UNIVERSIDADE EDUARDO MONDLANE
FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

TRABALHO DE LICENCIATURA

REPLICAÇÃO ENTRE BASES DE DADOS ORACLE E OS MÉTODOS DE RESOLUÇÃO DE CONFLITOS

Caso Prático da MOZAL

Por Alexandre Alfredo Cumbe

Agosto, 2002

UNIVERSIDADE EDUARDO MONDLANE
FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

TRABALHO DE LICENCIATURA

Tema:

REPLICAÇÃO ENTRE BASES DE DADOS ORACLE E OS MÉTODOS DE RESOLUÇÃO DE CONFLITOS

Caso Prático da MOZAL

Estudante: Alexandre Alfredo Cumbe
Supervisor: Eng. Fernando Rafael J. Comolo

Maputo, Agosto, 2002

DEDICATÓRIA

Dedico este trabalho aos meus pais que me trouxeram ao mundo, a minha esposa e a minha filha pela compreensão durante longas horas de estudo.

AGRADECIMENTOS

Ao dr. Fernando Rafael Comolo, pela disposição e dedicação na supervisão desta tese.

Os meus agradecimentos estendem-se para os funcionários da biblioteca do DMI, MOZAL S.A.R.L., Oracle Corporation (África do Sul) e para aquelas pessoas que directa ou indirectamente contribuíram para o efectivação do trabalho.

Alexandre Alfredo Cumbe

DECLARAÇÃO DE HONRA

Declaro por minha honra que este trabalho é fruto da minha própria investigação, e que o mesmo foi realizado para ser submetido como trabalho de **licenciatura em informática** na Universidade Eduardo Mondlane.

Alexandre Alfredo Cumbe

Maputo, Maio de 2002

RESUMO

A MOZAL está dotada de uma infraestrutura de tecnologias e sistemas de informação bastante complexa e moderna para responder a sua necessidade de processamento de informação para a gestão dos seus processos de produção de alumínio. Vários servidores de bases de dados estão instalados nos vários locais da empresa trocando entre sí dados através da rede dos computadores.

Esta troca de dados é garantida pelos mecanismos de replicação de dados providenciados pelos Sistemas de Gestão de Bases de Dados instalados em cada servidor. A Oracle apresenta diferentes soluções para a replicação de dados que são apresentadas ao longo de trabalho. Replicação é o processo de cópia e manutenção de objectos de um esquema de base de dado para outros esquema em múltiplas bases de dados.

O ambiente de replicação é mais típico em transmissão de dados assíncrona, neste caso prováveis conflitos relacionados podem ocorrer, sendo assim, necessário tomar respectivos métodos de resolução dos conflitos. A importância da resolução de conflitos reside no facto de garantir a consistência de dados. Uma vez detectado um conflito, a sua resolução deve ser breve de forma a evitar o efeito da inconsistência em cascata, ou seja conflitos podem gerar outros conflitos e por aí em diante.

Os sistemas de Informação da MOZAL estão divididos em níveis, conforme as sua funcionalidades, nomeadamente sistema do nível 0, I, II, III. Esta classificação é de acordo com convenção da própria Mozal usada no grupo das empresas de fundição de metais pertecentes à Billiton. Neste trabalho são apresentados as infraestruturas de suporte para cada nível de sistemas e os interface entre os sistemas do mesmo nível e de níveis difentes. É nalguns destes interfaces que estão implementados os mecanismos de replicação de dados.

Este trabalho faz a avaliação da aquetectura ou modelo dos mecanismos de replicação e métodos de resolução dos conflitos implementados nos interface entre os sistemas que fazem a propagação de dados e propomos a alteração dos mecanismos por mais adequados.

Índice

Conteúdo **Pág.**

I. INTRODUÇÃO

1. DESCRIÇÃO DO PROBLEMA

Uma base de dados é uma colecção de dados duma organização (Loney, 1995). Esta colecção pode ser compartilhada ou integrada. Uma organização, e para o seu melhor desempenho das suas funções tem como um dos recursos importantes os dados, que são a base para informação.

A ideia de criação de uma base de dados é de permitir que cada sector duma organização aceda facilmente aos dados da organização de uma forma compartilhada. Para que este acesso seja disciplinado é importante e necessária a introdução de um sistema de gestão de base de dados (SGBD).
Geralmente a implementação de uma base de dados tem algumas implicações em relação aos aspectos de integridade, segurança e consistência dos dados.

Um sistema completo de gestão de base de dados, além de permitir um acesso concorrido e compartilhado dos dados armazenados, que é a sua principal função, deverá necessariamente proporcionar outras facilidades e funções adicionais que permitam a organização ganhar uma vantagem competitiva, aumentando a qualidade do serviço prestado.

Uma das funções adicionais não menos importante que o SGBD deve proporcionar relaciona-se com o controlo e protecção dos dados. Aqui, se entende que o sistema deverá reunir recursos e técnicas capazes de proteger os dados contra perigos diversos (deliberados ou acidentais), á vários níveis, que incluam o controle da recuperação, o uso concorrido do acesso, consistência, segurança e integridade dos dados.

Com o desenvolvimento das tecnologias de informação, as tendências de globalização e concorrência dos mercados mundiais, a localização geográfica das organizações em relação aos clientes constitui uma das fortes razões para o desenvolvimento de SGBD distribuídos ou centralizados mas com interfaces no mundo. É neste contexto que o SGBD ORACLE introduziu um mecanismo de replicação de dados entre bases de dados. A replicação não é mais do que uma cópia de determinados dados de uma

base de dados para outra na hora exacta(Delmolino, 1995). Porém esta cópia deve ser fiel de forma a garantir a integridade dos dados. Com a eventualidade de dados poderem ser inseridos, apagados ou actualizados de forma não apropriada surgem no entanto os métodos de resolução de conflitos.

É neste contexto que a implementação de tipos de replicação em oracle, requer estudos cuidadosos, pois nele se encontram envolvidos recursos. Este estudo deve equacionar de forma equilibrada todos recursos e infra-estruturas envolvidos.
Este trabalho concentrar-se-á na técnica e nos diferentes tipos de replicação bem como nos métodos de resolução de conflitos a eles associados implementados na MOZAL.

2. OBJECTIVOS

2.1. Objectivo Geral

O objectivo geral deste trabalho é estudar a problemática sobre a replicação de dados entre bases de dados oracle bem como os métodos de resolução de conflitos derivados da replicação na MOZAL.

2.2. Objectivos Específicos

- Fazer um estudo teórico sobre os diferentes métodos de resolução de conflitos no âmbito da replicação dos dados.
- Identificar problemas e aspectos que afectam a replicação de dados entre sistemas de gestão de base de dados ORACLE.
- Apresentar o modelo de o modelo de Replicação de Dados e Resolução de Conflitos da Mozal.
- Avaliar o modelo de Replicação de Dados e Resolução de Conflitos da Mozal.

3. METODOLOGIA DO TRABALHO

O presente trabalho será descritivo e explicativo acompanhado de exemplos (pequenos programas e sintaxes de comandos) que representarão a implementação do objecto em estudo. Será também baseado na descrição e análise da situação real existente na MOZAL S.A.R.L na qual existem 8 bases de dados ORACLE, versão

8.0.5.1.0 pertencentes a diferentes níveis de produção e que trocam dados entre eles usando a replicação .

Fará parte da metodologia também a consulta de documentos de instalação, operação e manutenção bem como bibliografias ligadas á SGBD.

Este trabalho é composto por cinco capítulos, sendo que o primeiro apresenta a motivação na origem do trabalho e a oportunidade deste estudo na resolução do problema de replicação de dados na Mozal. O segundo capítulo apresenta a teoria da replicação dando ênfase nos conceitos, parâmetros de configuração e os diversos tipos de replicação em Oracle, dá também os métodos de resolução no caso de ocorrência de conflitos da replicação. No capítulo três descreve-se a configuração dos sistemas da Mozal, os seus níveis e interfaces entre estes, com destaque aos interfaces com recurso a replicação através da oracle e o modelo ideal para a replicação. No capítulo quatro é feita a avaliação da aplicabilidade do modelo de replicação proposto para os sistemas da Mozal. Por último, o capítulo cinco apresenta as conclusões e recomendações do trabalho.

4. BREVE HISTORIAL

Depois do surgimento do sistema de gestão de bases de dados em oracle, e de outros sistemas de gestão de bases de dados, a troca de informação entre diferentes sistemas foi sempre um problema por resolver. Com sistemas ligados em rede, houve a necessidade de se evitar a intervenção do homem no transporte da informação em medias, devido por um lado a volatilidade da informação, confidencialidade e insegurança que os medias (periféricos) apresentam, e por outro lado custos e tempo que uma determinada informação levaria para ser enviada de um ponto para o outro.
Antes do surgimento do sistema de replicação em oracle, vários utilitários automáticos foram e ainda são usados para a transferência da informação de uma base de dado oracle para outra, a saber:

4.1. Exportação e importação

Um utilitário sempre presente desde a introdução da bases de dados oracle. Este utilitário permite fazer a transferência de dados de uma base de dados oracle para a

outra com a mesma versão, ou de uma inferior para uma superior (Gosset, Jang, 1999). A exportação cria um ficheiro compactado que é descodificado e executado na importação.

Este método apresenta algumas desvantagens :

- Tem que ser feita para toda a tabela e nunca uma parte dos dados da mesma.
- Os dados têm que ser importados para serem visíveis.
- Qualquer alteração durante e após a exportação não será visível depois da importação, o que resulta em inconsistência dos dados da base de dados.

4.2. Criação da tabela como selecção

Este método cria uma nova tabela usando dados e estrutura seleccionados duma tabela existente. Qualquer "constraint" e outras formas de integridade de referência não são transferidas. Tabela com o mesmo nome da nova não deve existir para a ocorrência da replicação com sucesso.

4.3. Comando de cópia do SQL*PLUS

Este é usado para a cópia de dados de uma tabela para outra noutra base de dados. Também pode ser usado para a transferência de tipos de dados que não são suportados pela replicação em Oracle.

4.4. Trigger

Esta cópia acontece no tempo real. É uma espécie de gatilho que dispara executando certas operações quando ocorre uma modificação.

Contudo estes métodos de replicação requerem por um lado a intervenção manual ou auxílio dos comandos do sistema operativo, e por outro que a base de dados de destino e rede física estejam funcionais para a sua efectivação.

Além destes utilitários ainda existem em uso métodos manuais para replicação de dados, que envolvem dispositivos periféricos para a sua realização. Para estes utilitários os seus métodos de resolução dos conflitos são manuais.

O sistema de gestão de bases de dados em Oracle, introduziu pela primeira vez na sua versão 7, em 1993, o utilitário próprio para gestão da replicação em Oracle.

Resultante disto surgem os métodos de resolução de conflitos automáticos (Loney, 1995).

II. TEORIA DA REPLICAÇÃO E RESOLUÇÃO DOS CONFLITOS EM ORACLE

Os dados são uma vantagem principal duma organização moderna. É amplamente sabido como as bases de dados tornaram-se numa potente ferramenta para concorrência pelos clientes. Muitas decisões do negócio são baseada em informação colectada nas bases de dados, entretanto, a gestão dos dados como um recurso a partir do qual se atinge a eficiência operacional, a efectividade de gestão e o fluxo das vantagens competitivas são um importante objectivo da organização.

Quando uma organização estiver geograficamente dispersa, ou tiver processos operacionais dispersos, poderá decidir em instalar uma base de dados distribuída em vários computadores. Uma Base de Dados Distribuída é uma simples base de dados lógica cuja as suas estruturas físicas se localizem em diversos computadores dispersos em diferentes locais e que estão conectados através de uma rede de comunicação de dados. Uma base de dados distribuída requer múltiplo sistemas de gestão de bases de dados correndo em vários pontos remotos.

A ferramenta tecnológica essencial para garantir estes aspectos são DBMS que mantém os dados numa base de dados integrada e fornecem o acesso controlado aos dados. Existem várias DBMS á saber: Oracle, Informix, MS-SQL-Server, SysBase, e outros. Para este trabalho o DBMS a considerar será a Oracle.

Numa organização onde haja um ambiente heterogéneo e uma complexa relação interdepartamental ou com diversos servidores de bases de dados podem ser aplicadas algumas funcionalidades tais como:

- Partilha de dados.
- Acesso concorrente de dados.
- Replicação de dados.
- Resolução de conflitos na replicação de dados.

O objectivo deste capítulo será estudar os mecanismos para implementação de replicação de dados entre vários servidores de bases de dados e apresentar os respectivos métodos de resolução dos conflitos na eventualidade de emergência

usados na base de dados Oracle. Para tal, temos que apresentar os conceitos básicos associados a este estudo, a seguir os tipos, as técnicas de replicação, as formas de transmissão de dados replicados, e depois os métodos de resolução dos conflitos.

1. TERMINOLOGIA/CONCEITOS BÁSICOS

1.1. As Principais Componentes da Replicação

Num ambiente de replicação de dados, são indispensáveis os seguintes componentes(Bobrowski, Smith, 1997).

Utilizador da administração: É necessária um único utilizador de administração da replicação, como medida de segurança. É usando esta conta que se pode criar a replicação, configurar e remover a replicação. Este utilizador deve ter privilégios para apagar, inserir, modificar e executar qualquer operação sobre os objectos replicados. É comum a utilização do utilizador "repadmin".

Propagador e recebedor: São únicos utilizadores que devem ser criados em cada ponto de replicação através dos quais dados replicados serão enviados e recebidos.

Objecto replicado: Todo o objecto existente em múltiplas bases de dados localizados em diferentes servidores e que pode ser propagado entre eles. Normalmente os objectos básicos replicados são tabelas, porém podem ser suportados visões, pacotes, funções, procedimentos, "Triggers", índices e sinónimos.

Grupos de replicação: Oracle gere os objectos replicados usando os grupos de replicação. Estes são constituídos por um conjunto de objectos lógicos pertencentes a um dono(SCHEMA) necessários para suportar uma determinada aplicação da base de dados. Um objecto replicado só pode ser membro de um único grupo. O grupo de replicação facilita a gestão de objectos replicados pois são tratados como um todo.

Ponto de Replicação: Chama-se Ponto de replicação (Replication site) a qualquer uma de entre múltiplas bases de dados que suporta determinado grupo de

replicação(6). Existem dois tipos de ponto de replicação: Ponto Mestre, Ponto Snapshot.

1. Ponto Mestre: Aquele que mantém uma cópia completa de todos os objectos pertencentes a um grupo de replicação. Todos os pontos mestres num ambiente de replicação multi-mestre comunicam um com outro para a propagação dos dados modificados no grupo de replicação. Todo o grupo de replicação deve ter um e só um único ponto mestre da sua definição. Este funciona como um ponto de controlo da gestão do grupo de replicação e dos objectos replicados.

2. Ponto Snapshot: Suporta um snapshot que pode ser apenas para leitura ou para leitura e modificação, pertencente a uma tabela básica definida no ponto mestre. A tabela snapshot pode conter todos ou parte dos dado de uma tabela definida no grupo de replicação.

Snapshot (foto instantânea): É uma cópia total ou parcial de uma tabela pertencente a um dono lógico local numa base de dados remota. Os *snapshots* podem ser simples ou complexos. São simples quando são resultados de uma única instrução de selecção sem a inclusão de outras instruções nas clausulas *"where"* ou *"having"*. São complexos quando são resultados de uma única instrução de selecção, que contém outras instruções de selecção nas clausulas *"where"* ou *"having"*.

Ligadores de bases de dados (database link): Este é criado com o propósito de estabelecer a comunicação entre duas bases de dados. Especifica os caracteres de conexão via SQL*Net (nome do serviço), utilizador assim como a senha a serem usados na base de dados remota. O ligador de base de dados pode ser privado ou público.

Catálogo de replicação: Todos os locais ou pontos mestres assim como snapshots devem ter um catálogo de replicação. Um catálogo de replicação é um conjunto distinto de dicionário de dados, tabelas e visões que mantém toda a informação administrativa referente aos objectos e grupos de replicação nesse ponto. Todo servidor participante em um ambiente de replicação pode automatizar a replicação dos

objectos nos grupos de replicação usando a informação contida no seu catálogo de replicação. O catálogo da replicação é criado após a criação da base de dados através do programa 'catrep.sql'. Este programa localiza-se normalmente no directório $/ORACLE_HOME/rdbms/admin no Unix, %ORACLE_HOME/rdbms/admin[1] no Windows(95,98,NT e 2000).

O gestor de replicação API e as solicitações administrativas: Para a configuração e gestão da replicação todos os servidores participantes, utilizam uma aplicação programada para interface chamada (API). Esta aplicação consiste num conjunto de pacotes escritos em SQL e ou PL/SQL que contém procedimentos e funções encapsulados que podem ser evocados pelo administrador para a realização do seu trabalho.

Uma solicitação administrativa é um pedido de execução de um procedimento ou função do API. Por exemplo para a criação de um grupo de replicação pode se evocar o procedimento DBMS_REPCAT.CREATE_ _MASTER_REPGROUP. Onde DBMS_REPCAT[2] é o nome do pacote de oracle e CREATE_MASTER_REPGROUP é o procedimento dentro do corpo do pacote.

Transação em deferido: Um ou vários RPC (remote procedure calls) submetidos como uma única transação local e que será propagada e executada como uma transação na base de dados de destino.

Linhas mestres na utilização do API: Algumas funções e ou procedimentos só permitem ser chamados no ponto de definição quando se trata de uma configuração multi-mestre.

Para a realização de certas tarefas administrativas, deve–se primeiro suspender a replicação para os grupos, antes de se chamar procedimentos ou funções de API. A ordem na qual diferentes procedimentos e funções são chamadas é muito importante para a gestão sem erros da replicação .

