], as réplicas são armazenadas em repositórios de dados, e não necessariamente em banco de dados. A propagação das atualizações ocorre através da reconciliação do *log*, com os repositórios de dados se comunicando por conexões *peer to peer* [16]. Além disso, não existe ordem total na execução das transações. A consistência

global dos dados não é totalmente garantida. Dependendo do grau de complexidade da reconciliação dos *log's* de cada aplicação, eventuais conflitos que não são solucionados devem ser tratados pela aplicação, pois o sistema não dispõe de tal recurso. Isso em alguns casos poderá exigir que operações contidas em um determinado *log* sejam desfeitas.

No esquema de Replicação em Duas Camadas proposto por Gray [4], a arquitetura de banco de dados é baseada no modelo Mestre / Escravo de controle de consistência [16]. Nele, todas as transações somente são validadas se forem atualizadas na cópia mestre, segundo critérios de aceitação estabelecidos pelo sistema. Esta é uma forma de consistência bastante utilizada comercialmente, mas não atende a forma como é desejado o controle de consistência da solução proposta na seção 5. Possivelmente poderá ser utilizado como uma das opções do modelo de consistência na solução a ser projetada.

Em Dataman [5], [6], [7], [8], [9] é proposto um modelo de banco de dados relacional mais maduro e não requer que o cliente mantenha uma cópia de toda a relação ou banco de dados, ao contrário do que ocorre em BAYOU. Além disto, a resolução de conflitos é transparente para a aplicação. O sistema pode ser especialmente projetado para prover automaticamente a resolução de conflitos.

A solução proposta, que será apresentada na seção 4, se assemelha mais ao proposto em Dataman. Estamos criando o conceito de *Objeto Réplica*, que representa o conjunto de dados a serem replicados nas unidades móveis, e contém referências a forma de reconciliação da réplica no banco de dados. Será utilizado o esquema otimista para tratar os problemas de consistência, bem como diversos modelos de reconciliação a serem escolhidos no momento da criação do *Objeto Réplica*. Tal flexibilidade resulta em um modelo dinâmico de escolha de modelos de reconciliação, permitindo de maneira fácil a introdução, a posteriori, de novos modelos ao sistema. Essa característica, não observada nos esquemas e modelos estudados acima, é a grande motivação dessa solução proposta.

4. Solução

A solução proposta trata a integração assíncrona de instâncias de banco de dados em esquemas homogêneos. Consideremos diferentes instâncias de um banco de dados distribuídas por diversos nós. Inicialmente, utilizaremos o mesmo esquema homogêneo em todas as instâncias do banco de dados para reduzir o grau de complexidade envolvido. É desejável que tais nós trabalhem com operações desconectadas, resultando em um modelo assíncrono de replicação de dados. A atualização das réplicas é sempre permitida, mesmo estando o nó desconectado do sistema. Devido a essa característica, o modelo de replicação adotado é o otimista, pois utiliza consistência fraca e necessita que eventuais conflitos resultantes de atualizações realizadas nas réplicas, sejam resolvidos na reconexão dos nós móveis ao sistema. Com isso, a idéia é proporcionar alta disponibilidade de uso dos dados aos nós, mesmo estando estes trabalhando de forma desconectada.

A seguir encontram-se relacionamos os aspectos da replicação de dados que podem ser utilizados como uma classificação geral das várias alternativas de replicação de dados. Detalhes das formas de replicação podem ser encontradas em Abdelsalam [16].

Visibilidade dos Dados :

- *Global* : todas as alterações realizadas sobre dados devem ser propagadas a

todas as instâncias do banco de dados que armazenam as réplicas desses dados. Tal opção visa proporcionar uma completa disponibilização das atualizações de dados em todo o sistema. O custo relacionado ao processo de propagação é alto, pois mensagens devem ser trocadas entre todas as instâncias do banco de dados, toda vez que inserções e alterações forem realizadas.

- *Parcial* : não há obrigatoriedade que as atualizações de dados sejam propagadas a todas as instâncias de banco de dados que armazenam as réplicas dos dados. O custo envolvido na propagação das atualizações é bem menor que o existente no modelo de visibilidade global, pois somente a instância global de banco de dados responsável pela sincronização das atualizações será atualizada.

