FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Segurança da Informação

VICTOR DE MORAES CARVALHO

UMA ANÁLISE DE MECANISMOS DE SEGURANÇA, DISPONIBILIDADE E DESEMPENHO EM BANCO DE DADOS ORACLE

Americana, SP

2016

FACULDADE DE TECNOLOGIA DE AMERICANA

VICTOR DE MORAES CARVALHO

UMA ANÁLISE DE MECANISMOS DE SEGURANÇA, DISPONIBILIDADE E DESEMPENHO EM BANCO DE DADOS ORACLE

Trabalho de Conclusão de Curso submetido à FATEC – Faculdade de Tecnologia de Americana como parte dos requisitos necessários para a obtenção do Grau de Tecnólogo em Segurança da Informação. Sob a orientação do Professor Ms. José Mario Frasson Scafi.

Americana/SP

2016

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS

Dados Internacionais de Catalogação-na-fonte

C329u CARVALHO, Victor de Moraes

Uma analíse de mecanismos de segurança, disponibilidade e desempenho em banco de dados Oracle / Victor de Moraes Carvalho. – Americana: 2016.

58f.

Monografia (Curso de Tecnologia em Segurança da Informação). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.

Orientador: Prof. Ms. José Mario Frasson Scafi

1. Oracle - banco de dados I. SCAFI, José Mario Frasson II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU:681.3.07

Victor de Moraes Carvalho

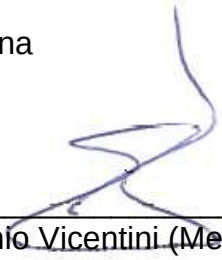
Título: UMA ANALISE DE MECANISMOS DE SEGURANÇA, DISPONIBILIDADE E DESEMPENHO EM BANCO DE DADOS ORACLE

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Segurança da Informação pelo CEETEPS/Faculdade de Tecnologia – FATEC/Americana.
Área de concentração: Segurança da Informação

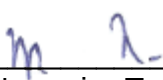
Americana, 09 de dezembro de 2016.

Banca Examinadora:

_____
José Mario Frasson Scafi (Presidente)
Mestre
Fatec Americana



_____
Eduardo Antonio Vicentini (Membro)
Mestre
Fatec Americana



_____
Rodrigo Nogueira Tofani (Membro)
Especialista
Fatec Americana

Dedico este trabalho a minha família
que sempre esteve ao meu lado durante todo o
desenvolvimento do TCC, também dedico a
todos que acreditaram em mim, e
especialmente ao meu orientador que teve toda
a paciência e me direcionou para o caminho
certo. Obrigado a todos!

AGRADECIMENTOS

Agradeço a todos os que me ajudaram na elaboração deste trabalho: Meus professores, amigos pessoais e de trabalho e meu irmão, e também a minha mãe e namorada que colaboraram de forma especial na elaboração do trabalho.

“A nossa maior fraqueza está em desistir. O caminho mais certo para vencer é tentar mais uma vez”.

Thomas Edison

RESUMO

O objetivo deste trabalho é mostrar os recursos fornecidos pela Oracle para manter um banco de dados disponível e seguro. Podem ser atribuídos aos princípios básicos para garantir a segurança da informação no que diz respeito à confidencialidade, integridade e disponibilidade. Esses recursos podem ser divididos por confidencialidade: Oracle *datavault*, *profiles* e *roles* (impede que o dado seja acessado por quem não tem direitos, tornando-os assim confiáveis); por Integridade: *database firewall* (impossibilita que o dado seja modificado ou que haja a tentativa de modificação); e por último, disponibilidade: Oracle RAC, *dataguard* e RMAN (mantém o dado disponível o tempo todo). Cada recurso, trata de uma parte especifica em relação à segurança da informação, e ao longo do trabalho será explicado como funciona cada um e a importância de utilizá-los para proteger os dados de uma empresa.

Palavras chaves: Oracle; banco de dados; segurança da informação.

ABSTRACT

The purpose of this paper is to show the features provided by Oracle to keep a database available and secure. They can be attributed to basic principles to ensure information security with respect to confidentiality, integrity and availability. These resources can be divided by confidentiality: Oracle datavault, profiles and roles (prevent the data from being accessed by those who do not have rights, thus making them reliable); By Integrity: database firewall (it prevents data from being modified or attempted modification); and finally, availability: Oracle RAC, dataguard and RMAN (keep data available). Each resource is a specific part of information security and throughout the work will be explained how each works and a use value for the data of a company.

Keywords: Oracle; database; security information.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

1. INTRODUÇÃO

Este trabalho apresenta alguns recursos de segurança que a Oracle disponibiliza para a proteção de seu serviço gerenciador de banco de dados (SGBD), nem todos seus recursos servem somente para os produtos de sua fabricação, mas também são capazes de serem utilizados com outros serviços gerenciadores de banco de dados (SGBD). Entretanto a maioria dos recursos aqui mencionados funciona somente com o Oracle.

O conteúdo aqui exposto tem como proposito explicar o que é, e para que serve determinado produto, será mostrado qual sua utilização, como ele pode servir para aumentar a segurança de um banco de dados. O trabalho possui bastante teoria para o entendimento de cada tecnologia, como ela funciona, porque é necessário saber o que se quer proteger e o mais importante que é como proteger, qual ferramenta vai suprir as necessidades de uma determinada empresa ou cliente.

Um grande estímulo para a criação deste trabalho de segurança em banco de dados Oracle foi o CID (Confidencialidade, Integridade e Disponibilidade). As informações são o que a empresa possui de mais importante, então manter estas informações confiáveis, integras e disponíveis é essencial. Cada uma das tecnologias abordadas durante o trabalho tem um importante papel nestes itens.

Antes de começar a falar sobre o serviço gerenciador de banco de dados (SGBD) Oracle, no capitulo 2 é abordado o conceito de banco de dados relacional, que forma uma base para entender melhor a estrutura do Oracle. Cada subcapítulo do segundo capítulo demonstra e explica em mais detalhes o uso e o conceito básico de cada recurso, capítulo 2 o que são *profiles* e *roles*, como podem ser utilizadas para proteger um banco de dados, assim como no capítulo 6 sobre *dataguard* e 7 Oracle RAC, ferramentas para alta disponibilidade, sendo que o RAC ainda pode ser utilizado para ganho de desempenho.

O capítulo 8 possui um exemplo prático de Oracle RAC, mostrando seu recurso de redundância a falhas e balanceamento de cargas. Para finalizar o trabalho, o capítulo 4 apresenta as considerações finais, sobre todo o conteúdo aqui apresentado.

2. ORACLE BANCO DE DADOS RELACIONAL

Toda organização possui informações que devem ser armazenadas e gerenciadas de forma que atenda às suas necessidades. Por exemplo, a área de recursos humanos de uma empresa deve coletar e manter registros de todos seus empregados. Essas informações devem estar disponíveis àqueles que precisam, neste caso todos os funcionários que atuam diretamente com a área de recursos humanos.

De acordo com Ashdown e Kyte (2013) um sistema de informação é um sistema formal para armazenar e processar informações. Então um sistema de informação poderia ser um conjunto de caixas de papelão contendo pastas com seus conteúdos, assim como regras de como armazenar e recuperar estas pastas. No entanto, a maioria das empresas hoje em dia usa um banco de dados para automatizar seus sistemas de informação. Segundo Ashdown e Kyte (2013) um banco de dados é uma coleção organizada de informações tratadas como uma unidade. O propósito de um banco de dados é coletar, armazenar e recuperar informações que são geradas pelos sistemas de informação, seja ele um ERP (*Enterprise Resource Planning*) de grande porte ou um pequeno sistema da empresa para armazenar horário de entrada e saída de seu funcionário quando ele utiliza seu crachá para passar na catraca da portaria.

Codd (1970) definiu o modelo relacional da seguinte forma o modelo relacional é baseado na teoria matemática de conjuntos. Hoje, o modelo de banco de dados mais aceito e utilizado é o modelo relacional.

Um banco de dados relacional é um banco de dados que está em conformidade com o modelo relacional. Que segundo a Ashdown e Kyte (2013) este modelo possui as seguintes características: estruturas: objetos bem definidos que armazenam ou acessam os dados de um banco de dados. Operações: ações claramente definidas que permitem as aplicações manipularem os dados e estruturas de um banco de dados. Regras de integridade: regras de integridade dominam operações sobre os dados e estruturas de um banco de dados.

A forma que um banco de dados relacional armazena seus dados é em um conjunto de relações simples. Onde está relação é um conjunto de tuplas, e uma tupla é um conjunto desordenado de valores de atributos.

A definição utilizada por Ashdown e Kyte (2013) para definir uma tabela é a seguinte uma tabela é uma representação bidimensional de uma relação na forma de linhas (tuplas) e colunas (atributos). Onde as linhas de uma tabela têm os mesmos conjuntos de colunas. Um banco de dados relacional é um banco de dados que armazena dados em relações (tabelas). Por exemplo, um banco de dados relacional poderia armazenar informações sobre os funcionários da empresa em uma tabela de funcionários, uma tabela de setor e uma tabela de contracheque.

O modelo relacional é a base para um sistema de gerenciamento de banco de dados relacional (SGBD). Um SGBD move os dados para um banco de dados, armazena e recupera os dados para que as aplicações possam manipulá-los.

Conforme especificado por Ashdown e Kyte (2013) um SGBD distingue entre os seguintes tipos de operações: operações lógicas – neste caso, a aplicação especifica o conteúdo necessário. Por exemplo, uma aplicação solicita um nome de funcionário ou adiciona um registro de funcionário a uma tabela. Operações físicas – o SGBD determina como as coisas devem ser feitas e executa a operação. Por exemplo, depois que uma aplicação consulta uma tabela, o banco de dados pode usar um índice para localizar as linhas solicitadas, ler os dados na memória e executar muitas outras etapas antes de retornar um resultado para o usuário. O SGBD armazena e recupera dados para que as operações físicas sejam transparentes para as aplicações de banco de dados.

O banco de dados Oracle é um SGBD que implementa recursos orientados a objeto, como tipos definidos pelo usuário, herança e polimorfismo, ele é chamado de sistema de gerenciamento de banco de dados relacionado a objetos (SGBDRO). O Oracle *Database* estendeu o modelo relacional a um modelo relacionado a objetos, permitindo armazenar modelos de negócios complexos em um banco de dados relacional.

3. ESTRUTURA DO ORACLE

Para compreender um pouco melhor de como e para que os recursos de segurança do Oracle podem ser de primordial importância ao negócio do cliente, é necessário entender o básico da estrutura do Oracle, este capítulo busca sanar estas.

Um servidor de banco de dados Oracle consiste em um banco de dados e pelo menos uma instância de banco de dados, comumente referida como simplesmente uma instância. Como uma instância e um banco de dados estão tão intimamente conectados, o termo banco de dados Oracle é usado às vezes para se referir tanto à instância quanto ao banco de dados.

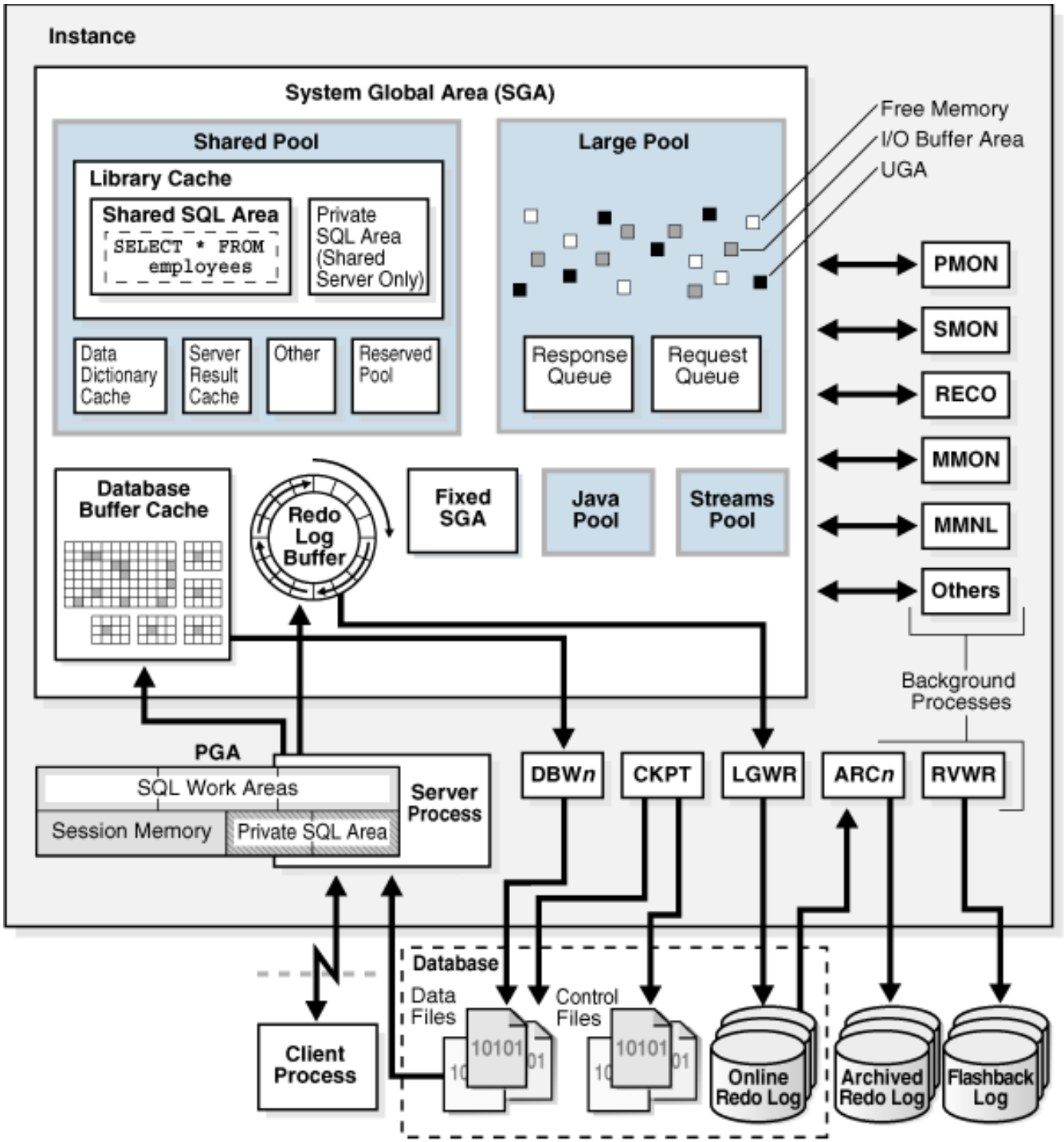
Neste sentido mais estrito, os termos têm os seguintes significados de acordo com Ashdown e Kyte (2013) um banco de dados é um conjunto de arquivos, localizados no disco, que armazenam dados. Esses arquivos podem existir independentemente de uma instância de banco de dados. Enquanto Instância de banco de dados é um conjunto de estruturas de memória que gerenciam arquivos de banco de dados. A instância consiste em uma área de memória compartilhada, chamada de área global do sistema (SGA – *System Global Area*) e um conjunto de processos em segundo plano. Uma instância pode existir independentemente dos arquivos de banco de dados.

A figura 1 ilustra um banco de dados e sua instância. Para cada conexão de usuário com a instância, um processo cliente (*client process*) executa a aplicação. Cada processo cliente (*client process*) está associado ao seu próprio processo de servidor (*server process*). O processo do servidor (*server process*) por sua vez tem sua própria sessão de memória particular (*private memory session*), conhecida como a área global do programa (*Program Global Area - PGA*).

Um banco de dados pode ser considerado tanto a partir de uma perspectiva física quanto lógica. Os dados físicos podem se dizer que são "visíveis" a nível de sistema operacional. Como por exemplo, os utilitários do sistema operacional, neste caso o comando *ls* e *ps* do *Linux*, que podem listar arquivos e processos de banco de dados. Já os dados lógicos que pode ser exemplificada como tabelas, elas são significativas somente para o banco de dados. Uma instrução SQL (*Structured Query*

Language) pode listar as tabelas em um banco de dados Oracle, mas um comando do sistema operacional não.

Figura 1 – Estrutura do Oracle



Fonte: Ashdown e Kyte (2013)

De acordo com Ashdown e Kyte (2013) o banco de dados tem estruturas físicas e estruturas lógicas. Como estas estruturas são separadas, é possível gerenciar o armazenamento físico de dados sem interferir com o acesso as estruturas de armazenamento lógico. Por exemplo, renomear um arquivo de banco de dados físico não renomeia as tabelas nas quais os dados estão armazenados.

É sempre difícil falar de termos básicos sobre a estrutura do Oracle porque dificilmente se fala de um termo sem acabar citando outro. Será importante conhecer estes termos porque serão utilizados ao longo deste trabalho, alguns traduzidos e outros sem traduzir porque a tradução literal é inviável.