[1] ORACLE_HOME é uma variável configurada durante a instalação do Oracle
[2] Todos os pacotes com prefixo DBMS são instalados automaticamente durante a instalação do oracle

Bicha de serviço: É um conjunto de códigos em PL/SQL agendado para ser executado periodicamente por um processo em *background*. Todas as transações são submetidos á replicação sobre forma de pedidos de serviço. Quando agendamos um serviço a ser executado ele é colocado numa bicha a fim de ser executado na hora programada conforme a frequência da sua execução que é especificado por uma unidade de tempo. Contudo a bicha de serviço pode ser usada também para executar outras actividades que não sejam da replicação mas que tenham que ser executados periodicamente.

1.2. Variáveis de um serviço

Job Number: Identificador numérico assignado ao serviço. Quando se submete um serviço, oracle gera e atribui automaticamente o número do serviço. Este número é gerado automaticamente através da sequência SYS.JOBSEQ. Uma vez o serviço assignado ao número de serviço este nunca muda mesmo quando se faz uma exportação e importação.

What: O nome do procedimento compilado a ser executado, podendo este conter uma série de parâmetros separados por ponto e vírgula.

Next_date: A próxima vez que se pretende que o serviço seja executado, se este for nulo o serviço será removido da fila de espera.(15)

Interval: É a função usada para calcular a próxima vez que queremos executar o serviço. Esta função envolve o NEXT_DATE seus nos cálculos.
Se por uma razão qualquer o serviço falhar durante a sua execução, oracle tentará 16 vezes a sua execução, de acordo com intervalo especificado, fim das quais irá marcar o serviço como estando quebrado, produzindo um ficheiro com todas as tentativas fracassadas registadas e que pode ser visto nos erros locais.

No_parse: É usado quando se pretende submeter um serviço antes da criação dos objectos envolvidos. Pode tomar valores de falso ou verdadeiro.

Inicializando o background processo SNP: Para a simplificação das tarefas administrativas, Oracle providencia uma configuração que permite refrescar todos os snapshots ou tabelas automaticamente. Os parâmetros abaixo explicados devem ser configurados e inicializados no ficheiro dos parâmetros(anexo A) em cada servidor sobre o qual os snapshots serão replicados e refrescados. Contudo existem outros parâmetros opcionais.

Job_queue_processes: Especifica o número dos background processos do tipo SNPn por servidor participante, onde o n varia de 0 á 9 ou de A á Z. Na configuração do grupo de refrescamento para actualização de snapshots automaticamente, este parâmetro deve ser colocado num valor maior ou igual á 1. Quanto maior for o número de grupos refrescamento que são accionados ao mesmo tempo maior será a quantidade dos processos SNP. Este parâmetro também é usado por oracle para processar pedidos submetidos pelo pacote DBMS_JOB.

Job_queue_interval: Especifica o intervalo de tempo em segundos entre o estado activo e não activo dos SNP background processos. Por regra o valor 60 é estabelecido por oracle para ambientes típicos de replicação.

2. REPLICAÇÃO DE DADOS

Uma das opções mais usadas para distribuição dos dados é o armazenamento separado de cópias de dados nas bases de dados em vários pontos. A replicação permite que um Sistema de Informação duma organização seja acessível num determinado ponto mesmo que os outros estejam indisponíveis e a actualização e sincronização seja feita a posterior.

Replicação é o processo de cópia e manutenção de objectos de um esquema de base de dado para outros esquema em múltiplas bases de dados. O ambiente de replicação é mais típico em transmissão de dados assíncrona e consequentemente aos métodos de resolução de conflitos. Contudo a replicação pode ser também adoptada em transmissão sincronizada. (Delmolino, 1995).

A base de dados Oracle providencia três tipos de replicação:

- Replicação multi-mestre (replicado de qualquer ponto).
- Snapshots actualizáveis.
- Configuração híbrida.

2.1. Replicação Multi-mestre

Neste tipo de replicação existe a possibilidade de actualização dos dados no objecto mestre numa base de dados, assim como nos objectos replicados noutras bases de dados. Neste caso todo o conteúdo da tabela é replicado, não sendo possíveis replicações parciais de uma tabela ou qualquer outro objecto. A figura 2.1. mostra 2 pontos interligados para replicação. Em cada ponto mestre deve conter a cópia do objecto replicado. A replicação multi-mestre pode ser configurada para um único ponto mestre ou para vários pontos mestres. (Gosset, Jang 1999).
A replicação multi-mestre permite a cópia completa de todos tipos de objectos de um schema com a excepção de sequências, database links e clusters.

No momento em que a tabela é criada através do catálogo da replicação, um conjunto de objectos de suporte da replicação para essa tabela serão também criados. Um trigger interno do tipo "*after row , for each row*" também será criado para capturar qualquer operação que for realizada na tabela. Este *trigger* será responsável por colocar um pedido nas tabelas das transações em deferidos para que estas acções sejam propagadas para os pontos remotos. Este pedido corresponde a execução de um procedimento embutido no pacote no ponto remoto. Este pacote tem procedimentos para assegurar todas as modificações (inserção, actualização ou remoção). Além desta capacidade os pacotes possuem uma inteligência para assegurar que não ocorra a re-replicação. Um serviço configurado na bicha de serviço é executado periodicamente para fazer a difusão das transações em deferido para pontos remotos (para todos duma só vez).

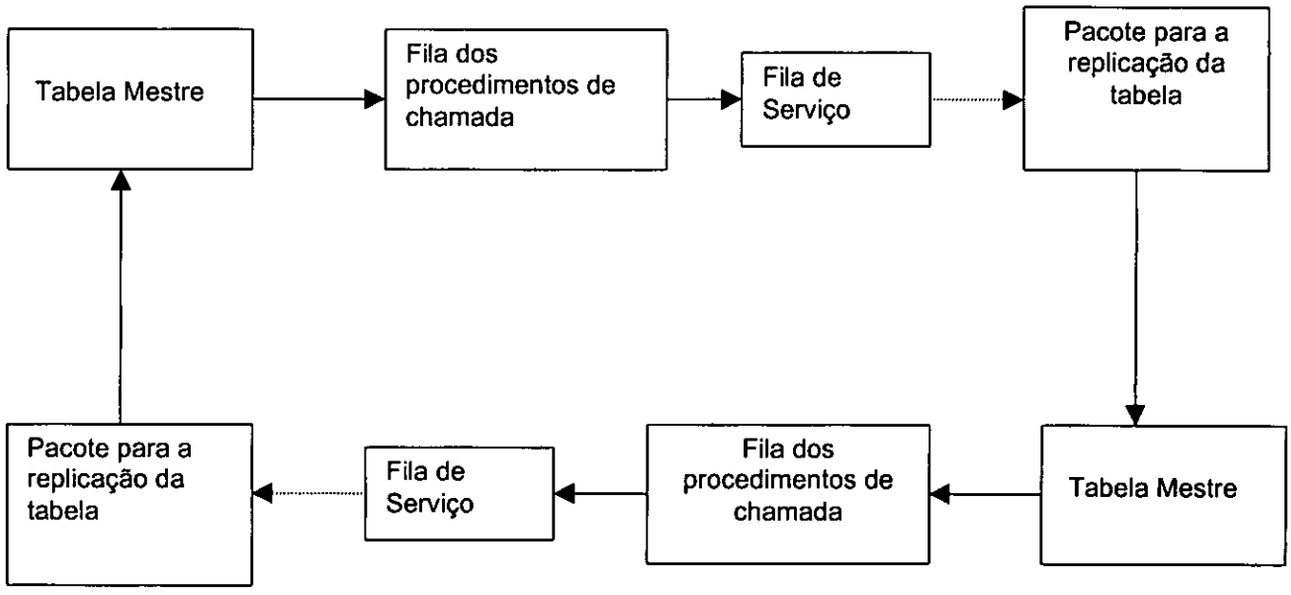


Figura 2.1. Replicação Multi-mestre

Para a criação do ambiente de replicação multi-mestre deve serem seguidos os passos:

1. Desenhar o ambiente de replicação avançada[1], organizar os objectos a replicar e colocá-los em grupos uniformes.
2. Usar a ferramenta ORM[2] para a configuração das componentes da replicação multi-mestre.
3. Conectar se aos ponto mestres e criar os objectos a serem replicados, correspondentes para cada ponto.
4. Garantir os privilégios necessários para os utilizadores das aplicações acederem os dados em cada ponto.

2.2. *Snapshots* actualizáveis

Este tipo de replicação é uma extensão das facilidades que são providenciadas pelo sistema de gestão das bases de dados Oracle sobre a criação e manuseamento dos *snapshots* (Delmolino, 1995).Nesta configuração os pontos snapshots (*snapshots sites*) não contêm as tabelas replicadas, mas em sua substituição tem os *snapshots* que podem ser actualizáveis ou não como mostra a figura 2.2.

[1] É uma técnica de implementação da replicação descrita mais a frente.

[2] ORM – Oracle Replication Manager (anexo H)

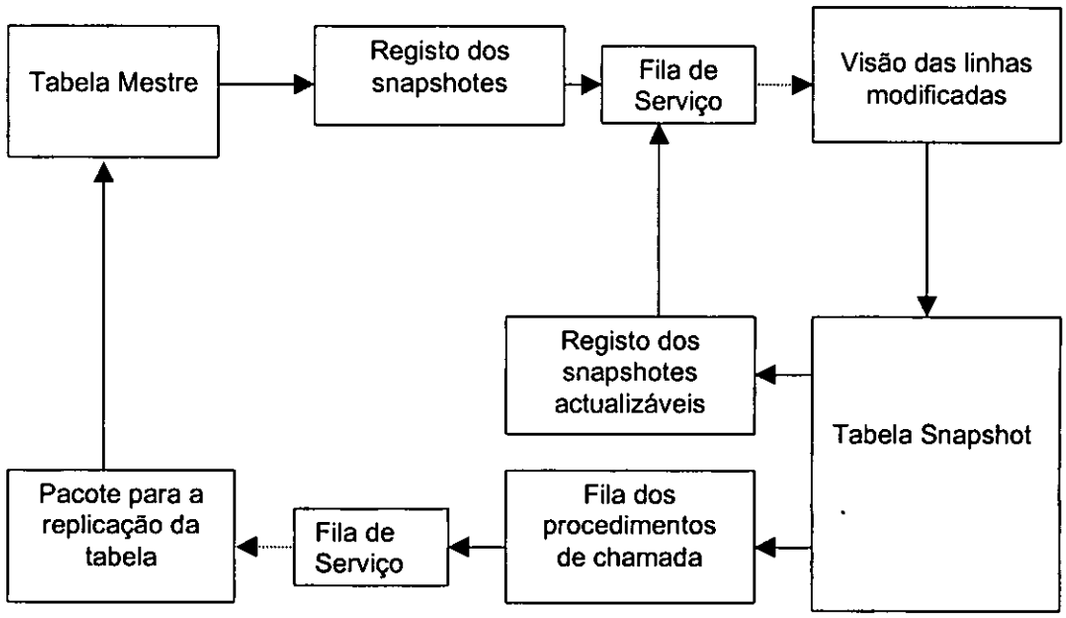


Figura 2.2. Snapshots Actualizáveis

Os *snapshots* actualizáveis têm as seguintes propriedades:

- São sempre tabelas de refrescamento rápido.
- Oracle propaga as mudanças realizadas nos *snapshots* para a tabela mestre remota correspondente. Se for necessário propaga também para outros pontos mestres em forma de cascata.
- Processo de refrescamento dos snapshots actualizáveis é semelhante aos de simples leitura.
- Os *snapshots* actualizáveis também usam tabelas mestres, visões e indexes a semelhança dos de simples leitura.

O funcionamento deste método está ligado a utilização dos *snapshots logs*. Toda a tabela mestre deve ter associada a si um eficiente *snapshot log* para o refrescamento rápido do respectivo snapshot. Se a tabela não tiver o *snapshot log* só pode ser refrescada totalmente o que se assemelha de certa forma a configuração multi-mestre. Para o caso dos *snapshots actualizáveis* cada um destes deve ter também associado a si um *snapshot log.*

A Oracle cria o *snapshot log* para a tabela mestre na mesma base de dados que esta. Este é tratado como se fosse uma tabela normal. Quando a transação altera qualquer informação na tabela mestre, um trigger interno do tipo "*after row , for each row*" é

accionado e insere estas alterações na tabela *snapshot log*. No snapshot log encontra-se a identificação das linhas (*rowid*) que foram modificados, o tipo de modificação (actualização, inserção, ou remoção) ocorrida na tabela mestre bem como a informação acerca dos *snapshots* que tem e os que não tem estas actualizações que reflictam as alterações na tabela mestre. O *snapshot log* está sempre associado a uma tabela mestre o que significa que uma tabela mestre só pode conter um e só único snapshot log. Contudo se a tabela tiver mais de que um snapshot definido sobre ela em outros pontos mestres estes deverão usar o mesmo snapshot log.

2.2.1. Gestão do espaço para o snapshot log

Oracle identifica e reconhece automaticamente quais as linhas do *snapshot log* que já foram utilizados no refrescamento e remove-os dos *logs* para que estes não cresçam infinitamente. Caso existam muitos *snapshots* a usar o mesmo *snapshot log* oracle espera até receber a confirmação de actualização de todos os snapshots antes de remover as respectivas linhas do *log*.

Como resultado da forma como oracle faz a limpeza dos logs, algumas situações indesejáveis que fazem os *logs* crescerem infinitamente, podem ocorrer, especialmente quando vários snapshots estão baseados numa única tabela mestre. Estas situações podem ser derivados a:

- *Snapshots* que não foram configurados para fazerem o refrescamento automático e que não foram manualmente refrescados por muito tempo.
- *Snapshot* que tem um intervalo de refrescamento pouco frequente (de 365 dias) em relação aos dos outros.
- Uma avaria na rede física, que tenha impedido actualizações de um ou mais *snapshots* por longo período.
- Uma avaria na rede que tenha impedido o ponto mestre da actualização do seu dicionário de dados reflectindo que um *snapshot* foi removido.

2.2.1.1. Remoção das linhas nos logs

Deve se tentar manter no *snapshot log* o menor número possível de linhas de forma a minimizar o espaço usado na base de dados.

Para a remoção das linhas e ganhar espaço para outras novas deve se:

- Refrescar os snapshots que estão associados ao log para permitir que oracle remova as linhas.
- Remover manualmente os registos nos logs que se mostram desnecessários.

2.2.3. Usando snapshots só de leitura(*read-only snapshots)*

Nesta configuração as aplicações podem interrogar(seleccionar) os *snapshots* como se de tabelas ou visões se tratasse. Contudo as aplicações não devem emitir comandos de inserção, actualização ou remoção. Se tal acontecer oracle irá retornar um erro. As actualizações só podem ocorrer na tabela mestre. Para a criação do ambiente de replicação *read-only-snapshot* devem serem seguidos os passos:

1. Desenhar o ambiente de replicação básica[1]. Decidir quais serão as tabelas mestres que devem ser replicadas usando os *snapshots*, e quais as bases de dados que requerem tais *snapshots*.
2. Em cada ponto *snapshot* criar os esquemas e os database links para suportar os *snapshots*.
3. No ponto mestre deve ser criado o *snapshot log* para o refrescamento automático dos *snapshots*.
4. Criar os *snapshots* em cada ponto *snapshot*.
5. Garantir privilégios necessários para os utilizadores das aplicações acederem aos dados dos *snapshots*.

2.3. Configuração híbrida ou mista

Esta é uma configuração sofisticada que pode ser obtida através da combinação dos dois métodos acima discutidos. (Delmolino, 1995)

Esta configuração suporta a replicação total e/ou parcial de uma tabela conforme pode ser visto na figura 2.3.