Arquitetura de Sincronização do Banco de Dados :

- *Central* : arquitetura muito utilizada comercialmente. Um nó central é responsável pela sincronização das atualizações realizadas nas instâncias de banco de dados. Tal nó está sempre apto a receber as conexões dos nós móveis. Dependendo do modelo de replicação adotado, o nó central será responsável também pela propagação das atualizações dos dados às demais instâncias de banco de dados envolvidas. Se desejado, armazenará a instância global do banco de dados, pois recebe todas as atualizações realizadas nas demais instâncias de banco de dados do sistema.
- *Distribuída* : considera que todos os nós tem o mesmo poder para atualizações de dados, devendo estas serem repassadas às demais instâncias de banco de dados nos momentos da reconexão dos nós móveis. Tal esquema permite grande disponibilidade de atualização, mas em contra-partida, o custo envolvido para sincronizar as atualizações das réplicas é considerável.

Critério para Replicação :

- *Nenhum* : todos os dados deverão ser replicados em todas as réplicas existentes no sistema. Tal situação resulta em excesso de mensagens de rede, necessárias para a propagação das atualizações, e custo adicional para controle de atualizações conflitantes. Num ambiente assíncrono de replicação, tal proposta não é desejada, devido a baixa e instável banda de comunicação dos nós móveis. Uma vez que todas as atualizações sejam propagadas, cada réplica representará o mesmo estado global consistente do banco de dados do sistema.
- *Predicado* : são critérios que realizam a classificação e escolha dos dados a serem replicados nas diversas instâncias de banco de dados existentes no sistema. A diferenciação entre os dados existentes em tais instâncias é resultado de um predicado, as quais definem as características comuns aos dados pertencentes a uma mesma instância de banco de dados. Diferentes predicados podem ser usados para definir os dados de uma instância. Assim, um mesmo predicado pode ser usado em instâncias de banco de dados diferentes. Caso as atualizações conflitantes entre dados pertencentes a instâncias distintas de banco de dados não sejam propagadas pelo sistema, teremos um estado global inconsistente do banco de dados.

O conceito de localidade está muito relacionado à computação móvel, pois tem-se uma previsão do conjunto de dados envolvidos nas próximas interações do nó móvel no sistema. Por exemplo, ao visitar uma cidade um vendedor sabe antecipadamente quais

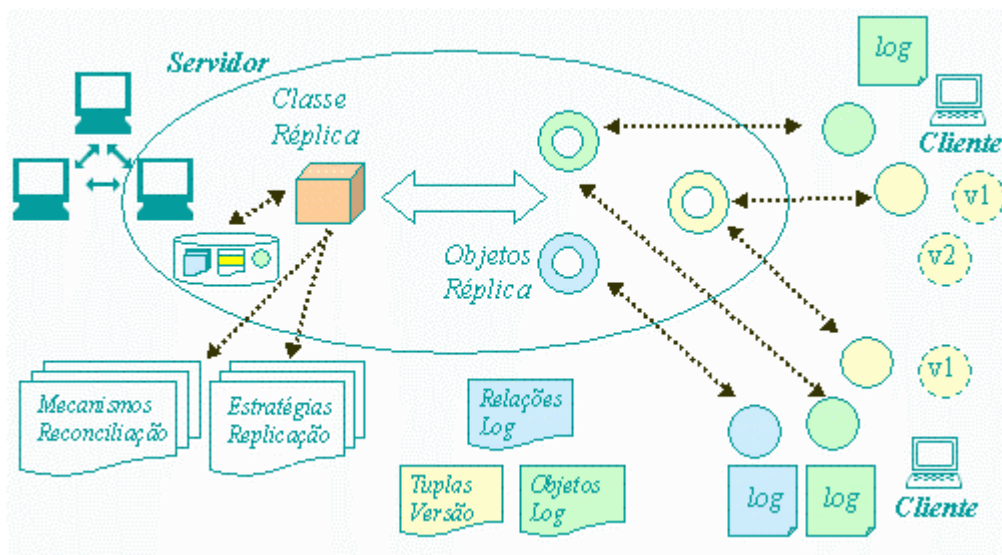
são os possíveis clientes e fornecedores envolvidos nas próximas transações comerciais a serem realizadas. Assim, um bom predicado de replicação a ser adotado é a localização do vendedor. Dessa maneira, prevendo-se as cidades que farão parte do roteiro de viagem, o conjunto de dados relacionados as possíveis transações comerciais serão replicados em sua instância de banco de dados local.

