BANCO DE DADOS

ADMINISTRAÇÃO

ORACLE

Prof. Marcos Alexandruk

SUMÁRIO

INTRODUÇÃO

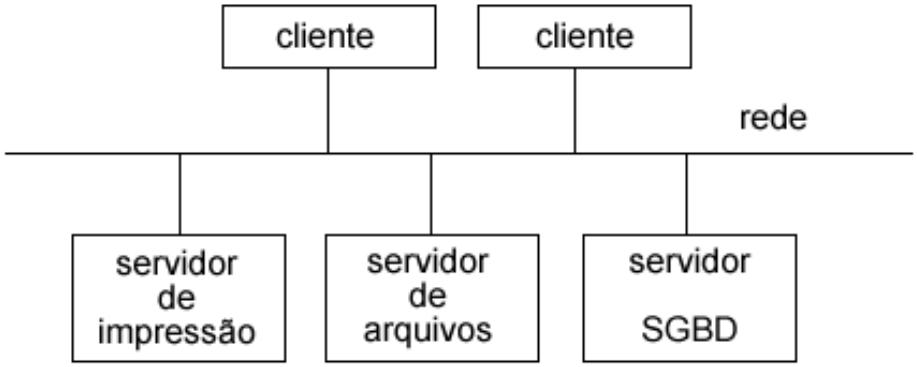
BANCO DE DADOS

- "Conjunto de programas que manipulam arquivos de dados." – Scott Martin (projetista da Oracle).

Arquitetura Cliente/Servidor (duas camadas)

A arquitetura cliente/servidor foi desenvolvida para trabalhar com ambientes computacionais, nos quais um grande número de PCs, estações de trabalho e servidores especializados (de arquivos, de impressão, de banco de dados, etc.) e outros equipamentos estão conectados via rede.

As máquinas clientes oferecem aos usuários as interfaces apropriadas para utilizar os servidores, bem como o poder de processamento para executar as aplicações locais.

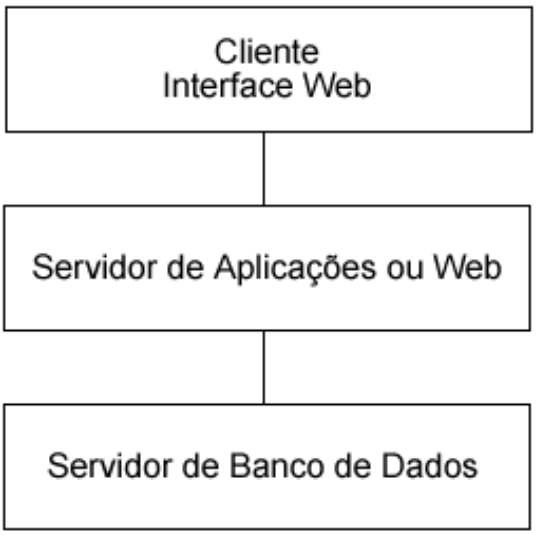


Arquitetura lógica de duas camadas cliente/servidor

Arquitetura Cliente/Servidor (três camadas)

Muitas aplicações para Web utilizam a arquitetura de três camadas, que possui uma camada intermediária entre o cliente e o servidor de banco de dados.

Essa camada intermediária, ou *camada de meio*, é, algumas vezes, chamada de servidor de aplicações ou servidor Web. Esse servidor desempenha um papel intermediário armazenando as regras de negócio (procedimentos ou restrições) que são usadas para acessar os dados do servidor de banco de dados.



Arquitetura lógica cliente/servidor de três camadas

DBA – Administrador de Banco de Dados

As principais funções do DBA são:

- Definir a estrutura de armazenamento e a estratégia de acesso;
- Servir de elo com os usuários;
- Definir controles de segurança e integridade;
- Definir a estratégia de backup e recuperação;
- Monitorar o desempenho.

ADMINISTRAÇÃO DE BANCO DE DADOS

A administração de banco de dados, seja ela manual, via SQL*Plus, ou visual, via OEM (Oracle Enterprise Manager), baseia-se em sua maioria, nas visões de dados, que podem ser de três tipos, diferenciando-se pelos prefixos:

- USER_... – objetos cujo proprietário *(owner)* é o usuário conectado.
- ALL_... – objetos de propriedade do usuário conectado, além daqueles aos quais o usuário tem privilégios para acesso.
- DBA_... – todos os objetos do banco de dados.

BANCO DE DADOS ORACLE

Um banco de dados é uma coleção de dados relacionados utilizada por uma ou mais aplicações informatizadas.

Fisicamente, um banco de dados Oracle é um conjunto de arquivos em algum lugar do disco. O local físico desses arquivos é irrelevante para as funções do banco de dados, mas não para seu funcionamento.

Logicamente, o banco de dados Oracle é dividido em um conjunto de contas de usuário conhecido como *schemas*. Cada schema está associado a um ID de usuário. Sem um nome de usuário, senha e privilégios válidos, não é possível acessar informações do banco de dados.

ORACLE 9i SERVER: OVERVIEW

O Oracle Server é composto de duas partes básicas:

- Banco de Dados (Database)
- Instância (Instance)

O Banco de Dados representa os arquivos físicos que armazenam os dados.

A Instância é composta pelas estruturas de memória[1] e pelos processo de segundo plano (background)[2].

Cada Banco de Dados pode estar associado a uma Instância. É possível que múltiplas Instâncias acessem um Banco de Dados, isto ocorre na configuração conhecida como RAC Real Application Cluster.

O Banco de Dados possui uma estrutura física e uma estrutura lógica.

As estruturas lógicas representam os componentes que você pode ver no Banco de Dados Oracle (tabelas, índices, etc.) e as estruturas físicas representam os métodos de armazenamento utilizado internamente pelo Oracle (os arquivos físicos). O Oracle mantém separadamente as estruturas lógicas e físicas. Por isso, as estruturas lógicas podem ser idênticas independentemente da plataforma de hardware ou do sistema operacional.

ESTRUTURAS LÓGICAS

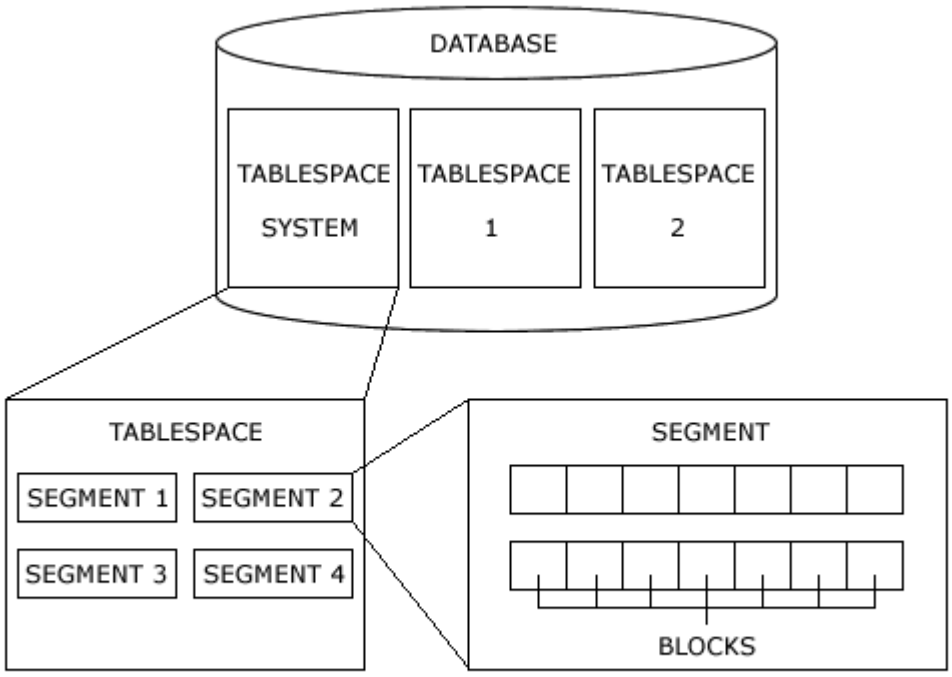
O Oracle divide o Banco de Dados em unidades menores para gerenciar, armazenar e recuperar os dados de forma eficiente. Os itens a seguir apresentam um overview das estruturas lógicas. Posteriormente, apresentaremos mais detalhes de cada uma delas.

- TABLESPACES: O Banco de Dados está logicamente dividido, no nível mais alto, em estruturas menores chamadas Tablespaces. O DBA pode utilizar as Tablespaces para organizar melhor o Banco de Dados (por aplicação, por função, por departamento, etc.). Esta divisão lógica ajuda a administrar uma parte do Banco de Dados sem afetar outras. Cada Banco de Dados pode ter uma ou mais Tablespaces. Quando você cria um novo Banco de Dados, o Oracle pelo menos uma Tablespace: SYSTEM.
- BLOCKS: O Bloco é a menor estrutura de armazenamento do Oracle. O tamanho de um Bloco é normalmente um múltiplo do tamanho de um Bloco do Sistema Operacional. Um bloco de dados corresponde a um número específico de bytes. Seu tamanho é baseado no parâmetro DB_BLOCK_SIZE e determinado quando o Banco de Dados é criado.
- EXTENTS: Formam o próximo nível da estrutura lógica. São formados por um agrupamento de blocos contíguos.
- SEGMENTS: Um Segmento é composto por um conjunto de Extents alocados para uma estrutura lógica: tabela, índice, etc. Quando uma dessas estruturas é criada, o Oracle aloca um segmento contendo pelo menos um Extent, que por sua vez deverá conter pelo menos um Bloco. Um Segmento pode estar associado a uma única Tablespace. A figura a seguir apresenta a relação entre Tablespaces, Segments, Extents e Blocks.

[1] As estruturas de memória são conhecidas como SGA (System Global Area) e PGA (Program Global Area).

[2] Os principais processos de segundo plano (background) são: Database Writer (DBW0), Log Writer (LGWR), System Monitor (SMON), Process Monitor (PMON), Checkpoint Process (CKPT).

Um Schema[3] é uma estrutura lógica utilizada para agrupar objetos em um Banco de Dados. Porém, um Schema não está relacionado diretamente a uma Tablespace ou qualquer outra estrutura lógica apresentada até o momento. Os objetos pertencentes a um Schema podem estar em diferentes Tablespaces, e uma Tablespace pode "conter" objetos pertencentes a vários Schemas. Os objetos de um Schema podem ser: tabelas, índices, synonyms, procedures, triggers, etc.



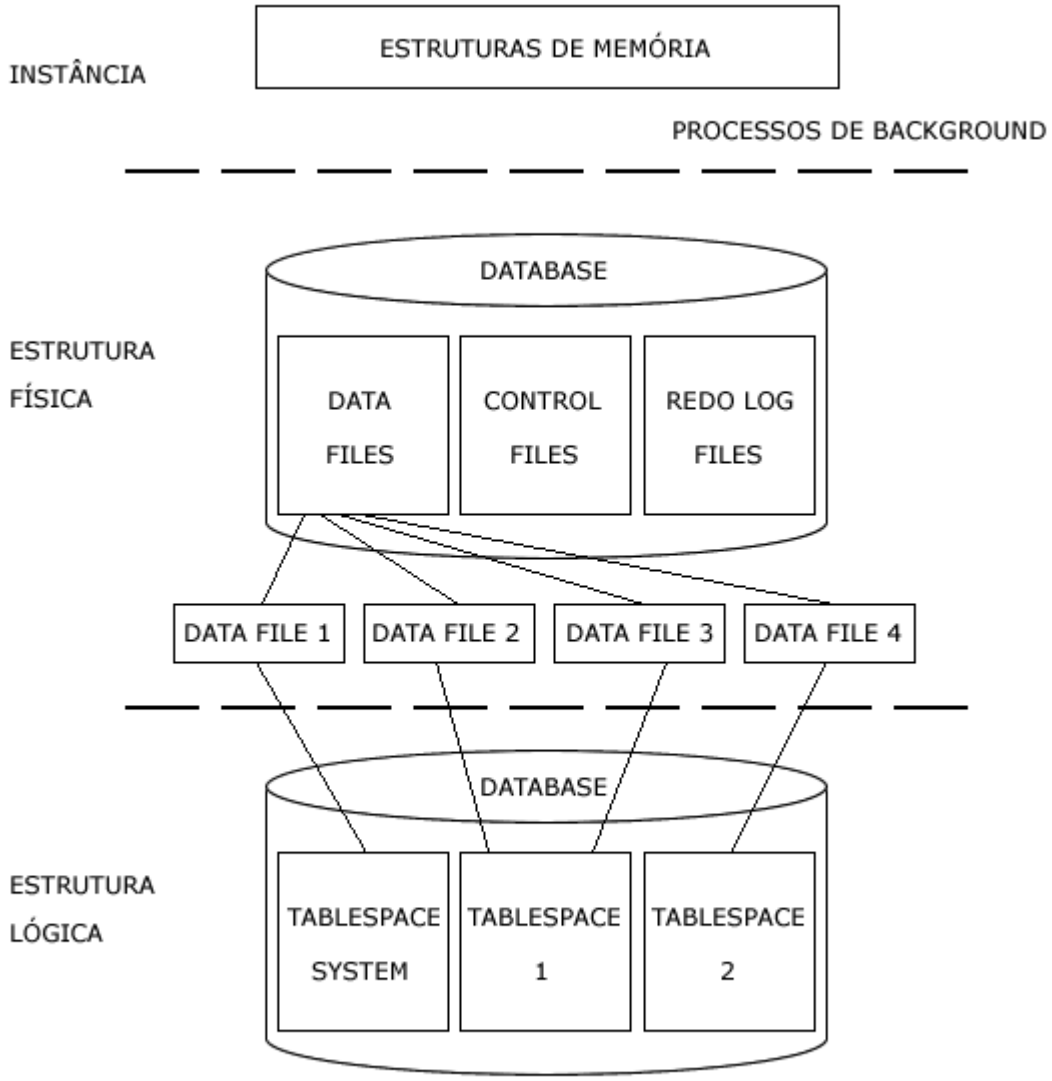
ESTRUTURA LÓGICA

[3] O Banco de Dados Oracle é dividido em um conjunto de contas de usuário conhecido como Schemas. Cada Schema está associado a um ID de usuário. Sem um nome de usuário, senha e privilégios válidos, não é possível acessar informações do Banco de Dados.