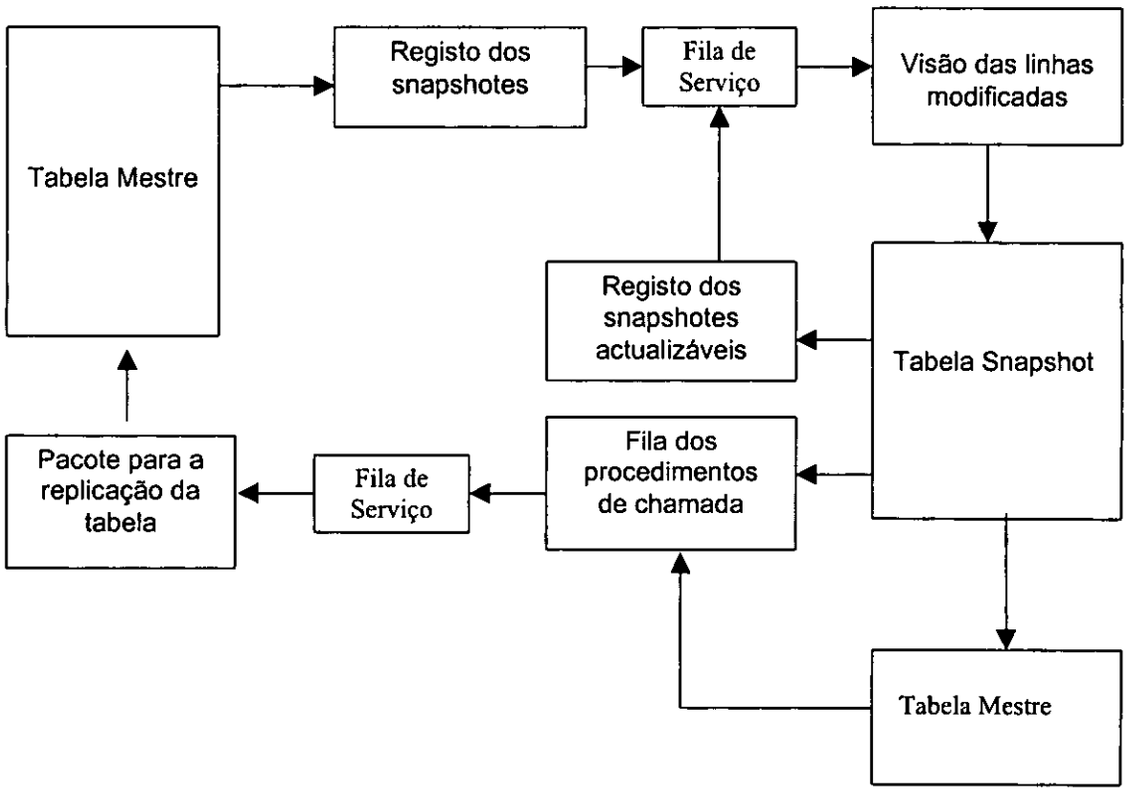


Figura 2.3. Configuração Híbrida

Esta configuração combina as grandes diferenças entre a replicação multi-mestre com a dos *snapshots* actualizáveis nomeadamente:

- Os mestres replicados devem conter dados correspondentes a tabela ou objecto completo a ser replicado enquanto os *snapshots* suportam partes das tabelas que podem ser restringidas através da combinação de linguagens de divinação de dados e linguagens de manipulação de dados.
- A replicação multi-mestre permite a reprodução das mudanças feitas pelas transações a medida que elas ocorrem, enquanto os *snapshots* actualizáveis são orientados para o subconjunto ou seja, agrupam as transações ocorridas em um lote e propagam-nas mais tarde.
- Caso hajam conflitos na configuração multi-mestre podem ser detectados e resolvidos nos vários pontos mestres, enquanto nos *snapshots* só podem serem detectados e resolvidos no ponto mestre.

[1] É uma técnica de implementação da replicação descrita mais a frente

3. TÉCNICAS DE IMPLEMENTAÇÃO

Para a configuração e implementação de um ambiente de replicação Oracle RDBMS providencia dois tipos de técnicas: técnicas básicas e avançadas. Os critérios usados para a selecção da técnica a implementar são o volume de dados, o tipo de replicação, a disponibilidade da rede física, o tipo de transmissão, número de pontos envolvidos, regras do negócio, Software e hardware de suporte.

3.1. Técnicas básicas

3.1.1. Replicação básica

Replicação básica é aquela cujo os utilizadores só podem ler os dados replicados provenientes da tabela mestre. Neste contexto as aplicações podem ler dados nos pontos locais contudo em caso de actualizações estas só podem ocorrer no ponto mestre que é remoto em relação aos pontos locais. Também é chamada de *snapshot replication*, quando os objectos replicados usam *snapshots*.(Gosset, Jang 1999)

3.1.2. *Primary ownership*

Primary ownership (dono primário) também chamado de "dono estático" é o modelo da replicação de dados que é suportado pelo *read-only snapshot*. Esta é a única técnica que não permite a ocorrência de conflitos, pois só é permitido num único servidor (ponto) a possibilidade de actualização ou inserção. Além de controlar o dono dos dados ao nível da tabelas ou outros objectos replicados, as aplicações podem ter permissões de actualização, inserção ou remoção a certas linhas (particionamento horizontal) ou colunas (particionamento vertical) das tabelas replicadas com base em cada ponto num certo instante.

3.2. Técnicas avançadas

3.2.1. Replicação avançada (*advanced replication*)

É aquela na qual os utilizadores podem actualizar os dados replicados em qualquer ponto que participa na replicação(Bobrowski, Smith 1997).

3.2.2. *Dynamic Ownership*

O modelo do *Dynamic Ownership* (dono dinâmico) é menos restrito comparado com o do dono primário ou estático. Neste modelo a capacidade de actualização de dados

nas tabelas move-se de ponto para ponto. Contudo o mecanismo garante que um ponto é que tenha privilégio de fazer actualização de dados num determinado instante.

3.2.3. *Shared Ownership*

Os modelos do ponto primário e do ponto dinâmico são muito seguros pois reduzem a zero a possibilidade de ocorrência de conflitos, contudo são muito restritivos e que dificilmente são implementáveis em ambientes mais flexíveis. Estes ambientes utilizam o método de *Shared Ownership*(dono compartilhado), que por sua vez permite que as actualizações sejam realizadas em qualquer ponto e momento. Contudo este método deve ser implementado com a ajuda de métodos de detecção e resolução de conflitos ou ainda com métodos de notificação.

3.2.4. Replicação Procedural

Quando um sistema de replicação avançada é procedural, os procedimentos que replicam os dados são responsáveis pela verificação da integridade desses dados, isto significa que os procedimentos devem ser desenhados com a componente da detecção e resolução de conflitos embutido. Este tipo de implementação é usado tipicamente quando a base de dados tiver que transferir grandes volumes de serviços em lote.

4. FORMAS DE TRANSMISSÃO DE DADOS

Chama-se de transmissão de dados ao processo através do qual Oracle transfere os dados replicados de um ponto para outro. Para a propagação dos dados replicados, Oracle providencia dois tipos de transmissão, nomeadamente transmissão síncrona e transmissão assíncrona (Bobrowski, Smith 1997).

4.1. Transmissão síncrona

Esta ocorre quando uma aplicação actualiza uma réplica de uma tabela, e ao mesmo tempo com a mesma transação actualiza todas as réplicas da mesma tabela em outros pontos. Consequentemente a transmissão síncrona de dados é também chamada de replicação em tempo real. É normalmente implementada através do uso de trigger nas tabela básica para as replicas e vice-versa.

Vantagens

- Todas réplicas das tabelas são iguais em todos os pontos da replicação a qualquer momento.
- Qualquer actualização é propagada em tempo real.
- Não há necessidade de implementação de métodos de resolução de conflitos.
- Não existe a possibilidade da ocorrência de inconsistência em nenhum momento.

Desvantagens

A principal desvantagem deste tipo de transmissão de dados é de que se um dos pontos não estiver disponível por falhas da rede física, base de dados ou hardware , os pontos disponíveis também manter-se-ão bloqueados até a resolução completa do problema no ponto em causa. Além disso, aumenta o tráfego na rede física, já que cada transação tem que ser transmitida através da rede como tal. Este método é pouco utilizado.

4.2. Transmissão assíncrona

Transmissão assíncrona ocorre quando uma aplicação actualiza uma réplica local de uma tabela, guardando a informação da replicação numa fila de espera local, para a sua propagação a outros pontos de replicação num tempo mais tarde. Consequentemente a transmissão assíncrona é chamada de replicação do tipo armazena - propaga (store-forward). Este é o método no qual se baseia a replicação em oracle.

Vantagens

- Se um ponto contendo tabelas réplicas não estiver disponível, isso não impedirá a troca de informação entre os outros, e logo que este estiver disponivel será actualizado para que esteja ao nível dos outros.
- As transações são armazenadas numa fila de espera e podem ser transmitidos em deferido em intervalos de tempo que convier a cada ponto. Podem ser programados serviços(job) que facilitem a monitoração de pontos remotos.
- Reduzem o tráfego na rede, muitas transações podem ser agrupadas em um único lote de serviço (batch job), podendo se escolher tempos mortos para a sua propagação.

Desvantagens

- Existe a possibilidade da ocorrência de inconsistência temporária entre as tabelas durante o tempo que esperam pela troca das actualizações.
- É obrigatório a implementação de métodos de resolução de conflitos, para garantir a consistência de dados.
- Estes conflitos ou inconsistência só serão detectados quando ocorrer a transmissão.
- Exige trabalhos de administração e configuração adicionais.

4.3. Propriedades

- As aplicações fazem a actualização das tabelas através do uso de linguagem de manipulação de dados. Atraves de triggers e pacotes programados no ponto de origem, oracle coloca na fila de espera o RPC[1]para replicar a transação a outros pontos no sistema.
- De acordo com intervalos programados, o sítio de origem puxa a transação em deferido da fila de espera para os pontos de destino.
- De acordo com intervalos programados o ponto de origem limpa as transações que foram puxados com sucesso para os sítios de destino.
- Diferentes pontos podem ter iguais ou diferentes intervalos.
- A ordem em que as transações são submetidos é preservada durante a execução em pontos remotos.
- A frequência da propagação pode ser controlada em vários pontos e pode ser realizada de forma independente uma da outra. Esta propagação pode ser controlada através da alocação de prioridades para cada sítio no caso de ela ser controlada em vários sítios nomeadamente:
    - Ponto de origem
    - Um dos pontos de destino
    - Um ponto qualquer apenas de controlo.

O link de destino é especificado quando a transação em diferido é submetida na fila de espera ou quando esta é registada por uma ferramenta própria de administração.

Durante a propagação toda a integridade de referência e outros "constraints" são preservados.

4.4. Erros de propagação

Se a propagação falhar por motivos da indesponibilidade da base de dados de destino, problemas na rede ou pelo facto do tablespace de destino onde reside a tabela se encontrar sem espaço, a transação respectiva permanecerá na fila de espera para a sua execução mais tarde.

Outros tipos de erros que podem ocorrer tais como (violação por duplicação do índice), durante a execução no sistema de destino resultarão em:

- Transação será cancelada (*rolled back*).
- A mensagem de erro e a informação da transação em diferido serão colocado numa tabela de registos de erros na base de dados de destino.

Evocando procedimentos em PL/SQL ou usando OEM o administrador da base de dados, poderá re-submeter a transação que tenham falhado, após a correcção do erro.

4.5. Ligações programadas

Ligações programadas são necessárias para a propagação de dados replicados de um ponto para o outro. Em replicação entre vários mestres cada ponto mestre requer uma ligação programada para mover dados para outros mestres. Adicionalmente um ponto snapshot actualizável também precisa de uma ligação programada para mover dados para o ponto mestre correspondente.

As ligações programadas requerem os seguintes parâmetros: ligador de base de dados a ser usado para a propagação de dados, Next Date (próxima data) tempo inicial para puxar as mudanças do ponto local para o de destino, expressão de intervalo que é intervalo automático no qual os dados deverão ser puxados. Estes parâmetros são detalhados no anexo B.

[1] Remote procedure call

5. OCORRÊNCIA DE CONFLITOS DE DADOS

Os conflitos podem ocorrer em qualquer ambiente de replicação que permita actualizações concorrentes dos mesmos dados em pontos diferentes. Entende-se aqui, que se duas transações originárias de pontos diferentes actualizarem ou removerem a mesma linha quase ao mesmo tempo ocorrerá um conflito. Durante a configuração do ambiente de replicação é importante prever o tipo de conflitos que podem ocorrer. Quando existe conflito e é detectado, os dados não convergem até a sua resolução total(Loney, 1995).

5.1. Tipos de Conflitos de replicação

Oracle possui facilidades que permitem detectar para posterior resolução de três tipos de conflitos. Conflito de actualização, Conflito de unicidade e conflito de eliminação.

5.1.1 Conflito de actualização

O conflito de actualização ocorre na replicação, quando há uma tentativa de actualização numa linha da tabela num ponto e num outro ao mesmo tempo. Entende–se que neste caso duas transações originárias de pontos diferentes, actualizam a mesma linha quase ao mesmo tempo.

5.1.1.1. Sequência de eventos durante a actualização

Durante a actualização ocorrem de forma sequencial os seguintes eventos:

- Uma linha é actualizada num ponto mestre.
- Um trigger interno é accionado de forma a criar um pedido de inserção na fila de espera das transações em deferido.
- O pedido de actualização é propagado para o ponto de destino, mais tarde.
- Um procedimento de actualização do pacote no ponto de destino é executado de forma a fazer actualizações nas linhas deste ponto.

5.1.2. Conflito da unicidade

O conflito da unicidade ocorre quando a replicação de uma linha tenta violar a integridade da entidade (Chave primária[1] ou único constraint). Entende–se aqui, que

[1] Oracle permite uma coluna ou combinação destas como chave primária

duas transações originária de dois pontos diferentes tentam cada uma inserir uma linha nas respectivas tabelas réplicas com o mesmo valor da chave primária.

5.1.3. Conflito de eliminação

O conflito de eliminação ocorre quando há duas transações originárias de pontos diferentes, sendo que uma tenta eliminar uma linha que a outra está eliminando ou actualizando ao mesmo tempo.

5.2. Formas de evitar conflitos

Em caso dos requisitos da aplicação assim o exigirem sempre deve se desenhar um sistema de replicação que evite a ocorrência de conflitos. As formas para conflitos são desenhadas de acordo com o tipo de conflito. Em circunstâncias em que estas formas não se apliquem satisfatoriamente, podemos recorrer aos métodos de resolução de conflitos e/ou de notificação, descritos na secção 7 e 8 deste capítulo.

5.2.1. Evitar conflitos de unicidade

Os conflitos de unicidade podem ser evitados através do uso de sequências, que geram chaves primárias mutuamente exclusivas de forma sequencial, ou através do uso de chaves compostas que incluam o número e ou o nome do ponto. Contudo estas soluções podem se tornar problemáticas a medida que o número de pontos aumentar, ou quando o número de linhas nas tabelas replicadas crescer(Thorpe, 2000).

5.2.2. Evitar conflitos de actualização

Este é o tipo de conflito que é difícil de evitar, contudo é possível minimizar com o uso de :

- Ponto primário como dono.
- Ponto dono dinâmico.

Para evitar com sucesso o conflito de actualização usando o método de ponto dono dinâmico as aplicações devem garantir que:

- Só o dono da linha é que pode actualizar a linha no momento.
- A linha não pode pertencer a dois ou mais donos ao mesmo tempo.

- Conflitos em termos de ordenamento podem ser resolvidos em qualquer ponto.

5.2.3. Evitar conflitos de eliminação

Sempre que se revelar impossível evitar que este tipo de conflito ocorra, podem ser usados vários métodos para se evitar a remoção de dados. Um dos métodos é não usar nenhuma instrução de remoção imediata nas aplicações, marcando apenas os registos para a sua remoção á posterior. Esta eliminação pode significar enviar os dados para tabelas históricas, tornando possível a sua recuperação em caso de necessidade.

6. DETENÇÃO DE CONFLITOS DE DADOS

A importância da resolução de conflitos reside no facto de garantir a consistência de dados. Uma vez detectado um conflito, a sua resolução deve ser breve de forma a evitar o efeito da inconsistência em cascata, ou seja conflitos podem gerar outros conflitos e por aí em diante (Bobrowski, Smith 1997).

6.1. Como oracle detecta diferentes tipos de conflitos

O ponto mestre de recepção num ambiente de replicação detecta conflitos de actualização, unicidade e de eliminação da seguinte forma:

1. O ponto de recepção detecta o conflito de actualização ao verificar alguma diferença entre o valor antigo, o valor novo da linha replicada(valor antes da modificação) e o valor corrente.
2. O ponto de recepção detecta o conflito de unicidade se ocorrer uma violação do único constraint durante a inserção ou actualização numa linha replicada.
3. O ponto de recepção detecta o conflito de eliminação se não puder localizar a linha para uma inserção ou eliminação por causa da ausência da chave primária ou da linha no seu todo.

6.2. Para o conflito de actualização

Oracle utiliza os grupos de colunas para detectar e resolver conflitos de actualização. Um grupo de colunas é um agrupamento lógico de uma ou varias colunas, numa tabela replicada. Toda a coluna numa tabela replicada é parte de um grupo de

colunas singular. Na configuração do ambiente de replicação num ponto de definição mestre, podem ser criados grupos de colunas e aplicar a cada uma delas um determinado método de resolução de conflitos.

Os grupos de colunas permitem e facilitam a designação de diferentes métodos de resolução de conflitos para cada grupo de tipos de dados. Frequentemente os dados de tipo numérico são assignados métodos de resolução de conflitos aritméticos, enquanto os de tipo caracteres são assignados á métodos de tempo da ocorrência(timestamp).

A selecção de colunas para a formação de grupos de colunas requer um estudo cuidadoso, pois colunas que tem uma certa dependência devem fazer parte do mesmo grupo de colunas de forma a garantir a consistência do dos dados. (Bobrowski, Smith 1997).

6.3. Para o conflito de unicidade

Em muitos casos deve se desenhar sistemas de replicação e aplicações que impossibilitam a ocorrência deste tipo de conflito. A sua implementação é facilitada pelo facto deste poder ser feito na linguagem de definição de dados sem recorrência a muitos e complicados exercícios em códigos de programação. Oracle providencia alguns métodos de resolução de conflitos para a unicidade.

6.4. Para conflito de eliminação

Na maioria dos casos é difícil desenhar aplicações que evitem a ocorrência deste tipo de conflito. Este tipo de conflito é restritivo para cada tipo de aplicação. Oracle não providencia nenhum método pré definido para a detenção deste tipo de conflitos reservando-se assim deste modo o uso de métodos definidos pelo usuário. Desta forma é claro que em muitos casos deve se recorrer a grandes exercícios em códigos de programação.

7. MÉTODOS DE RESOLUÇÃO DE CONFLITOS

Para a resolução automática de conflitos pode ser assignado um ou vários métodos de resolução de conflitos providenciados pela oracle. Para o uso destes métodos não é necessário o recurso ao código de programação, contudo abre-se também a possibilidade do usuário definir os próprios métodos que satisfaçam a sua aplicação e regras dos seus negócios. (Delmolino, 1995).