Dependendo dos predicados escolhidos para a escolha dos objetos a serem replicados nas instâncias de banco de dados, teremos dois tipos possíveis de réplicas. São elas :

- *Réplicas Disjuntas* : uma réplica pode ser considerada disjunta quando o conjunto de dados representados por ela não participa de nenhuma outra réplica que não seja cópia dela própria, isto é, o resultado do predicado usado para a seleção do conjunto de dados representados pela réplica, não coincide com os demais conjuntos de dados selecionados pelos outros predicados. Assim um mesmo conjunto de dados só poderá ser representado por réplicas que utilizaram um mesmo predicado. Em algumas situações, dependendo do predicado utilizado, somente teremos a ocorrência da réplica em uma única instância do banco de dados. Neste caso, teremos a forma mais simplificada de atualização de dados, pois como não ocorrem conflitos de atualização, não é necessário a utilização de *Mecanismos de Reconciliação*. É desejado, quando tivermos várias réplicas disjuntas de um dado no sistema, realizar a propagação das atualizações as demais réplicas.
- *Réplicas Sobrepostas* : nesta, diferentes predicados selecionam conjuntos de dados que apresentam partes comuns, isto é, existem intersecções não vazias nos conjuntos de dados selecionados pelos diferentes predicados. Assim os dados representados pelas réplicas que utilizaram um mesmo predicado, podem fazer parte do conjunto de dados representado por outra réplica que utilizou um outro predicado para seleção de dados. Atualizações realizadas sobre réplicas sobrepostas, devem ser sincronizadas para manter o banco de dados no estado consistente.

Nesse trabalho realizamos a integração assíncrona de instâncias de banco de dados que utilizam esquemas homogêneos em sua estrutura. Estaremos utilizando para isso predicados de replicação, para a criação de réplicas nas instâncias de banco de dados. É desejado que a solução apresentada utilize a visibilidade parcial. Predicados serão utilizados produzindo apenas réplicas disjuntas em diferentes instâncias do banco de dados. A arquitetura de sincronização do banco de dados escolhida será a central, ao invés da distribuída. O intuito dessa escolha é simplificar a solução proposta, restringindo o controle de reconciliação a um único nó.

Na figura abaixo apresentamos de maneira genérica os conceitos envolvidos na solução proposta, que serão descritos detalhadamente nas seções seguintes.



4.1 Estrutura da Classe / Objeto Réplica

Para tratar o banco de dados como uma “massa” de dados que pode ser dividida criou-se um novo conceito, que define como agrupar os dados envolvendo-os em unidades referenciáveis, denominadas *Classe / Objeto Réplica*.

A *Classe / Objeto Réplica* é capaz de encapsular um conjunto de dados. Esse conjunto de dados pode ser apenas uma instância de um determinado dado, uma coleção de tuplas ou colunas de uma relação (tabela), ou um objeto complexo no paradigma de orientação a objetos.

Conforme definido no item anterior, estaremos utilizando a arquitetura centralizada para sincronização do banco de dados. No nó central estará localizada a instância global do banco de dados do sistema, isto é, a instância do banco de dados que conterá todos os dados existentes no sistema. As instâncias do banco de dados localizados nos nós móveis conterão apenas os dados replicados, que foram selecionados conforme os critérios de escolha especificados pelos predicado. A instância global do banco de dados sofrerá o reflexo de todas as operações realizadas nas demais instâncias do banco de dados. O intuito de termos essa instância global é facilitar eventuais pesquisas e processamentos, que necessitem de todos os dados do sistema.

Segue abaixo a descrição dos elementos que irão compor a unidade computacional criada na solução proposta :

- *Classe Réplica* : representa o *meta-modelo* para a criação das réplicas, incluindo a estrutura de representação dos dados do banco, e a implementação das *Estratégias de Replicação* e dos *Mecanismos de Reconciliação*. Em todo o sistema existe apenas uma única *Classe Réplica*, localizada no nó central da arquitetura de sincronização, que utilizará a instância global do banco de dados, também localizada no nó central, para controle e gerência das réplicas.

A principal funcionalidade da *Classe Réplica* é criar os *Objetos Réplica*, que serão armazenados localmente no nó central. Para cada *Objeto Réplica* será gerado um *identificador da réplica*, que será transmitido juntamente com os dados representados no *Objeto Réplica*, com o intuito de recuperar a referência desses dados ao *Objeto Réplica* responsável pela sincronização de suas atualizações.