Agora segue uma breve descrição dos termos do Oracle:

- *Log* de Alerta (*Alert log*): Um arquivo de texto que o banco de dados gerencia mensagens de erros e estado. O *log* de alerta pode ser muito útil quando é necessário determinar a natureza de uma falha.

- *Redo Logs* Arquivados (*archived redo logs*): Quando um banco de dados está em modo *ARCHIVELOG*, então são gerados *archived redo logs* cada vez que o banco de dados troca o *online redo log* pelo processo LGWR. *Archived redo logs* são utilizados durante uma recuperação do banco de dados. Copias dos *archived redo logs* podem ser criadas em mais de dez diretórios diferentes, definido pelo parâmetro do *Oracle LOG_ARCHIVE_DEST_n* no arquivo de parâmetro do banco de dados.

- Arquivo de controle de *backup* (*backup control file*): é o *backup* do arquivo de controle que é gerado através do comando: *alter database backup controlfile to 'file_name'* ou então *alter database backup controlfile to trace.*

- Ponto de checagem (*checkpoint*): É um evento do banco de dados que limpa os blocos (usados) sujos da memória e escrevem eles para o disco.

- Bloco (*block*): A unidade mais atômica de armazenamento no Oracle. O tamanho padrão do bloco é determinado pelo parâmetro *DB_BLOCK_SIZE* no arquivo de parâmetros do banco de dado, e são permanentemente estabelecidos quando um banco de dados é criado.

- Consistência do banco de dados (*database consistency*): Implica que cada objeto no banco de dados está consistente com o mesmo ponto no tempo.

Significa que os dados nos *datafiles* do banco de dados estão consistentes no mesmo ponto no tempo. Isto também significa que os arquivos de controle do banco de dados estão sincronizados com os cabeçalhos do *datafile* do banco de dados.

- *Datafile* do banco de dados (*Database Datafile*): Uma entidade física que está relacionada a uma *tablespace*. Um banco de dados consiste de pelo menos um *datafile* (que poderia ser atribuída a *tablespace SYSTEM*), e a maioria dos bancos de dados consiste de vários *datafiles* diferentes. Onde uma *tablespace* pode conter vários *database datafile* associados, e um *database datafile* só pode conter uma *tablespace* associada a ele.

- Arquivo de parâmetros do banco de dados (*Database Parameter File*): Contém a informação de configuração das instâncias e do banco de dados e são dois principais arquivos: o *init.ora* que é um arquivo de texto e o *spfile.ora*, que permite definições permanentes dos parâmetros do banco de dados através do comando *alter system*.

- *Flash Recovery Area* (*FRA*): Uma área opcional configurada no disco usada para armazenar vários arquivos relacionados a recuperação. Arquivos de *backup* do RMAN, *archived redo logs*, *online redo logs*, e *control files* podem ser armazenadas nesta área.

- *Granule*: É uma unidade contínua da memória do Oracle. Todas as alocações de memória da área global do sistema (SGA – *System Global Area*) são arredondadas para perto das unidades *granule*. O tamanho de um *granule* depende completamente do tamanho esperado da SGA, e isto pode ser 4MB ou 16MB. Uma SGA é maior que 128MB e costumam ultrapassar quando o Oracle usa *granule* de tamanhos maiores. O número de *granule* alocado ao banco de dados é determinado quando o banco é inicializado.

- Instância (*Instance*): A coleção de processos e memória do Oracle. Quando a SGA (memória) é alocada e cada um dos processos dele estão inicializados e executando com sucesso, então a instância é considerada como inicializada. Perceba que apenas por uma instância estar executando não significa que o banco de dados esteja aberto (*open*). Uma instância está associada a somente um banco de dados.

- *Online Redo Logs*: Quando um *redo* é gerado, ele é fisicamente armazenado no *online redo log* do banco de dados. O Oracle necessita que pelo menos dois *redo logs* sejam criados para que o banco de dados opere. Estes *online redo logs* podem ter várias cópias espelhadas para a proteção do *redo*. Isto é conhecido como multiplexação do *redo log*. Quando um *online redo log* é preenchido por *redo*, o Oracle substitui pelo próximo *online redo log*, esta operação é conhecida como *log switch*.

Cada *online redo log file* contém um *log* de sequência numérica associada que unicamente identifica ele e, se ele é arquivado, ele é associado ao *archive redo log file*. Pode-se encontrar o *log* de sequência numérica do *online redo log* dando um *select* na tabela *V$LOG*. O número de sequência de um *archive redo log* pode ser encontrado na tabela *V$ARCHIVED_LOG* ou na tabela *V$LOG_HISTORY*.

Adicionalmente, um *online redo log* (e um *archived redo log*) contém um intervalo do sistema de mudanças numéricas (*SCNs – System Change Numbers*) que são únicas para o *redo log*. Durante uma recuperação, o Oracle aplica o desfazer (*undo*) nos *archived/online redo log* para sequenciar o número do *log*.

- Processos (*Proccesses*): Os programas que fazem o atual trabalho do banco de dados *Oracle*. Na versão 11G têm cinco processos necessários entre outros.

- *Redo*: É um registro de todas as mudanças feitas a um banco de dados. Para quase qualquer mudança no banco de dados, um registro *redo* associado é gerado.

- *Schema*: O dono de vários objetos lógicos no Oracle, como tabelas e índices, e é sinônimo a um usuário.

- SGA (*System Global Area*): É uma área de memória compartilhada que é alocada pelo Oracle quando ele inicializa. A memória no SGA pode ser compartilhada por todos os processos do Oracle.

- *System Change Number* (SCN): Um contador que representa o atual estado do banco de dados a um determinado momento. Assim como o contador na VCR, como tempo de processos, o SCN aumenta. Cada SCN automaticamente representa um ponto na existência do banco de dados. Então, às 11 A.M., o

database SCN pode ser 10ffx0 (4351 decimal), e às 12 P.M., ele pode ser 11f0x0 (4592 decimal).

- *Tablespace*: Uma entidade física-lógica. Ela é uma entidade lógica porque é alocada onde os objetos lógicos do Oracle (assim como tabelas e índices) são armazenados. Ela é uma entidade física porque é criada em cima de um ou mais *datafiles*. Um banco de dados deve obter pelo menos uma *tablespace*, a *tablespace SYSTEM*, entretanto a maioria dos bancos de dados possuem várias *tablespaces* diferentes.

- *Trace files*: Gerados pelo banco de dados em um número diferente de situações, incluindo erro de processos. Cada processo do banco de dados também gera seu próprio *trace file*. *Trace files* podem ser importantes quando é necessário resolver uma falha essencial do banco de dados.

4. O QUE SÃO PROFILES E ROLES

De acordo com a Jeloka (2012) *Profile* é um conjunto de limites que contém a quantidade de recursos do banco de dados que um usuário pode utilizar. Ou seja, um usuário pode ser ligado a um *profile* (em português significa ‘perfil’) e este *profile* já estar devidamente configurado para as funções que este usuário deverá desempenhar dentro da empresa, por exemplo: o DBA (*database administrator*) da empresa cria um *profile* chamado financeiro e adiciona todos os acessos que o usuário deverá ter para desempenhar corretamente seu cargo, então para cada pessoa contratada para a área financeira basta ligar o usuário ao *profile* financeiro e ele já estará com todos os acessos devidamente configurados.

Segundo a Jeloka (2012) *roles* são os objetos criados no banco de dados e que pode ser associado a um determinado *schema*. Uma *role* é um agrupamento de privilégios ou de outras *roles* e é utilizado para facilitar a administração de privilégios do banco de dados. Assim sendo, uma *role* (em português significa ‘função’) é criada para restringir o acesso dos usuários no banco de dados, e uma *role* pode ser ligada a outras *roles* e cada *role* deve ter um nome independente não podendo haver outra *role* com o mesmo nome. O interessante da *role* é que ela tem senha, reforçando que o usuário não conseguirá modificar sua *role* para obter mais acesso.

Juntando estes dois recursos do Oracle e montando uma boa estrutura departamental dentro da empresa é possível aumentar muito a segurança dentro do seu banco de dados permitindo assim com que cada área faça seu serviço sem interferir na outra.

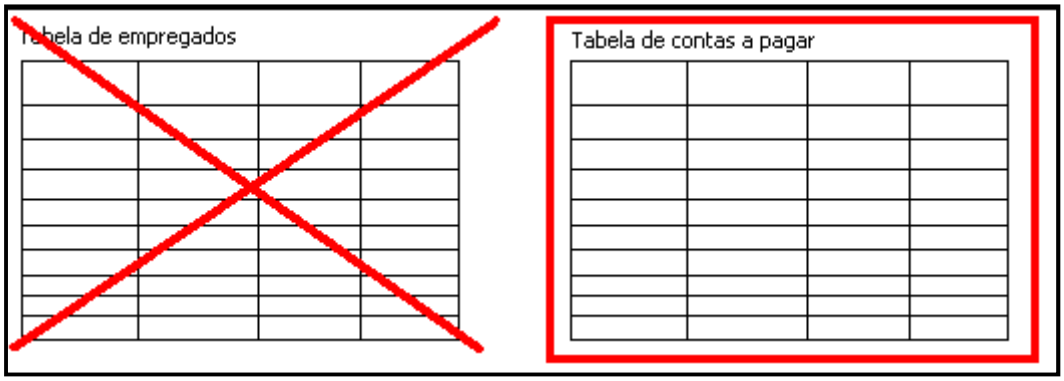
Quando uma *profile* é criada no Oracle seria como se estivesse criando um grupo no *Linux*, e para esse *profile* é possível ligar as suas *roles* que seriam equivalentes as ACLs (*Access Control List*) ligadas ao grupo no *Linux*, ou seja, criando um *profile* financeiro nele pode-se ligar várias *roles* limitando o acesso de todos os usuários que fizer parte do *profile* financeiro.

Por exemplo, é possível ligar ao *profile* do financeiro acesso a tabela contas a pagar e retirar o acesso a tabela de empregados, os empregados do financeiro não precisam saber quais são os usuários que a empresa possuí e nem o salário deles.

Este é só um exemplo básico, porque há muito mais que é preciso proteger de usuários mal-intencionados. Porém isto será abordado mais adiante nos próximos capítulos. A figura 2 ilustra o exemplo acima mencionado.

Também é importante ressaltar que as permissões negativas são mais efetivas que as permissivas, assim sendo, se um usuário possui tanto acesso como restrição a uma tabela, no final não será possível que ele acesse a tabela.

Figura 2 – Exemplo de role e profile



Fonte: Próprio autor.

5. MOTIVAÇÃO PARA A CRIAÇÃO DO ORACLE DATABASE FIREWALL

Enquanto houver casos publicados de perdas ou roubos de *laptops* com informações pessoais identificáveis (*Personally Identifiable Information – PII*), então as tentativas de roubar grandes quantidades de informação através de ataques a servidores se tornarão cada vez mais comum.

Conforme a Verizon (2010, apud BEDNAR, 2012) os relatórios de investigação de violação a dados realizado em 2010 revelou que 98% dos dados violados vêm dos servidores. Lançando ataques com sucesso em grandes repositórios pode resultar em um dia muito lucrativo para o "predador" e isso sem dizer que ambientes de aplicação, *data warehouses* e bancos de dados em geral estão se tornando maiores e mais críticos para as operações dos negócios e, portanto, um alvo tentador.

É fato que hoje em dia o crime organizado está se tornando um grande problema, ainda mais em ataques cujo foco é a violação de dados, onde índices mostram que ataques internos ainda representam uma alta taxa.

Segundo a Verizon (2010, apud BEDNAR, 2012) os relatórios de investigações de violação a dados também notaram que abuso de privilegio e *hacking* eram os mais comuns meios que violações ocorriam e frequentemente alavanca o número de credenciais perdidas e vulnerabilidades de injeções de SQL para ganhar acesso não autorizado.

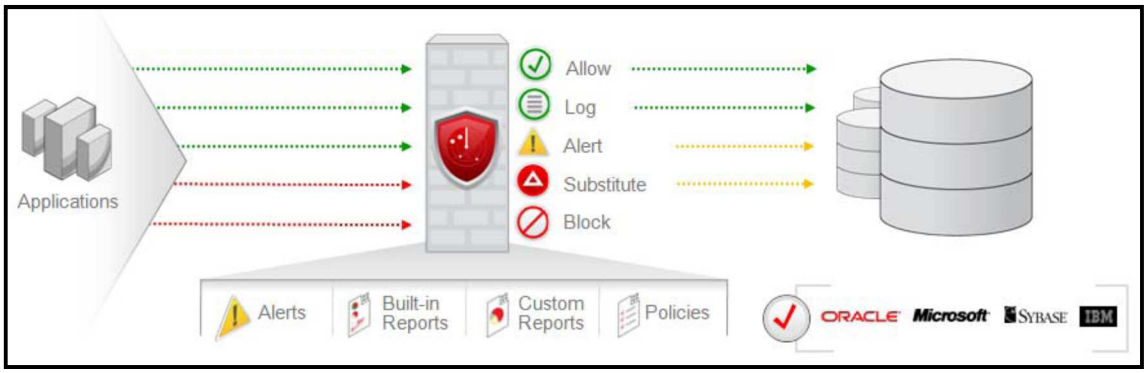
Para manter os dados seguros dentro de um servidor são necessárias várias camadas de proteção onde ambas as funções são o foco, técnicas e administrativas. Sem mencionar medidas de precaução simples como: desabilitar contas que não são utilizadas e proibir que ocorra compartilhamento de contas administrativas para aumentar o nível de segurança (que é muito difícil de controlar). Então soluções como a criptografia de e privilégios de controle de contas dentro do banco de dados acabam se tornando partes muito importantes para a segurança das aplicações e da informação. Mesmo assim acaba não sendo possível de monitorar as requisições SQLs enviadas ao banco de dados, uma vez que este comando vem de uma conexão interna segura. E com o Oracle *Database Firewall* se torna possível um controle de

segurança do perímetro, que é uma primeira camada de proteção sobre os bancos de dados Oracle ou outros SGBDs.

5.1. Conhecendo o Oracle database firewall

Conforme a Bednar (2012) o Oracle *database firewall* é uma solução ativa e de tempo real que fornece políticas de *white list* (lista branca), *black list* (lista negra) e *exeption list* (lista de exceções), monitoração e gerenciamento. Esta barreira de proteção ajuda a reduzir o risco de perda de dados, e aumenta o controle de acesso aos dados, e ajuda a gerenciar mudanças nas regulações, todas estas funções podem ser vistas na figura 3.

Figura 3 – Overview Oracle Database Firewall



Fonte: Bednar (2012)

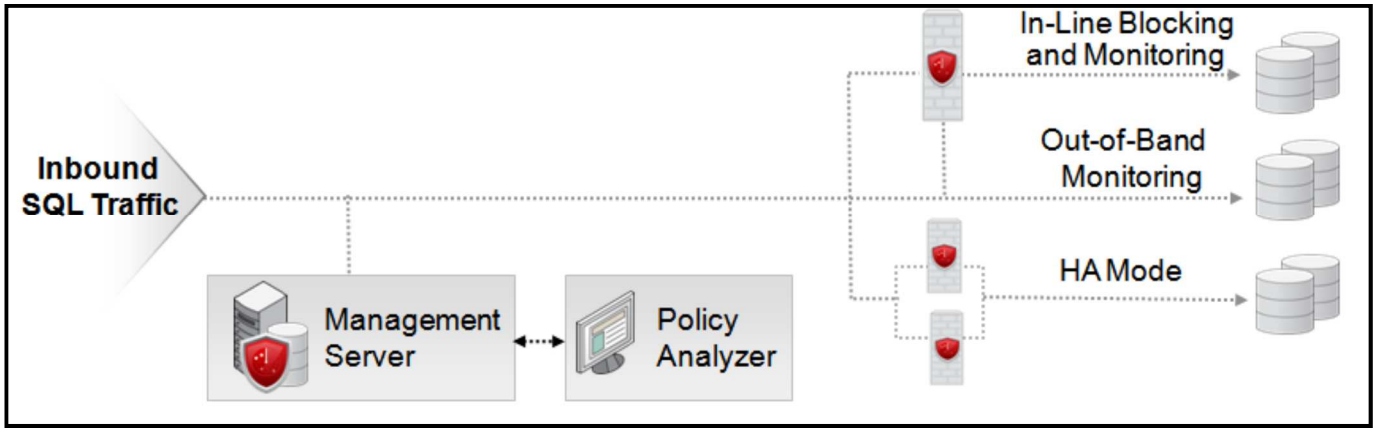
Diferente dos tradicionais *Firewalls* de SQL baseados em identificar SQLs fora da política usando estratégias como expressões regulares, comparação de *string*, e comparação de *schema*, o Oracle *database firewall* entrega uma segurança de *firewall* mais inteligente, permitindo políticas que se adaptam rápida e precisamente. As organizações podem escolher implantar o *Firewall* do Oracle em modo de bloquear como uma política de reforço ao sistema para proteger seus bancos de dados ativos, ou então somente em modo de monitoração das atividades no banco de dados como um adicional para auditoria ou como finalidade de gerência.

