Administração de Banco de Dados

aula 1

Prof. Marcos Alexandruk

aula 1

Principais funções do DBA

Estruturas físicas do Oracle (overview)

Estruturas lógicas do Oracle (overview)

Principais funções do DBA

- **O DBA (Database Administrator) é responsável por criar e manter o banco de dados conforme os modelos conceitual e lógico definidos de acordo com as necessidades de cada organização.**

- **Principais funções do DBA:**

  - **Definir o esquema interno:** Elaborar o projeto físico, isto é, a partir do projeto lógico, definir como os dados serão representados no banco utilizando para isso a DDL (Data Definition Language).

  - **Zelar pela segurança e integridade:** Definir as restrições de segurança e integridade utilizando os mecanismos que restringem ou permitem acesso aos dados de acordo com "papéis“ (roles).

  - **Realizar o monitoramento e o ajuste de desempenho:** Monitorar o desempenho para certificar-se de que o banco de dados executa todas as funções importantes correta e rapidamente.

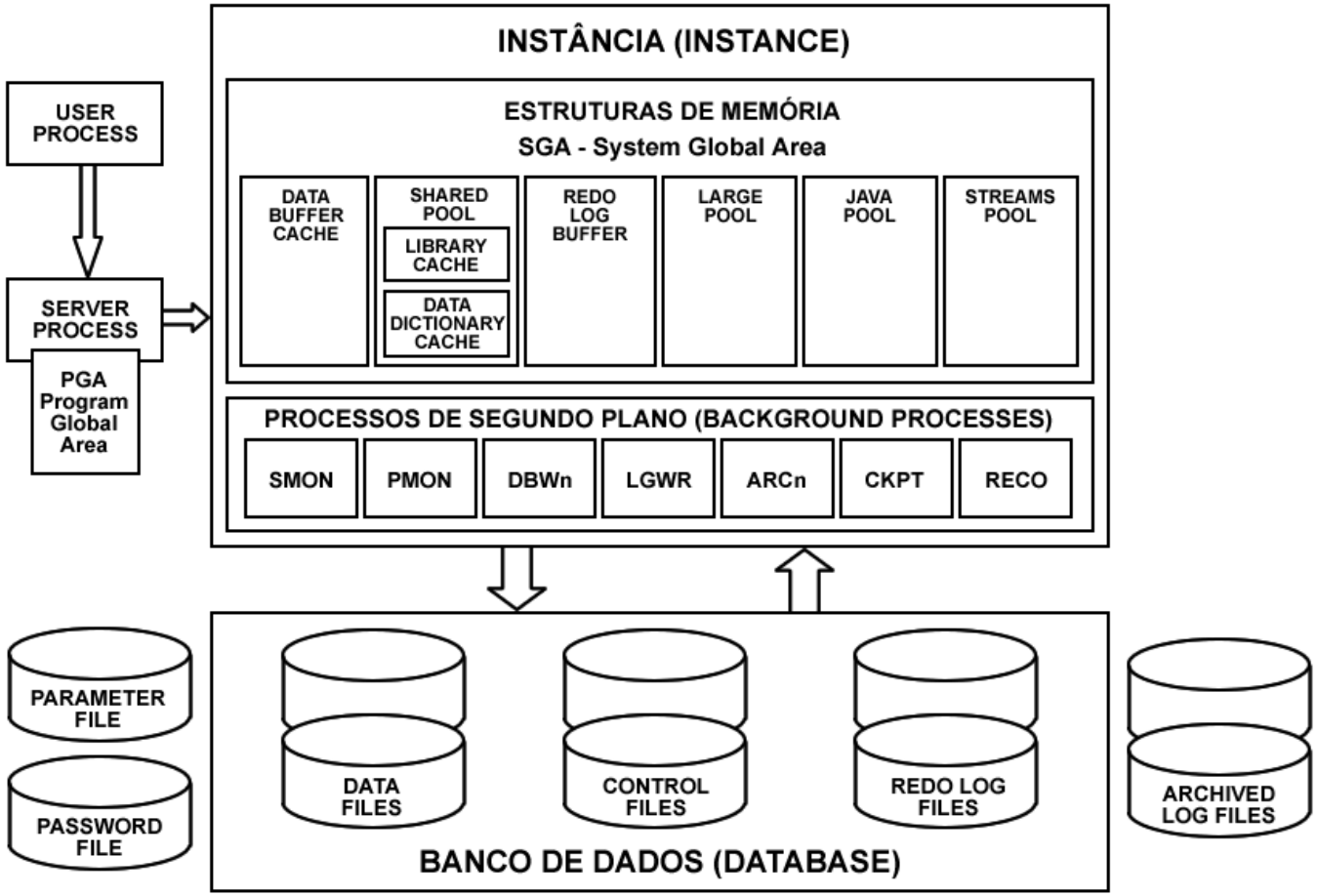
  - **Atender as demandas dos usuários e dos seus sistemas:** Garantir que os dados necessários estejam disponíveis, prestar consultoria em projetos de aplicações e, quando necessário, fornecer treinamento técnico.

Principais funções do DBA

- **Para que o DBA possa desempenhar satisfatoriamente suas funções é fundamental que conheça profundamente as estruturas físicas e as estruturas lógicas que compõem um SGBD (Sistema de Gerenciamento de Banco de Dados).**

Estruturas físicas

Estruturas físicas do Oracle Database 11g



Estruturas físicas

- Um servidor Oracle é composto basicamente pela **instância** e pelo **banco de dados**.