ESTRUTURAS FÍSICAS

A estrutura física de um Banco de Dados consiste em três tipos de arquivos:

- DATA FILES: Contêm todos os dados do Banco. Cada Banco de Dados é formado por um ou mais Data Files. Cada Data File está associado a uma única Tablespace. Uma Tablespace pode consitir de um ou mais Data Files.
- REDO LOG FILES: Gravam todas alterações nos dados do Banco. O Oracle possui dois ou mais desses arquivos, porque eles são gravados de forma cíclica. Através deles pode-se obter informações sobre os dados alterados. São fundamentais nas operações de recovery. É aconselhável manter cópias múltiplas destes arquivos de preferência em discos diferentes.
- CONTROL FILES: Cada Banco de Dados Oracle tem pelo menos um Control File. Eles mantêm informações sobre a estrutura física do Banco de Dados. O Oracle manter múltiplas cópias destes arquivos e recomenda-se esta prática. O Control File contém o nome do Banco de Dados e o timestamp de sua criação, bem como os nomes e localização de todos os Data Files e Redo Log Files.



ESTRUTURA FÍSICA

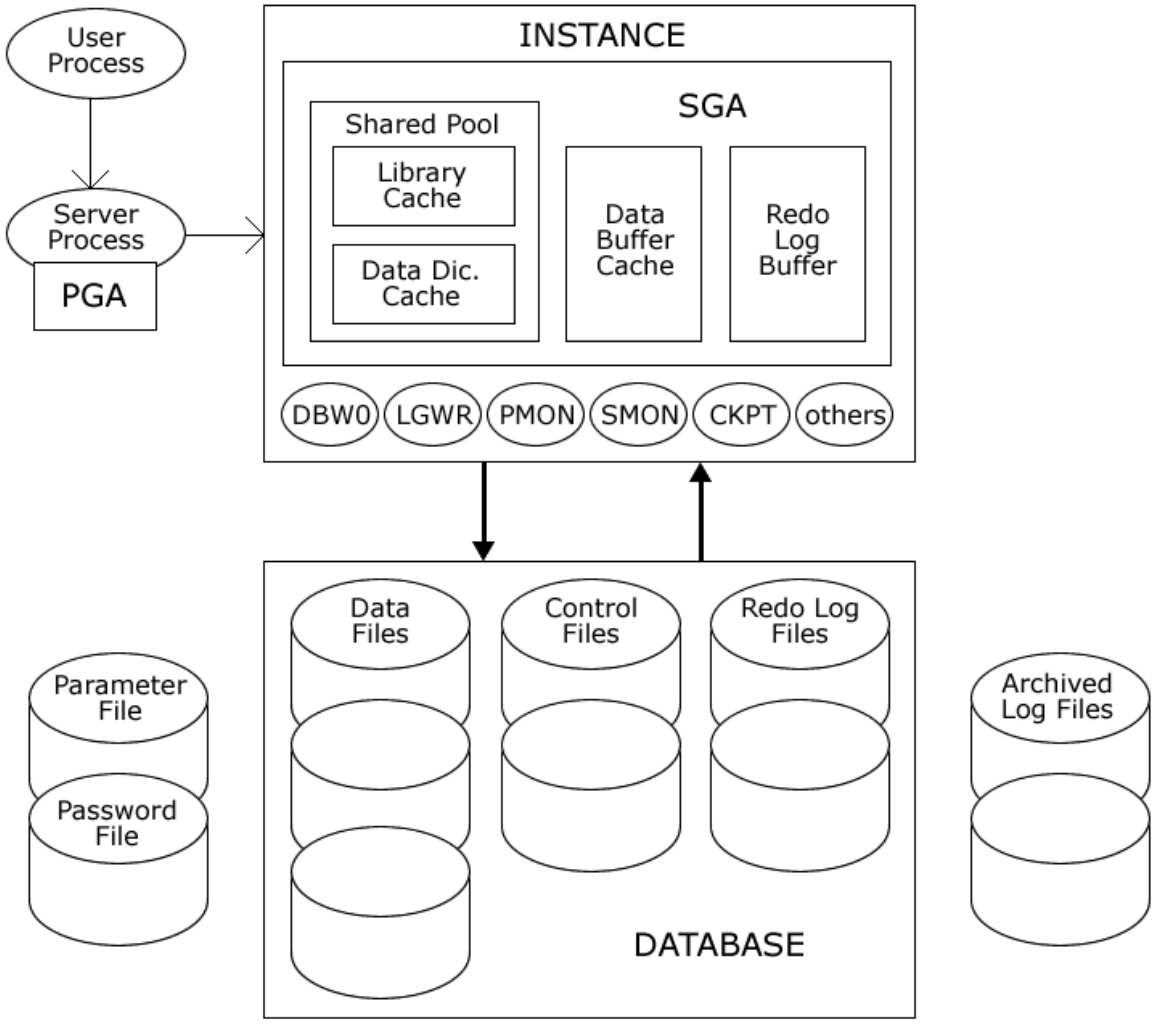
ARQUITETURA FÍSICA

A arquitetura de banco Oracle inclui estruturas lógicas e físicas que compõem o Banco de Dados.

O Servidor Oracle é um RDBMS (Relational Database Management System) ou SGBDR (Sistema de Gerenciamento de Banco de Dados Relacionais) que oferece uma abordagem aberta, abrangente e integrada ao gerenciamento de informações.

COMPONENTES PRINCIPAIS

Há vários processos, estruturas de memória e arquivos em um servidor Oracle. No entanto, nem todos esses itens são usados durante o processamento de uma instrução SQL. Alguns são utilizados para melhorar o desempenho do banco de dados, garantir que o mesmo possa ser recuperado caso ocorra um erro de software ou hardware, ou executar outras tarefas necessárias para a manutenção do banco de dados. O servidor Oracle consiste em uma INSTÂNCIA Oracle e em um BANCO DE DADOS Oracle.



ORACLE - ARQUITETURA FÍSICA

INSTÂNCIA

É a combinação dos processos de segundo plano e das estruturas de memória (SGA). A instância deve ser iniciada para acessar os dados do banco de dados. Toda vez que uma instância é iniciada, uma SGA (System Global Area – Área global do sistema) é alocada e os processos de segundo plano do Oracle são iniciados.

Informações importantes podem ser obtidas através da visão V$INSTANCE.

```
SELECT INSTANCE_NAME FROM V$INSTANCE;

INSTANCE_NAME
-------------
ORCL
```

Memórias

PGA (Program Global Area)

Região privativa de cada sessão na qual são armazenados dados temporários, tais como: variáveis ou endereços de rotinas.

Cada conexão provoca a criação de uma pequena área na PGA.

SGA (System Global Area)

SGA é uma área de memória usada para armazenar informações de bancos de dados que são compartilhadas pelos processos do banco de dados. Ela contém dados e controla informações para o servidor Oracle. Está alocada na memória virtual do computador no qual o servidor Oracle foi instalado. A SGA consiste em diversas estruturas de memória:

- Shared Pool: é usado para armazenar instruções SQL mais executadas recentemente (Library Cache / Shared SQL Area) e os dados do dicionário de dados (Dictionary Cache) mais usados. Essas instruções SQL podem ser submetidas por um processo de usuário, ou, no caso de procedimentos armazenados, lidas do dicionário de dados. A ocupação ocorre segundo um algoritmo LRU (Least Recently Used), isto é, os mais acessados ficam por mais tempo.
- Data Buffer Cache: é usado para armazenar em blocos (Data Block Buffers) os dados mais usados recentemente. Os dados são lidos e gravados nos arquivos de dados.

BLOCO: representa a menor unidade manipulável para o banco.

Cada bloco pode estar em quatro estados:

- LIVRE: Ainda não foi preenchido com algum dado proveniente de disco. No instante imediatamente posterior à abertura do banco, só há blocos livres.
- OCUPADO: Já foi preenchido. A ocupação também acontece segundo um algoritmo LRU (Least Recently Used).
- SUJO: Já foi preenchido e alterado, portanto deve ser gravado em disco brevemente.
- ROLLBACK: Possui um dado que poderá ser reaproveitado caso a transação que o alterou seja encerrada sem gravação.

- Redo Log Buffer: é usado para controlar as alterações efetuadas no banco de dados pelo servidor e pelos processos de segundo plano. Registra todas as transações 'comitadas'. Trata-se de uma lista circular, cujo conteúdo é gravado periodicamente nos Redo Log Files. As transações não 'comitadas' residem em áreas de rollback (memória ou disco).

A visão V$SGA revela diversas informações sobre a configuração da SGA:

```
SELECT * FROM V$SGA;

NAME                                     VALUE
-------------------------------- ------------
Fixed Size                              49152
Variable Size                        12906496
Database Buffers                      2048000
Redo Buffers                            73728
```

Processos de segundo plano (background)

Têm como finalidade principal integrar as estruturas de memória aos arquivos em disco.

Esses processos executam funções comuns que são necessárias para as solicitações de serviço de usuários simultâneos, sem comprometer a integridade e o desempenho do sistema. Eles consolidam funções que, de outra forma, seriam tratadas por diversos programas Oracle executados para cada usuário. Os processos de segundo plano executam tarefas de E/S e monitoram outros processos Oracle, para oferecer maior paralelismo, o que aumenta o desempenho e a confiabilidade.

PROCESSOS

Pequenos programas que executam tarefas específicas: integração entre as estruturas de memória e os arquivos em disco, conexão ao servidor, etc.

PROCESSO USUÁRIO

Cada vez que ocorre uma conexão, dispara-se um processo usuário executado na estação cliente. Eles têm a função de encaminhar ao servidor as requisições do cliente.

PROCESSO SERVIDOR

Recebe as requisições de processos usuários e as encaminha ao servidor. Podem ser dedicados (um para cada cliente) ou não. Neste caso, implementa-se o recurso MTS (Multi-threaded Server) que permite compartilhar os processos servidores com vários usuários.

Dependendo da configuração, uma instância Oracle pode incluir vários processos de segundo plano, no entanto cada instância inclui estes cinco processos de segundo plano fundamentais:

- O Database Writer (DBW0) grava os dados alterados do Data Buffer Cache nos Data Files.
- O Log Writer (LGWR) grava as alterações registradas no Redo Log Buffer nos Redo Log Files.
- O System Monitor (SMON) verifica a consistência no banco de dados e, se necessário, inicia a recuperação do banco de dados quando ele é aberto.
- O Process Monitor (PMON) disponibiliza recursos se um dos processos Oracle falhar. Por exemplo, limpa os processos de usuário que falharam e libera os recursos que o usuário estava usando.
- O Checkpoint Process (CKPT) atualiza as informações de status do banco de dados nos Control Files e nos Data Files, sempre que as alterações efetuadas no Data Buffer Cache ficam registradas no banco de dados de forma permanente.

  O último checkpoint pode ser verificado na visão: V$DATABASE.

```
SELECT NAME, CHECKPOINT_CHANGE# FROM DATABASE;

NAME CHECKPOINT_CHANGE
---- ------------------------------------------
ORCL                                    402987

ALTER SYSTEM CHECKPOINT;

System altered.

SELECT NAME, CHECKPOINT_CHANGE# FROM DATABASE;

NAME CHECKPOINT_CHANGE
---- ------------------------------------------
ORCL                                    402988
```

O processo seguinte, não é imprescindível, mas recomendável:

- ARCH (Archiver) copia o conteúdo de Redo Log Files para arquivos localizados em HDs ou fitas. Como a gravação nos Redo Log Files é cíclica, o processo impede que algum registro de transação se perca.

Para ver uma lista completa com os nomes de todos os processos de segundo plano disponíveis em seu banco de dados utilize a seguinte consulta:

```
SELECT NAME FROM V$BGPROCESS;
```

ARQUIVOS

Três conjuntos de arquivos armazenam os dados fisicamente e controlam as diversas funções do banco de dados Oracle:

- Data Files
- Control Files
- Redo Log Files

Esses arquivos devem estar presentes, abertos e disponíveis antes que quaisquer dados possam ser acessados.

Data Files

Armazenam os dados (tabelas), os índices e as áreas temporárias e de rollback.

Estão sempre ligados a um tablespace.