5.2. O funcionamento do Oracle database firewall

O Oracle *database firewall* é instalado na rede tanto em uma *bridge* (ponte) ou em uma porta de extensão e monitora toda solicitação de transação SQL. Essas

requisições SQL são processadas por um mecanismo (Grammar-based analysis engine) que decompõe e caracteriza o SQL. Além de puramente olhar a solicitação SQL, as políticas podem avaliar fatores como endereço IP, tempo, e o nome do programa.

Figura 4 – Recursos Oracle database firewall



Fonte: Bednar (2012)

Um único Oracle *database firewall* pode monitorar e proteger vários bancos de dados de uma vez, e ele pode ser implantado em vários cenários conforme ilustrado na figura 4, estes cenários são:

- *In-Line network blocking mode*: *In-line* significa que o trafego SQL passa pelo *firewall* do Oracle e é analisado antes de ser enviado para o banco de dados ou bloqueado.

- Out-of-band passive network monitoring: Out-of-band significa que o tráfego SQL é copiado para o *firewall* do Oracle e ao mesmo tempo é enviado diretamente ao banco de dados geralmente por meio de uma porta de extensão.

- *Heterogeneous*, *multi-database*, *enforcement*: Um dispositivo pode suportar banco de dados de vários fornecedores diferentes simultaneamente, por exemplo: Oracle 8i, Oracle *database* 10g, s*ybase database*, SQL *server*.

- Implementações combinadas: in-line e/ou out-of-band pode ser combinado com um local de servidores para monitorar somente os agentes de conexão local.

Também há a opção de alta disponibilidade (*High Availability* – HA). Onde é recomendado que seja implementado dois Oracle *database firewalls* para que a monitoração de SQL não seja interrompida.

5.3. O que é white list

De acordo com a Bednar (2012) *white list* é um conjunto de instruções SQL aprovadas que podem ser enviadas para o banco de dados. O *firewall* compara o trafego SQL com os itens aprovados na *white list* e baseado no que foi definido na política ele escolhe entre bloquear, substituir ou alertar a instrução SQL.

O *firewall* pode ser configurado para bloquear todos os eventos que estão fora da política, e estas implementações podem ser feitas da seguinte maneira:

- Bloquear a instrução SQL.
- Modificar a requisição usando a substituição de instrução SQL (SQL *statement subistitution*)
- Alertar todas as instruções SQL fora da política e adicionalmente bloquear ou substituir.

Em vários casos, a melhor solução é aplicar a substituição de instrução, porque deixa o *firewall* mais transparentes a ataques de hackers e também fica transparente para a aplicação.

Em termos simples a substituição de instruções é um processo que pega uma instrução que está fora da política e substitui para uma nova instrução que não vai retornar dado algum, conforme a tabela 1 mostra.

Tabela 1 – Database firewall substituição de SQL

Requisição Original (fraudulenta)	Requisição substituida	Resposta do banco de dados (resultado)
Select * from tbl_users;	SELECT * FROM tbl_users WHERE 'a' = 'b';	No records found.
DROP TABLE tbl accounts;	SELECT * FROM aaabbbccc;	Error. Table not known.
UPTADE tbl_accounts SET accounts = '123' WHERE user = 'Fred';	SELECT DUAL SET 'fred';	Error. Incorrect Syntax

Fonte: Bednar (2012)

5.4. O que é black list

Em adicional a *white list* que é um modelo de reforço positivo, o *firewall* também suporta o modelo de *black list* que possibilita especificar qual instrução SQL bloquear e assim como a *white list* também é capaz de permitir uma requisição especifica, baseado pelo endereço IP, hora do dia e também pelo programa.

5.5. O que é Exception list

As políticas de exceção são um complemento para as *white* e *black lists*, permitindo que sejam criadas políticas para atividades em especifico. Por exemplo, a lista de exceção pode ser usada para permitir que um administrador conecte remotamente para diagnosticar um problema de desempenho.

5.6. Para que serve o monitor baseado em host

O Oracle *database firewall* também disponibiliza um monitor baseado em host ou agentes que monitoram os bancos de dados. Este monitor de host envia as informações para o *firewall* com a finalidade de monitoração, controle de *login* e alertas. A tabela 2 mostra as características e operações do monitor de banco de dados do *firewall*.

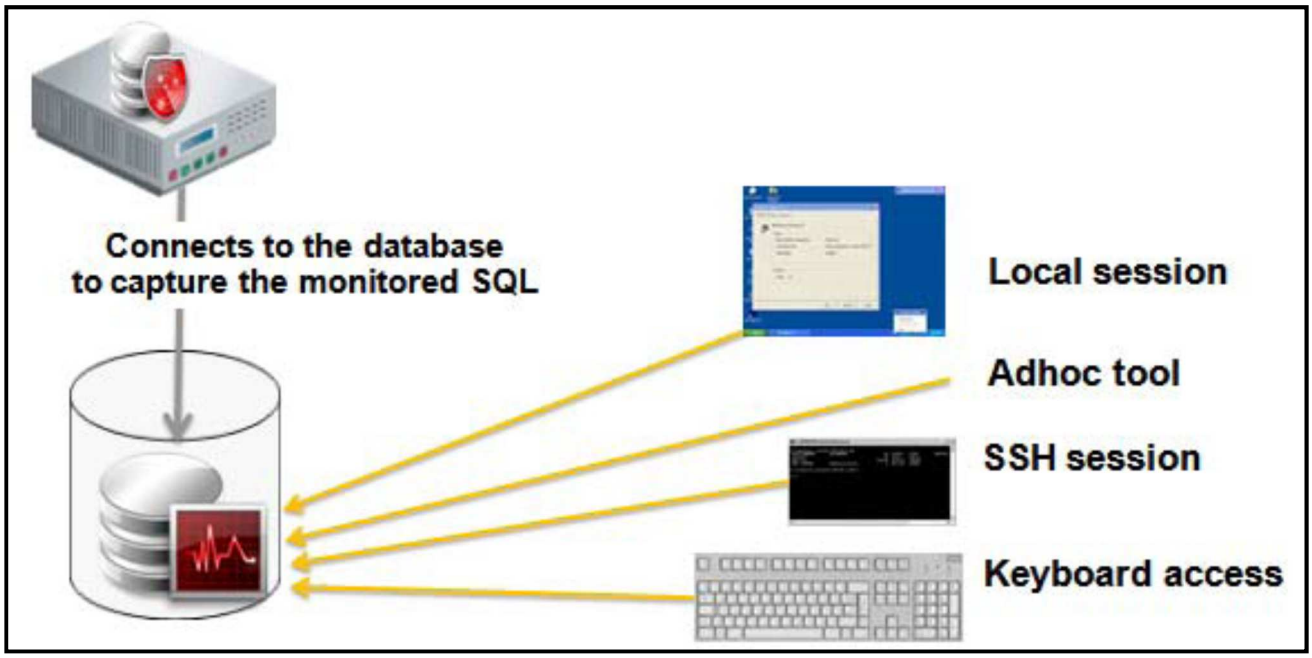
Tabela 2 – Database firewall tipos de monitores

Tipo de monitor	Metodos de operação
Monitor Remoto	Software de monitoramento (agente) é instalado no sistema operacional do host. O agente monitora o trafego da rede destinada a um ou mais schemas ou catálogos do banco de dados. Este agente captura todas as transações SQL e envia de volta para o firewall para obter alertas em tempo real, observações pós-evento e monitoração de relatórios. Um monitor remoto é usado quando não é possível implementar o Oracle Database Firewall frente ao host para capturar o trafego SQL.
Monitor Local	Tabelas adicionais são instaladas dentro dos bancos de dados monitorados para capturar o trafego SQL que são originados da fonte (de dentro do banco de dados) e tem acesso direto ao banco de dados, assim como console de usuários, batchs, scripts e Jobs que rodam diretamente dentro do servidor de banco de dados. Dessa forma o Firewall coleta os dados enviando querys para o banco de dados em intervalos regulares, e então usa os dados da mesma forma que as instruções geradas pelos usuários do banco de dados. Dependendo da politica desenvolvida, as instruções podem ser registradas e/ou gerar alertas.

Fonte: Bednar (2012)

A figura 5 mostra o funcionamento do Oracle database firewall com monitor baseado em host.

Figura 5 – Database firewall monitor baseado em host



Fonte: Bednar (2012)

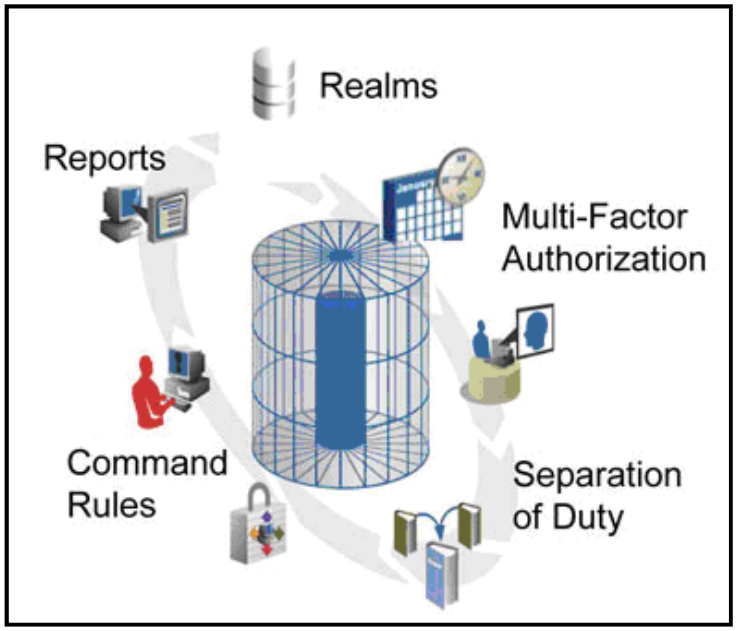
6. RECURSO DE RESTRIÇÃO ORACLE DATABASE VAULT

Conforme a Huey (2014) o Oracle *database vault* restringe o acesso de qualquer usuário a áreas especificas em um banco de dados Oracle, inclusive usuários que possuem permissões de administrador. Por exemplo, é capaz restringir acessos administrativos a tabelas com o salário dos empregados, tabela de relatórios médicos, ou outras informações privadas.

Tornando assim possível que sejam aplicados controles de acessos mais refinados aos dados privados de maneiras diferentes. Assim a proteção das instâncias do banco de dados Oracle acaba se tornando mais rígida e reforça as melhores práticas padrões de divisão de cargos de usuários com acessos mais fortes. O mais importante é que, mesmo protegendo os dados de usuários privilegiados (administradores), ainda assim eles são capazes de gerenciar o Oracle.

Com o Oracle *database vault*, a tarefa de tratar os problemas de segurança mais complicados de hoje em dia acaba se tornando mais simples, prático e fácil, sendo estes problemas: ataques internos, atender requisitos do cumprimento das regulamentações, e impor a separação de deveres.

Figura 6 – Visão geral do Oracle database vault



Fonte: Huey (2014)

A figura 6 ilustra todas as proteções que o Oracle database vault disponibiliza para deixar um banco de dados mais seguro. A tabela 3 descreve a função de cada um destes recursos.

Tabela 3 – Recursos do database vault

Recurso	Descrição
Domínios	São limites que age juntamente ao banco de dados Oracle como um firewall para prevenir que usuários com acessos extras acessem os dados da aplicação
Regras de comando	São regras que controlam a execução de comandos de banco de dados.
Fatores	São parâmetros do ambiente (endereço IP, métodos de autenticação) que podem ser usadas junto com as Regras de Comando e dos Domínios para criar um caminho seguro até os dados, definindo quem, quando, onde e como as aplicações, dados e bancos de dados são acessados.
Separação de Funções	O mínimo de privilégios de controle dentro do banco de dados que separa as principais ações administrativas externas (gerenciamento de conta, administração de segurança e administração do banco de dados)
Relatórios	São relatórios ligado a segurança que fornece detalhes de tentativas de violações do domínio e de outros reforços de controle do Database Vault

Fonte: Huey (2014)

6.1. Como o datavault ajuda com as regulamentações

Muitos regulamentos contêm temas em comum que requer um forte controle sobre o acesso a dados privativos, assim como a separação de funções. Enquanto muitas requisições regulamentares são procedimentos da instituição, soluções técnicas são necessárias para mitigar os riscos associados a itens como acessos não autorizados e modificação de dados.

A tabela 4 demonstra todas as regulamentações onde o database vault se adequa de forma que torna fácil serem cumpridas regras de auditorias e regulamentações internas da empresa.

Tabela 4 – Regulamentações que datavault se encaixa

Regulamentação	*Requisitos*	É aplicável ao Database Vault?
Sarbanes-Oxley seção 302	Prevenir modificações não autorizadas aos dados	Sim
Sarbanes-Oxley seção 404	Prevenir modificações aos dados e acesso não autorizado	Sim
Sarbanes-Oxley seção 409	Prevenir negação de serviço e modificações não autorizadas	Sim
Gramm-Leach-Bliley	Prevenir acesso não autorizado e modificação não autorizada	Sim
HIPAA 164.306, 164.312	Prevenir acesso não autorizado aos dados	Sim
Basel II – Internal Risk	Prevenir acesso não autorizado aos dados	Sim
CFR Part 11 (FDA)	Prevenir acesso não autorizado aos dados	Sim
Japan Privacy Law	Prevenir acesso não autorizado aos dados	Sim
PCI – Requerimento 7	Restringir o acesso dos dados do titular do cartão de pessoas que não precisam conhecer	Sim
PCI – Requerimento 8.5.6	Prover a capacidade de restringir o acesso aos dados do titular do cartão ou bancos de dados seguindo os seguintes critérios: - Endereço IP/Mac Adress - Aplicação/serviço - Contas de usuário/Grupos	Sim

Fonte: Huey (2014)

6.2. Como funciona um domínio datavault

Administradores de banco de dados e outros usuários privilegiados têm a função crítica de gerenciar o banco de dados. *Backup* e recuperação, melhoria de desempenho, e alta disponibilidade são apenas algumas tarefas do dia-a-dia que os

usuários privilegiados executam. De qualquer forma, a capacidade de prevenir que estes usuários vejam dados de aplicações privadas se tornou cada vez mais importante. Consolidação da aplicação e direitos de visualização de dados requer um forte controle sobre os acessos aos dados privativos do financeiro, recursos humanos, ambulatório, etc. O Oracle *database vault* como mencionado anteriormente previne que os usuários privilegiados vejam as informações geradas por estas aplicações devidas aos seus poderosos privilégios. Segundo a Huey (2014) o Domínio do Oracle *database vault* pode ser utilizado para proteger uma aplicação por inteira ou então proteger um grupo especifico de tabelas dentro da própria aplicação, provendo assim uma flexibilidade maior e um cumprimento adaptável de segurança.

6.3. Para que servem os fatores e regras de comando

De acordo com a Huey (2014) as regras de comandos do Oracle *database vault* permitem que multi-fatores de controle de autorização sejam estendidos além das funções (*roles*) tradicionais do banco de dados. Usando regras de comandos e multi-fatores de autorização, o acesso ao banco de dados pode ser restringido a uma sub-rede especifica ou a um servidor de aplicação, criando um caminho seguro para os acessos aos dados. O *database vault* fornece vários fatores embutidos, como endereço IP, que pode ser utilizado individualmente ou em combinação com outros fatores para aumentar significantemente o nível de segurança para a aplicação existente.

As regras de comando proporcionam a capacidade de facilmente anexar políticas de segurança aos comandos mais comuns de ser utilizado no banco de dados. Elas permitem que aumente os controles internos e que se aplique as melhores práticas para a empresa e configurações de políticas mais seguras. Por exemplo, uma regra de comando pode ser utilizada para prevenir que qualquer usuário, até mesmo o DBA ou o dono da aplicação, de excluir alguma tabela da aplicação no ambiente de produção.

6.4. O que é separação de função

De acordo com Huey (2014) a Separação de função Oracle *database vault* permite abordagens sistemáticas para a segurança que aumentam os controles dentro

do banco de dados e ajuda aos requerimentos das regulamentações, ele cria três tipos de responsabilidades distintas e separadas dentro do banco de dados onde a tabela 5 explica a função de cada uma.

Tabela 5 – Oracle datavault separação de função

Responsabilidade	Descrição
Gerenciador de contas	Um usuário com a responsabilidade de gerenciar contas pode criar, deletar ou modificar os usuários do banco de dados. Os usuários privilegiados existentes serão impedidos de fazer este gerenciamento de contas.
Administração de segurança	A responsabilidade da administração de segurança é conceber que usuários estejam aptos a se tornarem administradores de segurança (Database Vault Owner) do banco de dados. Um administrador de segurança pode controlar domínios, regras de comandos, fatores, e executa vários relatórios de segurança específicos do Oracle Database Vault.
Administrador de banco de dados	A responsabilidade do administrador do banco de dados é permitir que usuários com privilégios de DBA continuem a realizar as tarefas de gerenciamento e manutenção associados ao banco de dados assim como backup e recuperação, instalação de patches, e melhoria na performance sem ter acesso aos dados assegurados do negocio.