- A **instância** envolve uma área de memória compartilhada conhecida como **SGA (System Global Area)** e os **processos de segundo plano (background)**.

Estruturas físicas

- **Principais processos de segundo plano (background):**
  - **SMON (System Monitor)**
  - **PMON (Process Monitor)**
  - **DBWn (Databse Writer)**
  - **LGWR (Log Writer)**
  - **ARCn (Archiver Process)**
  - **CKPT (Checkpoint Process)**
  - **RECO (Recovery Process)**

Estruturas físicas

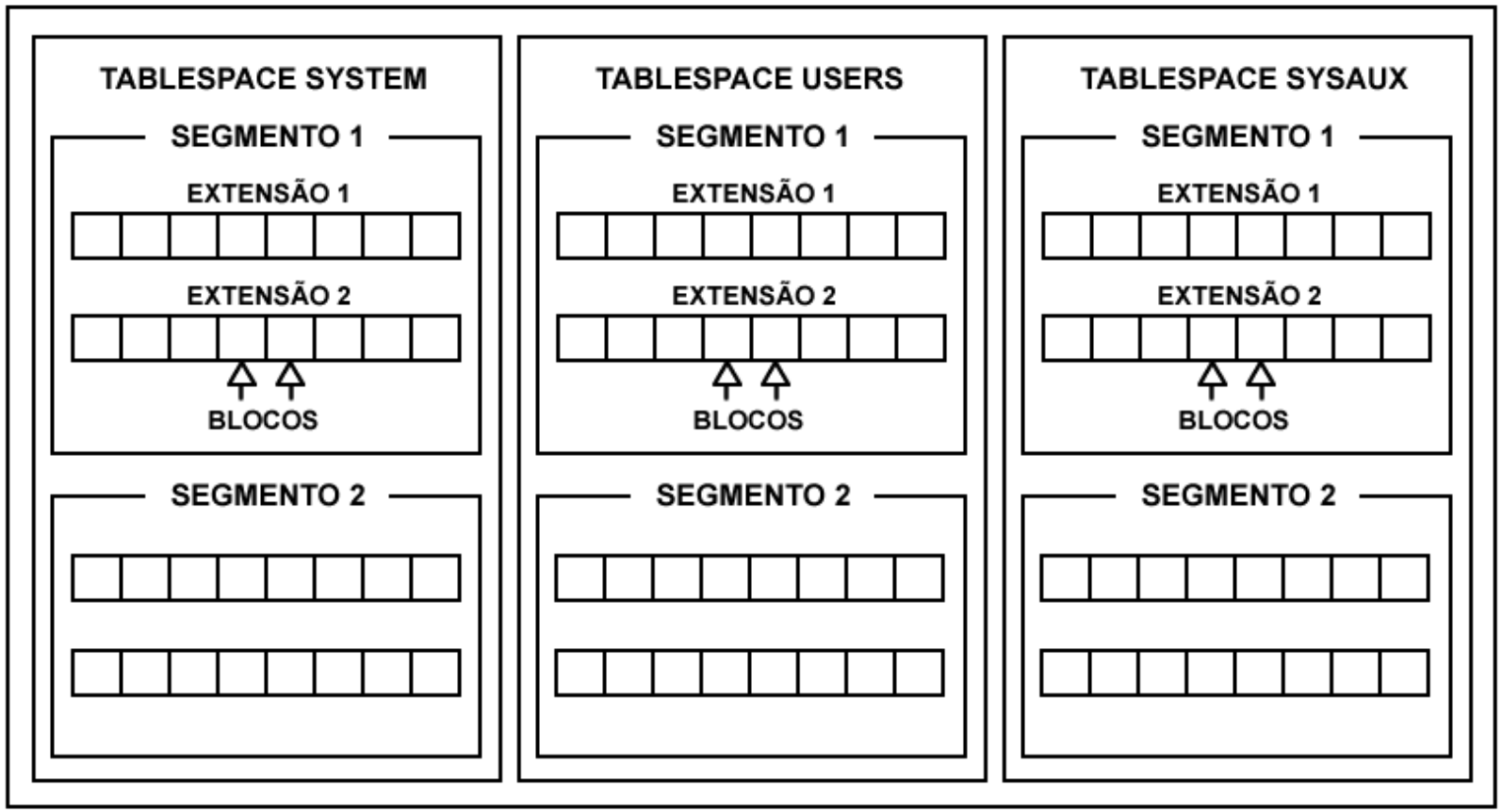
- **PGA (Program Global Area):**
  - **As sessões de usuários também precisam alocar memória no servidor.**
  - **Essas áreas de memória conhecidas como PGA (Program Global Area), diferentemente da SGA, não são compartilhadas.**

Estruturas físicas

- **Arquivos que compõem o banco de dados:**
  - **DATA FILES (arquivos de dados)**
  - **CONTROL FILES (arquivos de controle)**
  - **REDO LOG FILES (arquivos de log on-line)**
- **Arquivos externos ao banco de dados que também são necessários:**
  - **PARAMETER FILE (arquivos de parâmetro de instância)**
  - **PASSWORD FILE (arquivos de senhas)**
  - **ARCHIVED LOG FILE (arquivos de redo log arquivados)**
  - **TRACKING FILE (arquivos de alerta e rastreamento)**