- *Objeto Réplica* : cada “dado” a ser replicado nos nós móveis possuirá seu respectivo

Objeto Réplica, que armazenará todas as informações necessárias para realizar a sincronização das atualizações realizadas sobre as réplicas, utilizando o *meta-modelo* fornecido pela *Classe Réplica*. Fazem parte dessas informações : a referência à *Estratégia de Replicação* e ao *Mecanismo de Reconciliação* utilizados, a composição da réplica (representada pelo conjunto de dados encapsulados), a localização dos componentes da réplica no banco de dados global, a identificação da réplica. De acordo com a *Estratégia de Replicação*, as referências de localização dos nós móveis também deverão ser armazenadas. Pois se a visibilidade desejada for a global, as atualizações sofridas por uma réplica deverão ser repassadas as demais instâncias de banco de dados possuidoras da réplica.

- *Extração / Integração*, um serviço que tem a responsabilidade de separar e integrar os dados replicados nos nós móveis do sistema, conforme certas necessidades. Por exemplo, a redefinição dinâmica dos *Objetos Réplica*, visando a modificação do conjunto de dados representados pela réplica, e troca do *Mecanismo de Reconciliação* e da *Estratégia de Replicação* utilizada.

4.2 Estratégias de Replicação

Devido ao modelo assíncrono de integração de réplicas, as estratégias de replicação a serem utilizadas serão aquelas relacionadas com a consistência fraca. Estaremos propondo o uso de vários níveis de consistência fraca, pois é desejado que a solução proposta apresente flexibilidade na escolha de tais estratégias.

Dependendo da visibilidade dos dados e dos tipos de réplicas existentes, vislumbramos inicialmente apenas as seguintes variações das estratégias de replicação. Novas variações de estratégias de replicação poderão ser adicionadas posteriormente a solução.

Estratégia 1 : Réplicas disjuntas e visibilidade parcial

Nessa estratégia, as réplicas existentes nas instâncias do banco de dados são disjuntas, isto é, os predicados utilizados pelas réplicas selecionam conjuntos de dados que não apresentam dados comuns. A ocorrência de mais de uma réplica dependerá da necessidade de alocação de dados.

A visibilidade das réplicas é a parcial, o que resulta na não propagação das atualizações às demais réplicas do dado em questão. Nos casos onde tenhamos diversas réplicas do mesmo conjunto de dados no sistema, faz-se necessário a existência do nó central, que tratará de garantir o controle global sobre a atualização das réplicas. Assim, apenas a instância global do banco de dados, localizado no nó central, seria atualizada após a propagação da atualização realizada na réplica pelo nó móvel.

Essa estratégia poderá ser utilizada principalmente nos casos onde os predicados de replicação produzam réplicas totalmente disjuntas, não necessitando que suas atualizações sejam propagadas às demais réplicas eventualmente existentes. Tal estratégia ressalta a característica de localidade do dado e apresenta o menor custo relacionado à propagação de atualizações. Entretanto, apresenta certo grau de inconsistência nas demais réplicas do dado, pois como a propagação das atualizações não se faz necessária, eventuais dados desatualizados estarão sendo utilizados nos demais nós móveis. Tal situação é aceitável dependendo do dado e aplicação envolvida. No entanto, é necessário em algum momento, atualizar as réplicas com os valores da instância mais atualizada. Tal processo pode ocorrer com certa periodicidade, em

momentos pré-estabelecidos e controlados que não representem grandes ônus a conexão do nó móvel.

Estratégia 2 : Réplicas disjuntas e visibilidade global

Nesta estratégia as réplicas são disjuntas, mas a visibilidade é global. Assim, as atualizações deverão ser propagadas às demais réplicas do dado existentes no sistema. Aqui, o nó central, após realizar a reconciliação das atualizações em sua instância global do banco de dados, tratará de propagar as alterações realizadas às demais réplicas do dado. Para isso, aguardará a reconexão dos nós móveis que armazenam as réplicas envolvidas, e repassará informações necessárias para a atualização das réplicas.

O custo da propagação relacionado a tal estratégia é alto, devido a necessidade da realização da sincronização das atualizações para as demais réplicas do objeto. Essa constitui uma característica importante, pois garante que após serem realizadas todas as propagações de atualização, as instâncias do banco de dados existentes no sistema estarão apresentando o mesmo estado global consistente. O que não é observado na estratégia anterior, pois mesmo após propagar a atualização ao nó central, teremos ainda instâncias do banco dados inconsistentes em alguns nós móveis do sistema.