Fonte: Huey (2014)

A extensibilidade do Oracle *database vault* permite que a separação de função seja personalizada para os requerimentos do negócio. Por exemplo, é capaz de subdividir a responsabilidade de administração do banco de dados em *backup*, desempenho e aplicação de *patch* (pacotes). Se for em empresas pequenas, pode se unificar as responsabilidades, ou então nomear uma conta diferente para cada responsabilidade, tornando mais fácil a atividade de auditoria.

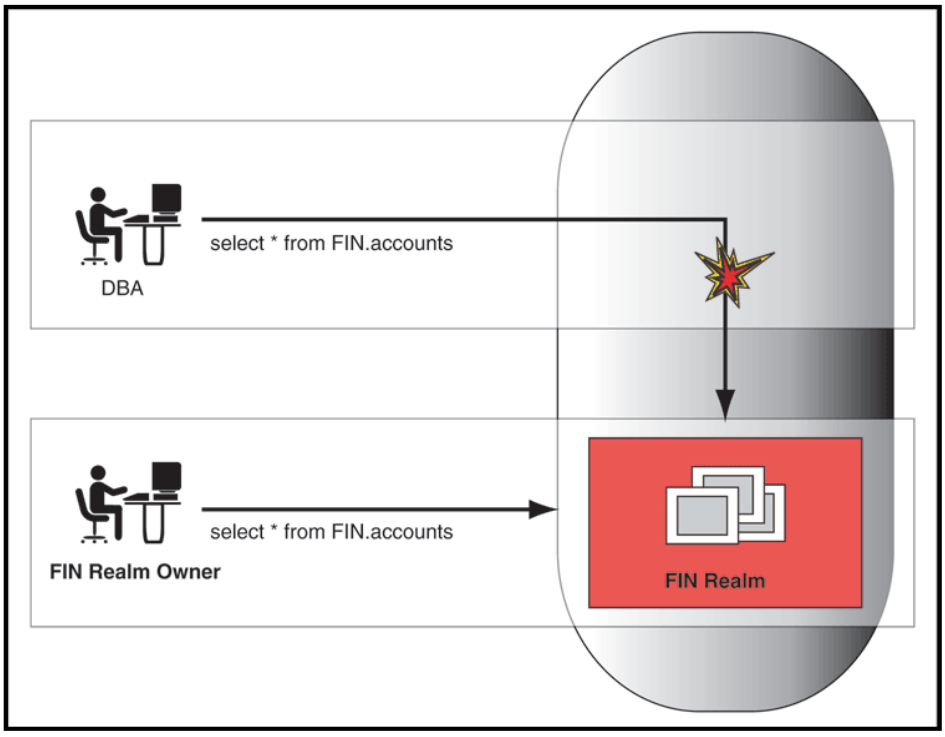
6.5. Como o oracle database vault soluciona ameaças internas

Por muitos anos, *worms*, vírus, e intrusos externos (*hackers*) foram percebidos como as grandes ameaças dos sistemas de computadores, e pode-se perceber que agora também é preciso tomar cuidado com o lado interno. Infelizmente, o que por

várias vezes é negligenciado é uma chance em potencial para usuários confiáveis ou usuários privilegiados roubar ou modificar dados.

O Oracle *database vault* protege contra estes ataques internos usando os recursos anteriormente mencionados e ilustrado na figura 7. Combinando todas estas ferramentas de segurança o acesso aos bancos de dados, aplicações e informações confidenciais se tornam mais confiável. Combinando regras com fatores é possível definir sobre quais condições o banco de dados pode executar os comandos, e o controle de acesso aos dados podem ser feitos pelo domínio.

Figura 7 – Overview do *Oracle* datavault



Fonte: Huey (2014)

E com o Oracle *database vault*, também é possível gerar vários relatórios que permite ao administrador ser informado a cada tentativa de acesso aos dados bloqueados por domínios. Por exemplo, se um DBA tenta acessar os dados de uma aplicação protegida por um domínio, o Oracle *database vault* vai bloquear este acesso e vai ser criado um registro de auditoria que pode ser facilmente visualizado usando o relatório de violação de domínio.

7. UTILITÁRIO DE BACKUP E RESTORE RMAN

Segundo Freeman e Hart (2010) o RMAN é uma aplicação *stand-alone* que faz uma conexão cliente no banco de dados Oracle para acessar os pacotes internos de *backup* e recuperação.

Ou seja, ele não é nada mais do que um interpretador de comando que quando recebe um comando envia uma requisição remota (*Remote Procedure Call* – RPCs) para ser executada no banco de dados.

Instalado como parte dos utilitários do banco de dados, O RMAN inclui extração de dados (*data pump*), SQL**Loader*, *DBNEWID*, e *dbverify*.

O utilitário RMAN é feito de duas partes: um arquivo executável e um arquivo chamado *recover*.*bsq*. Este arquivo *recover*.*bsq* é praticamente a biblioteca do RMAN, de onde o arquivo executável extrai os comandos para criar as chamadas PL/SQL, ou seja, o *recover*.*bsq* é o cérebro por trás de toda a operação. Estes dois arquivos estão ligados e logicamente formam o RMAN, então se algum desses arquivos for de versão diferente então o RMAN não funciona.

O RMAN serve a propósitos distintos, ordenados e previsíveis como: executar chamadas PL/SQL que serão remotamente executadas, a linguagem de comando do RMAN é única e necessita de alguma prática, ele essencialmente possui uma lista de todas as coisas necessárias para se fazer *backup*, restauração, ou recuperação do banco de dados, ou então para manipular estes *backups* de alguma maneira. Estes comandos são interpretados por um tradutor e comparado aos blocos PL/SQL do arquivo *recover*.*bsq*, e então repassa essas requisições para o banco de dados para obter a informação baseada no que foi pedido. Se o comando necessita de operações de I/O (em outras palavras, um comando de backup ou de recuperação), então quando esta informação retorna, o RMAN prepara outro bloco de procedimentos e passa de volta ao banco de dados. Estes blocos são responsáveis por engajar as chamadas ao sistema ao SO para operações de leitura e escrita especifica.

7.1. A necessidade de backup e restauração

É necessário lidar com duas áreas diferentes quando se trata de criar e executar um plano de redundância quando um banco de dados fica indisponível, no caso a primeira é alta disponibilidade, que logo anda bem próximo a segunda área, que no caso é o *backup* e a restauração. Nos próximos capítulos veremos cada uma em mais detalhes.

7.2. Alta disponibilidade

A alta disponibilidade implica em uma arquitetura que impede que o usuário fique parcialmente ou totalmente protegido de uma falha no sistema (banco de dados, rede, *hardware*, *software*, etc.). As soluções de alta disponibilidade podem ser incluídas como *drivers* espelhados, arquiteturas RAID (*redundant array of independent disks*), *cluster* de banco de dados, *schemas* de *failover*, e claro, *backup* e restauração. A alta disponibilidade adiciona custos à arquitetura global do banco de dados fora os custos da solução de *backup* selecionada. O RMAN não é parte da solução de alta disponibilidade, mais é uma parte muito importante para que a alta disponibilidade seja alcançada, porque é a parte que garante que os dados serão recuperados caso haja uma falha no *hardware* (HD, *Storage*).

As soluções de alta disponibilidade são as seguintes:

- Oracle d*ata guard.*
- Oracle *real application cluster.*
- RAID e discos espelhados.

7.3. Tipos de backup e restauração

O RMAN disponibiliza dois tipos primários de *backups*: *Offline* (*cold*) e *Online* (*hot*).

Os *backups offline* são realizados com o banco de dados parado (*down*), o que significa que durante o *backup* o sistema fica indisponível para o usuário. Por outro lado, também há o *backup online* que é realizado enquanto o banco de dados está

funcionando. Antes de se aplicar uma política de *backup* é necessário que seja levantado as necessidades do usuário e da empresa.

7.4. O que são modo archivelog e noarchivelog

De acordo com Freeman e Hart (2010) um banco de dados Oracle pode rodar em dois modos, que por padrão o banco de dados é criado no modo *noarchivelog*. Este modo permite que o banco de dados execute operações normais, mais não fornece a capacidade de realizar uma operação de recuperação *point-in-time* e nem um *backup online*. No modo *archivelog* o banco de dados faz uma cópia de todos os *redo logs* através de um processo chamado *ARCH*, para um ou mais diretórios de destino.

É necessário reparar que uma vez que o banco estiver funcionando em modo *archivelog*, o banco vai suspender suas atividades uma vez que todos os *online redo logs* estiverem sendo usados. O banco de dados fica suspendido até esses *online redo logs* serem arquivados. Então uma configuração errada do banco de dados quando estiver em modo *archivelog* pode eventualmente suspender seus serviços quando ele não puder mais arquivar seus atuais *redo logs*.

7.5. O que é um catalogo de recuperação

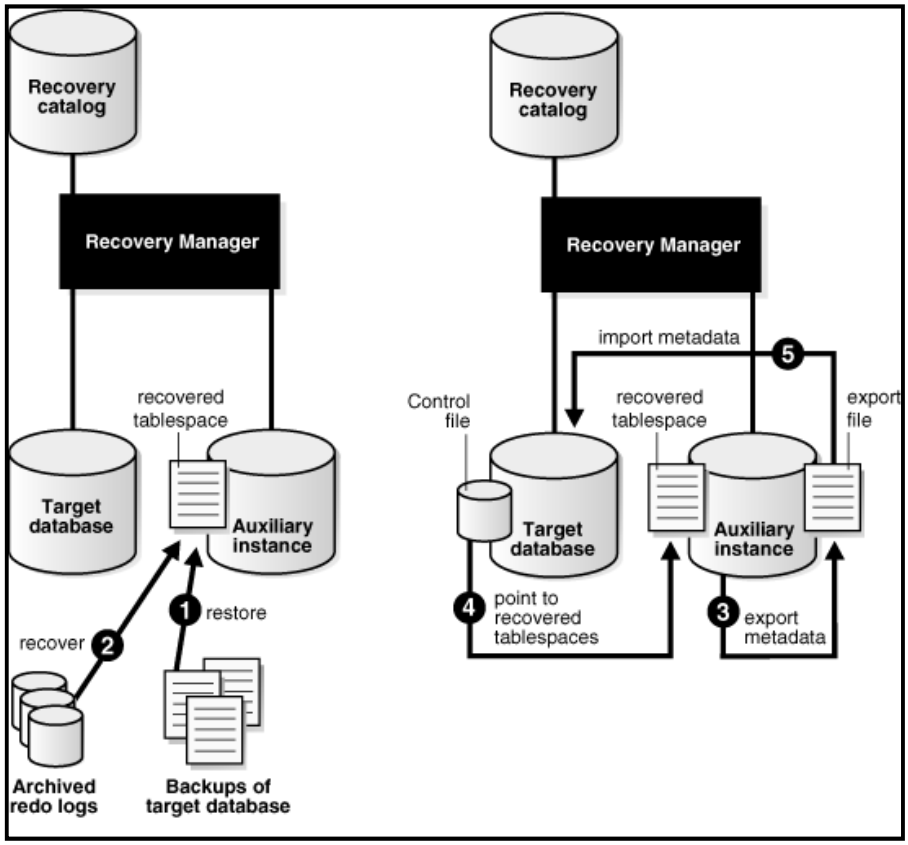
Conforme Freeman e Hart (2010) o catalogo de recuperação é um componente adicional do RMAN que armazena a informação do histórico de *backups* dos *backups* do RMAN. Diferentemente dos arquivos de controle do banco de dados, as informações do RMAN não são ligadas a uma base regular. Por isso, o histórico de informação do catalogo de recuperação tende a ser mais compreensivo e atualizado que o histórico de informação do arquivo de controle. O catalogo de recuperação traz com ele alguns benefícios tais como:

- Catalogo de recuperação armazena os *scripts* do RMAN.
- Oferece um catalogo único de repositório de informação do RMAN, tornando mais fácil de ver as informações dos *backups* realizados.
- Torna mais flexível a criação de relatórios, desde que já se possui um documento com o dia e *data* que ocorreu o *backup*.

Adicionalmente, o catalogo de recuperação é uma parte essencial do ambiente de backup do *data guard*. Nesta configuração, quando é feito o *backup* do banco de dados, o catalogo de configuração é considerado a informação mais atual, então ele acaba se tornando importante para a estratégia e também um ponto único de falha se não for gerenciado com atenção.

Quando conectado ao RMAN deve ser passado um parâmetro dizendo que deseja se conectar ao catalogo de recuperação. Por padrão o RMAN utiliza a opção *nocatalog*, que indica que o catalogo não será utilizado. Depois de usar o parâmetro de catalogo, é necessário indicar o *id* do usuário e a senha para o *schema* do catalogo de segurança que contem seus objetos (tabelas, índices, etc). A figura 8 demonstra como o *recovery manager* funciona e também papel do catalogo de recuperação.

Figura 8 – Como recovery manager funciona



Fonte: Freeman e Hart (2010)

7.6. Backups do rman ou scripts de backup

Por mais que *scripts* funcionem perfeitamente, alguns scripts cedo ou tarde acabam falhando mesmo que tenham sido criados com a maior perfeição. E isto pode causar dois graves problemas. Primeiro, quando um *script* falha por algum motivo, o *backup* também falha. Segundo, quando este *script* falha, alguém tem que consertar. E se este *script* foi desenvolvido por alguém que não está mais na empresa, ou foi transferido, se o *script* não possuir documentação ou comentários que ajude a entender como ele funciona e isto gera um grande problema, pois enquanto alguém trabalha para tentar compreender o que foi feito, o banco de dados pode estar parado, ou funcionando, entretanto, sem *backup*. Então aí vai um grande ponto a favor do RMAN, porque ele possui suporte da Oracle, e qualquer dúvida que se tenha sobre ele basta entrar no site oficial ou ligar para o suporte.

E claro que o RMAN também tem outros pontos positivos como:

- Detectar os blocos corrompidos e reporta.
- Fazer o *backup* do banco de dados *online* sem precisar colocar as *tablespaces* em modo de *backup hot* (que às vezes afeta o desempenho).
- Rastrear os novos *data files* e *trablespaces*, o que significa que não é necessário ficar adicionando ao script cada vez que um novo *data file* ou *tablespace* é criada.
- Realizar *backup* somente dos blocos de dados utilizados, tornando assim sua imagem menor que a gerada por um *script*.
- Fornecer compressão para as imagens de *backup*.
- Suportar estratégias de *backup* incremental.
- Fazer teste do *backup* sem ter que restaurar ele. O que não é possível nos *scripts* de *backup*.

7.7. Diferença entre recover e restore

No RMAN “recuperação” (*recover*) e “restauração” (*restore*) têm significados diferentes. Restauração implica em acessar um *backup* previamente feito e pegar um

ou mais objetos dele e restaura-los para algum local no disco. Uma recuperação é separada de uma restauração, então, a recuperação é o atual processo de fazer o banco de dados consistente até certo ponto no tempo para que ele possa ficar aberto, geralmente através da aplicação de redo (contidos no *online redo log* ou no *redo log* arquivado).

Segundo Freeman e Hart (2010) o banco de dados é específico sobre o estado dos dados dentro dele, e isto exige que o dado esteja consistente a uma dada situação no tempo em que o banco de dados está inicializando. Se o banco de dado estiver consistente então ele fica com estado aberto (*open*), caso contrário, ele não vai abrir (*not open*). Através do uso de segmentos de *rollback*, esta consistência é mantida enquanto o banco de dados está funcionando. Quando um banco de dados é desligado, normalmente, seus *datafiles* ficam consistentes novamente.

Infelizmente, uma inconsistência pode se arrastar dentro do seu banco de dados através de várias maneiras diferentes. Um shutdown anormal do banco de dados (como acontece com um *shutdown abort*, ou devido a uma queda de energia) vai fazer com que ele fique com estado inconsistente.

E claro que também há outras ameaças que podem afetar seu banco de dados. Assim como a perda dos datafiles do banco de dados através da perda de um HD, ou possivelmente perder o banco de dados inteiro. O problema também pode ser quando a restauração é feita mais o banco enxerga como se tivesse inconsistente. Ou pior ainda, se a restauração estiver com estado muito velho (10 horas antes de ocorrer o incidente ou mais).

Quando o Oracle é reiniciado após recuperar os *datafiles*, o Oracle detecta se aqueles *datafiles* estão inconsistentes, e para isto ele passa por três etapas: não montado (*nomount*), montado (*mount*) e aberto (*open*). Quando o banco de dados entra no estado aberto, ele checa o SCN no arquivo de controle com o SCNs de cada *datafile* do banco de dados. Se os SCNs não forem compatíveis, o Oracle sabe que está faltando alguma coisa e ele está inconsistente.

Outras checagens ocorrem durante a inicialização, e algumas dessas checagens não são de consistência. Por exemplo, o Oracle checa cada *datafile* procurando por um bit que indica que o *datafile* está em modo de *backup hot*. Se este

bit está setado, O Oracle não entra no estado aberto. Neste caso, um simples comando de *alter database datafile end backup* resolve o assunto.