7.1. Métodos de resolução de conflitos de actualização

Segundo(Bobrowski, Smith 1997) os métodos de resolução de conflitos de actualização oferecidos por Oracle e que podem ser assignados aos grupos de colunas são:

- Valor aditivo e valor médio
- Valor máximo e valor mínimo
- Valor do tempo cedo e valor do tempo tarde. (Earliest and latest timestamp value)
- Valor sobreposto e valor descartado
- Pontos prioritários e grupos prioritários

7.1.1. Funcionamento

7.1.1.1. Valor aditivo e valor médio

Os métodos aditivo e valor médio funcionam com um grupo de colunas de tipo numérico. O método aditivo adiciona a diferença entre os valores antigo e o novo no ponto de origem com o valor corrente do ponto de destino.

Valor corrente = valor corrente + (valor novo - valor antigo).

O método aditivo providencia a convergência de dados para qualquer número de pontos mestres envolvidos.

Para a resolução de conflitos o método do valor médio calcula a média entre o valor novo vindo do ponto de origem e o valor corrente do ponto de recepção.

Valor corrente=(valor corrente + valor novo)/2.

Este método não garante a convergência de dados se estiverem envolvidos mais do que um ponto mestre. É mais usado nos casos em que temos um ponto mestre e vários pontos snapshots.

7.1.1.2. Valor Mínimo e valor Máximo

Estes métodos consistem em fazer comparaçoes entre os valores novos provenientes do ponto de origem com os valores correntes do ponto de destino pertencentes a um grupo de colunas. Para o metodo de valor do valor minimo, se o valor novo for menor que o valor corrente, o valor novo proveniente do ponto de origem é aplicado no destino (Assumindo que não há nenhum outro conflito pendente para mesma linha). Se o valor novo para uma designada coluna for maior que o valor corrente, o conflito será resolvido por manter esta sem nenhumas alterações.

No caso de se verificar uma igualdade entre os dois valores (Exemplo, se a coluna designada não for a que causa conflitos), o conflito não será resolvido usando este método, e os valores manter-se-ão intactos. O método de valor máximo é similar ao método do valor mínimo descrito acima sendo a diferença de que o valor novo proveniente do ponto de origem só é aplicado no ponto de destino se este for maior que o valor corrente.

Este método não restringe o tipo de dados das colunas envolvidas, contudo se torna pesado para casos em que temos variáveis de tipo caracteres. Só é garantido convergência para mais do que um ponto se:

- Para o método de valor máximo, o valor novo da coluna for sempre crescente.
- Para o método de valor mínimo, o valor novo da coluna for sempre decrescente.

7.1.1.3. Valor do tempo cedo e valor do tempo tarde.

Os métodos do tempo mais cedo e do tempo mais tarde são uma variante dos métodos do valores máximo e mínimo, para os casos em que temos colunas de tipo caracter. Para o uso deste método é obrigatório acrescentar á tabela ou ao grupo de colunas, uma coluna do tipo DATE. Quando a aplicação faz a actualização de uma coluna num ponto deverá fazer a actualização também da coluna acrescentada com o

valor SYSDATE[1]. Para qualquer mudança realizada noutro ponto, o valor do tempo deve ser o do ponto de origem.

O método do tempo mais cedo aplica as mudanças provenientes do ponto com o tempo mais cedo enquanto que o do tempo mais tarde actualiza com base no ponto que tiver o tempo da ocorrência mais tarde. É sempre necessário designar um método para backup que deve ser accionado, caso hajam actualizações com tempo de ocorrência igual.

Outros factores a ter em conta para este método.

Quando se decide por este método deve se ter o cuidado de analisar como o tempo é gerido nos diferentes pontos onde se localizam os servidores da replicação. No caso da replicação cruzar deferentes fusos horários devem ser, por consenso, calculados e convertidos os tempos locais para o tempo Universal ou tempo médio de Greenwich (TMG).

De referir que se dois servidores não tiverem os seus relógios bem sincronizados as comparações dos tempos poderão redundar em erros.

As formas de minimizar a ocorrência destas anomalias podem ser:

- Uso da lógica por cada aplicação para sincronizar o tempo
- O administrador da base de dados pode criar um programa para as tabelas replicadas que faça a sincronização para todos os membros da replicação.

Este método não pode garantir a convergência de dados num ambiente de replicação com intervenção de mais do que dois pontos mestres.

7.1.1.4. Valor sobreposto e valor descartado

Estes métodos ignoram pura e simplesmente valores do ponto de origem ou do ponto de destino. O método de sobreposição substitui o valor corrente no ponto de destino com o valor novo vindo do ponto de origem. Inversamente o método do valor descartado ignora o valor vindo do ponto de origem. Estes métodos não garantem a convergência de dados no caso em que estão envolvidos mais do que um ponto

[1] variável do sistema que indica data no servidor local:dia, mês, ano, hora, minuto, segundo

mestre. Este método é útil quando se tem definido um simples ponto mestre e múltiplos pontos snapshots ou quando acompanhado de algumas facilidades de notificação de conflitos.

7.1.1.5. Grupos prioritários e ponto prioritários

Grupos prioritários permitem assignar níveis de prioridades para cada valor possível duma determinada coluna. Se Oracle detectar um conflito, ele actualiza a tabela cuja a coluna de prioridade tem valor menor usando os dados provenientes da tabela com um valor de prioridade maior. É necessário especificar prioridades para todos os valores possíveis das colunas prioritárias, além disso é também necessário designar qual será a coluna prioritária.

O ponto prioritário é um caso particular de um grupo prioritário. Com o ponto prioritário a coluna prioritária que designamos é automaticamente actualizada com o nome global da base de dados do ponto onde a actualização é originaria. Estes métodos são usados quando se parte de um princípio de que dados provenientes de um certo ponto são sempre os mais correctos.

Usando estes métodos a convergência de dados não é garantida para os casos em que estão envolvidos mais do que dois pontos mestres, contudo pode ser garantida a convergência com mais de dois ponto mestre, usando grupos prioritários quando o valor da coluna prioritária for sempre crescente.

7.2. Métodos de resolução de conflitos de unicidade

Oracle oferece os seguintes métodos de resolução de conflitos que podem ser assignados á chaves primárias ou aos "constraints" únicos(Delmolino, 1995):

- Anexar o nome global do sítio(site name) de origem ao valor da coluna vindo do mesmo ponto.
- Anexar o valor numérico da sequência(site sequence) gerada no sítio de origem ao valor da coluna vindo do mesmo ponto.
- Ignorar o valor da linha vinda do sítio da origem.

Os métodos acima definidos não garantem a convergência de dados em oracle mas, servem de elo de ligação para os mecanismos de notificação.

7.2.1. Nome do ponto e sequência de ponto anexados ao valor da coluna

Os métodos do nome do ponto ou sequência do ponto anexados funcionam através do acréscimo de uma sequência de caracteres na coluna que está gerando uma excepção(erro) do tipo DUP_VAL_ON_INDEX. Não obstante estes métodos permitirem que sejam inseridos ou actualizados dados sem violar a integridade de constraint único, não garantem a convergência de dados entre múltiplos pontos mestres. As discrepâncias verificadas devem ser manualmente resolvidos. Estes métodos devem ser usado acompanhados de um método de notificação. Estes dois métodos só podem ser usados para dados de tipo caracter. Estes métodos podem ser úteis quando a disponibilidade de dados for o item mais importante do que a própria precisão ou exactidão dos mesmos. Como forma de permitir que dados estejam disponíveis logo que sejam replicados deve-se:

- Seleccionar nome do ponto anexo ou a sequência do ponto anexa e usar um esquema de notificação para alertar a pessoa indicada para a resolução do conflito da duplicação em vez de registar o conflito.
- Quando o conflito de unicidade ocorre, o método de anexar o nome acrescenta o nome global da base de dados do ponto que origina a transação no valor da coluna replicada.
- De forma similar o método anexar a sequência acrescenta o número da sequência ao valor da coluna. O valor da coluna é truncado sempre que for necessário. Se o tamanho valor da coluna gerado exceder o comprimento da coluna o método de resolução de conflitos não resolve o erro.

7.2.2. Ignorar o valor da linha vinda do sítio da origem.

O método de ignorar, resolve o conflito de unicidade simplesmente descartando a linha vinda do ponto de origem que causa o erro. Este método não garante a convergência de dados entre múltiplos pontos mestres e deve ser usado acompanhado de uma facilidade de notificação. Ao contrário dos métodos de anexar este método minimiza a propagação de dados até que a sua precisão seja verificada.

7.3. Restrições nos métodos de resolução de conflitos

Os método de resolução de conflitos pré definidos em oracle não suportam o seguinte:

- Conflitos de eliminação.
- Mudanças nas colunas das chaves primárias (unique constraints)
- Valores nulos nas colunas que designamos para a resolução de conflitos
- Violação da constraints de integridade referencial

Para estas situações, o programador/analista deve determinar métodos de resolução de erros manualmente ou providenciar os chamados métodos de resolução de conflitos definidos pelo utilizador.

7.4. Métodos de notificação providenciados pelo Oracle

Como forma de acompanhamento do uso dos métodos de resolução de conflitos providenciados pelo oracle ou definidos pelo utilizador é importante considerar a utilização de registo e notificações de conflitos para completar a resolução de conflitos. Oracle permite a configuração duma tabela de replicação para chamar os métodos definidos pelo utilizador, a serem accionados, e para gravar as informações de conflitos ou notificar caso oracle não consiga resolver determinado conflito. Também é possível configurar grupos de colunas, constraintes, ou tabelas replicadas para a notificação de todos os conflitos ou apenas aqueles que oracle não consegue resolver.

7.5. Combinação de diferentes métodos de resolução de conflitos.

Indicando e combinado vários métodos de resolução de conflitos para grupo de colunas permitirá ao oracle resolver conflitos em várias alternativas nas quais um único método seria um fracasso. Quando se usa esta combinação, oracle executa estes métodos de acordo com a sequência indicada pelo usuário.

O algoritmo usado por oracle para a resolução do conflito de actualização é o seguinte:

Começando pelo primeiro grupo de colunas, o ponto de recepção examina cada coluna no grupo, de forma a determinar se foi mudado ou não, se for o caso, entao verifica se há algum conflito entre os valores antigo, novo e o corrente.Se não tiver

ocorrido nenhum conflito, oracle pode continuar passando a coluna do grupo seguinte. Se tiver ocorrido algum conflito, oracle chama o método de resolução de conflito que tiver sido assignado o menor valor da sequência no grupo de colunas.

Se o método de resolução de conflito resolver com sucesso o conflito, oracle conserva o valor apropriado das colunas de determinação de estado pendentes. Se o método não puder resolver o conflito, oracle continua com o próximo método até conseguir resolver o conflito ou até não haver mais métodos disponíveis.

Depois de avaliar todos os grupos de colunas e resolver com sucesso qualquer conflito, oracle guarda o valor novo da coluna. Se oracle não puder resolver o conflito com o uso dos métodos configurados o ponto de recepção regista a transação no seu catálogo de replicação como errada e não aplica qualquer mudança na linha da tabela local. Usam-se combinações de diferentes métodos de resolução pelas seguintes razoes:

- Para se aplicar métodos alternativos de resolução de conflitos quando o método preferido não poder resolver o conflito.
- Para receber uma notificação automática sobre a ocorrência de um conflito de replicação.

7.5.1. O uso de combinações de métodos como forma de segurança

Em certas situações o método de resolução de conflito escolhido para um grupo de colunas, constarint ou mesmo tabela pode não ter sucesso sempre. Neste caso tem que ser listada uma sequência de métodos alternativos de forma que oracle resolva automaticamente o conflito sem necessidade de uma intervenção manual.

Alguns métodos não podem garantir o sucesso da resolução em todas as circunstancias. Por exemplo o método de ultimo tempo da ocorrência pode falhar na situação em que exactamente ao mesmo tempo dois pontos distintos actualizam uma linha de uma tabela, contudo se este método for combinado com o de ponto prioritário como sendo o seu backup oracle resolverá este conflito sem intervenção manual

7.6. Métodos de resolução de conflitos definidos pelo usuário

Oracle permite que o utilizador defina o seu método de resolução de conflitos. O método de resolução de conflito definido pelo utilizador é um conjunto de código em PL/SQL que deve retornar falso ou verdadeiro. Verdadeiro deve ser retornado quando o método indicado tiver resolvido todos as mudanças conflituosas para um determinado grupo de colunas. Falso indica que o método de conflito falhou na resolução de determinado conflito e que deve continuar a usar outros métodos até que retorne verdadeiro ou os métodos disponíveis se esgotem. Se o método de conflito levantar uma excepção(erro), oracle para mesmo que existam métodos disponíveis não irá usá-los.

7.7. Os parâmetros dos métodos de resolução de conflitos.

Os parâmetros necessários para os métodos de resolução de conflitos definidos pelo usuário dependem do tipo de conflito que se pretende resolver e do tipode dados da coluna que está sendo replicada. Todos os métodos levam como parâmetros o valor novo, o valor antigo e o valor corrente(Bobrowski, Smith 1997).

.

O valor antigo representa o valor da linha no ponto de origem antes da realização de mudanças. O valor novo representa, o valor da linha no ponto de origem depois da realização de mudanças. O valor corrente representa o valor da linha equivalente no ponto de recepção. Oracle usa a chave primária ou outra chave especificada por SET_COLUMNS para determinar quais as linhas a comparar.

Para a implementação com sucesso e de forma fácil dos métodos de resolução de conflito definidos pelo usuário, oracle providencia algumas funções que são pré construídas, cabendo ao usuário carregar os respectivos parâmetros.

7.7.1. Para o conflito de actualização

Para o conflitos de actualização a função definida pelo utilizador deve aceitar os seguintes valores por cada coluna no grupo de colunas(anexo B ponto1):

- Valor antigo do ponto de origem. O modo deste parâmetro deve ser IN. Este valor nunca deve ser mudado.

- Valor novo da coluna do ponto de origem. O modo deste parâmetro deve ser do tipo IN e OUT. Se a função puder resolver o conflito com sucesso modificara o valor antigo pelo novo.
- Valor corrente que representa o valor corrente do ponto de recepção. O modo deste parâmetro deve ser IN.

7.7.2. Para o conflito de unicidade

Já que este ocorre como resultado de uma actualização ou inserção a função ou procedimento deve aceitar como parâmetros o valor novo da coluna vindo do ponto de origem, com o modo IN e OUT, o resultado final da função deve ser Boleano A função deve retornar verdadeiro se tiver descartado o valor ou falso caso contrario.

7.7.3. Para o conflito de eliminação

Para o conflito de eliminação a função deve aceitar o valor antigo da coluna no modo IN OUT para a respectiva linha. O parâmetro final para o método de resolução de conflito deve ser de tipo boleano. Se o método de resolução de conflito puder resolver o conflito com sucesso, irá modificar o valor antigo e assim sendo permitirá que oracle elimine o valor corrente da coluna. A função deve retornar verdadeiro se tiver descartado o valor ou falso caso contrario.

7.7.4 Restrições dos métodos de resolução de conflitos definidos pelos utilizadores

Devem ser evitados os seguintes tipos de declarações, que podem redundar em resultados imprevisíveis:

- Linguagens de definição de dados (exemplo create, alter, drop)
- Declarações de controle de transações (commit, rollback, savepoint)
- Declarações de controlo de sessões (por exemplo, ALTER SESSION).
- Declarações de controlo do sistema (por exemplo, ALTER SYSTEM).

7.8. Método de notificação definido pelo utilizador

O método de notificação definido pelo usuário é um conjunto de instruções que providencia a notificação de conflitos como complemento dos métodos de resolução. Estes procedimentos ou funções podem ser codificados de forma a notificar o

administrador da base de dados por meio de mensagem ao telefónica celular, pager ou e-mail(anexo B ponto 2.2)

Depois de se ter codificado o método de notificação, este deve ser assignado a um grupo de colunas (ou constraint) numa ordem específica, assim oracle poderá usá-lo caso qualquer conflito seja detectado e antes de tentar avançar para o método subsequente ou depois de oracle tentar sem sucesso resolver um conflito.

Para configurar uma tabela replicada com um mecanismo de notificação devem ser completados os seguintes passos:

- Criação de um registo para as notificações dos conflitos.
- Criação do programa para a notificação de conflitos.

Para a criação de um registo para as notificações é necessário, primeiro criar-se uma tabela na base de dados que irá registar as notificações(Anexo B ponto 2.1). Esta deve conter campos que o utilizador melhor achar que lhe podem facilitar a notificação e a fácil percepção. A criação da tabela é feita com base em linguagem de definição de dados. Não se deve replicar as tabelas de registo das notificações, pois estas representam também especificações de cada ponto onde se encontram. (O anexo B 2.2 mostra um programa de notificação de conflitos.)

III. SISTEMAS DE INFORMAÇÃO DA MOZAL

Os sistemas de informação da Mozal, estão divididos em níveis conforme o tipo de informação que providenciam. Os níveis variam desde os sistemas que comunicam directamente com os instrumentos até aos mais integrados como é o SAP. Existem vários interfaces para a comunicação entre eles e que todos são suportados por diferentes tipos de hardware e Software (Document ID MZAT-007, Dti CONSORTIUM, Novembro 1999].

1. INTRODUÇÃO A MOZAL

A Mozal representa o maior investimento estrangeiro em Moçambique no período pós independência (cerca de 1.3 biliões de dólares americanos), tem como accionistas a Billiton (Reino Unido) 47%, Industrial Development Corporation (RAS) 24%, Mitsubishi (Japão) 25% e Governo Moçambicano 4%.

A função principal da Mozal é de produzir alumínio primário. A funcionar na sua máxima capacidade instalada (primeira fase) esta fábrica vai produzir 250.000 toneladas de alumínio primário por ano na forma de lingotes de 22 Kg cada.