Informações sobre os data files são encontradas nas views V$DATAFILE (mais completa) e V$DBFILE (apenas dois campos).

```
SELECT * FROM V$DBFILE;
```

```
FILE# NAME
----- -----------------------------------
    2 C:\ORACLE\ORADATA\TESTE\USERS01.DBF
    3 C:\ORACLE\ORADATA\TESTE\INDX01.DBF
    4 C:\ORACLE\ORADATA\TESTE\TMP01.DBF
    1 C:\ORACLE\ORADATA\TESTE\SYSTEM01.DBF
```

Control Files

Armazenam a estrutura do banco de dados e seu sincronismo por meio do SCN (System Change Number). Sem eles não é possível inicializar o banco de dados.

Quando ocorre um checkpoint ou quando há alterações na estrutura do banco de dados, o arquivo de controle é atualizado

Informações sobre os control files são encontradas na view V$CONTROLFILE.

```
SELECT NAME FROM V$CONTROLFILE;
```

```
NAME
-----------------------------------------------
C:\ORACLE\ORADATA\TESTE\CONTROL01.CTL
C:\ORACLE\ORADATA\TESTE\CONTROL02.CTL
C:\ORACLE\ORADATA\TESTE\CONTROL03.CTL
```

Redo Log Files

Mantêm um histórico das transações efetuadas, permitindo refazer as operações no caso de perda de dados.

É exigido no mínimo dois destes arquivos, que são gravados de forma cíclica: conforme o espaço em um dos arquivos se esgota, procede-se à gravação no outro.

Se o banco de dados estiver no modo *Archiving*, quando um Redo Log File enche, antes de se passar para o próximo Redo Log File, é realizada uma cópia dele (.arc – Archive Log File). Caso contrário, o Redo Log File é sobrescrito.

Em ambiente de produção, o banco de dados sempre deve estar em Archiving Mode.

Informações sobre os redo log files são encontradas nas views V$LOG e V$LOGFILE.

```
SELECT * FROM V$LOGFILE;
GROUP# STATUS MEMBER
------ ------ --------------------------------
     2 STALE  C:\ORACLE\ORADATA\TESTE\REDO02.LOG
     1        C:\ORACLE\ORADATA\TESTE\REDO01.LOG
```

NOTA: O termo *stale* indica que o arquivo ainda não foi utilizado.
O Servidor Oracle utiliza outros arquivos que não fazem parte do banco de dados:

Parameter File

Define uma característica de uma instância Oracle como, por exemplo, parâmetros que dimensionam algumas estruturas de memória na SGA.

Password File

Autentica os usuários que tem permissão para inicializar e desativar uma instância Oracle.

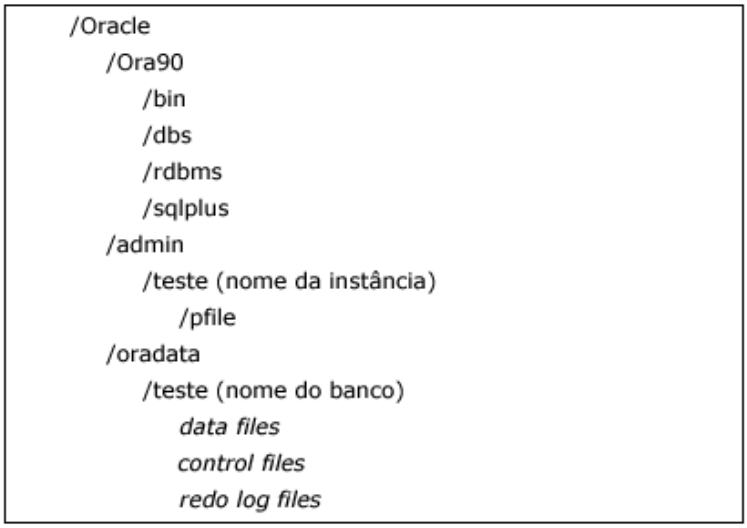
Archived Redo Log

São cópias off-line dos arquivos de redo log que podem ser necessários para a recuperação depois de falhas de mídia.

OFA (Optimal Flexible Architecture)

É composta por um conjunto de diretrizes que facilitam a manutenção do banco de dados, organizando os arquivos por tipo e uso.

Quando um banco é removido, os diretórios com o nome daquele SID são removidos de dentro dos diretórios admin e oradata.



Arquitetura OFA

Parâmetros

O funcionamento de uma instância é determinado por parâmetros que constam no arquivo INITsid.ORA. Os mais importantes são:

DB_BLOCK_SIZE	2.048 bytes
Tamanho do bloco, menor unidade manipulável. Para obter um valor ótimo, deve-se descobrir o tamanho do bloco do Sistema Operacional. O DB_BLOCK_SIZE deve ser igual ou múltiplo desse valor. No Windows NT, o tamanho é de 4096 (4KB). Para mudar esse parâmetro deve-se realizar a exportação completa e recriar o banco.	

DB_BLOCK_BUFFERS	1000
Tamanho do Database Buffer Cache: 1.000 X 2.048 = 2.000 KB. Por questões de performance, este valor deve representar, no máximo, 2% do espaço utilizado em disco pelo banco.	

SHARED_POOL_SIZE	11.534.336 bytes
Tamanho da Shared Pool Area.	

LOG_BUFFER	8.192 bytes
Tamanho do Redo Log Buffer.	

Etapas

Antes que uma instância esteja disponível e um banco de dados aberto, deve-se passar pelos seguintes estágios:

```
SQL*Plus: connect system/manager as sysdba
```

SHUTDOWN

```
shutdown
```

- Instância fechada
- Banco fechado

NOMOUNT

```
startup nomount
```

- Instância aberta
- Banco fechado

Ativam-se os processos background e aloca-se memória para a SGA. Após essa operação pode-se criar um Banco de Dados ou consertar Control Files.

MOUNT

```
alter database mount
```

- Instância aberta
- Banco parcialmente aberto

Lêem-se os Control Files e determina-se a localização dos demais arquivos. Recuperação e mudança da forma de arquivamento (processo ARCH) acontece após essa operação.

OPEN

```
alter database open
```

- Instância aberta
- Banco aberto

Abrem-se os Data Files e Redo Log Files. Usuários que não sejam DBA podem se conectar.

Estabelecendo uma conexão com um Banco de Dados

Três tipos de conexões distintas:

- O usuário estabelece logon no sistema operacional e inicia uma aplicação ou ferramenta que acesse o banco de dados neste sistema.
- O usuário inicia aplicação ou ferramenta em computador local e conecta-se através de uma rede a uma instancia de banco de dados. Esta é a configuração denominada Client/Server.
- Conexão de três camadas: o computador do usuário se comunica pela rede com uma aplicação ou servidor de rede. Exemplo: O usuário executa um browser (Internet Explorer, Opera, Firefox, etc.) para usar uma aplicação que reside em um servidor (Windows/IIS, Linux/Apache, etc.) que recupera dados de um banco de dados Oracle com host Unix em execução.

Processando uma consulta (SELECT)

As consultas (SELECT) são distintas dos outros comandos DML, pois retornam resultados (uma ou mais linhas), enquanto os outros comandos retornam apenas se houve êxito ou não. Em uma consulta existem três estágios principais:

1. Analisando uma instrução SQL

- Procura cópia da instrução SQL no Shared Pool.
- Valida a instrução SQL (sintaxe).
- Efetua pesquisas no dicionário de dados para validar tabelas e campos (semântica).
- Verifica os privilégios do usuário.
- Determina o plano de execução ideal para a instrução (índices).
- Armazena versão parseada (compilada) do comando.

Na fase de análise apenas são feitos verificações de erros que possam ocorrer antes da execução. Erros de conversão de dados ou informação duplicada de chave primária são tratadas na fase de execução.

2. Executando a instrução SQL

Neste ponto, o servidor Oracle possui os recursos necessários para a execução dos comandos, no caso de instrução SELECT, é preparado o processo de recuperação de dados. Identifica as linhas para extração.

- Leituras lógicas: blocos que já estão na memória (cache hit).
- Leituras físicas: blocos que não estão na memória. (cache miss)

3. Extraindo as linhas de uma consulta

Neste estágio as linhas são selecionadas e ordenadas se necessário e passadas pelo servidor ao usuário. Dependendo do número de linhas podem ser necessários um ou mais processo de extração.

Processando uma instrução DML (INSERT, UPDATE, DELETE)

Uma instrução DML requer apenas duas fases de processamento:

1. Analisando uma instrução DML (Igual ao processo consulta SQL)

- Procura cópia da instrução SQL no Shared Pool.
- Valida a instrução SQL (Sintaxe).
- Efetua pesquisas no dicionário de dados para validar tabelas e campos.
- Verifica os privilégios do usuário.
- Determina o plano de execução ideal para a instrução (índices).

2. Executando a instrução DML

- Se não houver blocos de rollback e dados no Data Buffer Cache, o processo de servidor fará sua leitura dos Data Files para o Cache de Buffer.
- Processo de servidor bloqueia as linhas que serão modificadas.
- No Redo Log Buffer, são registradas as alterações a serem feitas no rollback e dados.

- As alterações de bloco de rollback registram os valores dos dados antes de serem modificados, armazenam de forma que as instruções DML possam ser submetidas a rollback se necessário.
- As alterações dos blocos de dados registram os novos valores de dados.

- O Processo de servidor registra a imagem original do bloco de rollback e atualiza o bloco de dados. Essas duas alterações são efetuadas no cache de buffer do banco de dados. Qualquer bloco alterado no cache de buffer será marcado como buffer sujo, ou seja, os buffers que não são iguais aos blocos correspondentes no disco.

REVISÃO

1. Descreva a arquitetura de três camadas.
2. Quais as principais funções do DBA (Database Administrator)?
3. Tomando como base a arquitetura física, liste os componentes principais do Oracle.
4. O que é instância?
5. O que é PGA?
6. O que é SGA?
7. Quais os três conjuntos de arquivos? Descreva-os.
8. Quais são os processos de segundo plano (background) essenciais para o funcionamento do Oracle?
9. Qual a função do processo ARCH?
10. Fale sobre as diferentes formas do usuário conectar-se ao Oracle.

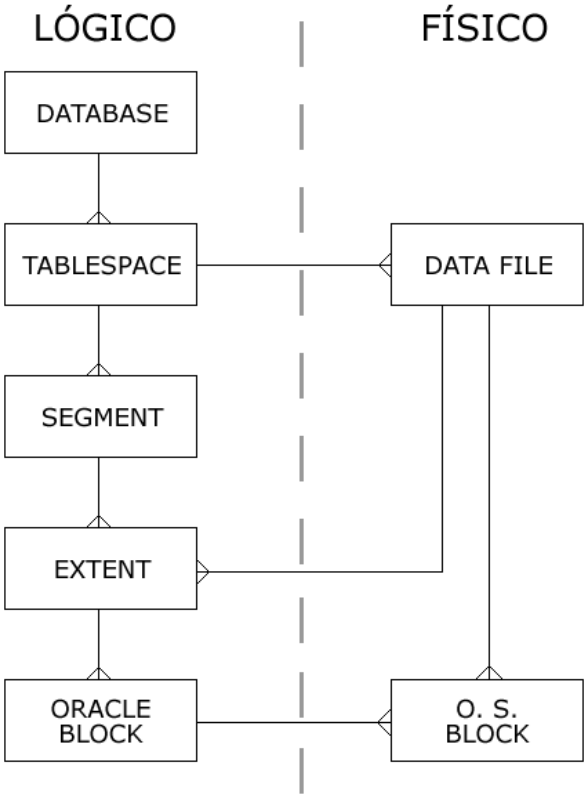
LABORATÓRIO

1. Informe o nome e o status da INSTÂNCIA.
2. Obtenha os seguintes parâmetros da SGA:
   Fixed Size, Variable Size, Database Buffers, Redo Buffers
3. Verifique quais processos de segundo plano (background) estão disponíveis.
4. Informe os nomes dos DATA FILES do Banco de Dados.
5. Informe os nomes dos CONTROL FILES do Banco de Dados.
6. Informe os nomes dos REDO LOG FILES do Banco de Dados.
7. Feche a INSTÂNCIA e o BANCO DE DADOS.
8. Abra somente a INSTÂNCIA e mantenha o BANCO DE DADOS fechado.
9. Monte o BANCO DE DADOS (sem abri-lo).
10. Abra o BANCO DE DADOS.

ARQUITURA LÓGICA

A estrutura lógica de um banco de dados Oracle inclui:

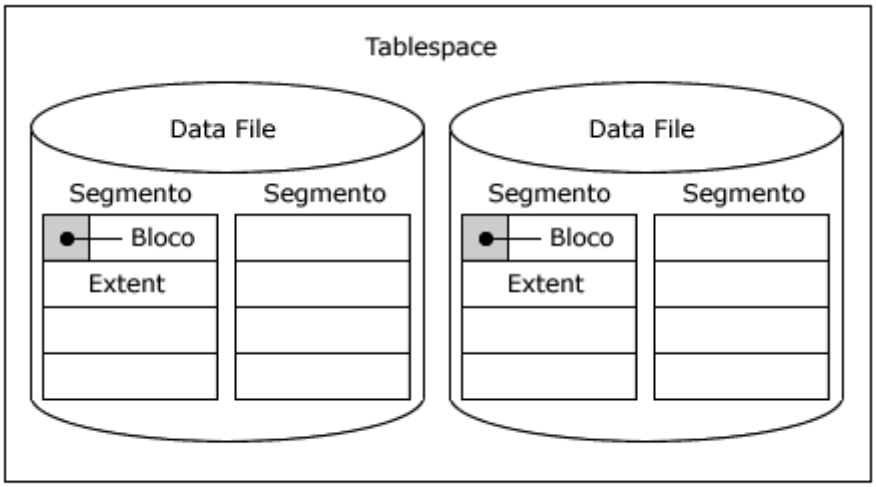
- tablespaces
- segmentos
- extensões
- blocos de dados



Arquitetura Lógica e Física do Banco de Dados

Os dados em um banco de dados Oracle são armazenados em TABLESPACES.