Estratégia 3 : Réplicas sobrepostas e visibilidade parcial

Nessa estratégia as réplicas existentes em diferentes instâncias do banco de dados são sobrepostas, isto é, permitem atualizações de forma concorrente sobre o mesmo conjunto de dados, o que resulta na possível ocorrência de mais um nível de atualizações conflitantes no momento da reconciliação. Nessa situação além da necessidade de reconciliação entre as réplicas de um mesmo *Objeto Réplica*, deveremos também repassar as atualizações para outros *Objetos Réplica* que representem os dados comuns.

Aqui, a visibilidade utilizada é a parcial, o que significa que as modificações deverão ser sincronizadas porém sem serem propagadas as demais as réplicas do dado. O nó central realizará a reconciliação das atualizações provindas das diferentes réplicas e tratará de atualizar sua instância global do banco de dados, que será a única instância do banco a receber as atualizações de todas as réplicas. Devido a visibilidade parcial, as demais instâncias do banco de dados poderão utilizar dados desatualizados, que foram modificados em outras réplicas do dado. Assim, o banco de dados composto pelas instâncias de banco de dados localizadas nos nós móveis terá um estado inconsistente, ao contrário da instância global localizada no nó central, na qual será a única instância consistente em todo o sistema.

O banco de dados aqui é decorrente da utilização de predicados de replicação que selecionam conjuntos de dados comuns, em diferentes instâncias do banco de dados.

Estratégia 4 : Réplicas sobrepostas e visibilidade global

Nesta estratégia, o nó central se encarregará de receber as atualizações realizadas nos nós móveis, atualizará sua instância global do banco de dados, aplicando o *Mecanismo de Reconciliação* (versão de dados e *log*) escolhido, e tratará de propagar as atualizações as demais réplicas, assim que os nós móveis, nas quais as armazenam, se reconectem ao sistema. A ocorrência de atualizações conflitantes é alta, devido as réplicas serem sobrepostas. Eventuais conflitos resultantes da reconciliação devem ser tratados, e possíveis notificações enviadas aos nós móveis, informando o sucesso ou falha na propagação das modificações. Caso a atualização não tenha obtido sucesso na

sincronização, as alterações realizadas pelo nó móvel deverão ser desfeitas. Não é desejada a ocorrência de tal situação, mas dependendo por exemplo do tempo de desconexão do nó móvel, torna-se impraticável a sincronização de tais modificações.

O custo relacionado a propagação das atualizações é alto, pois todas as réplicas do dado deverão ser atualizadas assim que o nó central sincronize a atualização em sua instância global do banco de dados.

As estratégias podem ser ilustradas na seguinte tabela :

Estratégias de Replicação	Visibilidade dos Dados		Tipos de Réplicas	
	Parcial	Global	Disjuntas	Sobrepostas
Estratégia 1	X		X	
Estratégia 2		X	X	
Estratégia 3	X			X
Estratégia 4		X		X

A existência de tal variedade de estratégias é fruto das diversas combinações existentes das abordagens de visibilidade de réplica, arquitetura de sincronização e tipo de réplica.

A mudança na escolha da estratégia de replicação do objeto está relacionada a evolução do objeto, nas instâncias do banco de dados. Por exemplo, se réplicas disjuntas tornarem-se sobrepostas, o sistema tratará de se adaptar a nova condição, através do serviço *Extração / Integração* que redefinirá o *Objeto Réplica*, modificando o conjunto de dados representados por ele.

Não é desejado a ocorrência de réplicas sobrepostas no sistema. Possíveis soluções para evitar essa situação seriam :

- A união dos conjuntos de dados resultantes dos predicados utilizados que resultaram nas réplicas sobrepostas, em um único conjunto de dados representado por um novo *Objeto Réplica*.
- A separação do conjunto de dados formado pela união dos resultados sobrepostos dos predicados, em novos *Objetos Réplica*, de forma a criarem réplicas disjuntas.