A tecnologia usada para a fundição de alumínio na Mozal é a AP30 da Pechiney (França), largamente reconhecida como a mais eficiente e ambientalmente aceitável tecnologia de fundição de alumínio do mundo.

Esta tecnologia é baseada no processo de Hall-Heroult, um processo electrolítico de fusão da alumina em alumínio metálico. O alumínio líquido da Redução é sangrado de cada célula electrolítica e transferido num tanque especial a quente até á Casa de Moldes.

1.1. Matérias primas vitais na produção de alumínio

A qualidade do metal obtido no processo de fundição electrolítica da alumina depende directamente do grau de pureza das matérias primas empregues no processo, dado que o processo não inclui a purificação do metal obtido. Por esta razão, a qualidade e quantidade das matérias primas adicionadas às células electrolíticas são controladas

regularmente e de acordo com o plano analítico em uso na Mozal. Esta matéria prima é descrita a seguir.

1.1.1. Alúmina

Alumina é o termo comercial do óxido de alumínio. Este deriva do tratamento químico da bauxite no processo conhecido como processo de Bayer. A alumina é produzida pela Worsley na Austrália e importada por via marítima para Moçambique.

1.1.2. Electricidade

A subestação na fundição transforma a electricidade de corrente alterna (AC) para corrente contínua (DC). A Mozal consome 13500 kWh de corrente eléctrica contínua por tonelada de alumínio produzido.

1.2. Departamentos da Mozal, função e área de responsabilidade

A Mozal SARL, é composta por dois tipos de departamentos a saber:

- Departamentos de produção: Redução, Serviços de redução e Manutenção, Laboratório e Ambiente, Fábrica de carbono, Casa de moldes e Sistema de gestão de stock-SPMS. Cada departamento de produção tem uma aplicação específica que suporta a produção e faz o interface com outros sectores.

- Departamento dos Serviços: Administração e finanças, Gestão de materiais, Gestão de sistemas informáticos, Engenharia e Recursos humanos.

Vamos descrever apenas os Departamentos de Produção porque, para além de serem específico para a Mozal, constituem o ambiente onde será realizado o estudo.

1.2.1. Departamento de redução

Este departamento tem como função produzir metal líquido de qualidade para a casa de moldes e maximizar a eficiência das caldeiras. As responsabilidades da Redução incluem todas as actividades relacionadas com a operação das caldeiras, delineamento e alinhamento destas e transporte do ânodo.

1.2.2. Fábrica de carbono

A função da fábrica de carbono é de fornecer ânodos rodados de qualidade ao departamento de Redução. As responsabilidades deste departamento estendem-se a todas as actividades relacionadas com as operações de todos os equipamentos do carbono desde os silos de armazenagem de piche e coque aos ânodos rodados, incluindo a reciclagem dos ânodos.

1.2.3. Casa de moldes

A função do departamento de moldagem, tal como o próprio nome indica é de produzir metal de qualidade para os clientes da Mozal. As responsabilidades deste departamento compreendem todas as actividades relacionadas com as operações do equipamento, incluindo transporte do metal vindo da redução, a moldagem do mesmo em pequenos lingotes, lavagem dos veículos de transporte, armazenagem dos lingotes de metal e transporte para porto.

1.2.4. Departamento de Laboratório e Ambiente

A função deste departamento é de assegurar análises exactas para todos os departamentos de produção a mínimo custo possível e produzir todos os dados técnicos requeridos dentro dum prazo estabelecido. As responsabilidades do departamento de laboratório e ambiente incluem todas as actividades relacionadas com a efectivação de análises para o controle do processo, das matérias primas, subprodutos ou produtos intermediários e produto final, bem como a monitorização do impacto de todas as operações no meio ambiente interno e externo.

1.2.5. Departamento de Serviços de redução

Este departamento tem como objectivo providenciar e controlar todos os serviços de auxílio ao departamento de redução. É composto por diversas unidades a saber: A manutenção e utilitários, subestação eléctrica, gestão de serviços e finalmente a gestão de matérias primas. A subestação é responsável pelo fornecimento da energia para a produção e consumo de toda a planta em geral. A unidade de manutenção e utilitários faz a gestão de instalações e equipamentos que não estão directamente ligados a nenhum dos sectores de produção, bem como as que servem os mesmos

sectores de produção. São exemplos o fornecimento da água potável, gasóleo, água para o combate á incêndios, manutenção de bombas de água, filtros de gás, meios circulantes e outros. A área de gestão de serviços é responsável pelo controlo dos serviços da produção na redução. Tais serviços incluem o tratamento das poeiras e gases das caldeiras de alumínio, filtros dos centros de tratamentos de gases e outros.

A unidade de gestão de matérias primas é usada para monitorar três tipos principais de matérias primas na Mozal que são Coque, Alumina e Pitch. Este material é monitorado em duas situações, a de entrada no porto da matola e nos silos da Mozal em biloluane. O transporte físico deste material do porto para a Mozal é feito através de camiões próprios.

1.2.6. Serviços de gestão de stocks

Esta é a única unidade da Mozal que funciona totalmente no porto do Maputo na sua terminal de cargas da matola. O objectivo desta unidade é de controlar, registar e monitorar os lingotes de alumínio que são enviados da casa de moldes para os clientes através do porto. Esta unidade também controla os lingotes que por qualquer motivo sejam rejeitados pelos clientes já no porto. A produção da Mozal foi concebida para ser escoada através do porto da matola.

2. NÍVEIS DOS SISTEMAS INFORMAÇÃO

Sistemas de Informação da Mozal são constituído por instrumentos geradores de dados, os dados, interfaces entre sistemas, comunicações, procedimentos, as pessoas, unidades organizacionais algumas geograficamente dispersas que interagem no processamento automático e manual de informação para gestão do seu negócio.

Os sistemas de informação da Mozal estão divididos em níveis, conforme as sua funcionalidades, nomeadamente sistema do nível 0, sistemas do nível I, sistemas do nível II e sistemas do nível III. Esta classificação é de acordo com convenção da própria Mozal usada no grupo das empresas de fundição de metais pertencentes à Billiton.

2.1. Sistemas de Nível 0

Designam-se de sistemas de nível 0 aos instrumentos instalados nas diversas unidades de produção tais como: tanques de agua, silos, robots, caldeiras, grupos geradores, tanques de combustíveis, compressores de ar e outros. Estes sistemas produzem sinais que são interpretados no sistema do nível 1. Estes sistemas processam informação na forma analógica.

2.2. Sistemas de nível I

São os sistemas informáticos que comunicam directamente com sistemas do nível 0, através de meios próprios tais como contadores, interruptores(alarmes), ficheiros de interfaces, scanners e outros. Os sinais que entram nos sistemas de nível I não podem ser carregados manualmente mas são introduzidos com ajuda dos instrumentos físicos dos quais compõem os sistemas do nível 0. Estes sistema recebem os dados na forma analógica e convertem-nos em forma digital para os sistemas do nível II. Os sistemas de nível I são chamados de PLC (programables Logic Controllers) e a sua configuração, monitoração e controle está sob a responsabilidade de um sector que se chama de "automação".

2.3. Sistemas de nível II

Os sistemas de nível II são responsáveis por colectar dados provenientes do nível I, na forma digital, e transformá-los em informação sob forma de relatórios. Os sistemas de nível II comunicam directamente com os do nível I através de um programa de interface chamado de "Insql gateway". Além de se comunicarem com o nível I, estes sistemas comunicam entre si, assim como com o nível III através da replicação avançada de dados em Oracle(oracle advanced replication).

2.4. Sistema de nível III

O sistema de nível III é basicamente composto pelo sistema de gestão integrada SAP.

O objecto de estudo em termos de técnicas de interface e troca de informação será concentrado no interface entre os sistemas de nível II entre si e com o nível III.

3. INFRAESTRUTURAS DE SUPORTE

Os sistemas do nível 0 são constituídos por equipamentos de diversas funcionalidades que não são objecto do presente estudo. Vamos apresentar a infra-estrutura em termos de hardware, Software e redes de comunicação que suportam os sistemas dos níveis I, II e III.

3.1. Sistemas de Nível I

3.1.1. Servidores

Os sistemas de nível I são compostos por cerca de 44 servidores HP, com processadores Pentium III, 333 Megahertz de velocidade, 128 Megabytes de memória "Ram" e 2*8 Gigabytes de disco duro. Estes servidores estão munidos de duas cartas de rede, sendo uma Ligada a rede da planta e outra a Industrial. Estes servidores têm o sistema operativo Windows NT, versão 4 e Wonderware InSql Server versão 6.5.

3.1.2. Clientes

Os computadores - clientes dos sistemas de nível I são compostos por: Micro-computadores de marca HP, com processador Pentium II, 8 Gigabytes de disco duro e 128 Megabytes de "Ram". Windows NT versão 4.0, Wonderware Intouch WindowViewer 7.0 a correr um aplicativo desenvolvido usando Wonderware Intouch WindowMaker.

3.2. Sistemas de nível II

3.2.1. Servidores

Existem configurados para este nível 8 Servidores HP, Serie D270 2.0 PA8000, com um processador de 160 Megahertz, 4*9 Gigabytes de disco duro e 528 Megabytes de "Ram". Estas máquinas tem activa uma carta de rede física que está ligada a rede da planta. O seu sistema operativo é Unix 10.20 da HP para 32 bits. Sistema de gestão de bases de dados Oracle versão 8.0.5.1.0 com: Opções de partição de objectos, replicação avançada, indexes do tipo Bit-mapped, database queuing, Instead-of triggers, parallel backup and recovery, execução paralela, parallel load e Point-in-time tablespace recovery.

3.2.2. Clientes

Os computadores clientes do nível II são munidos de: Computadores de marca compaq, IBM e HP, com processador pentium II e III, mínimo de 2 Gigabytes de disco duro e uma memória mínima de 32 Megabytes de "Ram". Estes possuem uma carta de rede que os liga a rede da planta.

Estão equipados de Windows NT 4.0, Oracle developer 2000 versão 2.0 (run times) – formas, relatórios, gráficos e adaptadores para o sql*net. Para correr as aplicações de nível II os clientes conectam se ao servidor NT onde residem os programas da aplicação através da rede física. Para a conexão entre os clientes e servidores onde residem as bases de dados, foi configurada uma autenticação através do ficheiro tnsnames(anexo G) que corre no SQL*NET. O ficheiro tnsnames de cada cliente deve conter o endereço, a porta, o protocolo de comunicação e o nome da base de dados do servidor onde reside a base de dados com a qual se pretende comunicar.

3.3. Sistema de nível III

Este nível é composto por 4 servidores onde corre o sistema de gestão integrada SAP. A infra-estrutura de Hardware e software é semelhante a dos sistemas de nível II. A diferença está no facto da aplicação residir no mesmo servidor onde se encontra a base de dados, e os clientes estarem ligados ao servidor através de um software que se chama SAP GUI, cuja a comunicação com a base de dados é feita através da aplicação.

4. INTERFACE ENTRE OS SISTEMAS DA MOZAL

4.1. Interface entre Redução e os Sistemas do Nível II e com o SAP

O sistema de nível dois da redução foi criado e fornecido por uma empresa francesa (Pichney), especializada na área de fundição de alumínio. Como forma de protecção da sua tecnologia, e tendo em conta as especificações dos acordos celebrados com a DTI (empresa fornecedora de todos os sistemas de nível II e III para toda a Mozal), nenhuma replicação directa (usando o OAR) seria configurado. A "Picheney" prontificou –se a fornecer os dados necessários através de um ligador da base de dados que faz a conexão com a segunda base de dados da redução. O sistema de

redução tem interface com os sistemas da casa de moldes, laboratório e ambiente e finalmente com o sistema de gestão integrado SAP através da casa de moldes.

4.1.1. Interface com a Casa de Moldes

O sistema da unidade de redução envia informação para a casa de moldes sobre o plano de previsão das caldeiras. Enviado três vezes por dia, ou seja uma vez para cada turno A, B e C, contém dados relacionados com as caldeiras antes de serem enviados para casa de moldes. Também são enviados os resultados das análises do metal a ser enviado para a casa de moldes. No interface entre estes dois sectores são enviados os números das panelas, que virão com o metal quente, incluindo a quantidade do metal.

4.1.2. Interface com a Fábrica de Carbono

O sistema da unidade do carbono vai ler os dados relacionados com os ânodos enviados para a redução. Além destes dados o carbono precisa dos registos dos ânodos devolvidos ou rejeitados. Esta informação é enviada em cada 10 minutos.

4.1.3. Interface com o SAP

Este interface será usado para o envio de quantidades de matérias primas consumidas e de produtos produzidos na redução para o sistema comercial SAP. A redução é responsável pela produção do alumínio líquido, que é depois transferido fisicamente através de camiões com panelas próprias para a casa de moldes.

As matérias primas consumidas na redução são alumina, ânodos rodados e electricidade. Os consumos e as produções são sumarizados e enviados todos os dias para o SAP.

4.1.4. Funcionamento

O interface entre a Redução e a Casa de Moldes é feito através de um pacote "KF21_JOBS" (anexo D), escrito em PL/SQL, que por sua vez é executado através de um serviço programado na base de dados da casa de moldes. Este pacote contém dois procedimentos que incluem um ligador de base de dados. Os procedimentos basicamente fazem uma selecção de dados na base de dados da Redução e inserem-nos do lado da Casa de Moldes. Se os dados tiverem sido inseridos com sucesso, o

pacote remove-os do lado da Redução, e caso contrário estes se mantém intactos. Este mecanismo é similar ao usado entre a Redução e o Carbono. O interface com o SAP é feito através do mesmo pacote. Os dados para o SAP são copiados para a Casa de Moldes e são propagados para o SAP através da replicação em oracle. A base de dados da Casa de Moldes serve de ponto de transição de dados da Redução para o SAP

4.2. Interface entre o Sistema do Carbono com os sistemas de nível II e com o SAP

4.2.1. Interface com o SAP

Este Interface é usado para o envio de níveis de stock de matérias primas e de produtos acabados, produzidos na área de carbono para o SAP. A área do carbono consiste em 3 áreas individuais que são, past plant, Baking Furnace e Rodding Shop. O Paste Plant é responsável pela produção de ânodos verdes, sendo estes cozidos nos fornos do Baking Furnace, e depois enviados para o Rodding Shop. Os ânodos são rodados e armazenados até ao seu consumo pela redução para produção do alumínio liquido. As matérias primas guardados no Paste Plant são coque, Pitch, ânodos verdes reciclados. Este interface foi programado para ser transmitido no último dia de cada mês. A fábrica de carbono como propagadora não tem interfaces com o nível II.

4.3. Interface entre o Sistema do Casa de Moldes com os sistemas de nível II e com o SAP

4.3.1. Interface com a Redução

A casa de moldes tem necessidade de enviar os dados para os serviços de redução referentes aos resultados dos planos de previsão das caldeiras. Estes resultados são enviados logo depois de gerados com base na informação recebida da redução. Também são enviados para redução informação sobre os pesos das panelas uma vez capturadas na casa de moldes.

4.3.2. Interface com o Sistema de Gestão de Stocks -SPMS

O sistema da casa de moldes envia para o sistema de gestão de stocks-SPMS informação acerca dos moldes logo que se termina com o processo da moldagem. É

também enviado o número do lingote, peso e a sua qualidade. Além desta informação a casa de moldes envia dados relacionados com os lingotes finais quando estes são aceites para serem despachados para o porto. Estes dados incluem os detalhes dos lingotes, a linha de moldagem do qual são originários e a identificação camião que os vai carregar da fábrica para o Porto.

4.3.3. Interface com o SAP

Este interface será usado para a o envio de níveis de stocks e de produtos produzidos na casa de moldes para o SAP. A casa de moldes é responsável pelo molde do alumínio liquido e lingotes. Estes lingotes são agrupados em bundles (44 lingotes = 1 bundle). Os níveis de stock são avaliados para o metal líquido dentro dos fornos sob forma de bons e maus lingotes produzidos, sendo estes últimos a serem rejeitados pelo controlo de qualidade. Esta informação é transmitida no último dia de produção de cada mês.

4.3.4. Funcionamento

O interface entre a casa de moldes e o sistemas de nível II da redução é feito através de um serviço que usa um ligador da base de dados. Este serviço foi configurado do lado do sistema de redução. Para a comunicação entre a casa de moldes com o SPMS e com o SAP foi utilizada a replicação em oracle.

4.4. Interface entre o Sistema dos Serviços de Redução com os Sistemas de nível II e com SAP

4.4.1. Do IBL para o SAP

Este interface é usado para a transferência de dados referentes ás matérias primas físicas dos navios para os silos da Mozal. Os dados sao gerados pelo sistema da logística (IBL) para o sistema de gestão SAP da Mozal. O Sistema de IBL é responsável pelas importações de matérias primas para a produção, que vêem directamente para o porto da Matola, (Terminal de cargas) e que depois são carregada em camiões cisternas para a fábrica da Mozal. O abastecimento em matérias primas é sumarizado todos os dias de produção e enviado ao SAP no fim do mês.

4.4.2. Da Subestação para o SAP

Este interface será usado para o envio dos dados relacionados com as quantidades de consumo em energia eléctrica total e por sector para o SAP. O maior destaque será dado ao consumo de energia no sector de redução já que esta é uma das principais matérias primas para produção do alumínio. A subestação é responsável pelo manuseamento da energia consumida por todos os sectores da MOZAL. As quantidades consumidas será acumuladas sendo estas enviadas no fim de cada mês via interface.