Se o banco de dados está inconsistente, é preciso determinar de que forma ele pode ser recuperado baseado nos *online redo logs*, ou se vai ser necessário os *redo logs* arquivados para isto. Se for possível recuperar com os *online redo logs*, então o Oracle vai executar uma recuperação de acidente (ou a recuperação de uma instância no caso de RAC). Se o Oracle não conseguir fazer o banco de dados ficar consistente com os online redo logs, então ele para a inicialização e pede a mídia de recuperação necessária.

Mídia de recuperação é a condição onde o Oracle precisa dos *redo logs* arquivados (*archive redo log*) para recuperar o banco de dados e torna-lo consistente se ele estiver em modo *ARCHIVELOG*. Caso estiver em modo *NOARCHIVELOG*, então a restauração/recuperação de um *backup* consistente do banco de dados provavelmente será necessária. Algumas vezes será necessário recuperar um banco de dados inteiro, assim como em alguns casos será necessário recuperar somente algumas bases de dados (*datafiles*).

7.8. Rman e data guard

RMAN e *data guard* são tecnologias complementares que junto formam uma solução de recuperação de desastres e alta disponibilidade muito forte. Os *backups* do RMAN podem ser usados para criar um banco de *standby* oculto, assim fornecendo uma fase inicial para a recuperação. Então, depois de criado o banco de *standby* e configurado ao *data guard*, o RMAN pode conectar a ele e começar a tirar *backups* que poderão ser utilizados no servidor primário. Desta forma, os recursos usados para tirar *backups* podem ser removidos dos bancos de dados de produção e passarem aos de *standby*.

Uma forma que facilita já que o banco de *standby* está completamente sincronizado ao banco primário, e ainda melhor, pois não impacta no desempenho do servidor de produção.

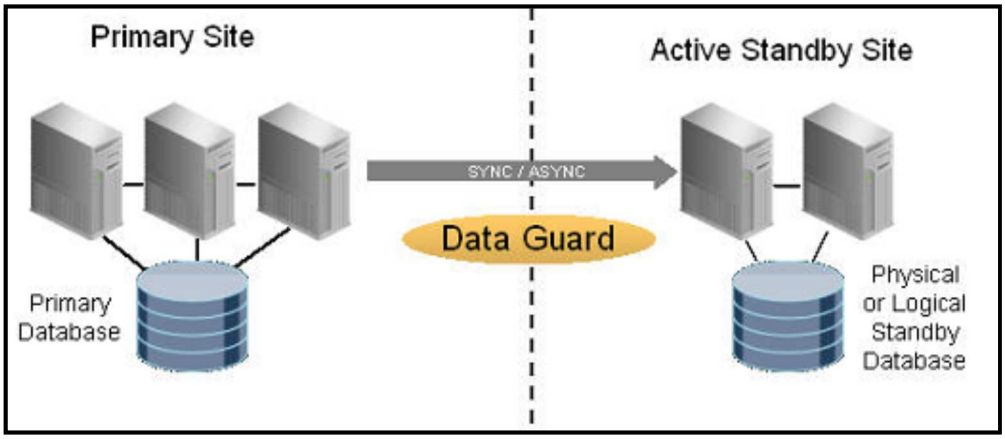
8. FERRAMENTA PARA ALTA DISPONIBILIDADE ORACLE DATA GUARD

O Oracle d*ata guard* mantém um ou mais banco de dados de *standby* para proteger os dados do Oracle de falhas, desastres, erros e corrupções. Fazendo o gerenciamento, monitoração e automatização da infraestrutura de *software*.

Segundo Meeks (2011) existem dois tipos de banco de dados em *standby*, um *standby* físico, onde ele usa a aplicação de Redo para manter uma cópia exata bloco a bloco do banco e o *standby* lógico, que usa a aplicação de SQL e contém a mesma informação lógica que o banco de dados primário, entretanto a organização física e a estrutura de dados podem ser diferentes conforme ilustrado na figura 9.

O administrador pode escolher entre manual ou automático para produzir um sistema de *standby* caso o servidor primário falhe.

Figura 9 – Visão geral do Oracle data guard



Fonte: Meeks (2011)

Os bancos de dados do *data guard* proporcionam um alto retorno no investimento também suportando queries *ad-hoc*, relatórios, *backups*, ou testes de atividade, enquanto fornece uma proteção a desastres. Especialmente:

- Opção *data guard* ativa: Permite que o *standby* físico do banco de dados seja usado por aplicações de somente leitura enquanto simultaneamente recebe atualizações do banco de dados primário. As queries executadas no *standby* ativo do banco de dados recebem resultados atualizados.

- *Snapshot Standby*: Permite que o *standby* físico do banco de dados esteja aberto a testes de leitura e escrita ou qualquer atividade que envolva leitura e escrita que venha de uma réplica dos dados da produção. Um *snapshot standby* continua a receber, entretanto não aplica as atualizações geradas pelo banco primário. Estas atualizações são aplicadas ao banco de dados em *standby* quando o snapshot *standby* é convertido de volta para standby físico. O banco primário é protegido o tempo todo.

- *Standby* Lógico: Este *standby* tem uma flexibilidade adicional de ser aberto a leitura e escrita, e enquanto os dados são gerenciados por aplicação SQL eles não podem ser modificados, e tabelas adicionais podem ser criadas ao banco de dados (já que a parte lógica é replicada e não a base de dados), e estruturas de índice local podem ser criadas para otimizar relatórios, ou então utilizar este banco de dados em *standby* como um data *warehouse*.

- Bancos de dados em *standby* podem ser usados para uma manutenção planejada. A manutenção primeiramente é feita no banco de dados em *standby*, logo após, o banco em *standby* assume o lugar do primário até as tarefas de manutenção sejam concluídas, o único tempo parado será até o servidor em *standby* assumir. Isto aumenta a disponibilidade e reduz o risco quando as atividades de manutenção em *hardware*, SO, aplicação de *patch*, atualização de versão, forem realizadas.

- Um *standby* físico, por ser uma réplica exata do banco primário, ele pode ser usado também para descarregar a carga de trabalho quando o banco primário estiver sobrecarregado executando rotinas de *backup*.

8.1. Funcionamento do Data guard

Segundo Meeks (2011) a configuração de um *data guard* inclui um banco de dados de produção, conhecido também como banco de dados primário, e mais de 30 bancos de dados de *standby*.

Desta forma sendo que os bancos primários e *standby* são conectados através de um serviço de rede da Oracle usando TCP/IP (não importa a localização). Um banco de *standby* inicialmente é criado da cópia do *backup* do servidor primário. O

banco de dados primário do *data guard* sincroniza com todos os outros bancos de dados em *standby* transmitindo o *redo* (a informação utilizada pelo Oracle para recuperar todas as transações) do banco de dados primário e aplicando nos bancos em *standby*.

8.2. Modos de proteção do Oracle data guard

De acordo com Meeks (2011) o *data guard* fornece três modos de proteção de dados para balancear custo, disponibilidade, desempenho e proteção de dados. Cada modo usa um método especifico de transporte de *redo*, e estabelecem regras que administram o comportamento da configuração do *data guard* toda vez que o banco de dados primário perder o contato com o *standby*. A tabela 6 descreve as características de cada modo.

Tabela 6 – Modos de proteção do data guard

Modo	Risco de perda de Dados	Transporte	Sem resposta do Banco de Standby, Então:
Proteção Máxima	Perda de dados Zero. Proteção a falhas dobrada.	SYNC	Banco de dados primário espera até receber a confirmação do banco de standby.
Disponibilidade Máxima	Perda de dados Zero. Ponto único de proteção a falhas.	SYNC	Banco de dados primário espera até receber a confirmação ou NET_TIMEOUT limite expire e então continua com o processamento
Desempenho Máximo	Possibilidade de perda mínima de dados	ASYNC	Banco de dados primário nunca espera pela confirmação do banco de standby

Fonte: Meeks (2011)

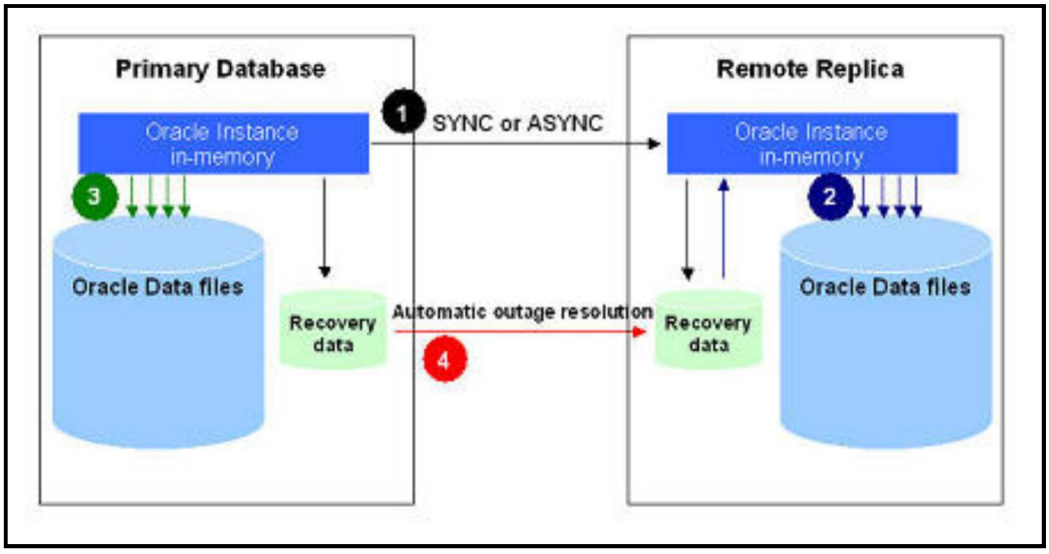
8.3. Serviços de transporte do Oracle data guard

Como os usuários dão *commit* em suas transações no servidor primário, o Oracle gera registros *redo* e escreve eles para um arquivo de *log online* local. Os

serviços de transporte do *data guard* transportam esse redo para os bancos de dados em *standby* tanto sincronamente ou assincronamente, onde é escrito no *redo log file* do servidor em *standby*. O *redo log* é transmitido de forma comprimida para evitar desperdício de banda.

Syncrhonous redo transport (SYNC) faz com que o banco de dados primário espere a confirmação do banco de dados *standby* de que o *redo log* já está no disco e somente depois disso será enviado uma confirmação de sucesso para a aplicação, provendo assim proteção zero à perda de dados. O desempenho do servidor primário é afetado pela soma do tempo requerida para que o I/O do arquivo de *redo log* e a transmissão pela rede sejam concluídos. Mas o *data guard* foi melhorado para reduzir o impacto ao banco primário durante o transporte SYNC. Agora o redo é transmitido para um *standby* remoto em paralelo com o *online log file* local no banco primário, efetivamente eliminando que o I/O do *standby* impacte no tempo total da transmissão. Isto permite uma separação geográfica ainda maior entre o banco primário e os de *standby* e uma configuração síncrona de perda de dados zero. Em redes de baixa latência pode ser reduzido o impacto da replicação SYNC do banco primário em quase zero, tornando atrativo um *standby* assíncrono complementar com um *standby* síncrono local para perda de dados zero, prevenindo contra falhas de componentes e falhas nos bancos de dados. Tudo isto está ilustrado na figura 10.

Figura 10 – Transporte de redo e aplicação de serviços



Fonte: Meeks (2011)

Conforme a Oracle o *asynchronous redo transport* (ASYNC) evita impacto ao desempenho do servidor primário enviando a confirmação de sucesso para a aplicação antes mesmo do banco *standby* receber o redo. As melhorias no *data guard* diminuíram os impactos de desempenho enviando diretamente o *log buffer* do banco primário (invés de enviar o *online redo log file*), diminuindo também o impacto na rede. O benefício de desempenho do ASYNC, entretanto, vem acompanhado de uma potencial perda de dados desde que não há garantias do recebimento do *redo log* pelos bancos em *standby*.

8.4. O que é recuperação rápida

Segundo Rich (2014) o *Fast-start* permite que o *data guard* automaticamente se recupere para um banco de dados em *standby* anteriormente escolhido sem precisar de nenhuma intervenção manual do administrador para invocar o *failover*.

O processo observador do *data guard* monitora continuamente o estado da configuração do *fast-start failover*. Se o processo observador e o banco de *standby* perde conexão ao banco primário, o observador tenta reconectar ao banco primário por um tempo previamente configurado antes de inicializar o *fast-start failover*. *Fast-start failover* foi projetado para assegurar que fora os três membros do *fast-start failover* – o banco primário, o banco *standby* e o observador - que pelo menos dois membros concordem com os principais estados de transição para evitar que um cenário de divisão ocorra. Uma vez que a falha no banco primário tenha sido reparada e montada, ele deve estabelecer comunicação com o processo observador antes de ficar em estado aberto (*open*). Quando for assim, vai ser informado que um *failover* ocorreu e o banco primário original é reintegrado como um servidor de *standby* do novo servidor primário (troca de função), um ótimo cenário para evitar perda de dados e alta disponibilidade.

8.5. Como funciona data guard e aplicação do redo

A opção *data guard* ativo possui vários recursos que aumenta a capacidade da aplicação de *redo* e do *standby* físico, incluindo:

- *Query* em tempo real: Permite acesso de somente leitura para um ou mais bancos de dados de *standby* físico para consultas, classificação, relatórios, acesso baseado em *web*, etc., enquanto a aplicação *redo* continua aplicando as mudanças recebidas do banco de dados primário. Em casos onde cargas de trabalho de apenas leitura podem ser isoladas das transações de leitura e escrita, o *data guard* ativo pode efetivamente dobrar a capacidade utilizando um *standby* físico que anteriormente estava inativo seguindo a função de *standby* (podem ser adicionados novos bancos de dados *standby* ativo para a configuração sem impactar em nada).

- *Active data guard service level agreements* (SLA): Pode ser implementado utilizando o parâmetro de sessão *STANDY_BY_MAX_DELAY*. O valor para este parâmetro especifica o limite para uma quantidade de tempo (em segundos) permitido para percorrer entre as mudanças que são aplicadas no servidor primário e quando elas podem ser consultadas no banco de *standby* ativo. O *standby* ativo retorna um código de erro se o limite for excedido. Aplicações entendem como se o banco estivesse desconectado e envia a solicitação para outro banco *standby* ou para o primário.

- *data guard* ativo permite a correção automática de blocos corrompidos. Em nível de blocos a perda de dados geralmente é intermitente, erros de I/O aleatórios, assim como corrupção de memória que é escrita no disco. Quando o Oracle descobre uma corrupção ele marca o bloco como mídia corrompida, escreve isto no disco e retorna um erro para a aplicação. Nenhuma leitura subsequente do bloco vai ser sucedida até que o bloco seja recuperado manualmente, entretanto se isto ocorre em um banco de dados primário que tem um banco de dados *standby* ativo, o bloco é recuperado automaticamente, transparente para a aplicação, usando uma cópia boa do bloco retirada do servidor de *standby*. Consequentemente, os blocos corrompidos do banco de *standby* são recuperados utilizando uma cópia boa dos dados do banco primário.

8.6. Validação dos dados do oracle

Uma das vantagens mais significantes do *data guard* é a possibilidade de usar os processos do Oracle para validar o *redo* antes de ser aplicado no banco de dados

de *standby*. O *data guard* é uma arquitetura acoplada livremente onde os bancos de dados de *standby* são mantidos sincronizados aplicando blocos redo, completamente separados de possíveis corrupções de arquivos de dados que podem ocorrer no servidor primário. *Redo* também é embarcado diretamente da memória (*system global area*), e então é completamente separado de qualquer corrupção de I/O no banco primário. Checagens de detecção de corrupção ocorrem no número de interfaces chaves durante o transporte e a aplicação do *redo*. O *software code-path* executado no banco de dados de *standby* também é fundamentalmente diferente do primário, efetivamente excluindo o banco de *standby* de erros de *firmware* e *software* que podem impactar no banco primário.

Segundo Meeks (2011) standby físico também utiliza o parâmetro: *DB_LOST_WRITE_PROTECT*. Se uma perda ocorrer na escrita enquanto um subsistema de I/O admite a finalização da escrita, enquanto na verdade essa escrita não ocorreu no *storage*, na leitura do bloco subsequente, o subsistema de I/O vai retornar a versão antiga do dado do bloco, que pode ser usado para atualizar outros blocos do banco de dados, assim corrompendo ele. Quando o parâmetro *DB_LOST_WRITE_PROTECT* é inicializado, então o banco de dados vai gravar um *buffer cache* de blocos lidos no *redo log*, e essa informação será utilizada pela aplicação de redo para determinar se houve perda de escrita, evitando *downtime*.