- Um TABLESPACE pode pertencer a somente um BANCO DE DADOS.
- Cada TABLESPACE consiste em um ou mais arquivos de dados: DATAFILES.
- Um TABLESPACE pode consistir em um ou mais SEGMENTOS.
- Os TABLESPACES podem ser colocados ON-LINE enquanto o banco de dados está em execução, e OFF-LINE para algumas atividades especiais.
- TABLESPACES podem ter o status READ-WRITE ou READ-ONLY.
- Existem tablespace de SYSTEM (Dicionário de Dados) e NON-SYSTEM (dados e aplicações dos usuários).
- DATAFILES são a implementação física dos TABLESPACES.
- Cada DATAFILE pertence a um único TABLESPACE.
- Os DATAFILES são formados por um conjunto de BLOCOS (menor unidade manipulável pelo Oracle).
- Os BLOCOS têm seu tamanho determinado na criação do BANCO DE DADOS e informado através do parâmetro DB_BLOCK_SIZE.
- Um BLOCO não pertence a mais de um EXTENT ou DATAFILE.



Estrutura lógica

O Oracle já vem com alguns TABLESPACES:

- SYSTEM
- UNDOTBS
- TEMP
- TOOLS
- USERS

Quando um objeto de banco de dados (tabela, índice, etc.) é criado, ele é designado a um TABLESPACE por meio dos defaults de usuário ou das instruções específicas. Um SEGMENTO é criado naquele TABLESPACE para conter os dados associados àquele objeto.

O espaço que é alocado ao SEGMENTO nunca é liberado antes de o SEGMENTO ser excluído, encolhido manualmente ou truncado.

Um SEGMENTO é composto por EXTENSÕES, que são conjuntos contíguos de BLOCOS do Oracle.

Depois que as EXTENSÕES existentes não podem mais conter novos dados, o SEGMENTO obterá outra EXTENSÃO para dar suporte às inserções adicionais de dados feitas no objeto.

O processo de extensão continuará continuamente até que não haja mais espaço disponível nos DATAFILES do TABLESPACE ou até que um número máximo interno de EXTENSÕES por SEGMENTO seja atingido.

Quando um DATAFILE é preenchido, ele pode ser estendido com base nas regras de armazenamento definidas para ele.

Tipos de segmentos disponíveis no Oracle:

- TABLE
- INDEX
- ROLLBACK
- TEMPORARY
- PARTITION
- CLUSTER

A quantidade de espaço usado por um SEGMENTO é determinada pelos PARÂMETROS DE ARMAZENAMENTO. Esses parâmetros são especificados quando um SEGMENTO é criado e podem ser alterado mais tarde.

Caso nenhum parâmetro específico de armazenamento seja dado no comando CREATE TABLE, CREATE INDEX, CREATE CLUSTER ou CREATE ROLLBACK SEGMENT, o banco de dados usará os parâmetros default de armazenamento para o TABLESPACE no qual ele deve ser armazenado.

Os valores default para os parâmetros de armazenamento de cada TABLESPACE podem ser consultados nas visões DBA_TABLESPACES:

```
SELECT * FROM DBA_TABLESPACES;
```

Quando um SEGMENTO é criado ele adquire pelo menos uma EXTENSÃO. A EXTENSÃO inicial armazenará os dados até não ter mais nenhum espaço livre.

> Você pode usar a cláusula PCTFREE para reservar dentro de cada BLOCO de cada EXTENSÃO uma PORCENTAGEM DE ESPAÇO que estará disponível para as atualizações (UPDATES) das linhas existentes do BLOCO.

Quando os dados adicionais são incluídos no SEGMENTO, este se estende obtendo uma segunda EXTENSÃO com o tamanho especificado no parâmetro NEXT.

O parâmetro PCTINCREASE foi criado para minimizar o número de EXTENSÕES das tabelas que podem crescer.

Um número diferente de zero para esse parâmetro fará com que o tamanho de cada EXTENSÃO sucessiva aumente geometricamente de acordo com o fator especificado.

Não há necessidade de usar um valor PCTINCREASE diferente de zero, a menos que o volume de dados da tabela seja diferente daquele que o seu projeto pedia.

> O PCTINCREASE não pode ser usado em TABLESPACES gerenciados LOCALMENTE.

O gerenciamento das EXTENSÕES podem ser feitos através de dois métodos:

- LOCALMENTE: nos TABLESPACES (a partir do Oracle 8i)
- Através do DICIONÁRIO DE DADOS

GERENCIAMENTO LOCAL

Nos TABLESPACES gerenciados localmente, o TABLESPACE gerencia seu próprio espaço mantendo um bitmap em cada DATAFILE dos BLOCOS livres e utilizados ou dos conjuntos de BLOCOS do DATAFILE.

Sempre que uma EXTENSÃO é alocada ou liberada para reutilização, o Oracle atualiza (UPDATE) do bitmap para mostrar o status novo.

GERENCIAMENTO ATRAVÉS DO DICIONÁRIO DE DADOS

Em um TABLESPACE gerenciado por dicionário sempre que uma EXTENSÃO é alocada ou liberada para reutilização em um TABLESPACE, a entrada apropriada é atualizada (UPDATED) na tabela do Dicionário de Dados.

CRIANDO UM TABLESPACE GERENCIADO LOCALMENTE

Para criar um TABLESPACE gerenciado localmente especifique a opção LOCAL na cláusula EXTENT MANAGEMENT:

```
CREATE TABLESPACE nome_do_tablespace
DATAFILE 'C:\Oracle\oradata\nome_do_datafile.dbf' SIZE 10M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 256K;
```

Se a cláusula UNIFORM SIZE for omitida, o default é AUTOALLOCATE.

O SIZE default para UNIFORM é 1MB.

Se você especificar LOCAL em um comando CREATE TABLESPACE, não pode especificar uma cláusula DEFAULT STORAGE, MINEXTENTS ou TEMPORARY.

Se você usar o comando CREATE TEMPORARY TABLESPACE para criar o TABLESPACE, pode especificar EXTENT MANAGEMENT LOCAL.

Os TABLESPACES gerenciados localmente podem assumir parte das tarefas de gerenciamento de espaço executadas pelos DBAs nos TABLESPACES gerenciados por dicionário. Por causa de sua arquitetura eles têm menos chances de se fragmentar e seus objetos têm menos chances de apresentar problemas relacionados a espaço.

Você não pode criar TABLESPACES SYSTEM gerenciados localmente no Oracle 8i ou 9i quando cria o banco de dados. Esses TABLESPACES também não podem ser convertidos posteriormente.

EXEMPLO:

```
CREATE TABLESPACE TBSP_PQ
DATAFILE 'C:\DATA_PQ.DBF' SIZE 10M
EXTENT MANAGEMENT LOCAL UNIFORM 1M;

CREATE TABLESPACE TBSP_MD
DATAFILE 'C:\DATA_MD.DBF' SIZE 100M
EXTENT MANAGEMENT LOCAL UNIFORM 4M;

CREATE TABLESPACE TBSP_GD
DATAFILE 'C:\DATA_GD.DBF' SIZE 1000M
EXTENT MANAGEMENT LOCAL UNIFORM 16M;
```

Se você tiver uma tabela pequena basta coloca-la no TABLESPACE TBSP_PQ. Se ficar muito grande, você pode move-la para o TABLESPACE TBSP_MD ou TBSP_GD.

CRIANDO UM TABLESPACE GERENCIADO ATRAVÉS DE DICIONÁRIO

SINTAXE:

```
CREATE TABLESPACE nome_do_tablespace
DATAFILE 'C:\Oracle\oradata\nome_do_datafile_1.dbf' SIZE int [K|M] [REUSE]
   [AUTOEXTEND OFF|ON
      NEXT    int [K|M]
      MAXSIZE int [K|M|UNLIMITED],
DATAFILE 'C:\Oracle\oradata\nome_do_datafile_2.dbf' SIZE int [K|M] [REUSE]
MINIMUM EXTENT tamanho [K|M]
[LOGGING|NOLOGGING]
DEFAULT STORAGE (
   INITIAL     int [K|M]
   NEXT        int [K|M]
   MINEXTENTS  int
   MAXEXTENTS  int
   PCTINCREASE int [0-100] )
[ONLINE|OFFLINE]
[PERMANENT|TEMPORARY];
```

DATAFILE

- Indica um ou mais arquivos que constituirão o tablespace.
- Deve-se especificar o path (caminho completo), o tamanho em KB (default) ou MB e a possibilidade de reaproveitar um arquivo existente (reuse).
- Raramente um tablespace ocupará menos que 1 MB.

AUTOEXTEND

- Um arquivo pode ser automaticamente expandido por um valor fixo até um limite ou indefinidamente. Esta cláusula não é recomendada.
- Se você não especificar esta cláusula, os arquivos (datafiles) não serão estendidos automaticamente.

LOGGING

- Informa-se a criação dos segmentos do tablespace, que serão registrados no Redo Log por default. A opção NOLOGGING realiza o contrário.

DEFAULT STORAGE

- Parâmetros de armazenamento:

INITIAL

- Tamanho do primeiro extent alocado (em bytes). Default: 10 KB.

NEXT

- Quantos bytes serão alocados no momento da expansão. Não há garantia de que sejam contíguos aos primeiros. Default: 10 KB.

MINEXTENTS

- Quantos extents deverão ser alocados na criação de um segmento. Default: 1.

MAXEXTENTS

- Quantidade máxima de extents alocados por segmento. Recomenda-se que não passe de 10. Default: 121.

PCTINCREASE

- Porcentagem de crescimento aplicada na expansão. Default: 50.

ONLINE | OFFLINE

- Determina se a tablespace estará disponível imediatamente após a criação ou não (OFFLINE).
- O tablespace SYSTEM precisa estar ONLINE para dar permissão aos usuários.
- É necessário estabelecê-lo OFFLINE quando se deseja realizar uma tarefa como backup ou como manutenção das tabelas.

PERMANENT | TEMPORARY

- Determina se os dados que constarão do tablespace serão permanentes ou temporários (neste caso, serão dados transitórios, resultados de agrupamentos ou seleções).

EXEMPLO:

```
CREATE TABLESPACE TESTE
  DATAFILE 'C:\ORACLE\DATA\TAB1.DBF' SIZE 1 M,
           'C:\ORACLE\DATA\TAB2.DBF' SIZE 1 M
  DEFAULT STORAGE (INITIAL 100 K
                   NEXT     50 K
                   MINEXTENTS 1
                   MAXEXTENTS 10
                   PCTINCREASE 100)
  EXTENT MANAGEMENT DICTIONARY;
```

NOTA: O diretório destino já deve existir.

extents	tamanho	expande*	novo
1	100	0	100
2	100	50	150
3	150	100	250
4	250	200	450
5	450	400	850

* PCTINCREASE aumenta em 100% o valor anterior.

Para comprovar a criação:

```
SELECT TABLESPACE_NAME FROM DBA_TABLESPACES;

TABLESPACE_NAME
----------------------------------------
SYSTEM
USER_DATA
ROLLBACK_DATA
TEMPORARY_DATA
TESTE
```

ALTERANDO UM TABLESPACE

Incluir mais um data file:

```
ALTER TABLESPACE TESTE
   ADD DATAFILE 'C:\ORACLE\DATA\TAB3.DBF' SIZE 1 M;

Tablespace altered.

SELECT FILE_NAME, BYTES, BLOCKS, STATUS
   FROM DBA_DATA_FILES
   WHERE TABLESPACE_NAME = 'TESTE';

FILE_NAME                BYTES   BLOCKS STATUS
------------------------ ------- ------ ---------
C:\ORACLE\DATA\TAB1.DBF  1048576   512  AVAILABLE
C:\ORACLE\DATA\TAB2.DBF  1048576   512  AVAILABLE
C:\ORACLE\DATA\TAB3.DBF  1048576   512  AVAILABLE
```

Mudar parâmetro de armazenamento:

```
ALTER TABLESPACE TESTE
   DEFAULT STORAGE (MAXEXTENTS 20 PCTINCREASE 50);

Tablespace altered.

SELECT TABLESPACE_NAME, MAX_EXTENDS, PCT_INCREASE
  FROM DBA_TABLESPACE
  WHERE TABLESPACE_NAME = 'TESTE';

TABLESPACE_NAME        MAX_EXTENDS   PCT_INCREASE
---------------------- ------------- ------------
TESTE                           20           50
```

ELIMINANDO UM TABLESPACE

```
DROP TABLESPACE name
  [INCLUDING CONTENTS [AND DATAFILES] | CASCADE CONSTRAINTS];
```

INCLUDING CONTENTS

- Realiza a exclusão mesmo se houver algum segmento. A operação fracassará, caso existam segmentos de rollback ou temporários ativos. A opção AND DATAFILES apaga os arquivos Data Files.