4.5. Interface entre Laboratório e os Sistemas do Nível II

Existem em funcionamento Interfaces com os seguintes sectores de produção na empresa: carbono, casa de moldes, serviços de redução, redução e meio ambiente.

Os dados trocados com estes sectores são basicamente resultados das amostras submetidos por estes sectores ao laboratório para analises. A submissão é feita no sector do laboratório.

Para o interface com a casa de moldes e carbono é usado o mecanismo da replicação de dados em Oracle(OAR). Entre o laboratório e o sistema do meio ambiente é usado um trigger enquanto para o envio de dados para o sistema da redução é usado um ligador de base de dados (database link).

4.5.1. Configuração

Já que a comunicação necessária entre o laboratório e as outras unidades de produção é numa direcção, isto é do Laboratório para esses sectores, então para os sectores que usam a replicação foram criados utilizadores de interface ou seja propagador e recebedor com as mesmas contas no Laboratório e nos sectores de destinos.

Para os sectores que não usam replicação em oracle, para o sistema da redução foi criado um utilizador que recebe e guarda os dados a serem copiados. O mesmo acontece para o utilizador do meio ambiente o qual foi criado um trigger, que logo depois da publicação de um resultado da análise duma amostra, carrega

automaticamente para a aplicação do meio ambiente. A aplicação do meio ambiente e do laboratório estão no mesmo servidor.

Foram criadas tabelas de interface que guardam todos os dados a serem replicados para os sectores de destino. O sistema de laboratório escreve os resultados das análises nestas tabelas da base de dados do laboratório. Depois deste processo a replicação transfere estes dados para áreas idênticas de interface nos sectores de destino. As aplicações de destino são responsáveis pela interpretação, processamento e consolidação destes resultados nas respectivas tabelas das bases de dados.
Oracle Database Link é usado para o estabelecimento da comunicação entre o Laboratório e o sistema da redução. O laboratório escreve o resultado das análises para tabelas de interface residentes na base de dados do laboratório. O sistema da redução tem o acesso a estes resultados através do database link, e uma conta separada com permissões para ler e apagar nestas tabelas de interface. O sistema da redução é responsável pela extracção de dados nas tabelas de interface da base de dados do Laboratório.

4.6. Interface do SAP para os sistemas de nível II

Este sistema envia apenas dados para o sistema de gestão de stocks que se localiza no porto. Os dados transmitidos são os pedidos de encomendas dos lingotes de alumínio.
A semelhança dos outros interface entre o nível III e o nível II, usa-se a replicação em oracle para a transmissão de dados. Este interface ocorre todos os dias de trabalho.

4.7. Interface entre Sistema SPMS com SAP

4.7.1 Para o SAP

O SPMS envia para o SAP a confirmação da satisfação total ou parcial dos pedidos de encomenda.
É também parte deste interface a informação sobre os lingotes de alumínio confirmados pelos accionistas como tendo sido embarcados e pagos no porto.

IV. AVALIAÇÃO DO MODELO IMPLEMENTADO NA MOZAL

1. RESUMO DA REPLICAÇÃO CONFIGURADA NA MOZAL

O interface entre o nivel2 e o sistema de gestão integrada SAP é feito com base na replicação multi-mestre que funcionará da seguinte maneira (ver Anexo E): Os dados são inseridos pela aplicação nas tabelas do nível 2, cujo o dono e ao mesmo tempo o propagador depois, segundo um intervalo de tempo pré determinado pela variável próximo, os dados são replicados para uma tabela análoga na base de dados do sistema comercial SAP.

De acordo com a teoria da replicação multi-mestre, esta é dada em múltiplos sentidos, isto e, qualquer mudança feita num ponto será replicada no outro. Para minimizar o uso intenso dos recursos dos servidores e o risco de replicação no sentido sap nível II, os registos do lado do sap são copiados com o uso de um Trigger da base de dados para uma tabela de trabalho cujo o dono neste caso e o sap e não o recebedor. Esta tabela é usada pela aplicação para o processamento da informação. Nenhum processamento será feito nas tabelas do recebedor da replicação. Este modelo de replicação é também usado na replicação entre os sistema de nível II, com a excepção dos que usam directamente pacote ou um trigger. Para os sistemas que tem o interface directo com o sistema de redução é usado a replicação procedural.

2. MÉTODO DE RESOLUÇÃO DE CONFLITO IMPLEMENTADO

Em todos os interfaces dos sistemas, a replicação em oracle na Mozal, foi configurado o método de tempo mais tarde da ocorrência(latest timestamp value). Este método consiste na propagação dos dados cujo tempo da ocorrência é o mais recente possível.

3. PONTOS FORTES DA CONFIGURAÇÃO

Para a replicação, todos os servidores envolvidos no processo de replicação estão localizados no mesma zona geográficas ou seja a distancias não superior a 10 km.

- A existência de uma estrutura de rede redundante(primário e secundário), sendo que uma funciona quando se verifica a falha da outra e vice-versa.

- O sistema que gere o core-busniss que é o da redução está isolado de todos sistemas de produção, podendo este laborar mesmo com a paragem dos outros.
- Todas as aplicações tem um interface manual como backup. Ou seja todos os dados replicados podem ser gravados num media e inseridos com ajuda de ferramentas da oracle em caso da suspensão da replicação.
- Todas aplicações têm tabelas históricas, o que permite a recuperação e reenvio dos dados caso haja uma remoção acidental de um lado da replicação.
- Todos os servidores tem como sistema operativo o UNIX da HP na sua versão 10.20 e com o sistema de gestão de bases de dados ORACLE versão 8.0.5.1.0
- A Mozal tem um contrato de manutenção com os fornecedores do Software referenciado nos parágrafos anteriores e que inclua o fornecimento de ferramentas de administração .
- A existência de técnicos com experiência em ORACLE e HP-UNIX.

Para o método de resolução de conflitos, os dados só fluem numa direcção, e para o caso bidireccional optou-se por construir duas replicações diferentes, sendo uma para cada direcção. Todos os servidores se encontram na mesma zona geográfica e que o "time zone" não constitui motivos para preocupação.
Os relógios de todos os servidores envolvidos estão sincronizados com base no relógio dum sistema da subestação.

4. PONTOS FRACOS DA CONFIGURAÇÃO

O grande problema da configuração da replicação escolhida na Mozal é o facto de esta ser multi-master, o que significa que para cada mudança em uma linha da tabela mestre, é replicado para as tabelas remotas o conteúdo de toda a tabela, o que provoca sérios problemas de sobrecarga na rede física, quando se tratarem de tabelas com mais de 1000 registos.
Este modelo contém dois pontos de administração e consequentemente dois pontos de possíveis erros, isto é, nesta configuração são criados esquemas separados para os dois pontos de replicação o que por sua vez requer uma dupla administração. É necessário calcular os parâmetros de armazenamento, indexação, crescimento e administrar o crescimento e a remoção em cada um dos lados de forma separada.

O mesmo desenho apresenta graves motivos de segurança, já que para cada um dos lados da replicação e apesar de ser em um só sentido os dados podem ser removidos pelos donos de cada esquema. Alem de poder remover os dados o dono de cada esquema pode remover as tabelas da replicação, provocando erros. Este tem sido um problema muito grande já que para casos de eliminação de registos, após a sua utilização é necessária que seja feita ao mesmo tempo em todas as tabelas corespondentes.

O sistema de redução foi desenhado é construído por empresa francesa encarregue de fornecer a tecnologia de alumínio e por motivos legais (protecção da patente), não pode colocar a disposição da Mozal qualquer acesso ao pessoal administrador da Mozal.

A ausência total ou parcial da documentação sobre os modelos e as configurações actuais que pode ser resultado do não comprimento dos prazos por parte do empreiteiro e pela pressão dos accionistas.

O administrador do sistema da redução não se encontra em maputo e esta é feita de forma remota, o que provoca demoras caso haja problemas por resolver.

A falta de empresas ou organizações em maputo que tenham já implementado a replicação de dados em oracle o que dificulta a troca de experiências e ou celebração de parcerias nesta matéria.

A inexistência de uma representação da oracle em maputo, o que de certa forma cria problemas para o caso de necessidades urgentes de suporte.

A replicação com o porto da motala, local que dista do resto da empresa de cerca de 15 Km, cuja a linha de rádio apresenta problemas constantes de avarias e flutuações, sendo que a replicação seja necessária 24 horas por dia.

A existência de alguns "bugs" reconhecidos pela oracle para a replicação na versão 8.0.5.1.0 (ex: 519980; 1034224.6 no http://metalink.oracle.com/- 22-10-2001).

Para o interface entre o sistema de redução com o SAP, usa-se um pacote(replicação procedural) que selecciona os dados e armazena na base de dados da casa de moldes e desta replica para o SAP. Contudo este esquema apresenta deficiências devido ao envolvimento de um ponto intermédio, que poderia ser eliminado. Caso haja qualquer problema entre a casa de moldes e a redução seja da ligação ou da base de dados este iria se reflectir no interface entre a redução e o SAP. Este ponto intermédio aumenta também as tarefas do administrador, já que tem que velar por uma ligação a mais e providenciar um espaço para acomodar os dados que vêem da unidade de redução para o SAP.

Para o método de resolução de conflitos, se por um motivo qualquer o administrador do sistema operativo, alterar o relógio da máquina da subestação, a fabrica vira um caos em termos de dados em todos interfaces.

5. MÉTODOS PROPOSTOS PARA CADA INTERFACE

Para a replicação entre o laboratório e os sistemas da casa de moldes e carbono. Entre estes sistemas seria aconselhável a configuração da replicação usando read-only-snapshot (ver Anexo F). O master site deve se localizar no sistema do laboratório e os snapshots sites seriam os sistemas de carbono e da casa de moldes. Este modelo tem o seu sustento no facto de os dados replicados serem resultados das amostras recentes não havendo necessidade de replicar a tabela inteira como acontece actualmente. O volume das amostras é muito ínfimo rondando aos 40 a 50 amostras diárias.

Entre o sistema do laboratório e o sistema do meio ambiente, a configuração de um trigger, bem se ajusta já que estes sistemas residem no mesmo servidor e na mesma base de dados não havendo necessidade da utilização da rede física ou de propagação em deferido.

O sistema do laboratório tem um database link configurado com o sistema da redução que também encontra o seu sustento no facto de por forca dos acordos não poder ser possível fazer qualquer configuração no sistema da redução, pois se tal fosse possível

aconselharse-ia a utilização da replicação com base no read-only-snapshot, com o master site localizado no sistema do laboratório e o snapshot site seria localizado no servidor do sistema de redução.

Para a replicação entre o sistema da redução e o SAP deve ser configurado uma replicação com o uso de read-only-snapshot. O master site deve ser localizado do lado da redução e os snapshot site deve estar do lado do sistema de gestão integrada SAP.

Na replicação entre o Sistema de Gestão de Stocks no Porto e o SAP deve ser configurada uma replicação avançada com o uso de snapshots actualizáveis. Esta deve ser configurada de forma que o snapshots site esteja localizado lado do SAP e o ponto Mestre do lado do Sistema de Gestão de Stocks do Porto.

Para a replicação entre a subestação de energia e o sistema de gestão SAP deve ser configurada uma replicação básica com o uso read-only-snapshots, que deve ter o master site o sistema da subestação e o snapshot site o sistema de gestão SAP.

Na replicação entre o Sistema da Casa de Moldes e o Sistema de Gestão de Stocks deve se optar pela replicação básica com o uso de read-only-snapshot, com o sítio mestre localizado no sistema da casa de ,moldes e o snapshot site no sistema de gestão de stocks. A mesma configuração deveria ser usada para a replicação entre a casa de moldes e o sistema da redução. (O Anexo F mostra a replicação proposta.)

V. CONCLUSÕES E RECOMENDAÇÕES

1. CONCLUSÕES

O objectivo principal deste trabalho era estudar a problemática sobre a replicação de dados entre bases de dados oracle bem como os métodos de resolução de conflitos derivados da replicação na MOZAL. Este foi integralmente alcançado no final deste estudo demonstrando um conjunto de constraints nos métodos de replicação em cada interface dos sistemas bem como a desvantagem do uso do método de resolução dos conflitos.

A Oracle providencia três tipos de replicação de dados nomeadamente a configuração multi-mestre, snapshot actualizáveis e configuração híbrida. Cada tipo deve ser aplicado dependo das características de interface entre os sistemas. Consoante o tipo da replicação são destinguidos diferentes métodos de resolução de conflitos.

Para a configuração do ambiente da replicação em Oracle devem ser considerados os seguintes aspectos: volume de dados envolvidos, a frequência de propagação de dados, o tipo de dados, o tipo de objectos, as regras do negócio, as versões do RDBMS, hardware e as comunicações.

A Mozal possui sistemas de informação classificados em níveis conforme o tratamento de dados. O modelo de replicação de dados adoptado na Mozal faz-se à necessidade de estabelecer as ligações entre os diferentes sistemas do nível II e III. Na prática, o modelo apresenta na maioria das ligações entre sistemas a replicação multi-mestre. Em outras ligações recorreu-se a um pacote para fazer a ligação entre as base de dados. Este pacote resolve por si só a ocorrência de conflitos. E para o primeiro caso, aplicou-se o método de ultimo tempo da ocorrência.

O modelo de replicação de dados implementado na Mozal apresenta inconvenientes relacionados com custos de administração, segurança, acesso, falta de documentação. Também apresenta vários aspectos positivos.

2. RECOMENDAÇÕES

Para a resolução dos problemas do modelo implementado na Mozal será preciso fazer:

- Substituição do Radio Link que faz a ligação entre a Mozal e o Porto de por meio de comunicação física mais eficiente, por exemplo a fibra óptica.
- Upgrade da versão do RDBMS para 8.i (8.1.7) que resolve os bugs da Oracle referenciados neste estudo.
- Configuração da replicação com base em snapshot no lugar da multi-mestre. Estes podem ser de simples leitura ou de leitura e escrita.
- Interceder junto dos Consultores implementadores do projecto para a entrega da documentação actualizada e correcta.
- Upgrade do hardware e do sistemas operativos
- Formação
- Manutenção do método de resolução de conflitos, mesmo com as alterações propostas

BIBLIOGRAFIA

Bibliografia Referenciada

1. Gosset , Scott & Jang, Susan (1999), **ORACLE8 Avanced Replication** – Oracle Corporation. **[Gosset, Jang (1999)]**
2. Thorpe, Heidi (2000), **ORACLE 8*i* Tuning and Administration,** Addison-Wesley Pub. **[Thorpe (2000)]**
3. Delmolino, Dominic J. (1995), **Strategies and Techniques for using Oracle7 Replication,** Osborne McGraw-Hill. **[Delmolino (1995)]**
4. Bobrowski, Steve & Smith, Gordon (1997), **Oracle8 Replication - Oracle Corporation. [Bobrowski, Smith (1997)]**
5. [Document ID MZAT-007, Dti CONSORTIUM, Novembro 1999].

Bibliografia Consultada

6. Ault, Michael (2000), **Oracle8I Administration and Management,** Addison-Wisley
7. Koletze, Peter & Dorsey, Paul (1999), **Oracle Designer Handbook,** 2nd Edition, Oracle Press.
8. Date, C. J. (2000), **An Introduction to Database Systems**, 7th Edition, Addison-Wesley.
9. McFadden, F. R.; Hoffer, J. A. & Prescott, M. B. (1999), **Modern Database Management**, 5th Edition, Addison-Wesley.
10. Singh, Lave; Leigh, Kelly & Zafian, Joe (1997), **Oracle 7.3 Developer´s Guide** - Sum Publishing.
11. Loney, Kevin (1995), **Oracle DBA Handbook,** Osborne McGraw-Hill.
12. Loney, Kevin & Koch (2000), **ORACLE 8*i* The Complete Reference,** Osborne McGraw-Hill.
13. Lewis, Jonathan (2001), **ORACLE 8*i* Building efficient Databases,** Addison-Wesley Pub.
14. Niemiec, Richard J.; Bradley, Joe Trezzo & Brown, D. (2000), **Oracle Performance Tuning Tips & Techniques,** Osborne McGraw-Hill.
15. Feuerstein, Steven; Dye, Charles & Beresniewicz, John (1998), **Oracle Built-In Packages,** O'Reilly & Associates.