8.7. Serviço de aplicação do data guard

O serviço de aplicação lê o *redo* de um banco *standby*, valida, e então aplica no banco de *standby* (passo dois na figura de *redo transport*) usando tanto aplicação redo (*standby* físico) ou aplicação SQL (*standby* lógico). Notando que os serviços de aplicação e de transporte são completamente independentes. O estado ou desempenho de uma aplicação no *standby* não causa impacto no transporte do *redo* ou no desempenho do banco primário. O transporte do *redo* é uma parte muito importante para um ponto de recuperação, uma exposição potencial para perda de dados. Tudo que impactar negativamente no transporte vai aumentar potencialmente a perda de dados. Se algo impactar negativamente durante um transporte síncrono, então o tempo de resposta será afetado, a aplicação será afetada, o usuário será

afetado, ou seja, por isto é importante isolar a aplicação de serviço do transporte de *redo*, para aumento de desempenho, tempo de resposta rápido e proteção aos dados.

8.8. Função do standby físico com aplicação de redo

Um *standby* de banco de dados físico aplica o *redo* recebido do banco primário usando o *Managed Recovery Process* (MRP), uma extensão de conhecimento do *data guard* dos padrões de recuperação de mídia usados por todo banco de dados Oracle.

Conforme afirma Meeks (2011) um *standby* físico é idêntico ao banco primário em uma base bloco a bloco, e então, os *schemas* do banco de dados, incluindo índices, são os mesmos (tudo idêntico). O processo MRP é paralelo para maior desempenho.

Testes de desempenho conduzidos pela Oracle alcançaram a taxa de recuperação de 50MB/segundo pelo estilo de carga de trabalho OLTP, e 100MB/segundo por caminhos diretos. Aplicação de redo é o mais simples, rápido, e mais confiável método de gerenciar réplicas síncronas de um banco de dados primário.

8.9. Como funciona aplicação de SQL em standby lógico

A aplicação SQL mantém o *standby* lógico sincronizado transformando o *redo* do banco primário em instruções SQL e então aplicando ele no servidor de *standby* que é aberto a leitura e escrita. A aplicação SQL tem algumas restrições quanto aos tipos de dados, tipos de tabelas, e tipos de operações DDL e DML.

Deve-se usar a aplicação SQL quando se quer:

- Executar relatórios de aplicações que necessitam de acesso leitura e escrita no banco de dados de *standby*. Dados mantidos por aplicação SQL não podem ser modificados.
- Adicionar tabelas, *schemas* adicionais, índices e visualizações materializadas que não existem no banco de dados primário.
- Aplicar atualizações de maneira dinâmica para evitar *downtime*.

8.10. Resolução automática de atraso

Em casos onde o banco de dados primário e de *standby* ficam *off-line* (falha na rede ou nos servidores), e dependendo de qual modo de proteção utilizado, o banco de dados primário vai continuar processando transações e acumular um *backlog* do *redo* que não pode ser enviado ao *standby* até uma próxima conexão ser estabelecida. Enquanto neste estado, o *data guard* continua a monitorar a condição, e detecta quando a conexão estiver reestabelecida, e automaticamente ressincroniza o banco de dados de *standby* com o primário (passo três na figura de transporte de redo). Nenhuma intervenção administrativa é necessária enquanto os *archive logs* necessários para a ressincronização dos bancos de *standby* estiverem disponíveis no disco do banco primário. Em caso de uma interrupção estendida onde não se contém os *archive log* necessários para a sincronização, então é necessário voltar um *backup* utilizando a ferramenta RMAN.

9. INTRODUÇÃO AO ORACLE REAL APPLICATION CLUSTER

O Oracle *Real Application Cluster* (RAC) permite que banco de dados execute qualquer serviço ou aplicação, sem qualquer alteração por toda a área de servidores. Isto fornece o mais alto nível de disponibilidade e escalabilidade. Se um servidor nesta área de servidores falha o banco de dados continua funcionando somente com os servidores que ainda estão funcionando normalmente. Se for necessário mais poder de processamento, basta simplesmente adicionar outro servidor no pool sem precisar interromper o sistema e usuário.

Segundo Lundhild e Michalewicz (2010) RAC prove uma base para uma arquitetura de nuvem privada da Oracle.

Desta forma ajuda a diminuir os custos com hardware, sua flexibilidade ajuda também a diminuir custos com administração e a partir de versões mais atuais ele vem com uma arquitetura *Toad* que não vai ser discutida neste tópico, mas ajuda a suprir alguns pontos negativos que serão apontados mais à frente.

9.1. O que é o oracle real application cluster

O Oracle RAC possibilita o dimensionamento de aplicação além da capacidade de um único servidor, trazendo assim uma maior comodidade quanto à questão de *hardware* e provendo uma escalabilidade maior para o ambiente suportar a carga de trabalho de uma aplicação. Este é um dos recursos que a Oracle disponibiliza que pode manter o serviço principal da sua empresa *online* com um melhor desempenho e redundância perante a falha de *software* ou *hardware*.

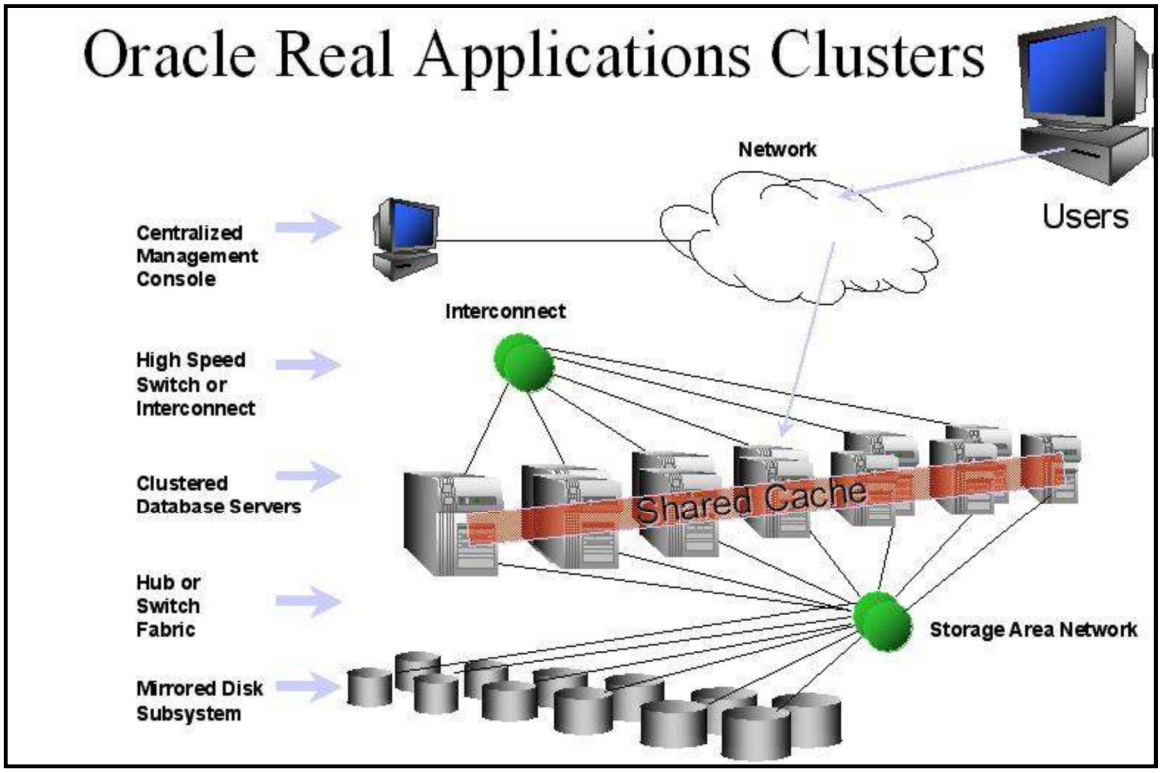
Segundo Scardine (2010) escalabilidade de um sistema é a função do consumo de recursos em relação à carga de trabalho. Idealmente, essa relação seria linear, ou seja, ao dobrar a carga de trabalho em determinado sistema também dobraria o consumo de recursos.

9.2. Arquitetura do oracle RAC

Um banco de dados Oracle RAC é um banco de dados *clusterizado*. Um *cluster* pode ser descrito como uma área de servidores independentes que cooperam como um único sistema. Essa área de servidores proporciona uma maior resistência a falhas. Em uma eventual falha do sistema, o cluster assegura alta disponibilidade aos usuários. Acesso a dados críticos não é perdido. Componentes de hardware redundante, como servidores adicionais, conexões de rede e discos (Hds), permitem que o cluster forneça uma maior disponibilidade. A arquitetura de *hardware* redundante evita que haja um ponto único de falha e fornece uma tolerância a falhas excepcional ilustrado na figura 11.

Com o Oracle *real application cluster*, o Oracle desacopla a instância do Oracle (a estrutura de processos e memoria executada no servidor que permite acesso aos dados) do banco de dados (as estruturas físicas residentes no *storage* que guardam os dados, conhecidas como *datafiles*).

Figura 11 – Arquitetura do Oracle RAC



Fonte: Lundhild e Michalewicz (2010)

Um banco de dados *clusterizado* é um único banco de dados que pode ser acessado por múltiplas instâncias e cada instância é executada em um servidor separado na área de servidores. Quando recursos adicionais são necessários, novos servidores e instâncias podem ser facilmente adicionados à área de servidores sem precisar parar a aplicação ou o banco de dados, ou seja, sem *downtime*. Uma vez que esta nova instância esteja funcionando as aplicações podem pegar seus recursos.

9.3. Para que serve a arquitetura de hardware

A arquitetura do Oracle RAC é uma arquitetura de compartilhamento de “tudo”. Todos os servidores na área de servidores compartilham o *storage* inteiro. O tipo de área de *storage* pode ser o *network attached storage* (NAS), ou o *storage area network* (SAN) ou então discos SCSI (*small computer system interface*), a escolha do método de *storage* é ditada pela escolha do *hardware* do servidor ou pelo *hardware* especificado pela aplicação (se é suportado ou não). Uma dica importante para escolher a área de *storage* é escolher um *storage* que proporciona I/O escalonáveis para a aplicação e um sistema de I/O que vai ampliar a medida que novos servidores são adicionados a área.

Um banco de dados Oracle RAC necessita de conexão de rede com a rede local (LAN) que o servidor de banco de dados está ligado à conexão da aplicação. A área dos servidores também requer uma conexão privada conhecida como “*the interconnect*” (por questões de alta disponibilidade a Oracle recomenda Interconectores redundantes). Este *interconnect* é usado pelo Oracle c*lusterware* para enviar mensagens entre os nós. O *interconnect* também é usado pelo Oracle RAC para implementar a tecnologia de fusão de *cache*.

Uma área de servidores é composta de mais de 1 servidor, cada um contendo uma conexão publica a LAN, uma conexão *interconnect*, e deve estar ligado a área compartilhada de *storage*. O Oracle c*lusterware* e o Oracle RAC suporta mais de 100 nós no cluster segundo a documentação oficial da Oracle. A configuração de *hardware* não precisa ser exatamente a mesma, mas cada servidor deve rodar o mesmo sistema

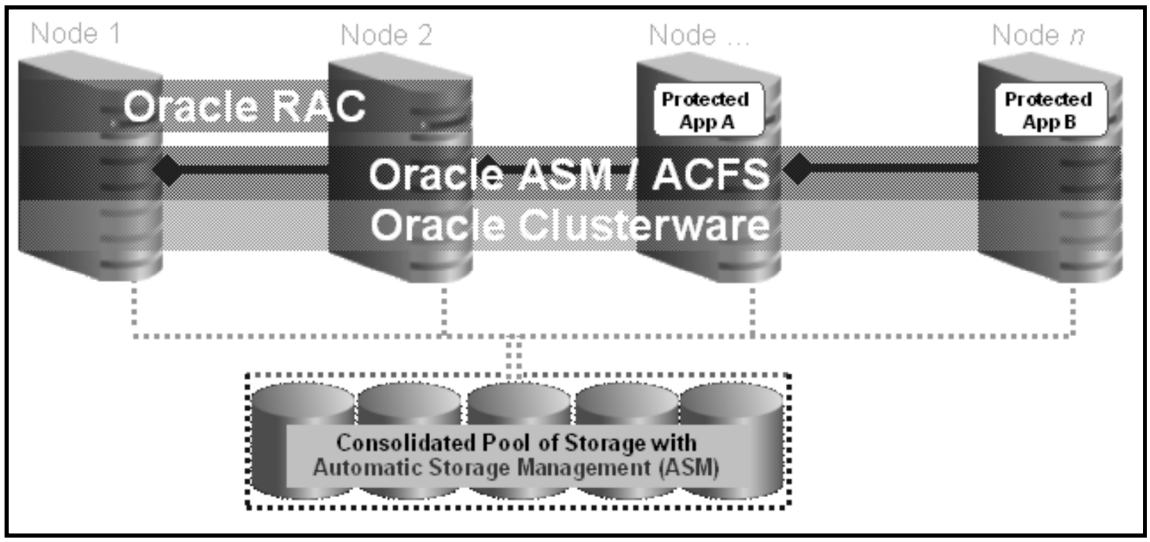
operacional e a mesma versão do Oracle, e todos os servidores deve suportar a mesma arquitetura. Exemplo: todos os servidores devem ser ou 32 ou 64 bits.

9.4. O que é clusterware

O Oracle *clusterware* é uma solução que é integrada e desenvolvida especificamente para banco de dados Oracle. Este recurso proporciona uma solução de cluster confiável e suporta qualquer aplicação. O Oracle *clusterware* é um pré-requisito para todas as implementações de Oracle RAC. Pode-se ter uma outra solução de *cluster*, entretanto se for utilizado o RAC ainda assim é preciso do Oracle *clusterware* para gerenciar todos os bancos de dados Oracle conforme ilustrado na figura 12.

Segundo Lundhild e Michalewicz (2010) o Oracle *clusterware* monitora e gerencia os bancos de dados Oracle RAC. Quando um servidor que está na área de servidores é inicializado, todas as instâncias, *listener* e serviços são automaticamente iniciados. Se uma instância falha, o Oracle *clusterware* vai automaticamente reinicia-la para que o serviço seja recuperado várias vezes antes mesmo do administrador perceber que estava parado.

Figura 12 – Visão geral do Oracle clusterware



Fonte: Oracle (2010)

Quando estiver registrando o serviço no Oracle *clusterware*, deve ser incluída a informação de como inicializar, parar e monitorar o processo. E também, podem-se

especificar servidores como candidatos para tomarem conta do recurso quando ocorrer uma falha. Com o recurso do *Oracle clusterware* é capaz de facilmente modelar planos de redundância para complexos cenários de *failover* e de restauração.

9.5. Escalabilidade

O Oracle *real application cluster* possui uma tecnologia única para dimensionamento de aplicação. Tradicionalmente, quando servidores de banco de dados ficavam sem capacidade, eles eram substituídos por servidores novos e mais potentes. Como os servidores estão cada vez mais potentes, então seus preços estão mais caros. Em bancos de dados usando Oracle RAC há algumas alternativas para aumentar a capacidade, essas aplicações que estão consumindo muito de um servidor podem ser migradas para uma área de servidores, também pode se manter o investimento no *hardware* que já possui e adicionar mais servidores à área de servidores ou então criar uma área de servidores se já não existir uma. Para adicionar mais servidores na área de servidores não é preciso a interrupção na aplicação e assim que o novo servidor já estiver funcionando dentro da área de servidores a aplicação já vai poder utilizar seus recursos.

A arquitetura do Oracle RAC acomoda automaticamente as rápidas mudanças nos requerimentos do negócio resultando em mudanças nas cargas de trabalho. Usuários de aplicações, ou clientes de meia camada de aplicação, conectado ao servidor em nome de algum serviço. O Oracle automaticamente balanceia o uso entre os múltiplos nós na área dos servidores. As instâncias do banco de dados Oracle RAC em diferentes nós subscrevem todos ou alguns subconjuntos de serviços de banco de dados. Isto prove aos administradores de banco de dados uma maior flexibilidade de escolher que aplicação especifica se conecta a um serviço de banco de dados pode conectar a alguns ou todos os nós. Os administradores podem sem trabalho algum adicionar capacidade de processamento conforme a demanda da aplicação cresce. A *Cache Fusion Architecture* do Oracle RAC imediatamente utiliza os recursos de CPU e memória dos novos integrantes do nó, assim os DBAs não precisam fazer este serviço manualmente.

Outro jeito de distribuir carga de trabalho em um banco de dados Oracle é através do recurso de execução paralela. A execução paralela (I.E *Parallel query* ou

Parallel DML) divide o trabalho de executar um comando SQL através de múltiplos processos. Usando o Oracle *cost-based optimizer* decisões inteligentes são tomadas com respeito nó-local e ao nó-externo.

Por exemplo, se uma *query* em particular requer seis *querys* para completar o trabalho e seis cpus estão disponíveis no nó local (o nó onde o usuário se conectou), então a *query* é processada somente usando os recursos locais. Isto demonstra um eficiente paralelismo no nó local e impede que a *query* saia usando o recurso dos outros nós. Agora se há somente duas CPUs disponíveis, então a *query* usa quatro CPUs de outro nó para terminar de executar. Desta maneira, é utilizado paralelismo entre ambos o nó-local e nó-externo para fornecer operações mais rápidas.