Estruturas lógicas

Estruturas lógicas do Oracle Database 11g



Estruturas lógicas

- **Tablespaces:**
  - **Tablespaces são estruturas lógicas que consistem em um ou mais arquivos de dados.**
  - **É importante associar as tabelas no momento em que são criadas a seus respectivos tablespaces para melhor organização do banco de dados.**
  - **O Oracle cria durante a instalação pelo menos dois tablespaces:**
    - **SYSTEM**
    - **SYSAUX**
  - **Há um tipo especial de tablespace no Oracle 11g denominado BIGFILE no qual é possível armazenar até 128 TB (terabytes).**

Estruturas lógicas

- **Blocos:**
  - **Os blocos de dados são as menores estruturas de armazenamento no banco de dados Oracle.**
  - **Um bloco pode ser constituído de um ou mais blocos do sistema operacional.**

Estruturas lógicas

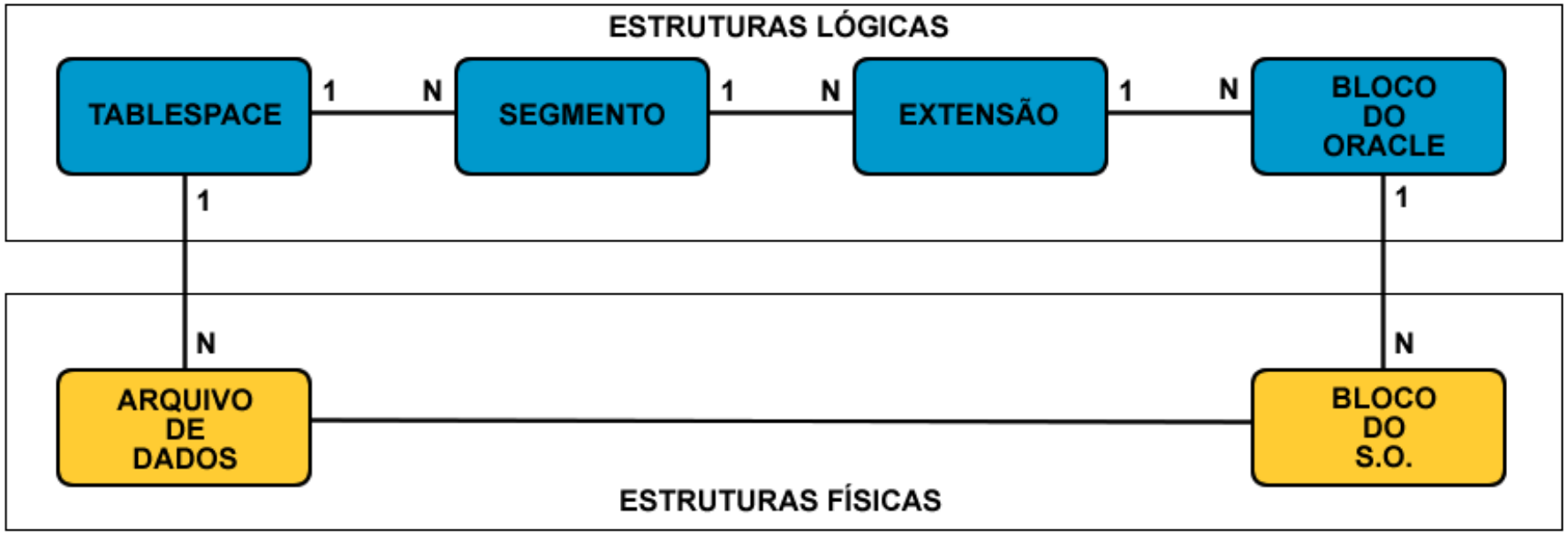
- **Extensões:**
  - **As extensões (extends) são formadas por um ou mais blocos.**
  - **Quando um objeto do banco de dados é expandido são alocadas mais extensões.**

Estruturas lógicas

- **Segmentos:**
  - **Os segmentos (segments) são o próximo nível de agrupamento lógico.**
  - **Um segmento é composto por um grupo de extensões e abrange um objeto do banco de dados (tabela, índice, etc.).**

Estruturas lógicas

Hierarquia de armazenamento lógico e físico



Aula 1: Revisão

- **Arquivos que compõem o banco de dados:**
  - **DATA FILES (arquivos de dados)**
  - **CONTROL FILES (arquivos de controle)**
  - **REDO LOG FILES (arquivos de log on-line)**
- **Arquivos externos ao banco de dados que também são necessários:**
  - **PARAMETER FILE (arquivos de parâmetro de instância)**
  - **PASSWORD FILE (arquivos de senhas)**
  - **ARCHIVED LOG FILE (arquivos de redo log arquivados)**
  - **TRACKING FILE (arquivos de alerta e rastreamento)**

Administração de Banco de Dados

Aula 2

Prof. Marcos Alexandruk

aula 2

Estruturas físicas do Oracle: Arquivos

- Arquivos de dados
- Arquivos de redo log
- Arquivos de controle
- Arquivos de log arquivados
- Arquivos de parâmetro de inicialização
- Arquivos de alerta e rastreamento
- Arquivos de backup
- Arquivos de senha