CASCADE CONSTRAINTS

- Elimina constraints relacionados a tabelas na tablespace corrente que estejam em outra tablespace.

```
DROP TABLESPACE TESTE;

Tablespace dropped.
```

```
SELECT TABLESPACE_NAME FROM DBA_TABLESPACE;

TABLESPACE_NAME
----------------------------------------
SYSTEM
USER_DATA
ROLLBACK_DATA
TEMPORARY_DATA
```

VERIFICANDO ESPAÇOS LIVRES NOS TABLESPACES

```
SELECT TABLESPACE_NAME, SUM(BYTES) FREE_SPACE
FROM DBA_FREE_SPACE
GROUP BY TABLESPACE_NAME;

TABLESPACE_NAME                FREE_SPACE
------------------------------ ----------
CWMLITE                          14680064
DRSYS                            12845056
EXAMPLE                            196608
INDX                             26148864
SYSTEM                           88276992
TOOLS                             4390912
UNDOTBS                         197853184
USERS                            26148864

8 linhas selecionadas.
```

DATA FILES

ALTERANDO AUTOEXTEND PARA 'OFF'

```
ALTER DATABASE
DATAFILE 'C:\ORACLE\DATA\TAB1.DBF'
AUTOEXTEND OFF;
```

ALTERANDO AUTOEXTEND PARA 'ON'

```
ALTER DATABASE
DATAFILE 'C:\ORACLE\DATA\TAB1.DBF'
AUTOEXTEND ON NEXT 100K MAXSIZE 2M;
```

ALTERANDO O TAMANHO DE UM DATA FILE

```
ALTER DATABASE
DATAFILE 'C:\ORACLE\DATA\TAB1.DBF'
RESIZE 2M;
```

REVISÃO

1. Tomando como base a arquitetura lógica, quais são os principais componentes do Oracle?
2. O que são TABLESPACES?
3. Um _____ pode pertencer a um único Banco de Dados.

   Cada TABLESPACE consiste em um ou mais DATAFILES.

   Um _____ é a menor unidade manipulável pelo Oracle.
4. Quais TABLESPACES são normalmente criados por ocasião da instalação do Oracle?
5. A quantidade de espaço usado por um segmento é determinado pelos _____.
6. O que acontecerá se nenhum parâmetro de armazenamento for informado num comando CREATE TABLE?
7. Que comando deve ser utilizado para obter-se os valores default para os parâmetros de armazenamento?
8. Quando um segmento é criado ele adquire pelo menos uma _____.
9. Que parâmetro quando diferente de zero fará com que o tamanho de uma extensão aumente geometricamente?
10. O gerenciamento das extensões pode ser feito através de dois métodos. Quais?

LABORATÓRIO

1. Obtenha as seguintes informações de cada TABLESPACE:
   - TABLESPACE_NAME
   - BLOCK_SIZE
   - INITIAL_EXTENT
   - NEXT_EXTENT
   - MIN_EXTENTS
   - MAX_EXTENTS
   - PCT_INCREASE
   - MIN_EXTLEN
   - STATUS [ONLINE|OFFLINE]
   - CONTENTS [PERMANENT|TEMPORARY]
   - LOGGING [LOGGING|NOLOGGING]
   - EXTENT_MAN [DICTIONARY|LOCAL]
2. Crie um TABLESPACE chamado TESTE com os seguintes parâmetros:
   - `DATAFILE: DATA1.DBF SIZE 20M`
   - `GERENCIAMENTO: LOCAL`
   - `EXTENTS: UNIFORM 512K`
3. Verifique se o TABLESPACE realmente foi criado.
4. O comando acima criou um arquivo com o nome DATA1.DBF?
5. Qual o tamanho do arquivo DATA1.DBF?
6. Adicione um DATA FILE chamado DATA2.DBF com tamanho igual a 10 MB ao TABLESPACE TESTE.
7. Obtenha as seguintes informações dos DATA FILES da TABLESPACE TESTE:
   - FILE_NAME
   - BYTES
   - BLOCKS
   - STATUS
8. Elimine o TABLESPACE TESTE.

9. O que aconteceu com o arquivo DATA1.DBF?
10. Elimine os arquivos DATA1.DBF e DATA2.DBF.
11. Crie um TABLESPACE chamado TESTE com os seguintes parâmetros:
    - DATAFILES: DATA1.DBF SIZE 2M e DATA2.DBF SIZE 2M
    - GERENCIAMENTO: DICIONÁRIO
    - DEFAULT STORAGE:
        - INITIAL: 100 K
        - NEXT: 50 K
        - MINEXTENTS: 1
        - MAXEXTENTS: 10
        - PCTINCREASE: 25
12. Verifique se o TABLESPACE realmente foi criado.
13. Altere os seguintes parâmetros de armazenamento default:
        - MAXEXTENTS: 20
        - PCTINCREASE: 50
14. Elimine o TABLESPACE TESTE.
15. Elimine os arquivos DATA1.DBF e DATA2.DBF.

SEGMENTOS

SEGMENTOS DE TABELA

São também chamados de SEGMENTOS DE DADOS, armazenam as linhas de dados associados a tabelas ou clusters.

Cada SEGMENTO contém um BLOCO de cabeçalho que serve como um diretório de espaço para o SEGMENTO.

Após adquirir uma EXTENSÃO, ele mantém a EXTENSÃO até o SEGMENTO ser excluído ou truncado.

NOTA: A exclusão das linhas de uma tabela por meio de um comando DELETE não tem impacto sobre a quantidade de espaço que foi alocada para aquela tabela.

O número de EXTENTS aumentará até que:

1. o valor de MAXEXTENTS seja atingido (se houve um definido)
2. a cota do usuário no TABLESPACE seja atingida
3. o TABLESPACE ficar sem espaço (se os DATAFILES não puderem se auto-estender)

Deve-se ajustar o parâmetro PCTFREE para minimizar a quantidade de espaço desperdiçado em um SEGMENTO DE DADOS.

O parâmetro PCTFREE especifica a quantidade de espaço que será mantida livre dentro de cada BLOCO. (O espaço livre poderá ser utilizado quando colunas com valores NULL forem atualizadas para conter valores ou quando atualizações para outros valores de linhas forçarem a linha a se estender.) A configuração adequada de PCTFREE é específica do aplicativo, uma vez que ela depende da natureza das atualizações que estão sendo executadas.

SEGMENTOS DE ÍNDICE

Para minimizar a disputa, os índices devem ser armazenados em um TABLESPACE separado de suas tabelas associadas.

Eles podem ser excluídos indiretamente se a tabela ou o cluster que eles indexam for excluído.

Os segmentos dos índices dos TABLESPACES gerenciados por DICIONÁRIO DE DADOS têm cláusula STORAGE que especificam os valores INITIAL, NEXT, MINEXTENTS, MAXEXTENTS e PCTINCREASE.

A opção REBUILD do comando ALTER INDEX pode ser utilizada para alterar as configurações STORAGE e TABLESPACE de um índice:

```
ALTER INDEX nome_do_indice REBUILD
  TABLESPACE nome_do_tablespace
  STORAGE (INITIAL 1M NEXT 1M PCTINCREASE 0);
```

NOTA: Durante o processo REBUILD, os índices antigo e novo existirão no banco de dados. Portanto, você deve ter espaço suficiente disponível para armazenar ambos os índices antes de executar o comando ALTER INDEX REBUILD.

SEGMENTOS DE ROLLBACK

Quando os usuários inserem, atualizam ou excluem os dados, o Oracle preserva as imagens dos dados anteriores à alteração.

Toda consulta que é iniciada antes do COMMIT da alteração verá a versão antiga dos dados, enquanto a sessão que faz a alteração verá a versão nova.

Para oferecer essa consistência de leitura, o Oracle oferece duas soluções:

- A primeira solução, os SEGMENTOS DE ROLLBACK, está disponível desde o Oracle 6.

- O Oracle 9i inclui o gerenciamento automático de UNDO como uma opção nova para a consistência de leitura dentro do banco de dados.

Pode-se utilizar os SEGMENTOS DE ROLLBACK ou os TABLESPACES de UNDO no Oracle 9i, mas não se pode usar ambos ao mesmo tempo.

Para selecionar um método de UNDO, utiliza-se o parâmetro UNDO_MANAGEMENT no arquivo de parâmetros INIT.ORA.

Se UNDO_MANAGEMENT estiver definido como MANUAL os SEGMENTOS DE ROLLBACK fornecerão o gerenciamento de UNDO.

Se UNDO_MANAGEMENT estiver definido como AUTO, um TABLESPACE de UNDO (também conhecido como UNDO GERENCIADO POR SISTEMA) fornecerá o gerenciamento de UNDO.

SEGMENTOS TEMPORÁRIOS

Armazenam dados temporários durante as operações de classificação, UNION, etc.

Cada usuário tem um TABLESPACE temporário especificado quando a conta é criada ou quando é alterada com ALTER USER.

No Oracle 9i, TABLESPACES permanentes não podem ser usados como temporários.

Quando um banco de dados é criado, pode-se especificar um TABLESPACE temporário default para todos os usuários. Em um banco de dados existente, pode-se alterar o TABLESPACE default com o comando ALTER DATABASE:

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE nome_da_tablespace;
```

Os usuários que foram indicados para um TABLESPACE default temporário antigo serão redirecionados para o novo TABLESPACE temporário default.

Para ver o TABLESPACE temporário atual execute a consulta:

```
SELECT PROPERTY_VALUE
  FROM DATABASE_PROPERTIES
  WHERE PROPERTY_NAME = 'DEFAULT_TEMP_TABLESPACE';

PROPERTY_VALUE
--------------
TEMP
```

Pode-se especificar um TABLESPACE como temporário com o comando CREATE TEMPORARY TABLESPACE. Um TABLESPACE temporário não pode ser utilizado para conter nenhum segmento permanente.

A primeira classificação a usar o TABLESPACE temporário aloca um SEGMENTO temporário dentro do TABLESPACE. Quando a consulta é concluída, o espaço usado pelo SEGMENTO temporário não é excluído. Em vez disso, o espaço usado pelo SEGMENTO temporário fica disponível para ser utilizado pelas outras consultas, permitindo assim que a operação de classificação evite os custos da alocação e liberação de espaço para os SEGMENTOS temporários.

TRANSAÇÕES

DEFINIÇÃO

Uma transação representa um conjunto de comandos cujo resultado será gravado de uma vez. Durante uma transação as linhas afetadas ficam presas (locked). Ela se inicia quando se executa o primeiro comando SQL e termina nos seguintes casos:

COMMIT	Grava efetivamente os comandos INSERT, UPDATE e DELETE.
ROLLBACK	Desconsidera os comandos INSERT, UPDATE e DELETE.
Fim da sessão	Ocorre um COMMIT implícito.
Comando DDL ou DCL	Provocam o fim da transação corrente: COMMIT implícito ou ROLLBACK (se houver qualquer problema durante a gravação definitiva).

Um comando SET TRANSACTION inicia uma transação com o Banco de Dados que pode ser **read only** ou **read write**. Ele somente pode ser usado após um COMMIT ou ROLLBACK.

SET TRANSACTION

```
SET TRANSACTION READ ONLY NAME 'tr1';
```

Cenário: Os usuários A, B e C estão conectados. O usuário A deseja emitir um longo relatório, envolvendo tabelas que estão sendo atualizadas por B e C. Neste caso, A deve iniciar sua transação com o comando acima para "fotografar" os dados no início da emissão do relatório e desconsiderar os COMMITs que B e C derem.

Em transações read write, somente um usuário pode alterar uma tabela a cada vez. Enquanto ele não encerrar sua transação, os demais usuários somente podem ver os dados como estavam antes da transação modificadora começar. Quando o comando COMMIT é executado, as alterações são gravadas, os valores antigos são perdidos e o lock desfeito.

COMMIT

Para efetivar as alterações:

```
INSERT INTO cliente (codigo,nome)
VALUES (1,'Antonio Alves');
COMMIT;

SELECT nome FROM cliente;

NOME
--------------------------
Antonio Alves

1 linha selecionada
```

ROLLBACK

Para descartar as alterações:

```
INSERT INTO cliente (codigo,nome)
VALUES (2,'Beatriz Bernardes');

SELECT nome FROM cliente;

NOME
--------------------------
Antonio Alves
Beatriz Bernardes

2 linhas selecionadas
```

```
ROLLBACK;

SELECT nome FROM cliente;

NOME
-------------------------
Antonio Alves
```

SAVEPOINT

Para inserir "marcas" a fim de possibilitar um rollback de apenas partes da transação:

```
INSERT INTO cliente (codigo,nome)
VALUES (2,'Beatriz Bernardes');
```

Estabelecendo um SAVEPOINT:

```
SAVEPOINT incluir_ok;
```

Incluindo outra linha:

```
INSERT INTO cliente (codigo,nome)
VALUES (3,'Claudio Carvalho');

SELECT nome FROM cliente;

NOME
-------------------------
Antonio Alves
Beatriz Bernardes
Claudio Carvalho
```

Voltando ao ponto marcado:

```
ROLLBACK TO incluir_ok;

SELECT nome FROM cliente;

NOME
-------------------------
Antonio Alves
Beatriz Bernardes
```

Salvando as operações realizadas:

```
COMMIT;
```

Terminada a transação, a marca é eliminada. Criando uma marca com um nome já utilizado, simplesmente ocorre uma substituição.

LABORATÓRIO

1. Crie uma tabela conforme a estrutura abaixo:

   Nome da Tabela: ALUNO

```
RA      NUMBER      (9)      PRIMARY KEY
NOME    VARCHAR2    (40)
```

2. Insira a seguinte linha:

```
111222333,'Daniela Damasceno'
```

3. Utilize o comando COMMIT

4. Observe o resultado com o comando SELECT

5. Insira a seguinte linha:

```
222333444,'Ernesto Elizeu'
```

6. Observe o resultado com o comando SELECT

7. Utilize o comando ROLLBACK

8. Observe o resultado com o comando SELECT

9. Insira (novamente) a seguinte linha:

```
222333444,'Ernesto Elizeu'
```

10. Utilize o comando SAVEPOINT marca_ok;

11. Insira a seguinte linha:

```
333444555,'Fabio Fernandes'
```

12. Utilize o comando ROLLBACK TO marca_ok;

13. Observe o resultado com o comando SELECT

11. Insira (novamente) a seguinte linha:

```
333444555,'Fabio Fernandes'
```

14. Utilize qualquer comando DDL (CREATE, DROP) ou DCL (GRANT, REVOKE)

15. Utilize o comando ROLLBACK

16. Observe o resultado com o comando SELECT

SEGMENTOS DE ROLLBACK / UNDO

A versão 9i do Oracle introduziu a possibilidade de gerenciar os segmentos de Rollback automaticamente através do SMU (System Managed Undo).

Convencionou-se denominar Rollback Segment os segmentos criados sob gerência manual, presente até a versão 8i e ainda possível (embora não recomendável) na versão 9i. Os Undo Segments identificam os criados sob a gerência automática.

Essencialmente, ambos têm a mesma definição: uma área especial destinada a guardar conteúdos de blocos alterados por uma transação.

Quando uma transação (não READ ONLY) começa, é reservado uma entrada em um segmento de Rollback/Undo.

Uma transação não ocupa dois segmentos, porém um segmento pode conter várias entradas de transações.

Outros extents são alocados à medida que outros dados vão sendo alterados.

Na gerência manual (Rollback Segments), os seguintes parâmetros são especificados: INITIAL, NEXT, MINEXTENTS e MAXEXTENTS. (PCTINCREASE não é utilizado).

Um extent pode conter blocos de várias transações, entretanto cada bloco possui referência a uma única transação.

Quando uma transação não pode mais adquirir espaço dentro da extensão atual, o segmento de rollback olha para a próxima extensão para saber se pode continuar gravando a entrada do segmento de rollback nela.

Se a extensão atual for a última dentro do segmento de rollback, o banco de dados tentará estender a entrada para a primeira extensão.

Entretanto, a próxima extensão já pode estar em uso. Neste caso, o segmento de rollback será forçado a adquirir uma nova extensão.

SEGMENTO DE ROLLBACK			
extensão 1	extensão 2	extensão 3	extensão 4
		transação 1 extensão 1	transação 1 extensão 2

A transação 1 se iniciou na extensão 3 e preencheu a extensão 4.

SEGMENTO DE ROLLBACK			
extensão 1	extensão 2	extensão 3	extensão 4
transação 1 extensão 3		transação 1 extensão 1	transação 1 extensão 2

Se a extensão 1 estiver disponível, a transação 1 a utiliza.

SEGMENTO DE ROLLBACK				
extensão 1	extensão 2	extensão 3	extensão 4	extensão 5
em uso		transação 1 extensão 1	transação 1 extensão 2	transação 1 extensão 3

Se a extensão 1 estiver em uso, o segmento de rollback se estenderá.

Quando uma transação termina, os extents que estavam ativos passam a estar inativos, isto é, são liberados para que outra transação possa utilizá-los.

O tamanho da área destinada a Segmentos de Rollback/Undo deve ser corretamente dimensionada, já que durante uma longa transação, se não for possível alocar uma extensão ocorrerá o seguinte erro:

```
ORA-1562: (unable to extend rollback segment)
```

Esta ocorrência provocará o rollback da transação em questão.

Para que um segmento de Rollback/Undo possa receber requisições de transações, ele deve estar ONLINE:

```
select segment_name, tablespace_name, status
from dba_rollback_segs;

SEGMENT_NAME  TABLESPACE_NAME  STATUS
------------- ---------------- -------
SYSTEM        SYSTEM           ONLINE
_SYSSMU1$     UNDOTBS          ONLINE
...
_SYSSMU10$    UNDOTBS          ONLINE
```

O comando acima foi disparado em um banco com SMU ativa.

Logo após a criação do Banco de Dados e sua primeira Tablespace, SYSTEM, cria-se um Segmento de Rollback também denominado SYSTEM. Caso nada se diga em contrário, será estabelecida a SMU e criada uma Tablespace UNDOTBS que armazenará Segmentos de Undo.

Estando a SMU ativa, comandos outrora utilizados para manipulação de Segmentos de Rollback ficam proibidos: CREATE, ALTER e DROP ROLLBACK SEGMENT, etc.

Os parâmetros abaixo indicam a presença da SMU:

```
SHOW PARAMETERS UNDO;

NAME                 TYPE      VALUE
-------------------- --------- --------
undo_management      string    AUTO
undo_retention       integer   900
undo_suppress_errors boolean   FALSE
undo_tablespace      string    UNDOTBS
```

NOTA: No exemplo acima undo_retention corresponde a 900 segundos.

Para alterar dinamicamente o parâmetro UNDO_RETENTION utilize o comando:

```
ALTER SYSTEM SET UNDO_RETENTION=1200;
```

FLASHBACK QUERIES

Estando a SMU ativa, sob determinadas circunstâncias, é possível recuperar o conteúdo de uma tabela cujas linhas tenham sido excluídas acidentalmente. Isto é possível através do novo recurso denominado Flashback Queries que aproveita dados coletados nos Extents ainda não expirados. Isto significa que as transações que os utilizaram foram concluídas a menos tempo do que informa o parâmetro undo_retention ou seus espaços ainda não foram ocupados.

A capacidade de recuperação restringe-se a cinco dias, sempre e quando Extents não tiverem sido reutilizados ou não tiver acontecido algum comando DDL sobre a tabela que se deseja recuperar.

Imagine que às 10:00 horas de uma manhã ensolarada, tenham sido eliminados todos os registros da tabela CLIENTES:

```
DELETE cliente;
COMMIT;
```

Agora, passados vinte minutos, descobrimos esta fatalidade.

"Voltaremos no tempo!" Como? Utilizando a rotina **enable_at_time**, presente no package **DBMS_FLASHBACK**:

```
execute dbms_flashback.enable_at_time (sysdate – 20/1440);
```

Este comando busca o estado das tabelas como estava há vinte minutos (um dia possui 1440 minutos).

Observe o resultado:

```
SELECT * FROM cliente;
```

IMPORTANTE: Restrições após habilitar uma janela no tempo:

Estão proibidos quaisquer comandos DDL, execute, update ou delete, ou seja, é possível apenas consultar as tabelas;

A única maneira de recuperar os dados consiste em criar um script que entre e saia do modo Flashback.

Crie uma tabela chamada AUX_CLIENTE para receber provisoriamente os dados e execute, a seguir, o bloco PL/SQL abaixo.

```
DECLARE
   CURSOR C_CLIENTE IS
      SELECT * FROM CLIENTE;
   V_LINHA CLIENTE%ROWTYPE;
BEGIN
   DELETE FROM AUX_CLIENTE;
   COMMIT;
   DBMS_FLASHBACK.ENABLE_AT_TIME(SYSDATE -20/1440);
   OPEN C_CLIENTE;
   DBMS_FLASHBACK.DISABLE;
   LOOP
      FETCH C_CLIENTE INTO V_LINHA;
      EXIT WHEN C_CLIENTE%NOTFOUND;
      INSERT INTO AUX_CLIENTE VALUES (V_LINHA.CODIGO, V_LINHA.NOME);
   END LOOP;
   CLOSE C_CLIENTE;
   COMMIT;
END;
```

A recuperação não precisa acontecer na mesma sessão responsável pela perda.

Concluída a execução dos comandos anteriores, é possível recarregar a tabela CLIENTE:

```
INSERT INTO cliente SELECT * FROM aux_cliente;
```

REVISÃO

1. Defina segmentos de rollback/undo.

2. O que acontece quando uma transação não pode mais adquirir espaço dentro da extensão atual?

3. O que acontece se, durante uma longa transação, não for possível alocar uma extensão?

4. Quando utilizamos o comando SHOW PARAMETERS UNDO; obtemos, entre outros, o seguinte parâmetro UNDO_RETENTION. O que ele informa? Seu valor é expresso em que unidade de tempo?

LABORATÓRIO

1. Consulte as seguintes informações sobre o segmento de UNDO de seu Banco de Dados:

1.1 Nomes dos segmentos
1.2 Nome do Tablespace
1.3 Status (online/offline)

2. Verifique se O SMU (System Managed Undo) está sendo utilizado.

3. Verifique o "tempo de vida" dos extents no SEGMENTO DE UNDO.

4. Altere dinamicamente o valar do parâmetro UNDO_RETENTION de um segmento para 36000.

5. Execute uma Flashback Query.

ÍNDICES

Estruturas especiais inseridas no Banco de Dados com o objetivo de melhorar o acesso às tabelas.

Índices são utilizados durante um SELECT com as cláusulas WHERE, ORDER BY e GROUP BY.

O Oracle cria automaticamente um índice do tipo UNIQUE ao criar uma PRIMARY KEY, o qual recebe o mesmo nome da CONSTRAINT.

TIPOS DE ÍNDICES

ÁRVORE B

Reduzem o I/O em disco utilizando uma estrutura de árvore-B (B*Tree) para localizar rapidamente os dados.

Ao varrer a árvore-B, ele identifica a chave e recupera o seu ROWID (ponteiro para acesso ao dado) localizando o registro rapidamente.

Os índices tipo árvore-B são mais utilizados para melhorar o desempenho dos comandos SELECT com colunas de valores exclusivos ou em sua grande parte distintos. É relativamente fácil para o Oracle manter este tipo de índice quando os dados são alterados em uma coluna indexada, tornando-o útil para aplicativos de transações on-line.

No entanto, esses índices não são muito recomendáveis para encontrar dados rapidamente nos comandos de SELECT em situações onde os valores na coluna indexada não são muito distintos.

Sintaxe geral:

```
CREATE [UNIQUE] INDEX nomeindice
   ON nometabela (nomecoluna [ASC|DESC] [,nomecoluna ...];
```

Quando criar índices compostos, coloque primeiro a coluna mais usada.

OPÇÕES	DESCRIÇÃO
UNIQUE	O índice não aceitará valores repetidos
[ASC|DESC]	Especifica a classificação ASC (ascendente), default, ou DESC (descendente)

Exemplo 1 (simples):

```
CREATE INDEX aluno_codturma_idx ON aluno (codturma);
```

Exemplo 2 (composto):

```
CREATE INDEX emp_dept_salario_idx ON emp (dept ASC, salario DESC);
```

ORDEM INVERTIDA

O índice de ordem invertida é o mesmo que o índice de árvore-B com uma exceção: os dados da coluna indexada são armazenados na ordem inversa. Assim, se um valor da coluna CODIGO é 1003, a coluna do índice de ordem invertida será 3001.

Os benefícios deste tipo de índice pode ser notado quando o comando SELECT contém termos WHERE que utilizam comparações de igualdade (SELECT * FROM itempedido WHERE codigo = 1003;), mas não em situações onde comparações de variação são utilizadas (SELECT * FROM itempedido WHERE codigo BETWEEN 1002 AND 1004;). Isto ocorre porque as linhas consecutivas são armazenadas longe umas das outras no índice.

Sintaxe geral:

```
CREATE INDEX nomeindice
   ON nometabela (nomecoluna) REVERSE;
```

Exemplo:

```
CREATE INDEX codprod_itempedido_idx ON itempedido (codprod) REVERSE;
```

ITEMPEDIDO		
CODPED	CODPROD	QTDPROD
1	1001	50
1	1002	40
1	1003	30
2	1004	40
2	1005	50
2	1006	30

ÍNDICE ITEMPEDIDO(CODPRODUTO)	
CODPROD	ROWID
1001	
2001	
3001	
4001	
5001	
6001	

BITMAP

O índice bitmap armazena uma entrada contendo cada valor distinto, os ROWIDs de início e fim (para indicar a variação de ROWIDs na tabela) e uma seqüência binária com a mesma quantidade de bits que a de linhas na tabela.

Índice BITMAP na tabela ALUNO (CAMPUS)			
Campus	ROWID início	ROWID fim	Nr bits = Nr linhas
Vila Maria	AAAAvQAAGAAAA3yAAA	AAAAvQAAGZZZZ3yZZZ	10000110
Memorial	AAAAvQAAGAAAA3yAAA	AAAAvQAAGZZZZ3yZZZ	01011000
Vergueiro	AAAAvQAAGAAAA3yAAA	AAAAvQAAGZZZZ3yZZZ	00100001

ALUNO			
RA	NOME	CURSO	CAMPUS
112233	Antonio Alves	TGSI	Vila Maria
223344	Beatriz Bernardes	TGRCI	Memorial
334455	Claudio Carvalho	TGSI	Vergueiro
445566	Daniela Damasceno	TGRCI	Memorial
556677	Ernesto Eliseu	TGRCI	Memorial
667788	Flavia Fernandes	TGRCI	Vila Maria
778899	Higino Hipolito	TGRCI	Vila Maria
889900	Isabel Inacio	TGRCI	Vergueiro

Os índices bitmap melhoram o desempenho em situações onde você seleciona dados de uma coluna cujos valores são muito repetidos e são pouco ou nunca alterados. Alterar o valor de uma coluna armazenada em um índice de bitmap requer que o Oracle trave o segmento inteiro armazenando o índice de bitmap para fazer a alteração.

Os índices de mapas de bits só estão disponíveis com a Enterprise Edition do Banco de Dados Oracle. - Oracle Referência para o BDA. p. 156

Sintaxe geral:

```
CREATE BITMAP INDEX nomeindice
   ON nometabela (nomecoluna);
```

Exemplo:

```
CREATE BITMAP INDEX aluno_campus_idx ON aluno (campus);
```

BASEADO EM FUNÇÃO

O índice baseado em função é desenvolvido para melhorar o desempenho de pesquisa, tornando possível definir um índice que funciona onde seu termo WHERE contém operações em colunas.

Por exemplo, imagine que você tem a tabela EMP com três colunas: ID, NOME e SALARIO. A coluna SALARIO tem um índice árvore-B nela. No entanto, se você emitir o comando:

```
SELECT * FROM emp WHERE (salario*1.05)>10000;
```

o Oracle irá ignorar o índice realizando um 'scaneamento' completo na tabela.

Os índices baseados em função são úteis nas situações em que os comando SQL contêm essas operações em seus termos WHERE.

Exemplo:

```
CREATE INDEX emp_func_idx ON emp (salario*1.05);
```

Para permitir o uso de índices baseados em função, você deve emitir os dois comandos ALTER SESSION a seguir:

```
ALTER SESSION SET QUERY_REWRITE_ENABLE = TRUE;

ALTER SESSION SET QUERY_REWRITE_INTEGRITY = TRUSTED;
```

OBTER INFORMAÇÕES

- Confirmar a existência do(s) índice(s):

```
SELECT INDEX_NAME FROM USER_INDEXES;
```

- Para obter outras informações:

```
SELECT INDEX_NAME, COLUMN_NAME, COLUMN_POSITION, COLUMN_LENGTH
   FROM USER_IND_COLUMNS
   WHERE TABLE_NAME='ALUNO';
```

MOSTRAR O USO DE ÍNDICES

```
ALTER INDEX aluno_codturma_idx MONITORING USAGE;

SELECT * FROM V$OBJECT_USAGE;
```

Nota: Se o índice acima for do usuário SCOTT, você deverá conectar-se ao Oracle como SCOTT. Se você conectar-se como SYS para pesquisar V$OBJECT_USAGE, não verá informações para o índice de SCOTT.

Situação 1 - O usuário faz a seguinte consulta:

```
SELECT * FROM aluno;
```

Ao realizarmos a consulta SELECT * FROM V$OBJECT_USAGE; o valor da coluna USED será NO, pois realizamos ou 'scaneamento' completo da tabela, não necessário utilizar o índice aluno_codturma_idx.

Situação 2 - O usuário faz a seguinte consulta:

```
SELECT * FROM aluno WHERE codturma=12;
```

O valor da coluna USED será YES, foi necessário utilizar o índice aluno_codturma_idx.

ELIMINAR UM ÍNDICE

```
DROP INDEX aluno_cod_turma_idx;
```

Nota: Este comando não elimina índices criados a partir de constraints.

TABELAS ORGANIZADAS POR ÍNDICE

Uma tabela organizada por índice (IOT) é um índice no qual uma linha inteira é armazenada, e não apenas os valores de chave da linha.

Em vez de armazenar um RowID para a linha, a chave primária é tratada como o identificador lógico da linha.

Deve-se especificar uma chave primária visto que os dados da tabela são classificados pela chave primária.

Exemplo:

```
CREATE TABLE t1 (
codigo NUMBER(4) PRIMARY KEY,
nome VARCHAR2(20))
ORGANIZATION INDEX;
```

EXERCÍCIOS

Criar as tabelas conforme script abaixo:

```
CREATE TABLE FUNC (
CODFUNC NUMBER(4),
NOMEFUNC VARCHAR2(40),
CPFFUNC VARCHAR2(12),
SALFUNC NUMBER(8,2),
CODDEPT NUMBER(4),
RAMALFUNC NUMBER(8),
CONSTRAINT FUNC_PK PRIMARY KEY(CODFUNC),
CONSTRAINT FUNC_CPFFUNC_UN UNIQUE(CPFFUNC));

CREATE TABLE PROJ (
CODPROJ NUMBER(4),
NOMEPROJ VARCHAR2(20),
INICIOPROJ DATE,
FINALPROJ DATE,
VALORPROJ NUMBER(12,2),
GERPROJ NUMBER(4),
CONSTRAINT PROJ_PJ PRIMARY KEY(CODPROJ),
CONSTRAINT PROJ_GERPROJ_FK FOREIGN KEY(GERPROJ) REFERENCES FUNC(CODFUNC));

CREATE TABLE PROJFUNC (
CODPROJ NUMBER(4),
CODFUNC NUMBER(4),
CONSTRAINT PROJFUNC_PK PRIMARY KEY(CODPROJ,CODFUNC),
CONSTRAINT PROJFUNC_PROJ_FK FOREIGN KEY(CODPROJ) REFERENCES PROJ(CODPROJ),
CONSTRAINT PROJFUNC_FUNC_FK FOREIGN KEY(CODFUNC) REFERENCES FUNC(CODFUNC));
```

CENÁRIOS

1. Observou-se que são realizadas consultas freqüentes na tabela FUNC utilizando como parâmetro o campo CPFFUNC. É necessário criar o seguinte índice:

CREATE INDEX CPFFUNC_IDX ON FUNC(CPFFUNC)? Por que?

2. O departamento de recursos humanos constantemente está emitindo relatórios conforme modelo a seguir com base na tabela FUNC:

```
NOMEFUNC                SALFUNC
----------------------- --------
ANTONIO ALVES           8500.00
CLAUDIO CARVALHO        5300.00
DANIELA DAMASCENO       4200.00
HIGINO HIPOLITO         3400.00
```

   Crie o índice adequado para este caso.

3. A gerência de operações faz várias consultas nas quais é necessário apresentar os dados ordenados pelo código do projeto (CODPROJ) a partir da data de início (INICIOPROJ) mais recente.
   O que você poderá fazer para melhorar o tempo de resposta a estas consultas?

4. A empresa tem cinco departamentos e um total de 650 funcionários. Não há previsão a curto e médio prazo de aumento no número de departamentos. Freqüentemente são emitidas listas com os nomes dos funcionários (NOMEFUNC) agrupados por departamento (CODDEPT).
   Crie um tipo de índice adequado a este caso.

5. Para atender as necessidades da recepcionista, foi criado um pequeno aplicativo capaz de consultar o ramal dos vários funcionários (RAMALFUNC). Para evitar o problema da divergência entre maiúsculas e minúsculas, as consultas são feitas utilizando a função UPPER.
   Exemplo: **`SELECT * FROM FUNC WHERE UPPER(NOMEFUNC) LIKE '%ANTONIO%';`**
   No entanto, a recepcionista tem repetido que a consulta "está muito lenta".
   Crie um índice para ajudar a 'pobre moça'.

6. O aplicativo também é capaz de informar nome do funcionário a partir do ramal interno. A empresa adotou uma solução de VoIP (Voz sobre IP) e para padronizar, os dois primeiros números referem-se ao país (Brasil = 55) os dois números seguintes 'apontam' a cidade (São Paulo = 11) e os quatro últimos são reservados para designar cada equipamento.
   Crie um índice adequado para consultas através do campo RAMALFUNC.

USUÁRIOS, PRIVILÉGIOS E PAPÉIS

USUÁRIO: pessoa previamente cadastrada que deseja acessar o Oracle.

PRIVILÉGIO: autorização para que o usuário acesse e manipule um objeto de banco de dados. Exemplo: um usuário pode ter o privilégio de selecionar tabelas, porém não modificá-las. Existem dois tipos de privilégios:

PRIVILÉGIOS DE SISTEMA: permissão de executar determinada ação em objetos de banco de dados. Existem mais de cem tipos de privilégios associados a ações de banco de dados. O nome do privilégio é praticamente o nome da ação que ele executa.

PRIVILÉGIO DE OBJETO: direito de executar uma determinada ação em um objeto específico, por exemplo, o direito de incluir uma linha em uma tabela. Eles não se aplicam a todos os objetos de banco de dados. Quando um usuário cria um objeto como uma tabela, esta só pode ser visualizada por ele. Para que outro usuário possa ter acesso a ela, é necessário que o proprietário conceda privilégios a ele ou um papel que irá acessar a tabela.

PAPEL: Um papel (role) é um grupo de privilégios associados a um nome que os identifica. Dessa forma, em vez de conceder vários privilégios a um único usuário, você pode criar um papel que recebe os privilégios e, em seguida, atribuí-lo ao usuário.

O Oracle 8i já tem preestabelecidos os seguintes usuários, senhas e papéis:

USUÁRIO	SENHA	PAPEL
SCOTT	TIGER	CONNECT e RESOURCE
SYSTEM	MANAGER	DBA
SYS	SYS	CONNECT, RESOURCE, DBA, EXP_FULL_DATABASE e IMP_FULL_DATABASE
DEMO	DEMO	CONNECT e RESOURCE

CRIAR NOVOS USUÁRIOS

1. Conectar-se como usuário system, senha manager:

```
CONNECT SYSTEM/MANAGER;
```

2. Criar o novo usuário:

```
CREATE USER aluno IDENTIFIED BY uninove;
```

Sintaxe completa:

```
CREATE USER nome_usuario
   IDENTIFIED [BY senha | externally]
   DEFAULT TABLESPACE nome_tablespace
   TEMPORARY TABLESPACE nome_tablespace_temporaria
   QUOTA numero_inteiro [K | M] ON nome_tablespace
   PROFILE nome_profile
   PASSWORD EXPIRE
   ACCOUNT [LOCK | UNLOCK]
```

PASSWORD EXPIRE: Expira imediatamente a senha, obrigando o usuário a especificar uma nova senha antes da primeira conexão.

3. Tentar conectar-se como novo usuário:

```
CONNECT aluno/uninove;
```

Ocorre um erro, pois o usuário não possui nenhum privilégio atribuído a ele.

DISTRIBUIR PRIVILÉGIOS

1. Conectar-se como usuário system, senha manager:

```
CONNECT SYSTEM/MANAGER;
```

2. Atribuir privilégio RESOURCE (criar tabelas, sequências, procedures, triggers, índices e clusters):

```
GRANT RESOURCE TO aluno;
```

3. Conceder privilégio connect ao usuário aluno:

```
GRANT CONNECT TO aluno;
```

4. Conectar usuário aluno ao Oracle:

```
CONNECT aluno/uninove;
```

5. Criar uma tabela:

```
CREATE TABLE teste
(codigo number(3));
```

CONCEDER UM PRIVILÉGIO DE OBJETO PARA UM USUÁRIO

Os privilégios concedidos podem ser:

PRIVILÉGIO	DESCRIÇÃO
SELECT	Seleciona dados de uma tabela ou de uma visão.
INSERT	Insere linhas em uma tabela ou em uma visão.
DELETE	Exclui linhas de uma tabela ou de uma visão.
UPDATE	Atualiza uma tabela (ou colunas especificadas).
INDEX	Cria ou exclui índices de uma tabela.
ALTER	Altera uma tabela.
ALL	Exerce todos os privilégios acima.

Objetos que podem conceder privilégios: *table, view, sequences, snapshots e synonym*.

1. Conceder privilégio para acessar a tabela cursos:

```
CONNECT SCOTT/TIGER;
GRANT SELECT ON cursos TO aluno;
```

2. Conectar usuário aluno e tentar acessar a tabela:

```
CONNECT ALUNO/UNINOVE;
SELECT * FROM cursos;
```

Ocorre um erro, pois o usuário deve fornecer o esquema* dessa tabela.

* Para cada usuário existe um esquema (schema). Um esquema é uma coleção de objetos: *tabelas, visões, sequências, sinônimos, índices, clusters, database links, snapshots, procedures, functions e packages.*

3. Acessar a tabela informando o esquema:

```
SELECT * FROM SCOTT.CURSOS;
```

VISUALIZAR TODOS USUÁRIOS

```
SELECT * FROM ALL_USERS;
```

CRIAR UM NOVO PAPEL (ROLE)

```
CREATE ROLE basico;
```

CONCEDER PRIVILÉGIOS E PAPÉIS A UM PAPEL

Um papel pode receber privilégios que podem ser específicos ou de outros papéis.