16. Aronoff, Eyal; Loney, Kevin & Sonawalla, Noorali (1999), **Advanced Oracle Tuning and Administration,** Osborne McGraw-Hill.
17. Shlaer, Sally & Mellor J. Stephen (1988), **Object –Oriented System Analysis Modeling the Word in Data**, Prantice-Hall
18. Coulouris, Dellimore & Kindberg(1994), **Distributed System**, 2nd Edition, Addison Wesley.
19. Ahmad, Ammar, Cheung(1994), **Replicated Data Management in Distributed Systems**, IEEE CS Press
20. Oracle7 Server. **Distributed System(1996)**. Relese 7.3, Oracle Press.
21. Oracle 7. **Distributed database technology & symmetric replication**, oracle press

Sites

1. http://safari1.oreilly.com (12/05/2002)
2. http://technet.oracle.com/products/oracle7/pdf/st_techsr.pdf (09/02/2001)
3. http://metalink.oracle.com(23/12/2000)
4. http://www.tusc.com(20/12/2001)
5. http://www.oracleprofessionalnewsletter.com(02/05/2002)
6. http://www.ixora.com.au(01/01/2001)
7. http://www.animatedlearning.com
8. http://www.confio.com(01/01/2001)
9. http://www.spawnware.com(01/05/2001)
10. http://www.dbazine.com(09/02/2001)
11. http://guidedogtech.com(01/03/2001)
12. http://www.stfb.com/foraclesource.html(12/10/2001)
13. http://orafocus.com(23/11/2001)
14. http://www.ubtools.com(23/05/2001)
15. http://www.orafaq.com(01/01/2001)
16. http://www.oraclenotes.com(12/12/2001)
17. http://www.dbasupport.com(30/03/2001)

ANEXO A

Ficheiro de parâmetros de Inicialização e Configuração da Base de Dados

```
# inclui os parâmetros de configuração da base de dados
ifile     = /oracle/app/oracle/admin/OBL/pfile/configOBL.ora

rollback_segments=
(ROLL01,ROLL02,ROLL03,ROLL04,ROLL05,ROLL06,ROLL07,ROLL08)

#####################################################################

# parâmetros de performance

db_files = 254     # Numero  total de ficheiros da base de dados

db_file_multiblock_r ead_count = 32     # Numero máximo de blocos possíveis de ler
db_block_buffers = 1600                 # Quantidade máxima de buffers

shared_pool_size = 30000000                    #Tamanho do shared_pool

log_checkpoint_interval = 9999999              #Intervalos entre dois checkpoints

processes = 120                         # Numero máximo de processos

dml_locks = 100                         # Numero de locks possíveis por sessão

log_buffer = 131072                            # Tamanho do  redo log buffer

sequence_cache_entries = 10                    # Numero máximo de caches que
podem ser definidos numa sequência

OPEN_CURSORS = 500                      #Numero máximo de cursores que podem ser
abertos

SESSION_CACHED_CURSORS=10     # Numero de cursores que podem ser
colocados na memória em cada sessão

remote_login_passwordfile= exclusive    #Conexão através do internal

compatible =8.0.6.2.0                          # Versões compatíveis para a
aplicação de patches

sort_area_size = 1572864                # tamanho da memória reservado para o
sorteamento

max_dump_file_size = 10240              # limite do  ficheiro da audição

log_archive_start = true                       #  Para se arquivarem os logs
```

```
log_archive_dest = /archive/                    #directório onde os logs serão
arquivados

log_archive_format = "T%TS%S.ARC"  # Formato a dar aos arquivos

global_names = FALSE                            # Uso de global names

# Parâmetros da replicação avançada

job_queue_processes                 = 4         # Numero de bichas de serviço (SNP)

job_queue_interval                  = 100  #Intervalos entre a execução das transacções
da bicha

parallel_max_servers                        = 10    # Numero máximo de s

distributed_lock_timeout            = 60   #Tempo de espera de um recurso fechado
nos sítios remotos

distributed_transactions            = 5             # numero máximo de transações
distribuídas

open_links                  = 4             #Numero máximo de database links activos
por sessão

snapshot_refresh_interval           = 60 # Tempo de refrescamento dos snapshotes
em segundos
```

ANEXO B

Parâmetros de Formas de Propagação de Dados

Enabled (Activo)

Verifica o activo imediato e caso seja afirmativo puxa os dados de acordo com o intervalo.

Propagação em paralelo (Parallel Propagation)

Verifica o tipo de propagação a ser utilizada. Se estiver activo a propagação é paralela e caso contrário a propagação é em serie.

Processos (Processes)

O número de processos em background que o ligador irá usar na propagação em paralelo. O valor por default é 0.

Demora em segundos (Delay Seconds)

O tempo total para o processo descansar depois de ter puxado a fila das transações em deferido locais, para o ponto remoto, mesmo que esta esteja vazia.

Tamanho do batch (Batch Size – apenas base de dado Oracle7)

Determina qual é a frequência do commit das transações quando estiver a puxar a bicha das transações em deferido locais, por default o valor 0 é assignado.

Parar no erro (Stop on Error)

Como reagir depois de detectar um erro durante o processo de puxar as transações em deferido. Se estiver activo significa que Oracle para de puxar as transações em deferido quando detectar um erro. Caso contrário oracle puxa, ignorando os erros.

Programando para puxar continuamente.

Normalmente puxa-se de forma assíncrona, ou seja não em tempo real. Contudo mesmo usando o mecanismo de replicação assíncrona é possível configurar uma ligação programada para simular uma replicação continua e em tempo real. Para tal é suficiente colocar o valor 500,000 para a variável "delay seconds"

Limpeza e manutenção da fila de espera

Linhas principais para a programação da limpeza das transações em deferidas na fila de espera.

Uma programação de limpeza determina como os mestres ou snapshots sites irão limpar certas transações em deferido da fila de espera.

Oracle cria um serviço que deve ser executado por intervalos de tempo especificados para a realização regular da limpeza local. Deve–se cuidadosamente analisar como vai ser feita esta limpeza antes de configurar.

Os factores que influenciam esta programação são o hardware, disponibilidade e tráfego da rede física, tamanho e espaço disponível na base de dados .

As seguintes variáveis são de configuração é obrigatória

próxima data (Next Date)

A próxima vez na qual a limpeza das transações bem sucedidas ocorrerá.

Interval Expression

Intervalo automático para a limpeza das transações bem sucedidos, na fila de espera de transações em diferidos locais.

Segmento de cancelamento (Rollback Segment)

O segmento de cancelamento a ser utilizado aquando da realização do trabalho de limpeza das transações em deferidos. Se não for especificado, oracle irá automaticamente assignar um rollback segment para cada limpeza .

Se a situação for tal que, os dados a serem eliminados for maior deve ser observado o tamanho e quantidade dos segmentos de cancelamento, caso contrário oracle irá produzir uma mensagem de erro, abortando a operação.

Demora em segundos (Delay Seconds)

Especifica tempo total no qual oracle continua a fazer consulta da fila de espera, mesmo se a fila estiver vazia. É importante para reduzir sobrecargas quando a limpeza ocorrer com muita frequência . Se houver necessidade de limpeza contínua é só fazer este parâmetro igual á 500000.

Considerações sobre tipos de dados nas tabelas replicadas

A replicação em oracle suporta a replicação de dados das tabelas com os seguintes tipos :NUMBER, DATE, VARCHAR2, CHAR, NVARCHAR2, NCHAR, RAW, ROWID.

Oracle também suporta replicação de tabelas com os seguintes tipos de grandes objectos: LOBs (BLOBs), caracteres LOBs (CLOBs), e caracteres nacionais LOBs (NCLOBs). O mecanismo de chamamento de procedimentos remotos usado na replicação propaga apenas quando inserções, actualizações ou remoção parciais ocorrem para este tipo de variáveis.

Oracle8 não suporta replicação de variáveis com tipo LOB no caso de alguns pontos utilizarem as versões Oracle7 na edição 7.3.

Oracle não suporta a replicação de dados das colunas do tipo LONG e LONG RAW. Oracle simplesmente omite as colunas contendo este tipo de variáveis nas tabelas de replicação.

Oracle também não suporta objectos com tipos definidos por utilizadores, externos ou baseados em tipos LOBs(BFILEs). Qualquer tentativa de configurar tabelas contendo estes tipos de variáveis como mestre vai terminar em erro.

Propagação em série e propagação em paralelo

Na propagação em serie, oracle propaga de forma assíncrona as transações replicadas, uma de cada vez, da mesma forma e sequência a que foram submetidos- First in-first out (FIFO). Esta é a opção por default na configuração da replicação em oracle.

Na propagação em paralelo, oracle propaga de forma assíncrona as transações replicadas usando o "múltiplo trânsito de fluxo paralelo sem paragem". Caso seja necessário, oracle ordena as execuções de transações dependentes de forma a garantir a integridade global da base de dados. A propagação em paralelo é uma opção de performance disponível apenas para oracle8, para que tal seja possível é necessário que todos os pontos usem esta versão.

ANEXO C

Metodos de resolucao e notificacao de conflitos definidos pelo usuario

1. Exemplos de um métodos de resolução de conflitos definidos pelo utilizador

Este método de resolução de conflito, e um exemplo do método pré construído do tipo MAXIMO. E de notar que contrariamente aos métodos standard, este pode conter valores nulos nas colunas usadas para a resolução de conflitos.

1. 1. A função do método máximo

```
-- Função do usuário similar ao método MAXIMO.
-- Se corrente for nulo ou corrente < novo, use o valor novo.
-- Se novo for nulo ou novo  < corrente, use o valor corrente.
-- Se ambos forem nulos, não ha resolução.
-- Não converge para 2 ou mais mestres excepto quando o valor sempre
--aumentar.

FUNCTION max_null_loses(old                  IN    NUMBER,
                        new                  IN  OUT  NUMBER,
                        cur                  IN    NUMBER,
                        ignore_discard_flag OUT    BOOLEAN)
  RETURN BOOLEAN IS
BEGIN
   IF (new IS NULL AND cur IS NULL) OR new = cur THEN
       RETURN FALSE;
   END IF;
   IF new IS NULL THEN
       ignore_discard_flag := TRUE;
   ELSIF cur IS NULL THEN
       ignore_discard_flag := FALSE;
   ELSIF new < cur THEN
       ignore_discard_flag := TRUE;
   ELSE
       ignore_discard_flag := FALSE;
   END IF;
   RETURN TRUE;
END max_null_loses;
```

1.2. A função do método aditivo

Este método de resolução de conflito e um exemplo do método pré construído do tipo ADDITIVO. é de notar que contrariamente aos métodos standard, este pode conter valores nulos nas colunas usadas para a resolução de conflitos.

```
-- Função do utilizador semelhante ao método aditivo.
-- Se old for null, old = 0.
-- Se new for null, new = 0.
-- Se curr for null, curr = 0.
-- new = curr + (new - old) -> Como o método ADITIVO.

FUNCTION additive_nulls(old                  IN    NUMBER,
                        new                  IN  OUT  NUMBER,
```

```
                        cur                    IN    NUMBER,
                        ignore_discard_flag OUT    BOOLEAN)
   RETURN BOOLEAN IS
   old_val NUMBER := 0.0;
   new_val NUMBER := 0.0;
   cur_val NUMBER := 0.0;
BEGIN
   IF old IS NOT NULL THEN
      old_val := old;
   END IF;
   IF new IS NOT NULL THEN
      new_val := new;
   END IF;
   IF cur IS NOT NULL THEN
      cur_val := cur;
   END IF;
   new := cur_val + (new_val - old_val);
   ignore_discard_flag := FALSE;
   RETURN TRUE;
END additive_nulls;
```

2. Exemplo de um programa para a Notificação

2.1. Criação de uma tabela de registo para as notificações

A tabela seguinte pode ser usada para registar notificações de conflitos provenientes de varias tabelas no sitio mestre.

```
CREATE TABLE conf_report (
 line          NUMBER(2),     --- usado para ordenar as mensagens
 txt           VARCHAR2(80),  --- mensagem da notificação
 timestamp     DATE,          --- Hora  do conflito
 table_name    VARCHAR2(30),  --- tabela na qual o conflito ocorreu
 table_owner   VARCHAR2(30),  --- Dono da tabela
conflict_type VARCHAR2(6)    --- INSERT, DELETE ou UNIQUE
)
```

2.2. Criação do pacote para a notificação

```
CREATE OR REPLACE PACKAGE notify AS
 -- Reporta a violação do conflito de unicidade na tabela clientes
 FUNCTION customers_unique_violation (
   first_name       IN OUT VARCHAR2,
   last_name        IN OUT VARCHAR2,
   discard_new_values IN OUT BOOLEAN)
 RETURN BOOLEAN;
END notify;
/

CREATE OR REPLACE PACKAGE BODY notify AS
Define a tabela em PL/SQL para guardar as notificações
 TYPE message_table IS TABLE OF VARCHAR2(80) INDEX BY BINARY_INTEGER;

 PROCEDURE report_conflict (
   conflict_report IN MESSAGE_TABLE,
   report_length   IN NUMBER,
   conflict_time   IN DATE,
   conflict_table  IN VARCHAR2,
```

```
   table_owner      IN VARCHAR2,
   conflict_type    IN VARCHAR2) IS
 BEGIN
   FOR idx IN 1..report_length LOOP
     BEGIN
       INSERT INTO sales.conf_report
         (line, txt, timestamp, table_name, table_owner, conflict_type)
         VALUES (idx, SUBSTR(conflict_report(idx),1,80), conflict_time,
                 conflict_table, table_owner, conflict_type);
     EXCEPTION WHEN others THEN NULL;
     END;
   END LOOP;
 END report_conflict;

--Este é o método de resolução de conflitos que será o primeiro a ser --
chamado
-- Se houver violacao de constraint unico na tabela customers
 FUNCTION customers_unique_violation (
   first_name  IN OUT VARCHAR2,
   last_nameIN OUT VARCHAR2,
   discard_new_valuesIN OUT BOOLEAN)
  RETURN BOOLEAN IS
   local_node  VARCHAR2(128);
   conf_report MESSAGE_TABLE;
   conf_time   DATE := SYSDATE;
  BEGIN
  -- Apanha o nome do sitio local
    BEGIN
      SELECT global_name INTO local_node FROM global_name;
    EXCEPTION WHEN others THEN local_node := '?';
    END;
  -- Gerando a mensagem para o ABD
  conf_report(1) := 'UNIQUENESS CONFLICT DETECTED IN TABLE CUSTOMERS ON ' ||
                    TO_CHAR(conf_time, 'MM-DD-YYYY HH24:MI:SS');
  conf_report(2) := '  AT NODE ' || local_node;
  conf_report(3) := 'ATTEMPTING TO RESOLVE CONFLICT USING ' ||
                    'APPEND SEQUENCE METHOD';
  conf_report(4) := 'FIRST NAME: ' || first_name;
  conf_report(5) := 'LAST NAME:  ' || last_name;
  conf_report(6) := NULL;
  --- Reportando oconflito
  report_conflict(conf_report, 5, conf_time, 'CUSTOMERS',
                  'OFF_SHORE_ACCOUNTS', 'UNIQUE');
  --- Nao abandonar o valor novo da coluna pois ainda é necessario       ---
para os outros metodos de resolucao de conflitos
  --- other conflict resolution Methods
  discard_new_values := FALSE;
  --- Indicacao de que oconflito nao foi resolvido.
    RETURN FALSE;
  END customers_unique_violation;
END notify;
/
```

ANEXO D

Pacote de Replicação entre a Casa de Moldes e a Redução

```
PACKAGE KF21_JOBS IS
/* É preciso delcarar um package para correr o procedimento da replicacao procedural*/
    Procedure PF01_COPY_REDUCTION;
End KF21_JOBS;

PACKAGE BODY KF21_JOBS AS
Procedure PF01_COPY_REDUCTION Is
    err_no          number;
    last_datpos     Date;
    last_indsdatpos Number (10);
    do_copy         Boolean;
    /* Cursor para apanhar todos os planos. */
    Cursor ALL_WO_SCHEDULE Is
                Select  DISTINCT DHSCH,
                                 DATPOS,
                                 INDSDATPOS
                           From  CAST_WO_SCHEDULE@RED_CHS
                       ORDER BY DHSCH DESC;
    Cursor SINGLE_WO_SCHEDULE (P_DHSCH       Date,
                               P_DATPOS      Date,
                               P_INDSDATPOS  Number)
                     Is Select  DHSCH,
                                CODPOT,
                                DATPOS,
                                INDSDATPOS,
                                METHEO,
                                AGE,
                                SCHED
                           From  CAST_WO_SCHEDULE@RED_CHS
                          Where  SCHED In ('S','D')
                            And  DHSCH      = P_DHSCH
                            And  DATPOS     = P_DATPOS
                            And  INDSDATPOS = P_INDSDATPOS;
    Cursor SINGLE_WO_PURITY (P_DHSCH  Date)
                    Is Select A.DHSCH,
                              A.CODPOT,
                              A.CODDET,
                              A.VALDET
                         From CAST_WO_PURITY@RED_CHS    A,
                              CAST_WO_SCHEDULE@RED_CHS  B
                        Where A.DHSCH  = P_DHSCH
                          And A.DHSCH  = B.DHSCH
                          And A.CODPOT = B.CODPOT
                          AND B.SCHED In ('S','D')
                       Order By A.DHSCH;
    Cursor WO_TAPMASS Is      Select  DHCAL,
```