Arquivos de dados

- Cada banco de dados Oracle contém pelo menos um arquivo de dados que corresponde a um arquivo físico do sistema operacional.
- Um tablespace pode ser composto por vários arquivos de dados, porém um arquivo de dados é membro de somente um tablespace.
- Um tablespace do tipo BIGFILE é composto por apenas um arquivo de dados.
- Caso seja configurado com o parâmetro AUTOEXTEND, o arquivo de dados poderá ser expandido automaticamente.
- O tamanho de um arquivo de dados estará limitado na prática pelo tamanho do disco no qual ele foi criado.
- O espaço em disco a ser ocupado pelo arquivo de dados pode ser limitado através do parâmetro MAXSIZE.
- Visto que é mais fácil mover arquivos "menores", é considerado uma boa prática limitar os arquivos de dados a 2 GB cada um.
- Deve-se observar também que alguns sistemas limitam o tamanho de seus arquivos a 2 GB.
- Os dados das diversas transações realizadas pelos usuários são finalmente gravados nos arquivos de dados (datafiles).
- Blocos de dados acessados com mais frequência são armazenados no cache de memória.
- Blocos de dados novos não são gravados imediatamente no arquivo de dados, porém, antes de concluir uma transação, as alterações são gravadas nos arquivos de redo log.

Arquivos de redo log

- Todos os dados adicionados, alterados ou removidos em tabelas, índices ou outros objetos do Oracle, têm uma entrada gravada no arquivo de redo log.
- São necessários no mínimo dois arquivos de redo log, pois o Oracle utiliza estes arquivos de maneira circular, isto é, após o preenchimento completo do primeiro, começa a utilizar o segundo e, após o preenchimento completo do segundo, volta a utilizar o primeiro, e assim por diante.
- O arquivo de redo log atual (necessário para recuperação da instância) é marcado como ACTIVE.
- Quando o arquivo não for necessário para recuperação da instância é marcado como INACTIVE.
- O arquivo seguinte reutilizado do início é marcado como CURRENT.
- Os arquivos de redo log são fundamentais para o processo de recuperação de falhas, sejam estas causadas por falta de energia ou outras falhas que ocorram no servidor.
- Quando a instância Oracle for reinicializada as entradas do arquivo de redo log são aplicadas aos arquivos do banco de dados para restaurar seu estado até o ponto em que ocorreu a falha.

Arquivos de controle

- Cada banco de dados Oracle tem pelo menos um arquivo de controle que mantém os dados da estrutura física do próprio banco de dados.
- As informações referentes a qualquer alteração realizada na estrutura do banco de dados são imediatamente enviadas ao arquivo de controle.
- Os arquivos de controle contêm o nome do banco de dados, quando este foi criado e os nomes e locais dos arquivos de dados e de redo log.
- Adicionalmente, os arquivos de controle contêm informações utilizadas pelo RMAN (Recovery Manager) e os tipos de backups executados no banco de dados.

O RMAN (Recovery Manager) é um programa independente do Oracle Database Server que proporciona uma forma segura e eficaz de realizar backup e recuperação dos dados.

Arquivos de log arquivados

- Um banco de dados Oracle pode operar em dois modos **archivelog** ou **noarchivelog**.
- Quando o banco está em **noarchivelog** os arquivos de redo log são reutilizados de maneira circular (conforme foi descrito).
- O modo **archivelog** envia um arquivo de redo log preenchido para um ou mais destinos, isto é, para um arquivo de log arquivado, que ficará disponível caso ocorra uma falha de mídia no banco de dados. Neste caso, se a unidade que contém os arquivos de dados for danificada, seu conteúdo pode ser recuperado até o momento anterior à falha.

Arquivos de parâmetro de inicialização

- Quando uma instância é inicializada o Oracle aloca a memória e abre um arquivo de texto denominado **init.ora** também conhecido como **PFILE** ou um arquivo de parâmetro de servidor conhecido como **SPFILE**.

- Os arquivos de parâmetro de inicialização especificam onde estão localizados os arquivos de rastreamento, arquivos de controle e arquivos de redo log.

- Especificam também os tamanhos das várias estruturas que compõem a SGA (System Global Area) e a quantidade de usuários que poderão se conectar simultaneamente ao banco de dados.

- Antes do Oracle 10g o arquivo **init.ora** era a única opção para especificar os parâmetros de inicialização da instância.

- No entanto, esta opção apresenta a seguinte **desvantagem**: se um parâmetro dinâmico de sistema for alterado com o comando ALTER SYSTEM, o DBA deverá editar o arquivo init.ora para que o novo valor do parâmetro entre em vigor na próxima vez que a instância for inicializada.

- Utilizando a opção SPFILE ao utilizar o comando ALTER SYSTEM este poderá alterar automaticamente um parâmetro de inicialização da instância, nenhuma alteração adicional deverá ser feita no arquivo SPFILE.

Arquivos de log de alerta e de rastreamento

- Quando ocorre algum erro, o Oracle geralmente grava as mensagens no arquivo de log de alerta e nos arquivos de rastreamento.

- O arquivo de log de alerta e os arquivos para os processos de segundo plano (background) localizam-se no diretório especificado pelo parâmetro de inicialização BACKGROUND_DUMP_DEST e contém mensagem de status de rotina e condições de erro.

- Arquivos de rastreamento também são criados para sessões ou conexões de usuários e são encontrados no diretório especificado pelo parâmetro de inicialização USER_DUMP_DEST.

Arquivos de backup

- Arquivos de backup podem ser gerados através do comando **copy** do sistema operacional ou com o **RMAN** (Recovery Manager).

- Utilizando o comando copy é possível copiar arquivos de dados, arquivos de redo log arquivados, dentre outros.