```
GRANT RESOURCE TO basico;
```

CONCEDER UM PAPEL A UM USUÁRIO

```
GRANT basico TO aluno;
```

VISUALIZAR OS PAPÉIS DO USUÁRIO ATUAL

```
SELECT * FROM USER_ROLE_PRIVS;
```

REVOGAR UM PRIVILÉGIO DE SISTEMA OU PAPEL CONCEDIDO

```
REVOKE basico FROM aluno;
```

REVOGAR UM PRIVILÉGIO DE OBJETO DE UM USUÁRIO

```
REVOKE SELECT ON cursos FROM aluno;
```

EXCLUIR UM PAPEL

```
DROP ROLE basico;
```

EXCLUIR UM USUÁRIO

```
DROP USER aluno <CASCADE>;
```

PROFILES

OBJETIVO: Controlar a utilização dos recursos disponíveis (tempo de CPU, tempo de conexão, etc.)

Alguns limites cujos valores podem ser atribuídos a um ou mais usuários:

SESSIONS_PER_USER	Sessões simultâneas. Algumas aplicações (exemplo: Oracle Forms) podem chamar outra, o que provocaria uma nova sessão.
CONNECT_TIME	Tempo de conexão em minutos. O usuário só percebe que seu limite foi esgotado ao tentar emitir um comando.
FAILED_LOGIN_ATTEMPTS	Quantas tentativas seguidas de login com insucesso causarão o bloqueio da conta.

Sintaxe:

```
CREATE PROFILE nome_profile LIMIT
   LIMITE_1 VALOR_1
   LIMITE_2 VALOR_2
   ...
```

Exemplo:

```
CREATE PROFILE teste LIMIT
   SESSIONS_PER_USER 1
   CONNECT_TIME 3;
```

Para comprovar a criação do profile:

```
SELECT RESOURCE_NAME, LIMIT
   FROM DBA_PROFILES
   WHERE PROFILE = 'teste':
```

Alterar um profile:

```
ALTER PROFILE teste LIMIT
   SESSIONS_PER_USER 2
   CONNECT_TIME 5;
```

Habilitar um profile:

```
ALTER SYSTEM SET RESOURCE_LIMIT=TRUE SCOPE=BOTH;
```

Associar um usuário a um profile:

```
ALTER USER nome_usuario PROFILE nome_profile;
```

AUTENTICAÇÃO VIA SISTEMA OPERACIONAL

1. Verificar os parâmetros de autenticação:

```
SHOW PARAMETERS AUTHENT
```

2. Alterar, se necessário, o parâmetro REMOTE_OS_AUTHENT para TRUE.

3. Criar um usuário que seja validado fora do Oracle (via sistema operacional):

```
CREATE USER OPS$ADMINISTRADOR IDENTIFIED EXTERNALLY;
```

4. Conectado como ADMINISTRADOR, realizar conexões com o Oracle:

```
CONNECT /
```

FUNÇÃO PARA VERIFICAR SENHA

A função a seguir verifica a utilização de senhas muito simples que poderiam comprometer a segurança do banco de dados:

```
CREATE OR REPLACE FUNCTION VERIFICA_SENHA
  (USERNAME      VARCHAR2,
   PASSWORD      VARCHAR2,
   OLD_PASSWORD  VARCHAR2)
   RETURN BOOLEAN IS
BEGIN
   IF PASSWORD = USERNAME THEN
      RAISE_APPLICATION_ERROR(-20001, 'SENHA IGUAL A USUÁRIO');
   END IF;
   IF LENGTH(PASSWORD) < 8 THEN
      RAISE_APPLICATION_ERROR(-20002, 'TAMANHO DA SENHA MENOR QUE OITO');
   END IF;
   IF NLS_LOWER(PASSWORD) IN ('oracle', 'banco','senha') THEN
      RAISE_APPLICATION_ERROR(-20002, 'SENHA MUITO SIMPLES');
   END IF;
   RETURN(TRUE);
END;
/
```

IMPORTANTE: A função VERIFICA_SENHA deverá ser criada abaixo do esquema **SYS**.

```
CREATE PROFILE nome_profile LIMIT
   PASSWORD_VERIFY_FUNCTION VERIFICA_SENHA;

ALTER SYSTEM SET RESOURCE_LIMIT=TRUE SCOPE=BOTH;

ALTER USER nome_usuario PROFILE nome_profile;
```

AUDITORIA

O Oracle tem a capacidade de fazer a auditoria das ações ocorridas no banco de dados.

Há três tipos de ações que podem ser auditadas:

- Tentativas de login
- Ações de banco de dados
- Acessos de objetos

A princípio são registrados os comandos bem sucedidos ou não, mas isso pode ser modificado quando cada tipo de auditoria é configurado.

Para ativar a auditoria, devemos ajustar o parâmetro estático AUDIT_TRAIL através do seguinte comando:

```
ALTER SYSTEM SET AUDIT_TRAIL=DB SCOPE=SPFILE;
```

Uma vez alterado o parâmetro, devemos reiniciar a Instância.

Para verificar as alterações utilize o comando:

```
SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME = 'audit_trail';
```

Os valores possíveis para AUDIT_TRAIL são os seguintes:

- DB Ativa a auditoria e grava na tabela SYS.AUD$
- OS Ativa a auditoria e grava no controle de auditoria do S.O. (sistema operacional).
- NONE Desativa a auditoria

A tabela SYS.AUD$ está associada ao tablespace SYSTEM e pode causar problemas de falta de espaço se os seus registros não forem periodicamente limpos. Por isso, os registros desta tabela devem ser arquivados e, em seguida, a tabela deve ser truncada:

```
TRUNCATE TABLE SYS.AUD$;
```

Devemos conceder ao usuário que fará as exclusões na tabela SYS.AUD$ o seguinte privilégio:

```
GRANT DELETE_CATALOG_ROLE TO nome_do_usuario;
```

AUDITORIA DE LOGIN

Objetivo: Auditar cada tentativa de conexão ao banco de dados.

Utilizar o seguinte comando para iniciar a auditoria das tentativas de conexão que resultem em sucesso ou não:

```
AUDIT SESSION;
```

Apenas para tentativas de conexão que resultem em sucesso:

```
AUDIT SESSION WHENEVER SUCCESSFUL;
```

Apenas para tentativas de conexão que resultem em falha:

```
AUDIT SESSION WHENEVER NOT SUCCESSFUL;
```

Os registros armazenados na tabela SYS.AUD$ podem ser visualizados com a visão de dicionário de dados DBA_AUDIT_SESSION:

```
SELECT OS_USERNAME, USERNAME, TERMINAL,
DECODE (RETURNCODE,'0','CONECTADO','1005','SUCESSO','1017','FALHA',RETURNCODE),
TO_CHAR(TIMESTAMP, 'DD-MM-YY HH24:MI:SS'),
TO_CHAR(LOGOFF_TIME, 'DD-MM-YY HH24:MI:SS')
FROM DBA_AUDIT_SESSION;
```

Para desativar a auditoria de sessão, use o comando NOAUDIT:

```
NOAUDIT SESSION;
```

AUDITORIA DE AÇÃO

Objetivo: Auditar qualquer ação que afete um objeto de banco de dados (tablespace, tabela, índice, etc.). As ações (create, alter e drop) que possam afetar esses objetos podem ser agrupadas durante a auditoria.

Alguns objetos que podem ser auditados: INDEX, PROCEDURE (PROCEDURE, FUNCTION, PACKAGE, LIBRARY), PROFILE, ROLE, SEQUENCE, SESSION, SYNONYM, TABLE, TABLESPACE, TRIGGER, USER, VIEW.

Para saber quais são as ações que podem ser auditadas utilize o seguinte comando:

```
SELECT NAME FROM AUDIT_ACTIONS;
```

Utilizar o seguinte comando para fazer a auditoria de todos os comandos que afetem as tabelas:

```
AUDIT TABLE;
```

Para desativar essa definição utilize o seguinte comando:

```
NOAUDIT TABLE;
```

Os registros podem ser visualizados com a visão DBA_AUDIT_OBJECT:

```
SELECT OS_USERNAME, USERNAME, TERMINAL,OWNER,OBJ_NAME,ACTION_NAME,
DECODE (RETURNCODE,'0','SUCESSO',RETURNCODE),
TO_CHAR(TIMESTAMP, 'DD-MM-YY HH24:MI:SS')
FROM DBA_AUDIT_OBJECT;
```

AUDITORIA DE OBJETO

Objetivo: Auditar ações de manipulação de dados (select, insert, update, delete) nos objetos.

Este tipo de auditoria pode ser feito por sessão ou por acesso:

BY SESSION

O registro de auditoria será gravado uma vez para cada sessão.

BY ACCESS

O registro de auditoria será gravado toda vez que o objeto for acessado. Usada de forma limitada para medir o número de ações separadas que ocorrem durante um intervalo de tempo específico. Visto que pode sobrecarregar os processos de gravação, quando a medição for concluída este tipo de auditoria deve ser revertido para BY SESSION.

Por exemplo, se um usuário executar três declarações UPDATE diferentes na mesma tabela, a auditoria BY ACCESS resultaria em três registros de auditoria (um para cada acesso de tabela). A auditoria do tipo BY SESSION resultaria na gravação de apenas um registro de auditoria.

Utilizar o seguinte comando para fazer a auditoria **por sessão** dos comandos UPDATE na tabela TESTE do usuário SCOTT:

```
AUDIT UPDATE ON SCOTT.TESTE BY SESSION;
```

Utilizar o seguinte comando para fazer a auditoria **por acesso** de todos os comandos INSERT na tabela TESTE do usuário SCOTT:

```
AUDIT INSERT ON SCOTT.TESTE BY ACCESS;
```

Os registros podem ser visualizados com a consulta da visão DBA_AUDIT_OBJECT:

```
SELECT OS_USERNAME, USERNAME, TERMINAL,OWNER,OBJ_NAME,ACTION_NAME,
DECODE (RETURNCODE,'0','SUCESSO',RETURNCODE),
TO_CHAR(TIMESTAMP, 'DD-MM-YY HH24:MI:SS')
FROM DBA_AUDIT_OBJECT;
```

IMPORTANTE: Para proteger o controle de auditoria em SYS.AUD$ utilize o seguinte comando:

```
AUDIT ALL ON SYS.AUD$ BY ACCESS;
```

BACKUP E RESTORE

O Oracle trabalha com o seguinte esquema de backup / recover:

Lógica	Exportação/Importação	O utilitário EXP pode copiar todos os objetos de um banco, de um determinado *schema*, *tablespace* ou tabela e armazena o resultado em um arquivo binário que pode ser lido pelo utilitário IMP.
Packages	DBMS_LOGMNR_D e DBMS_LOGMNR	Utiliza o recurso Log Miner. Por meio dele é possível investigar os comandos presentes nos Redo Log Files ou Archived Logs e até reverter seus efeitos.
	DBMS_FLASHBACK	A partir da leitura de segmentos de UNDO, permite obter uma fotografia dos dados em algum ponto do passado recente.
Tablespaces Transportáveis		Combinando com cópias físicas de arquivos, podem-se utilizar EXP e IMP para copiar *tablespaces* entre bancos.
Física	Cold	Estando o Banco e a Instância fechados, copia todos os arquivos, via S. O., para outro disco.
	Mista	O Banco deve estar fechado, os arquivos copiados fisicamente, porém, como o processo ARCH está ligado, a recuperação pode ser feita até um determinado instante.
	Hot	O Banco deve estar aberto e o processo Arch funcionando. Trabalha em nível de *tablespaces* que devem estar on-line. É mais indicado para bancos que nunca podem ficar indisponibilizados. A técnica é mais eficiente já que a SGA não precisa ser esvaziada.
Recovery Manager		Utiliza a ferramenta RMAN (Recovery Manager) que permite fazer cópias e recuperação até em nível de bloco. Como interage com o Arquivo de Controle da Instância, oferece inúmeras facilidades, como, por exemplo, informar há quantos dias na foi feito o backup de uma determinada *tablespace*.

EXP

EXPORTAÇÃO COMPLETA:

```
c:\>exp system/manager file=c:\temp\teste1.dmp full=y log=c:\temp\saida1.txt buffer=64000
```

EXPORTAÇÃO DE UM SCHEMA:

```
c:\>exp system/manager file=c:\temp\teste2.dmp owner=scott log=c:\temp\saida2.txt buffer=64000
compress=y
```

EXPORTAÇÃO DE DOIS SCHEMAS:

```
c:\>exp system/manager file=c:\temp\teste3.dmp owner=(scott,teste) log=c:\temp\saida3.txt
buffer=64000 compress=y
```

EXPORTAÇÃO DE UMA TABLESPACE:

```
c:\>exp system/manager file=c:\temp\teste4.dmp tablespaces=users log=c:\temp\saida4.txt
buffer=64000 compress=y
```

EXPORTAÇÃO DE UMA TABELA:

```
c:\>exp system/manager file=c:\temp\teste5.dmp tables=cliente log=c:\temp\saida5.txt
buffer=64000 compress=y
```

IMP

IMPORTAÇÃO COMPLETA:

```
c:\>imp system/manager file=c:\temp\teste1.dmp full=y log=c:\temp\saida1.txt buffer=64000
ignore=y commit=y
```

ignore: não é necessário que o banco esteja completamente vazio para fazer a importação

IMPORTAÇÃO DE UM SCHEMA:

```
c:\>imp system/manager file=c:\temp\teste2.dmp fromuser=scott touser=fulano
log=c:\temp\saida2.txt compress=y ignore=y commit=y
```

É necessário criar um usuário **fulano** que possua, no mínimo os papéis **connect** e **resource**.

IMPORTAÇÃO DE UMA TABELA:

```
c:\>imp system/manager file=c:\temp\teste5.dmp tables=cliente log=c:\temp\saida5.txt
buffer=64000 ignore=y commit=y
```