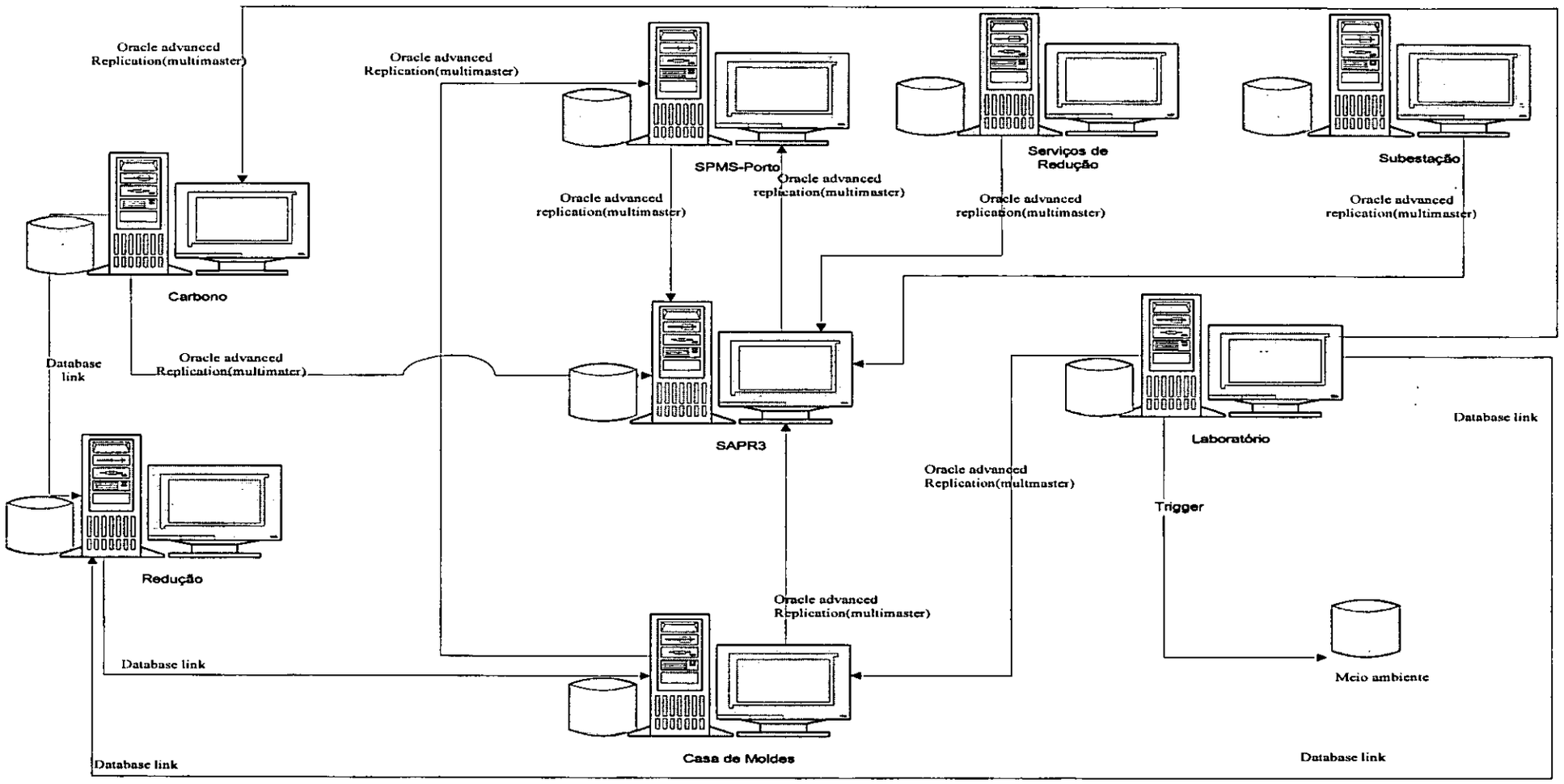
```
                                        Upper (CODLOG) CODLOG,
                                        DATPOS,
                                        INDSDATPOS,
                                        CODPHY,
                                        DHSUP,
                                        LAMEBT,
                                        TIMEBT,
                                    LAMFBS,
                                        TIMFBS,
                                        LAMFAS,
                                        TIMFAS,
                                        LAMEAO,
                                        TIMEAO
                                   From CAST_WO_TAPMASS@RED_CHS
                                order by dhcal;
    Cursor WO_POTMASS Is Select CODPOT,
                                        MPESON,
                                        Upper (CODLOG) CODLOG,
                                        DHCAL
                                   From CAST_WO_POTMASS@RED_CHS
                                order by dhcal;
    Cursor WO_POTMASS_SINGLE (LOGLADLE Varchar2, TAPDATE Date)
                                 Is Select CODPOT,
                                           MPESON,
                                           Upper (CODLOG) CODLOG,
                                           DHCAL
                                    From CAST_WO_POTMASS@RED_CHS
                                   Where DHCAL    = TAPDATE
                                     And LOGLADLE = Upper (CODLOG);
    Copy_Failed  Boolean;
    ERROR_GENERATED Exception;
Begin
    /* primeiro loop para todos os  PPS. */
    For PTD In ALL_WO_SCHEDULE Loop
        Copy_Failed := FALSE;
        If (nvl (last_datpos, sysdate)) = PTD.DATPOS And (nvl
(last_indsdatpos, 5) = PTD.INDSDATPOS) Then
            do_copy := FALSE;
        Else
            do_copy := TRUE;
        End If;
/* copia dos potes para cada PPS  */
        If do_copy Then
        For REC In SINGLE_WO_SCHEDULE (PTD.DHSCH, PTD.DATPOS,
PTD.INDSDATPOS) Loop
        Begin
Insert into CAST_WO_SCHEDULE (DHSCH,
                              CODPOT,
                              DATPOS,
                              INDSDATPOS,
```

```
                                    METHEO,
                                    AGE,
                                    SCHED)  Values
                                   (REC.DHSCH,
                                    REC.CODPOT,
                                    REC.DATPOS,
                                    REC.INDSDATPOS,
                                    REC.METHEO,
                                    REC.AGE,
                                    REC.SCHED);
          Exception
              When DUP_VAL_ON_INDEX Then
                  Update CAST_WO_SCHEDULE
                     Set DATPOS       = REC.DATPOS,
                         INDSDATPOS   = REC.INDSDATPOS,
                         METHEO       = REC.METHEO,
                         AGE          = REC.AGE,
                         SCHED        = REC.SCHED
                      Where DHSCH   = REC.DHSCH
                        And CODPOT = REC.CODPOT;
              When NO_DATA_FOUND Then
                  Null;
              When OTHERS Then
                  err_no := ke02_dbms_pipe.fe02_send_message
('KF21_JOBS','C',SQLERRM,'PF01_COPY_REDUCTION - 05','1',10);
Copy_Failed := TRUE;
Raise ERROR_GENERATED;
          End;
          End Loop;
          /* analises dos potes */
          For REC In SINGLE_WO_PURITY (PTD.DHSCH) Loop
          Begin
              Insert Into CAST_WO_PURITY (DHSCH,
                                          CODPOT,
                                          CODDET,
                                          VALDET) Values
                        (REC.DHSCH,
                         REC.CODPOT,
                         REC.CODDET,
                         REC.VALDET);
         Exception
              When DUP_VAL_ON_INDEX Then
                  Update CAST_WO_PURITY
                     Set VALDET   = REC.VALDET
                      Where DHSCH    = REC.DHSCH
                        And CODPOT   = REC.CODPOT
                        And CODDET   = REC.CODDET;
              When NO_DATA_FOUND Then
                  Null;
              When OTHERS Then
```

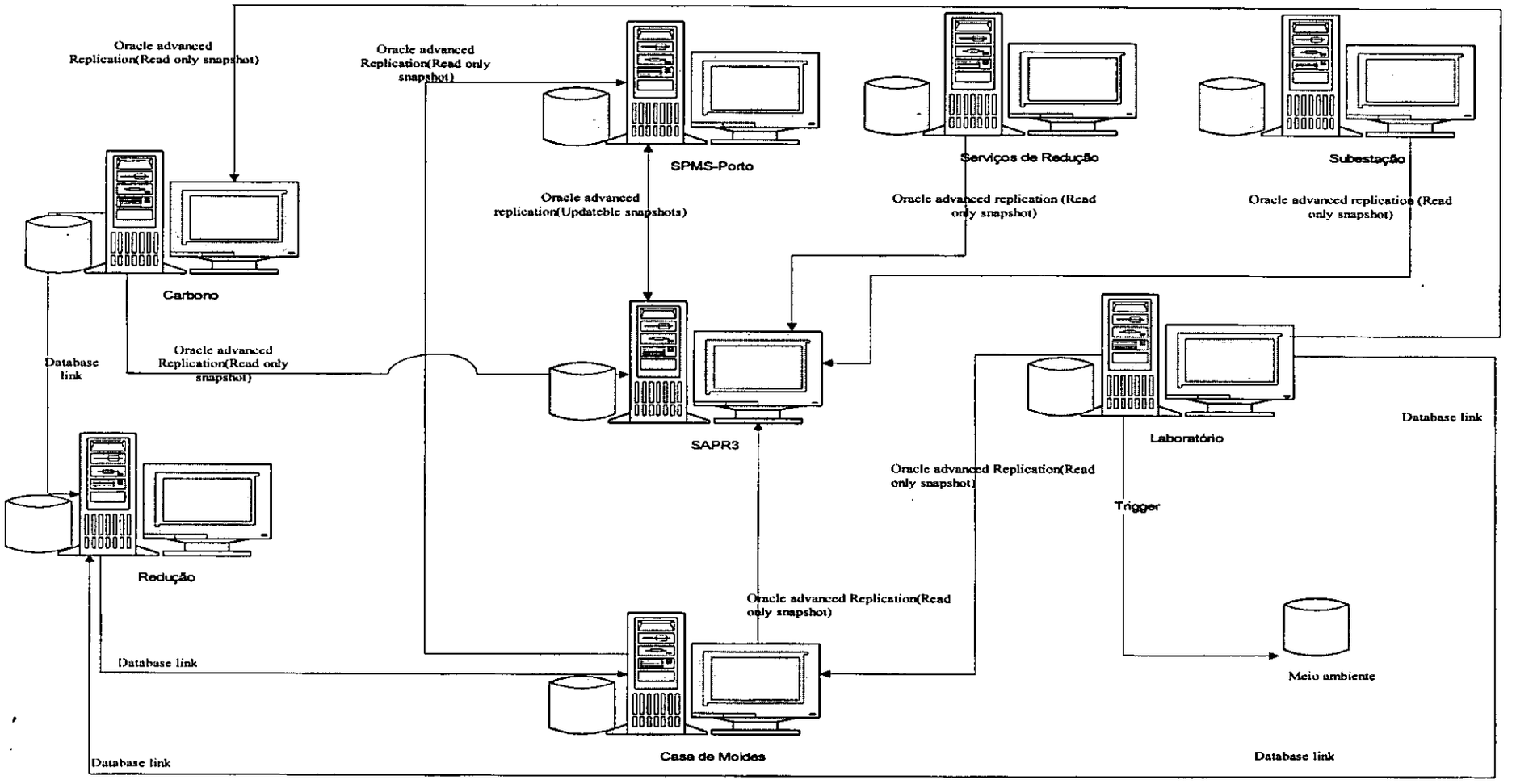
```
                err_no := ke02_dbms_pipe.fe02_send_message
('KF21_JOBS','C',SQLERRM, 'PF01_COPY_REDUCTION - 05','1', 10);
Copy_Failed := TRUE;
                Raise ERROR_GENERATED;
        End;
        End Loop;
        End If;

        If Not Copy_Failed Then
            Begin
                Delete From CAST_WO_PURITY@RED_CHS   where DHSCH
= PTD.DHSCH;
          Delete  From  CAST_WO_SCHEDULE@RED_CHS  where  DHSCH  =
PTD.DHSCH;
            Exception
                When OTHERS Then
                    err_no := ke02_dbms_pipe.fe02_send_message
('KF21_JOBS','C',SQLERRM,'PF01_COPY_REDUCTION - 01', '1',10);
err_no := ke02_dbms_pipe.fe02_send_message ('KF21_JOBS','C',
'Failed to delete records from ' ||
'CAST_WO_PURITY@ and CAST_WO_SCHEDULE@', 'PF01_COPY_REDUCTION -
01', '1',10);
Raise ERROR_GENERATED;
        End;
          Commit;
        Else
Rollback;
End If;
last_datpos      := PTD.DATPOS;
        last_indsdatpos  := PTD.INDSDATPOS;
End Loop;
Copy_Failed := FALSE;
For REC In WO_TAPMASS Loop
    Begin
        Insert Into CAST_WO_TAPMASS (DHCAL,
                                     CODLOG,
                                     DATPOS,
                                     INDSDATPOS,
                                     CODPHY,
                                     DHSUP,
                                     LAMEBT,
                                     TIMEBT,
                                     LAMFBS,
                                     TIMFBS,
                                     LAMFAS,
                                     TIMFAS,
                                     LAMEAO,
                                     TIMEAO)  Values
                            (REC.DHCAL,
                             REC.CODLOG,
```

```
                            REC.DATPOS,
                            REC.INDSDATPOS,
                            REC.CODPHY,
                            REC.DHSUP,
                            REC.LAMEBT,
                            REC.TIMEBT,
                            REC.LAMFBS,
                            REC.TIMFBS,
                            REC.LAMFAS,
                            REC.TIMFAS,
                            REC.LAMEAO,
                            REC.TIMEAO);
Exception
        When DUP_VAL_ON_INDEX Then
            Update CAST_WO_TAPMASS
               Set DATPOS      = REC.DATPOS,
                   INDSDATPOS = REC.INDSDATPOS,
                   CODPHY      = REC.CODPHY,
                   DHSUP       = REC.DHSUP,
                   LAMEBT      = REC.LAMEBT,
                   TIMEBT      = REC.TIMEBT,
                   LAMFBS      = REC.LAMFBS,
                   TIMFBS      = REC.TIMFBS,
                   LAMFAS      = REC.LAMFAS,
                   TIMFAS      = REC.TIMFAS,
                   LAMEAO      = REC.LAMEAO,
                   TIMEAO      = REC.TIMEAO
                Where DHCAL    = REC.DHCAL
                  And CODLOG   = REC.CODLOG;
When NO_DATA_FOUND Then
            Null;
When OTHERS Then
            err_no := ke02_dbms_pipe.fe02_send_message
('KF21_JOBS','C', SQLERRM, 'PF01_COPY_REDUCTION - 07','1', 10);
Copy_Failed := TRUE;
            Raise ERROR_GENERATED;
End;
commit;
End Loop;
For REC In WO_POTMASS Loop
    Begin
        Insert Into CAST_WO_POTMASS (CODPOT,
                                     MPESON,
                                     CODLOG,
                                     DHCAL) Values
                    (REC.CODPOT,
                     REC.MPESON,
                     REC.CODLOG,
                     REC.DHCAL);
Exception
```

```
        When DUP_VAL_ON_INDEX Then
            Update CAST_WO_POTMASS
               Set MPESON  = REC.MPESON
             Where CODPOT  = REC.CODPOT
               And CODLOG  = REC.CODLOG
               And DHCAL   = REC.DHCAL;
When NO_DATA_FOUND Then
            Null;
When OTHERS Then
            err_no := ke02_dbms_pipe.fe02_send_message
('KF21_JOBS','C', REC.CODPOT || ' ' ||REC.CODLOG || ' : ' ||
SQLERRM, 'PF01_COPY_REDUCTION - 06', '1',10);
End;
    End Loop;
If Not Copy_Failed Then
    Begin
        Delete From CAST_WO_POTMASS@RED_CHS;
        Delete From CAST_WO_TAPMASS@RED_CHS;
Exception
When OTHERS Then
            err_no := ke02_dbms_pipe.fe02_send_message
('KF21_JOBS','C', SQLERRM, 'PF01_COPY_REDUCTION - 02', '1', 10);
err_no := ke02_dbms_pipe.fe02_send_message ('KF21_JOBS','C',
'Failed to delete records from ' || 'CAST_WO_POTMASS@ and
CAST_WO_TAPMASS@', 'PF01_COPY_REDUCTION - 02', '1', 10);
Raise ERROR_GENERATED;
    End;
COMMIT;
Else
     /* faz o rollback se nao tiver inserido tudo, nao deve
apagar */
        ROLLBACK;
End if;
Exception
    When NO_DATA_FOUND Then
err_no              :=              ke02_dbms_pipe.fe02_send_message
('KF21_JOBS','C',SQLERRM,
'PF01_COPY_REDUCTION - 03','1',10);
    When DUP_VAL_ON_INDEX Then
 err_no             :=              ke02_dbms_pipe.fe02_send_message
('KF21_JOBS','C',SQLERRM,
'PF01_COPY_REDUCTION - 04','1',10);
End PF01_COPY_REDUCTION;
END KF21_JOBS;
```



ANEXO E: Replicacao configurada na empresa Mozal



ANEXO F: REPLICACAO PROPOSTA PARA EMPRESA MOZAL

Anexo G

O ficheiro tnsnames.ora

```
lab.world =
  (DESCRIPTION =
    (ADDRESS_LIST =
        (ADDRESS =
          (COMMUNITY = tcp.world)
          (PROTOCOL = TCP)
          (Host = 172.16.22.5)
          (Port = 1521)
        )
        (ADDRESS =
          (COMMUNITY = tcp.world)
          (PROTOCOL = TCP)
          (Host = 172.16.22.5)
          (Port = 1526)
        )
    )
    (CONNECT_DATA = (SID = lab)
    )
  )
env.world =
  (DESCRIPTION =
    (ADDRESS_LIST =
        (ADDRESS =
          (COMMUNITY = tcp.world)
          (PROTOCOL = TCP)
          (Host = 172.16.22.5)
          (Port = 1521)
        )
        (ADDRESS =
          (COMMUNITY = tcp.world)
          (PROTOCOL = TCP)
          (Host = 172.16.22.5)
          (Port = 1526)
        )
    )
    (CONNECT_DATA = (SID = lab)
    )
  )
POW.world =
  (DESCRIPTION =
    (ADDRESS_LIST =
        (ADDRESS =
          (COMMUNITY = tcp.world)
          (PROTOCOL = TCP)
          (Host = 172.16.22.4)
```

```
            (Port = 1521)
          )
          (ADDRESS =
            (COMMUNITY = tcp.world)
            (PROTOCOL = TCP)
            (Host = 172.16.22.4)
            (Port = 1526)
          )
      )
      (CONNECT_DATA = (SID = pow)
      )
  )
crb.world =
  (DESCRIPTION =
    (ADDRESS_LIST =
          (ADDRESS =
            (COMMUNITY = tcp.world)
            (PROTOCOL = TCP)
            (Host = 172.16.22.2)
            (Port = 1521)
          )
          (ADDRESS =
            (COMMUNITY = tcp.world)
            (PROTOCOL = TCP)
            (Host = 172.16.22.2)
            (Port = 1526)
          )
      )
      (CONNECT_DATA = (SID = crb)
      )
  )
```

Anexo H:
A ferramnta do Oracle Replication Manager