- Por outro lado, utilizando a ferramenta de backup RMAN é possível realizar cópias bit a bit dos arquivos de dados ou gerar backups completos ou incrementais de arquivos de dados, arquivos de controle, arquivos de redo log arquivados e SPFILE.

  Estes conjuntos de backups, legíveis somente pelo RMAN, geralmente terão um tamanho menor que os arquivos de dados originais, pois o RMAN não fará backup dos blocos não utilizados.

Arquivos de senha

- O arquivo de senha é utilizado para autenticar administradores do Oracle para tarefas tais como criar, inicializar ou efetuar shutdown em um banco de dados.
- Através do arquivo de senha são concedidos para os privilégios SYSDBA e SYSOPER.
- Outros usuários serão autenticados dentro do próprio banco de dados.
- É possível realizar a autenticação para privilégios SYSDBA e SYSOPER através do sistema operacional, neste caso não será necessário criar um arquivo de senhas e o parâmetro de inicialização REMOTE_LOGIN_PASSWORDFILE deverá ser definido como NONE.

Administração de Banco de Dados

Aula 3

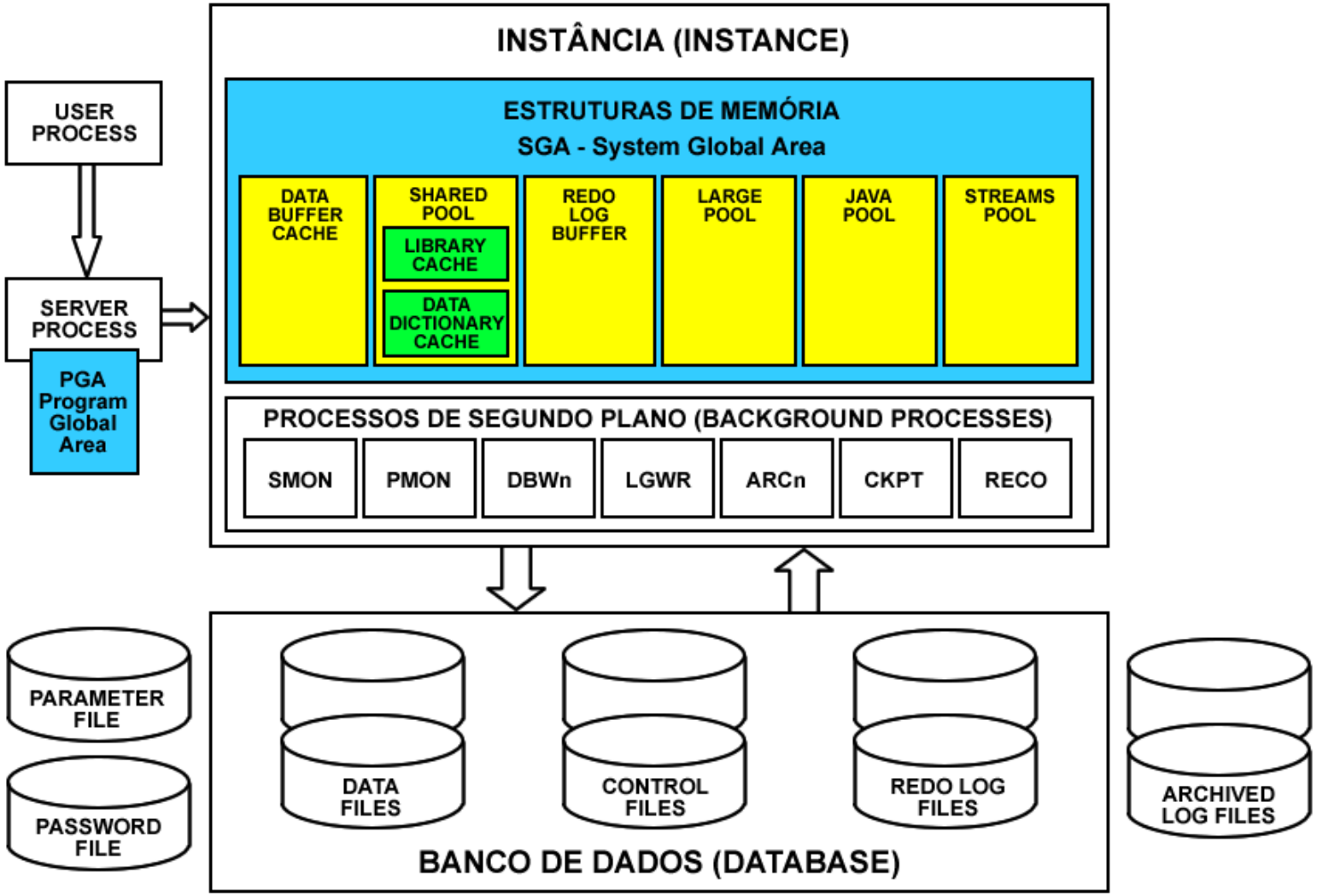
Prof. Marcos Alexandruk

aula 3

Estruturas físicas do Oracle: Memória

- SGA (System Global Area)
  - Buffer caches
  - Shared Pool
  - Library cache
  - Data dictionary cache
  - Redo log buffer
  - Large pool
  - Java pool
  - Streams pool
- PGA (Program Global Area)

SGA (System Global Area)



SGA (System Global Area)