9.6. Balanceamento de conexão

O serviço de rede da Oracle fornece o *Connection Load Balancing* para as conexões de banco de dados. De acordo com a Oracle o *Client-side load balancing*, balanceia as requisições de conexão através todos os *listeners* dentro do cluster, que é alcançada usando uma lista de endereço que se encontra na *string* de conexão do cliente. Caso o endereço não responder então será direcionado ao próximo servidor da lista. O *listener* está ciente de todas as instâncias dentro do *cluster* e de seus respectivos serviços. Assim baseado na meta definida pelo serviço, o *listener* escolhe a instância que melhor atenderá a meta e a conexão é roteada para a instância.

9.7. Função da área de servidores

Um banco de dados pode ser definido para funcionar dentro de uma área de servidores. Conforme informa a Oracle, a área de servidores é uma entidade lógica em um *cluster* que permite que o administrador aloque os recursos para uma aplicação especifica. Uma área de servidores pode ser definida por três atributos: min (há um número mínimo de servidores na área), *max* (o número máximo de servidores em uma área) e *importance* (proporciona a habilidade de dar uma importância relativa a diferentes áreas dentro do *cluster*). Oracle *clusterware* aloca servidores para as áreas definidas pelo usuário quando a reconfiguração tiver efeito.

Segundo Lundhild e Michalewicz (2010) o número de instâncias mantidas pelo banco de dados é definido pela cardinalidade da área de servidores.

As políticas podem ser definidas através do assistente de configuração do Oracle (DBCA).

9.8. Quais são os benefícios do oracle real application cluster

O Oracle r*eal* a*pplication* c*luster* é essencial para uma base de dados de alta disponibilidade. E também é um dos componentes principais do Oracle's *Maximum Availability Architecture* (Oracle MAA), que prove as melhores práticas para proporcionar um centro de dados de alta disponibilidade. Segundo a Oracle os benefícios do RAC são os seguintes:

- **Confiabilidade** – O banco de dados Oracle é conhecido também por sua confiabilidade. E o Oracle RAC coloca ele um passo à frente removendo um único ponto de falha. Se uma instância falha, as instâncias restantes na área de servidores continuaram abertas e ativas. O Oracle *Clusterware* monitora todos os processos do Oracle e imediatamente reinicia qualquer componente que tenha falhado.

- **Recuperabilidade** – O banco de dado Oracle inclui vários recursos que torna fácil recupera-lo de qualquer tipo de falha. Se uma instância falha em um ambiente Oracle RAC, logo é reconhecido por outra instância na área de servidores e a recuperação começa automaticamente. *Fast Application Notification* (*FAN*) e o *Fast Connection Failover* (*FCF*) ou *Transparent Application Failover* (*TAF*), estes recursos tornam mais fácil para as aplicações mascarar falhas nos componentes para os usuários (usuários não percebem se o sistema falha).

- **Detecção de Erro** – O Oracle *Clusterware* automaticamente monitora os bancos de dados do Oracle RAC assim como os outros processos do Oracle (ASM, *listener*, etc) e proporciona uma rápida detecção de problemas no ambiente. Também como dito anteriormente o *Clusterware* automaticamente corrige as falhas várias vezes antes que percebam que ouve uma falha. O *Fast Application Notification* (FAN) fornece a habilidade de a aplicação receber imediatamente uma

notificação de falha no componente do *cluster* com a finalidade de reemitir a transação antes que o erro vá para a superfície (chegue ao usuário).

- **Operações continuas** – Oracle *Real Application Cluster* proporciona serviço continuo para ambas interrupções, planejadas e não planejadas. Se um servidor (ou instância) falha, o banco de dado continua aberto e a aplicação é capaz de acessar os dados. A maioria das operações de manutenção feitas no banco de dado pode ser completada sem *downtime* e acaba sendo transparente para o usuário. Vários outros serviços de manutenção podem ser realizados com um *downtime* reduzido ou removido. *Fast Application Notification* (*FAN*) e o *Fast Connection Failover* auxilia a aplicação.

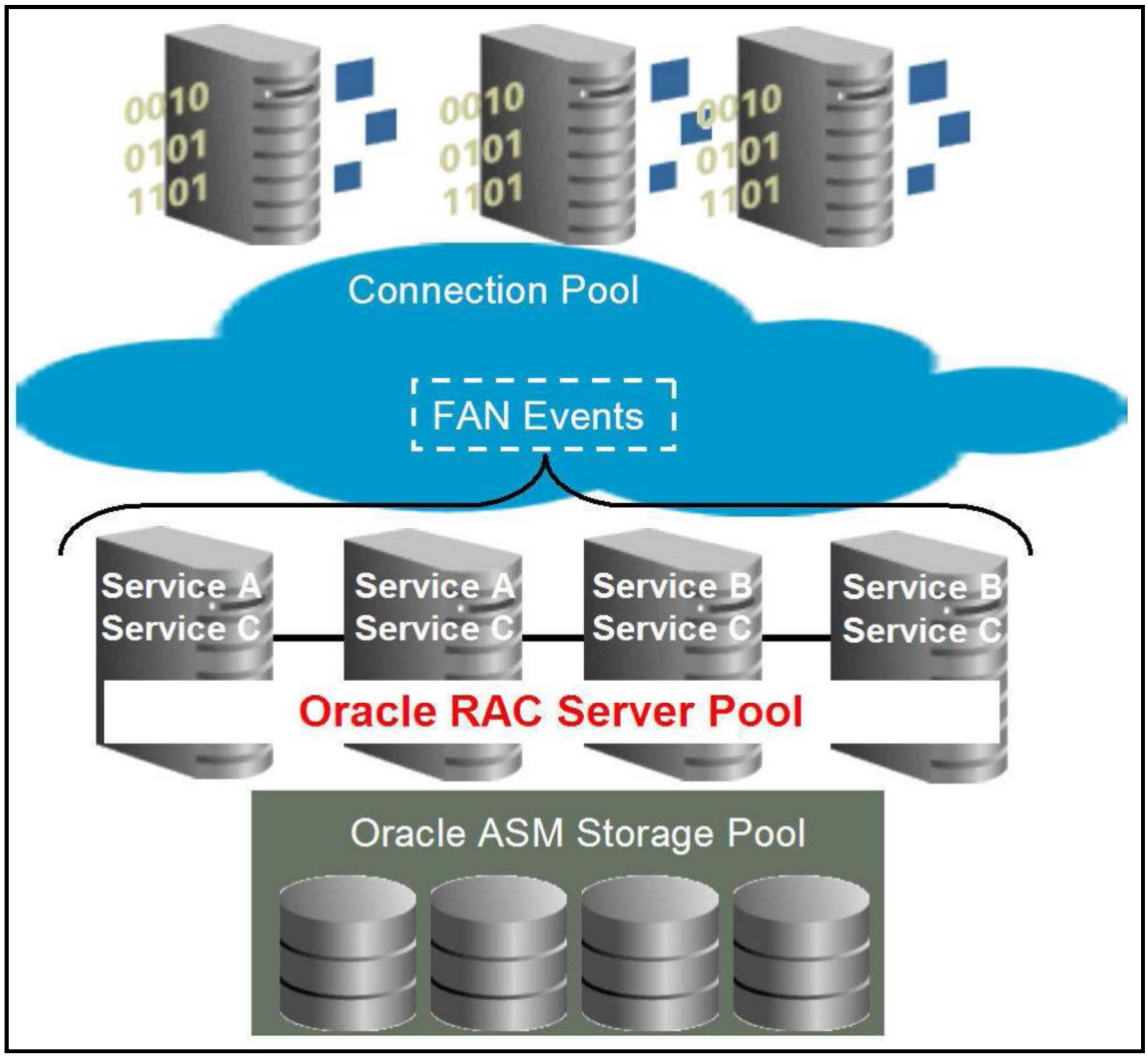
9.9. Serviço de Notificação Rápida

Segundo Lundhild e Michalewicz (2010) o Oracle *Fast Application Notification* (FAN) fornece integração entre o banco de dados do Oracle *RAC* e a aplicação. Isto permite que a aplicação fique ciente da atual configuração da área de servidores de qualquer horário determinado para que as conexões da aplicação sejam feitas apenas para aquelas instâncias que estejam atualmente abertas para responder as requisições da aplicação. O quadro *HA* do Oracle RAC posta um evento FAN imediatamente quando um estado muda dentro do *cluster* conforme ilustrado na figura 13.

Os clientes integrados recebem estes eventos e imediatamente reagem. Para eventos *DOWN*, a interrupção da aplicação é minimizada limpando as conexões há instância parada, transações em execução são interrompidas com um erro retornado para a aplicação. Aplicações fazendo conexões são direcionadas apenas para instâncias ativas.

Para eventos *UP*, novas conexões são criadas para permitir que a aplicação imediatamente utilize recursos extras disponíveis.

Figura 13 – Balanceamento de cargas em um Oracle RAC



Fonte: Lundhild e Michalewicz (2010)

10. ANÁLISE COMPARATIVA E EXPERIMENTAL

Os softwares utilizados para o experimento foram: *Virtual Box* para instalar todos os servidores virtuais, três servidores virtuais com Oracle *Linux* 5.8, *Asmlib* para *kernel* 2.6, Oracle *Grid Infraestructure* 11.2.0.1, Oracle *database* 11.2.0.1 (*release* 2), um servidor virtual *Windows server* 2012 com *active directory* e DNS instalado.

O recurso escolhido para experimento é o Oracle *Real Application Cluster*, o teste realizado serve para demonstrar que a alta disponibilidade realmente funciona, sendo assim, durante a execução da *query* de um usuário o nó (*node*) onde a atividade está sendo realizada será desligado, deixando o servidor onde está sendo processada a requisição indisponível, deve se notar qual o comportamento do Oracle RAC. O esperado para este cenário é que a *query* executada pelo usuário congele durante alguns segundos, enquanto a sua requisição é encaminhada a outro nó (*node*) disponível, assim como todas as demais sessões que estão ativas neste nó serão redistribuídas entre os nós que ainda permaneceram disponíveis.

Para funcionar o *transparent application failover* (TAF), o *real application cluster* (RAC) deve estar configurado de forma adequada e o client deve estar utilizando um driver compatível com essa tecnologia. Para esta demonstração será utilizado o Oracle *Call Interface* (OCI). O cliente se conectará ao banco através do serviço chamado *TSP*, abaixo está a entrada *tns: TSP=(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = rac-scan)(PORT = 1521)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = TSP.localdomain) (FAILOVER_MODE = (TYPE = SELECT) (METHOD = BASIC))))*

Para o *load balance* (balanceamento de carga) funcionar, o *domain name system* (*DNS*) deve estar devidamente configurado, de modo a permitir que o *SCAN* (*Single Client Access Name*) resolva para 2 ou mais endereços *IP* de forma rotativa, conhecido tipicamente como *Round Robin*. As conexões chegam ao *SCAN Listener* e através desses *IPs*, onde são então direcionados por *VIP*s (*Virtual IPs*) para o *Listener* local, que estabelece a conexão com a base de dados.

Para a demonstração do balanceamento de carga (*load balance*), foi utilizado um driver bastante comum em projetos, o JDBC (*Java Database Connectivity*), pois a aplicação foi desenvolvida em *java*, e este era um sistema de vendas online.

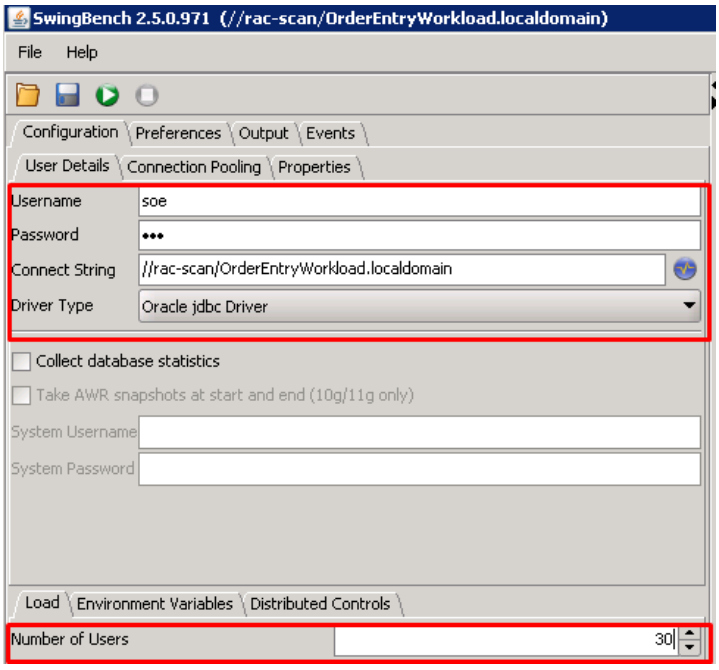
O teste realizado utilizou a conexão de 30 usuários, como o ambiente possui um *cluster* de 3 nós, o comportamento esperado é que 10 usuários devem se autenticar em cada servidor, desta forma demonstrando que o balanceamento de cargas está funcionando. O nome dado ao *SCAN* foi *rac-scan*, o nslookup para o *rac-scan* resolve para 3 *IP*s diferentes de forma rotativa, observe abaixo um exemplo:

C:\>*nslookup rac-scan*

Server: *isbrdc001.siselo.demo.net*
Address: 144.180.76.91
Name: *rac-scan.localdomain*
Addresses: 144.180.76.123
144.180.76.121
144.180.76.122

Abaixo seguem as figuras ilustrando o teste do balanceamento de cargas, onde 30 conexões foram criadas ao banco de dados, em seguida o teste para verificar onde estes usuários se conectaram.

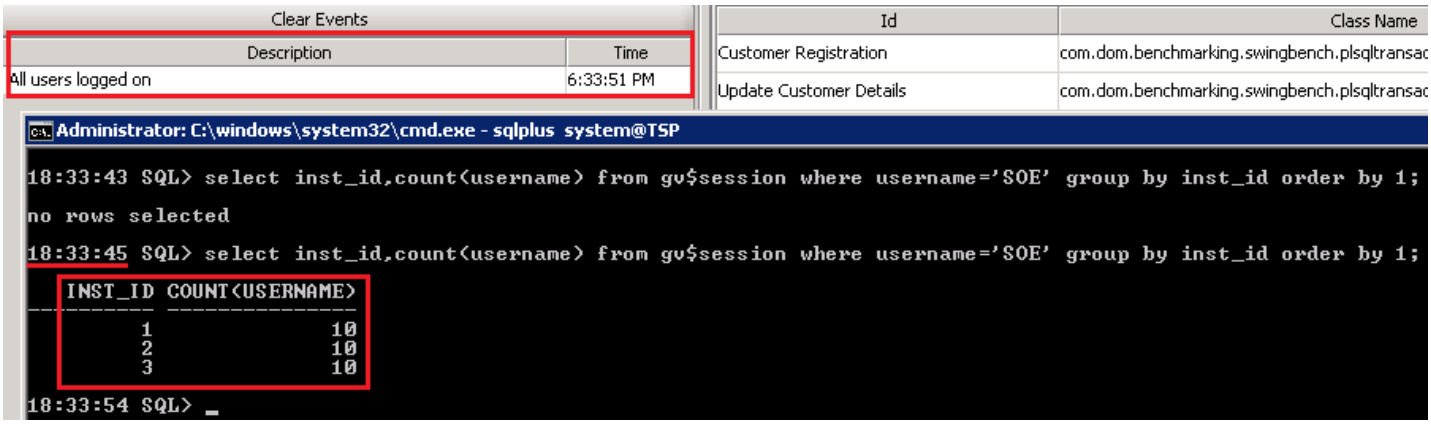
Figura 14 – Teste de balanceamento de cargas conectando 30 usuários



Fonte: Próprio autor

Ao conectar no Oracle RAC e verificar onde estas sessões de usuários se conectaram este foi o resultado.

Figura 15 – Balanceamento de cargas funcionando

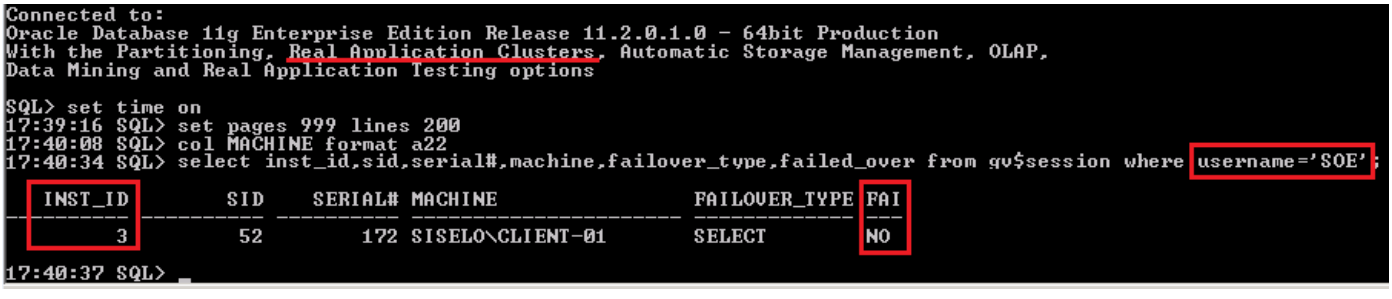


Fonte: Próprio autor.