Como todas essas situações estão previstas na unidade computacional proposta, a migração de uma estratégia para outra é totalmente adaptável. Essa é uma importante característica, pois constitui um modelo dinâmico de uso das estratégias de replicação de dados e alteração do conjunto de dados a serem replicados. Embora haja mais combinações, nesse trabalho iremos nos restringir apenas as quatro estratégias descritas no item 4.2, pois podem ser consideradas como formas normais de distribuição. Outras contribuições podem ser viabilizadas a partir das descritas neste trabalho.

4.3 Mecanismos de Reconciliação

Os *Mecanismos de Reconciliação* estão relacionados a forma como será realizada a sincronização das atualizações ocorridas nas réplicas durante seus períodos de desconexão. As instâncias do banco de dados serão atualizadas conforme a maneira definida pela estratégia de replicação escolhida, isto é, se as alterações sincronizadas no nó central deverão ser propagadas às demais réplicas do dado. Dependendo da estratégia, poderemos escolher o mecanismo mais adaptável e interessante no momento da integração.

A escolha do *Mecanismo de Reconciliação* ocorre no momento da criação da réplica. Uma vez definido o *Mecanismo de Reconciliação* e a *Estratégia de Replicação* envolvida, as réplicas tratarão de registrar todas as ações realizadas, conforme o modo escolhido, e os *Objetos Réplica*, realizarão a reconciliação utilizando os registros das alterações.

Utilizaremos inicialmente as duas formas de *Mecanismos de Reconciliação* abaixo descritas. Posteriormente novos *Mecanismos de Reconciliação* poderão ser adicionados à unidade computacional proposta. Esta é uma característica desejada nessa solução apresentada.

- *Versão de Dados* : Dependendo da *Estratégia de Replicação* escolhida, poderemos optar pelo método de multiversão de réplicas. Neste, as atualizações realizadas nos nós móveis, em seus períodos de desconexão, são representadas por versões de dados associadas as modificações ocorridas nas réplicas. Vários trabalhos [7] têm utilizado tal esquema de reconciliação de atualizações. Estudos realizados demonstram que sua utilização aumenta a probabilidade de sucesso da reconciliação das atualizações, reduzindo assim a ocorrência de conflitos no momento da reconciliação das réplicas.

As réplicas, localizadas nos nós móveis, tratarão de registrar as ações realizadas sobre seus dados. Na reconexão do nó móvel, tais registros deverão ser repassados ao *Objeto Réplica* responsável pela réplica em questão, para a realização da sincronização das atualizações. Algoritmos de multiversão tratarão os registros e decidirão se a propagação das modificações obteve sucesso ou falha. Os nós móveis receberão as notificações das tentativas da propagação realizadas.

- *Arquivo de Log* : Forma muito utilizada comercialmente pelos SGBDs. Representam o registro das transações ocorridas no banco de dados, com o intuito de garantir a integridade dos dados contra eventuais problemas que venham ocorrer. Ocasionalmente problemas com o banco de dados que resultem na inutilização de seus registros, podem ser corrigidos através da restauração da situação anterior do banco de dados, por meio de recuperação de um arquivo gravado em fita (*backup*), e re-execução do *log* de transações para a recuperação do estado do banco de dados imediatamente anterior ao momento de sua falha.

As réplicas, localizadas nos nós móveis, registrarão em seu arquivo de *log* local, a ocorrência das atualizações sofridas. Na reconexão ao nó central, tais registros serão repassados ao *Objeto Réplica* responsável pela sincronização das atualizações da réplica, que tratará de executar as ações registradas no banco de dados global. Eventuais conflitos decorrentes de atualizações serão tratados e notificações propagadas aos nós móveis envolvidos.

Algoritmos de reconciliação de logs deverão ser implementados, para realizem da melhor forma possível a reconciliação das atualizações por meio de arquivos de *log*.

Bayou [2], [3] propõe de forma análoga, o registro de atualizações em logs para posterior propagação das modificações aos demais repositórios de dados. Tal proposta, juntamente com a utilizada nos SGBDs, serão consideradas para a definição da estruturação do arquivo de *log* e para a implementação dos algoritmos necessários a este *Mecanismo de Reconciliação*.

5. Conclusões e Perspectivas

Os sistemas de arquivo tem sido modificados para permitir que clientes façam o *download* de informações, se desconectem, e depois se reintegrem. Já os banco de

dados não tem sido reprojatados para acomodar os clientes móveis. Assim, existe a necessidade de suportar tais clientes em banco de dados.