- A SGA é composta por um grupo de estruturas de memória compartilhada para uma instância Oracle.
- Quando a instância é inicializada é alocada para a SGA uma quantidade de memória que pode chegar até o limite especificado pelo parâmetro SGA_MAX_SIZE.
- Caso este valor não seja especificado, o Oracle ajustará automaticamente a memória até o limite especificado no parâmetro SGA_TARGET.
- O parâmetro SGA_TARGET é dinâmico, portanto, poderá ser alterado com a instância em execução.
- O Oracle 11g apresenta um novo parâmetro denominado MEMORY_TARGET que tem por objetivo equilibrar a memória disponível entre a SGA e a PGA.
- A SGA é alocada em unidades conhecidas como grânulos. Caso o tamanho da SGA seja igual ou menor que 128 MB cada grânulo terá 4 MB. Se o tamanho da SGA for superior a 128 MB cada grânulo terá 16 MB.
- A SGA (System Global Area) é composta por várias seções. Veja a seguir as principais seções que compões a SGA.

SGA (System Global Area)

Buffer Cache (Caches de Buffer)

- O Buffer Cache contém blocos de dados que foram lidos recentemente em uma consulta (SELECT) ou que foram alterados ou adicionados através de uma instrução DML (UPDATE, INSERT, etc.).
- O cache de buffer é alocado no momento que a instância é inicializada.
- Até o Oracle 8i não era possível redimensionar o cache de buffer sem reiniciar a instância.
- A partir do Oracle 10g o redimensionamento pode ser realizado manual ou automaticamente (conforme a carga de trabalho).

SGA (System Global Area)

Shared Pool (Pool Compartilhado)

- O Shared Pool contém dois "subcaches" principais:
  - **Library Cache:** área responsável por armazenar o código executado recentemente na sua forma analisada por parse para que ele possa ser reutilizado sem a necessidade de ser reanalisado. Isto melhora muito o desempenho quando são realizadas consultas ao banco de dados
  - **Data Dictionary Cache:** coleção de tabelas, pertencentes aos schemas SYS e SYSTEM, que contêm os metadados sobre o banco de dados (estruturas, privilégios e papéis (roles) dos usuários do banco). Os blocos de dados das tabelas do dicionário de dados são usados para ajudar no processamento de consultas dos usuários e outros comandos DML.
- O tamanho do **Shared Pool** é ajustado de modo dinâmico e pode ser gerenciado automaticamente.

SGA (System Global Area)

Redo Log Buffer (Buffer de Log de Redo)

- O Redo Log Buffer mantém as últimas alterações realizadas nos blocos de dados localizados nos arquivos de dados.
- Quando um terço do buffer estiver preenchido ou a cada três segundos, os registros de log de redo são gravados nos arquivos de log de redo.
- Uma transação confirmada não é considerada completa até que os registros do buffer de log de redo sejam gravados com sucesso nos arquivos de log de redo.

SGA (System Global Area)

Large Pool

- O Large Pool é uma área opcional utilizada para transações que interagem com mais de um banco de dados.
- O Large Pool torna disponível grandes blocos de memória para operações que demandam muita memória.
- O tamanho do Large Pool é controlado através do parâmetro dinâmico LARGE_POOL_SIZE.

SGA (System Global Area)

Java Pool

- O Java Pool disponibiliza memória para a JVM (Java Virtual Machine) do Oracle.
- Este recurso é necessário para processamento dos códigos e dados utilizados pelo Java dentro de uma sessão de usuário.

SGA (System Global Area)

Streams Pool

- O Streams Pool contém estruturas de dados e de controle necessários para gerenciar o compartilhamento de dados em um ambiente distribuído.
- O dimensionamento desta área é controlado através do parâmetro de inicialização STREAMS_POOL_SIZE.
- Caso este parâmetro esteja configurado como zero, será alocada memória para operações de streams a partir do Shared Pool, podendo consumir até 10% deste recurso.

PGA (Program Global Area)

- A PGA (Program Global Area) é a memória não compartilhada criada pelo banco de dados Oracle quando um processo servidor é iniciado.
- O acesso ao PGA é exclusivo para cada processo do servidor. (Há, portanto, uma PGA para cada processo do servidor.)
- Processos de segundo plano (background) também alocam suas próprias PGAs.
- A memória total usada por todas as PGAs individuais é conhecida como instância PGA.
- Os parâmetros de inicialização do banco de dados definem o tamanho da instância PGA e não as PGAs individuais.
- A PGA será alocada conforme a configuração de conexão do banco de dados:
  - Em uma configuração de servidor dedicado cada processo de usuário tem uma conexão exclusiva com o banco de dados.
  - Em uma configuração de servidor compartilhado diversos usuários compartilham uma conexão ao banco. (Haverá, neste caso, um consumo menor de memória no servidor, mas provavelmente afetará o tempo de resposta para as solicitações de usuários.)