Agora que o balanceamento de cargas foi testado e está funcionando de acordo com o ilustrado na imagem 15, as conexões foram divididas em 10 usuários para cada um dos nós.

O seguinte passo é testar se a alta disponibilidade está funcionando e se o *failover* realmente ocorre caso um nó ficar indisponível. Nesta etapa figura 16, existe um usuário chamado “SOE” que está conectado ao *RAC*, o comando executado verifica em qual nó do *RAC* ele se conectou, para este caso foi o nó 3.

Figura 16 – Verificar sessão do usuário no Oracle RAC

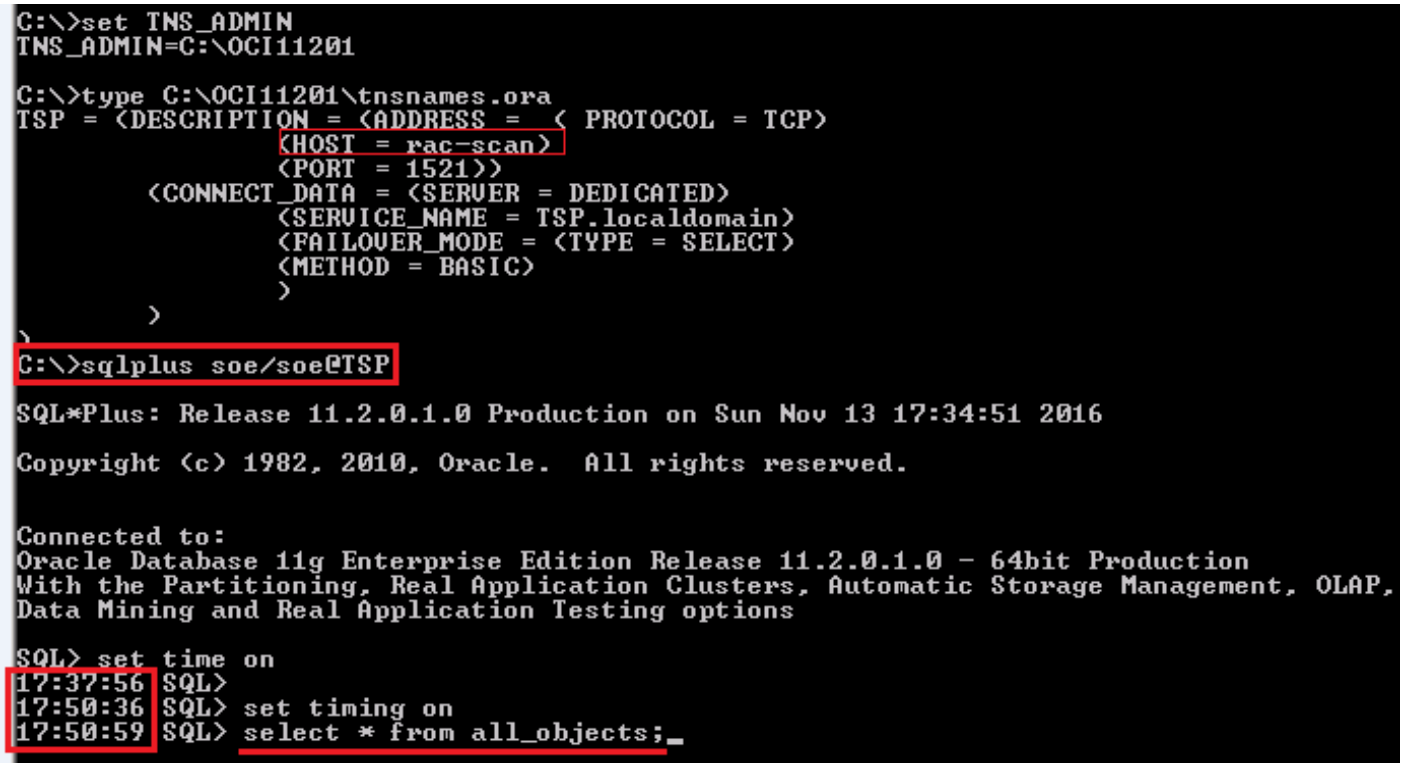


Fonte: Próprio autor.

Nesta atividade é verificado se a conexão será no realizada no *rac-scan*, que é o nome virtual que redireciona a conexão para um dos três nós do *RAC*. Uma vez que o usuário “SOE” se conectar ele será redirecionado a um dos nós do *RAC*, que conforme a figura 16, a conexão foi direcionada ao nó 3. É necessário prestar muita

atenção aos horários que são executadas cada atividade, as 17:37 o usuário estava conectado, 17:40 foi verificado onde o usuário se conectou e as 17:50 foi enviado o comando de *select* como ilustrado na figura 17.

Figura 17 – Conexão do usuário “SOE” no rac-scan e execução do select

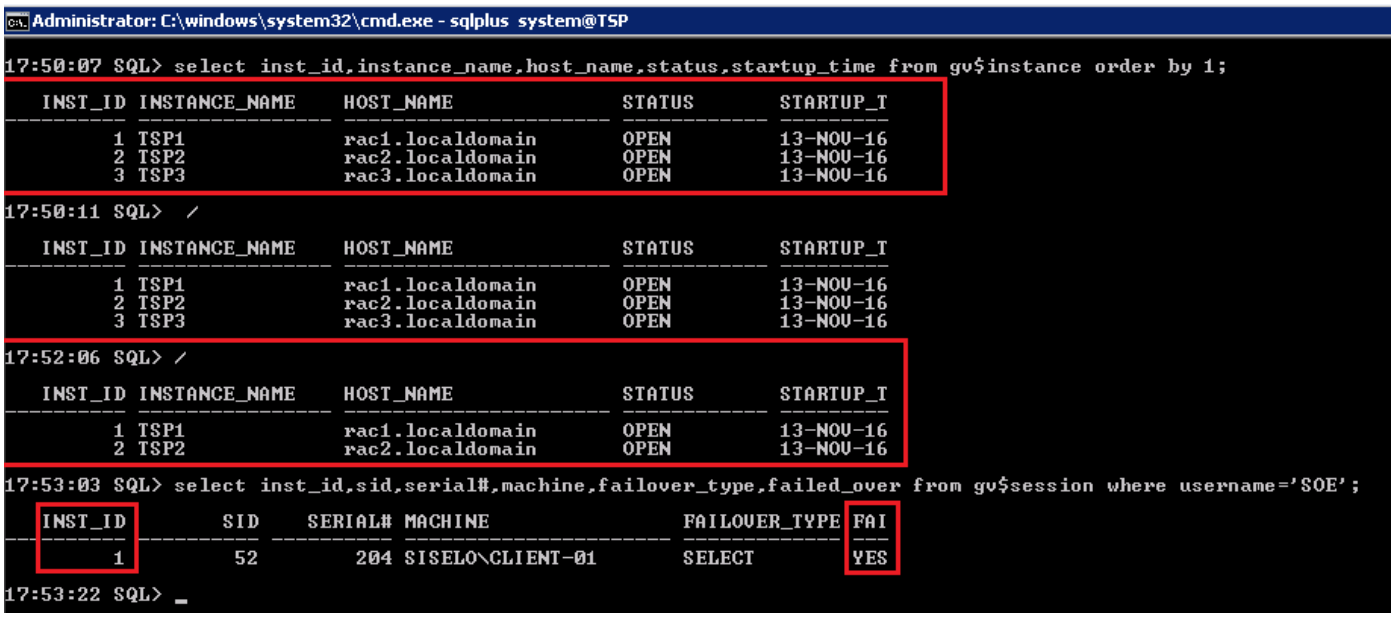


Fonte: Próprio autor.

Com o comando de *select* executando no nó 3 do *RAC*, o servidor foi desligado pelo virtual box, e as 17:52 o nó 3 ficou indisponível, neste momento o *select* do usuário “SOE” ficou congelado por alguns segundos, enquanto a sessão dele era encaminhada a outro nó ativo do *RAC*, os comandos abaixo mostram que o nó 3 sumiu da lista, pois não está mais disponível (a / no *sqlplus* repete o último comando que foi executado, no caso não é necessário digitar todo o *select* novamente, basta usar / e *enter* que o comando é executado novamente).

A segunda consulta executada é para verificar onde a sessão do usuário “SOE” foi parar, uma vez que o nó 3 foi desligado, onde o resultado indica que a sessão foi para o nó 1 e que ocorreu o *failover* durante uma operação de *select* de acordo com a figura 18.

Figura 18 – Failover da sessão do usuário “SOE”

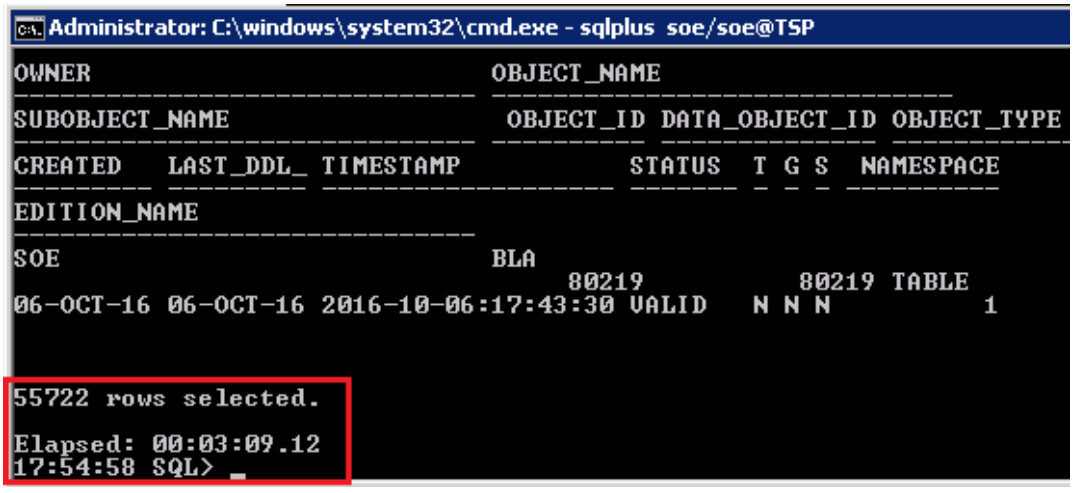


Fonte: Próprio autor.

Conforme ilustrado na figura 19, ao retornar a sessão do usuário “SOE”, mesmo após desligar o nó 3 e o *RAC* enviar a sessão ao nó 1, a consulta finalizou com sucesso.

Executando esta atividade, foi possível analisar e comprovar com mais clareza que tanto o balanceamento de cargas quanto a alta disponibilidade que a Oracle diz que o *Real Application Cluster* proporciona é verdadeiro e que todo o conceito se aplica na prática.

Figura 19 – Comando finalizado com sucesso após failover



Fonte: Próprio autor.

CONSIDERAÇÕES FINAIS

Considerando a importância da informação hoje em dia, fica difícil de mensurar quanto deve, ou quanto é possível gastar para protegê-la. Entretanto é até exagero se falar de proteção 100%, porque, por mais que um banco de dados, ou sistema esteja seguro, sempre haverá uma pequena brecha, e mesmo por menor que seja essa brecha, um dia alguém consegue passar. Não só em termos de computação, *hackerismo* bárbaro e ataques de vírus, mas também de pessoas que estão mal-intencionadas, a famosa 'engenharia social'. Ou seja, por mais que haja todos estas importantes camadas de proteção, é preciso tomar cuidado e se atualizar sempre. Porque o Oracle RAC pode deixar a empresa com um ambiente totalmente redundante e de alto desempenho, juntamente com o *database firewall*, *profiles* e *roles*, e com o *datavault* os dados ficaram tanto protegidos de ataque interno quanto externo, mas, mesmo assim seria besteira dizer que o ambiente está 100% protegido. Com uma infraestrutura deste porte, com certeza a empresa conseguirá manter seus dados seguros e disponíveis por um bom tempo. E hoje em dia como os usuários estão fazendo tudo pela *internet*, é importante manter um serviço *online*, e mais do que tudo assegurar que os dados dessas pessoas não sejam acessados (dados de cartão de crédito, endereço, nome completo, dados pessoais em geral, etc.), ainda mais que usuários só querem entrar nos sites e realizar suas transações normalmente sem se importar com os "bastidores". Uma empresa que ganha dinheiro com vendas *online*, então logo ela precisa ficar com seus dados 100% acessíveis 24 horas por dia, senão acaba perdendo vendas se o serviço ficar indisponível, pois o usuário quer comprar, mas não consegue então ele vai para outro *site*. Resumindo, é fundamental para que o negócio da empresa vá bem, ter uma alta redundância e disponibilidade e integridade em seus dados, e utilizando os recursos e dicas existentes neste trabalho, é possível tornar um sistema altamente disponível e seguro.

E para tal tarefa, foram apresentados os recursos durante todo o trabalho, o *firewall* para impedir que códigos maliciosos não sejam executados dentro de um banco de dado, o *datavault* que impede os dados de serem acessados por usuários que possuem um maior poder administrativo dentro do banco. O Oracle RAC, que além de aumentar a disponibilidade de um banco de dados ainda ajuda a aumentar o desempenho utilizando uma base de dados compartilhada para todos os servidores e

utilizando o hardware de cada servidor conforme a demanda, *dataguard* que diferentemente do RAC, ele não melhora o desempenho de um banco de dado e é uma solução mais simples, utilizando somente servidores em *cluster*. O RMAN que fornece os recursos de *backup* e recuperação do banco de dados, fazendo com que um possível plano de redundância seja criado. E por fim um recurso que já vem com o Oracle que são as *roles* e *profiles*, que bem configurados aumentam significantemente a segurança dentro de um banco de dado.

REFERÊNCIAS BIBLIOGRÁFICAS

ASHDOWN, Lance; KYTE, Tom. **Oracle database concepts, 11g release 2**. 2013. Disponível em: <http://docs.oracle.com/cd/E36909_01/server.1111/e25789.pdf>. Acesso em: 12 jun. 2016.

BEDNAR, Tammy. **Oracle white paper**, Oracle database firewall. Disponível em: <http://www.oracle.com/technetwork/database/focus-areas/security/ovw-oracle-database-firewall-1447166.pdf >. 2012. Acesso em: 22 abr. 2016.

CODD, Edgar F. **A Relational Model of Data for Large Shared Data Banks**. 1970. Disponível em: <https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf>. Acesso em: 25 jun. 2016.

FONSECA, Luiz Cláudio. **Trabalhando com Oracle database 10G**. 1. ed. Ciência Moderna, 2005, 272 p

FREEMAN, Robert G.; HART, Matthew. **Oracle RMAN 11G backup and recovery**. 1. ed. Estados Unidos, MC Graw Hill, 2010. 656 p.

HUEY, Patricia. **Oracle docs**: Oracle® Database Vault Administrator's Guide. 2014. Disponível em: <http://docs.oracle.com/cd/B28359_01/server.111/b31222.pdf>. Acesso em: 22 abr. 2016.

JELOKA, Sumit. **Oracle docs**: Oracle® database security guide 10*g* Release 2 (10.2). 2012. Disponível em: <http://docs.oracle.com/cd/B19306_01/network.102 /b14266.pdf>. Acesso em: 22 abr. 2016.

LUNDHILD, Barb; MICHALEWICZ, Markus**. Oracle White Paper**: Oracle Real Application Clusters (RAC) 11g Release2. 2010. Disponível em: <http://www.oracle. com/technetwork/database/clustering/overview/twp-rac11gr2-134105.pdf>. Acesso em: 12 mai. 2016.

MEEKS, Joe. **Oracle white paper**: Oracle data guard with Oracle database 11G. 2011. Disponível em: <http://www.oracle.com/technetwork/database/features /availability/twp-dataguard-11gr2-1-131981.pdf>. Acesso em: 22 abr. 2016

ORACLE. **Oracle Datasheet**: Oracle Real Application Clusters. 2010. Disponível em: <http://www.oracle.com/technetwork/database/clustering/overview/ds-rac-11gr2-134513.pdf.> Acesso em: 04 mai. 2016.

RICH, Kathy. **Oracle docs**: Oracle® Data Guard Concepts and Administration. 2014. Disponível em: < https://docs.oracle.com/cd/E11882_01/server.112/e41134.pdf>. Acesso em: 27 abr. 2016

SCARDINE, Paulo. **O que é escalabilidade**. Disponível em: < http://paulo.scardine. com.br/2010/07/o-que-e-escalabilidade.html>. 2010. Acesso em: 29 mai. 2016.