Neste documento descrevemos as motivações desse trabalho, bem como as necessidades tecnológicas existentes e os benefícios a serem alcançados com essa solução. Uma breve descrição da estrutura dos elementos que compõem a solução foi apresentada, baseando-se nas preocupações e anseios existentes atualmente.

As principais necessidades apresentadas nesse trabalho apontam para uma solução de *flexibilização das estratégias* de controle de réplica baseada na *visibilidade dos dados* e na *arquitetura de sincronização*. A escolha de uma determinada estratégia está intimamente vinculada com as formas de reconciliação que classicamente são conhecidas como *log* e versão de dados, conforme utilizadas em BAYOU [2], [3] e Dataman [5], [6], [7], [8], [9] respectivamente. Entretanto, há necessidade de *especializar o controle de réplica* no ambiente corporativo de dados, fazendo com que as soluções coexistam. O uso do conceito de desconexão de dados e transações, nos leva a criar uma unidade computacional que chamamos de *Classe / Objeto Réplica*, capaz de flexibilizar o controle das transações assíncronas envolvidas. Esse conceito também permite contemplar duas soluções clássicas para reconciliação : arquivo de *log* e versão de dados, que não são utilizadas de modo único, permitindo ao usuário o controle do dado de acordo com sua aplicação.

Referências

- [1] B. R. Badrinath and K. Ramamritham, Performance evaluation of semantics-based multilevel concurrency control protocols, ACM SIGMOD International Conference on Management of Data (37/206), pages 163-172, May 1990
- [2] A. Demers, K. Petersen, M. Spreitzer, D. Terry, M. Theimer and B. Welch, The Bayou architecture : Support for data sharing among mobile users, Proc. of the IEEE Workshop on Mobile Computing and Applications, pages 2-7, Dec. 1994
- [3] K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, A. J. Demers. Flexible update propagation for weakly consistent replication. Proceedings of the 16th ACM Symposium on Operation Systems Principles, Saint Malo, França, páginas 288-301, Outubro 1997
- [4] J. Gray, P. Helland, P. E. O'Neil and D. Shasha. The dangers of replication and a solution. Proceedings of ACM SIGMOD, pages 173-182, June 1996
- [5] S. H. Phatak. and B. R. Badrinath, Database server organization for handling mobile clients, Department of Computer Science Technical report DCS-TR-324, Department of Computer Science, Rutgers University, New Jersey, 1997
- [6] S. H. Phatak and B. R. Badrinath, An architecture for mobile databases, Department of Computer Science Technical report DCS-TR-351, Department of Computer Science, Rutgers University, New Jersey, 1998
- [7] S. H. Phatak and B. R. Badrinath, Multiversion reconciliation for mobile databases, Proc. of International Conference on Data Engineering (ICDE), Sydney, Australia, pages 582-589, Mar.'99.

- [8] S. H. Phatak and B. R. Badrinath, Data partitioning for disconnected client server databases, Proc. of International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'99), Seattle, Washington, Aug.'99.
- [9] S. H. Phatak and B. R. Badrinath, Conflict resolution and reconciliation in disconnected databases, Proc. of Mobility in Databases and Distributed Systems (MDDS), Florence, Italy, Sep.'99.
- [10] www.cloudscape.com, Cloudscape, Inc., Informix, Inc.
- [11] www.jdatastore.com, Borland[®], JdataStore[™] versão 3.5, Inprise Corporation
- [12] Richard Lenz, The “Virtual-Primary-Copy-Approach” compared to other approaches with weak consistent data replication, University of Erlangen – Nuremberg, 1996
- [13] Kistler, J., and Satyanarayanan, M. Disconnected operation in the Coda file system. Em *ACM Transactions on Computer Systems*, Vol. 10, No.1, páginas 3-25, fevereiro de 1992.
- [14] R. Guy, J. Heidemann, W. Mak, T. Page Jr., G. Popek, D. RotyMeier. Implementation of the Ficus Replicated File System. Em *The Proceedings of the Summer USENIX Conference*, páginas 63-71, Anaheim, CA, junho de 1990.
- [15] João Eduardo Ferreira and Marcelo Finger, Controle de Concorrência e Distribuição de Dados : a teoria clássica, suas limitações e extensões modernas, 12^a Escola de Computação, IME – USP, 2000
- [16] Abdelsalam A. Helal, Abdelsalam A. Heddaya and Bharat B. Bhargava, Replication Techniques in Distributed Systems, Kluwer Academic Publishers, 1996