Administração de Banco de Dados

Aula 4

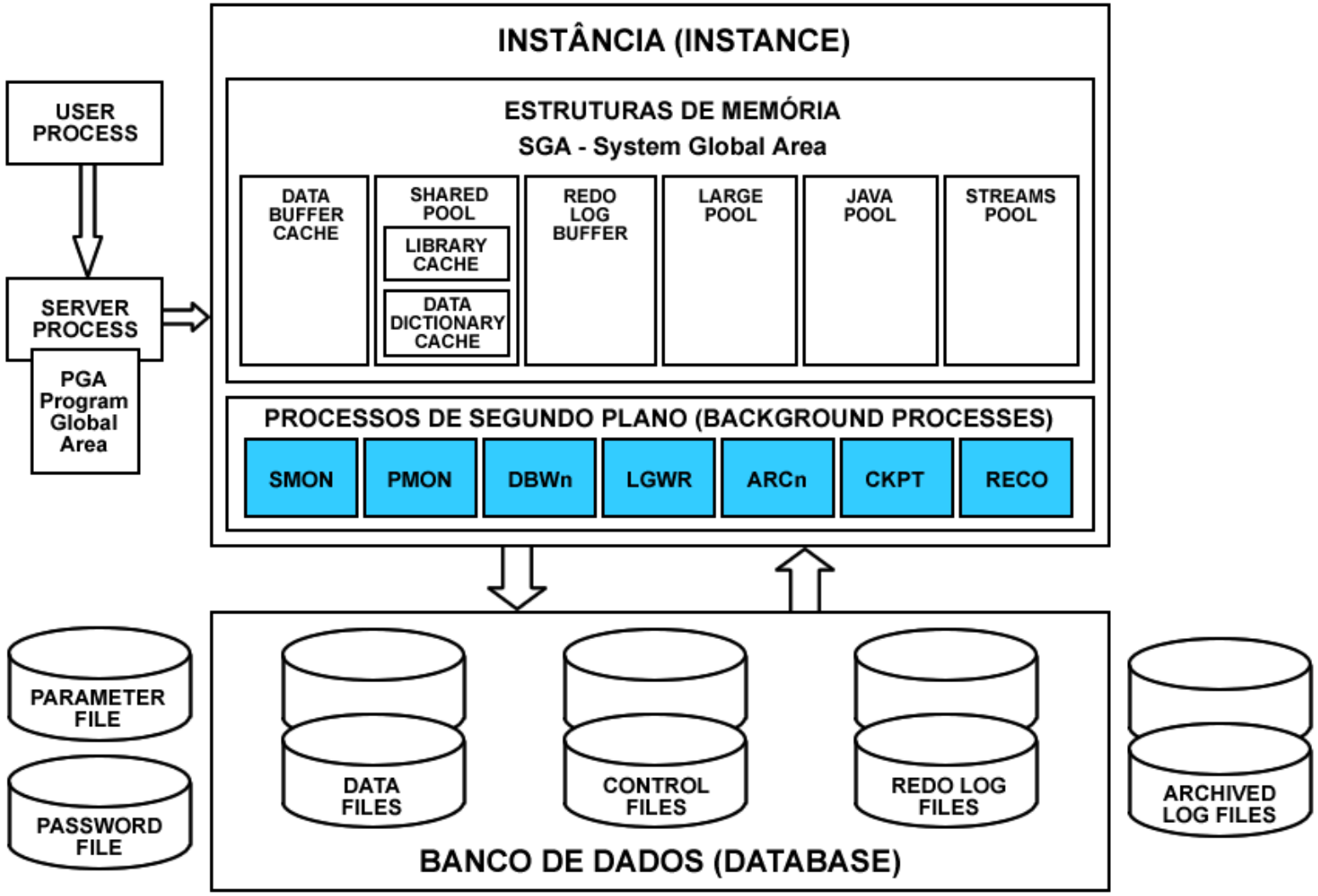
Prof. Marcos Alexandruk

aula 4

Estruturas físicas do Oracle: Background Processes

- SMON - System Monitor
- PMON - Process Monitor
- DBWn - Database Writer
- LGWR - Log Writer
- ARCn - Achiver Process
- CKPT - Checkpoint Process
- RECO - Recoverer Process

Background Processes



Background Processes

SMON (System Monitor)

- O System Monitor executa a recuperação de falhas aplicando as entradas dos arquivos de Redo Log online quando ocorrem erros causados por queda de energia ou falha da CPU.
- Durante o processo de reinicialização do sistema são eliminados os segmentos temporários em todos os tablespaces.
- Além disso, como parte de sua rotina, o SMON junta os espaços livres nos tablespaces gerenciados por dicionário.

Background Processes

PMON (Process Monitor)

- **O Process Monitor limpa o Cache de Buffer de Dados e outros recursos utilizados pelo usuário quando ocorre a falha de um processo de usuário ou se uma conexão de usuário falhar.**

  ***Por exemplo:* se uma sessão terminar de modo anormal o Process Monitor executará o rollback na transação ativa.**

- **O Process Monitor também interage com os listeners fornecendo informações sobre os status da instância para futuras solicitações de conexão.**

  Quando um cliente deseja estabelecer uma conexão com um servidor Oracle, ele geralmente primeiro contata o listener. O listener, então, cria um processo em primeiro plano (modo dedicado) ou atribui o pedido a um processo em primeiro plano existente (modo compartilhado).

  O cliente, em seguida, comunica-se diretamente com o processo de primeiro plano.

Background Processes

DBWn (Database Writer)

- O Database Writer grava blocos de dados novos ou alterados (também denominados de blocos "sujos") no Cache de Buffer nos arquivos de dados (datafiles).
- Uma instância pode ter até vinte Database Writers iniciados (de DBW0 a DBW9 e de DBWa a DBWj).
- O parâmetro de inicialização DB_WRITER_PROCESSES especifica o número de processos DBWn.
- O banco de dados seleciona uma configuração padrão apropriada para este parâmetro ou ajusta uma configuração especificada pelo usuário com base no número de CPUs e grupos de processadores.
- Há quatro circunstâncias que farão o DBWn gravar em disco: excesso de blocos "sujos", ausência de blocos "livres", um tempo limite de três segundos e quando há um checkpoint.

Background Processes

LGWR (Log Writer)

- O processo de background Log Writer é responsável pelo gerenciamento do Buffer de Redo Log.
- Uma transação somente será considerada bem sucedida quando o LGWR gravar com êxito as informações de redo, incluindo o registro do commit, nos Redo Log Files.
- Se o banco de dados tem um redo log multiplexado o LGWR grava as entradas de redo log para um grupo de Redo Log Files.
- Caso um dos arquivos de Redo Log for danificado, o LGWR gravará nos membros restantes do grupo de Redo Log Files e registrará o erro no log de alertas.
- Se todos os membros do grupo não puderem ser utilizados, o processo LGWR falhará e a instância ficará suspensa até que seja realizada a correção do problema.

Background Processes

ARCn (Archiver Process)

- Quando o banco de dados estiver no modo archivelog, o Achiver Process copiará os Redo Logs para um ou mais dispositivos de armazenamento (nos Arquivos de Logs Arquivados).
- A cópia será realizada sempre que um Redo Log for preenchido e começar o preenchimento do próximo Redo Log. (São necessários no mínimo dois arquivos de Redo Log, pois o Oracle utiliza estes arquivos de maneira circular.)
- É importante para o desempenho do banco que o processo de arquivamento termine antes que o Redo Log preenchido seja necessário novamente. (Os usuários não poderão concluir suas transações até as entradas sejam gravadas no Redo Log File e este não estará pronto para aceitar novas entradas, pois ainda estará sendo gravado em um ou mais dos Arquivos de Logs Arquivados.)

Background Processes

CKPT (Checkpoint Process)

- Um checkpoint é uma estrutura de dados que define um SCN (System Change Number) no segmento de redo do banco de dados. (SCN é um número em formato atômico mantido pelo Oracle para registrar quais alterações foram feitas no banco de dados.)
- Toda vez que um commit é realizado, um novo SCN é gerado marcando os arquivos para que o Oracle saiba onde e quanto de recuperação deverá ser aplicado em caso de falha. Isto é feito pelo processo CKPT.
- Os arquivos de controle são fundamentais neste processo, pois informam ao Oracle quais os SCNs corretos.
- Os checkpoints são registrados no arquivo de controle e em cada cabeçalho dos arquivos de dados (datafiles). Portanto, os checkpoints são um elemento essencial na tarefa de recuperação (recovery) do banco de dados.

Background Processes

RECO (Recoverer Process)

- O Recoverer Process é um processo de background que resolve transações distribuídas que estão pendentes por causa de uma falha de rede ou sistema em um banco de dados distribuído.
- O Recoverer Process utiliza as informações na tabela de transações pendentes para finalizar o status de transações em dúvida.
- Em determinados intervalos de tempo, o Recoverer Process local tenta se conectar a bancos de dados remotos e completar automaticamente o commit ou rollback da parte local das transações distribuídas pendentes.
- Todas as transações automaticamente resolvidas pelo Recoverer Process são removidas da tabela de transações pendentes.

Background Processes

Para saber mais: O que é um SCN?

- http://www.oracle.com/technetwork/pt/articles/database-performance/conceito-backup-e-recover-em-oracle-1384601-ptb.html

Administração de Banco de Dados

Aula 5

Prof. Marcos Alexandruk

Aula 5

Estruturas lógicas do Oracle:

- Tablespaces
- Blocos
- Extensões
- Segmentos

Tablespaces

TABLESPACE

- Um tablespace é composto por um ou mais arquivos de dados.
- Porém, cada arquivo de dados poderá estar relacionado a apenas um tablespace.
- Durante a instalação do Oracle 11g são criados pelo menos dois tablespaces: **SYSTEM** e **SYSAUX**.
- O Oracle 11g permite criar um tipo especial de tablespace denominado **BIGFILE** que pode ter até **128 TB** (terabytes).
- O DBA praticamente não terá trabalho para gerenciar os tablespaces do tipo BIGFILE, não precisará, por exemplo, preocupar-se com o tamanho ou com a estrutura dos arquivos de dados subjacentes.

Tablespaces

OMF - Oracle Management Files

- **Outro recurso disponível no Oracle 11g é o OMF (Oracle Management Files) que facilita ainda mais o gerenciamento dos arquivos de dados do tablespace.**

- **O DBA precisará apenas especificar um ou mais locais no sistema de arquivos onde os dados de controle e de redo log serão armazenados e o Oracle fará automaticamente o gerenciamento destes arquivos.**

Tablespaces

Tablespaces Temporários

- **Os tablespaces temporários, embora sejam mantidos "permanentemente" no servidor Oracle, são utilizados pelos segmentos temporários em operações de classificação de dados e para tabelas que existem apenas durante a sessão do usuário.**

Tablespaces

Gerenciamento de Tablespaces

- O gerenciamento dos tablespaces pode ser realizado através do dicionário de dados ou localmente.
- O gerenciamento das extensões dos tablespaces gerenciados por dicionário é registrado nas tabelas do dicionário de dados localizado no tablespace SYSTEM.
- Portanto, ainda que todas as tabelas sejam, por exemplo, criadas no tablespace USERS, o tablespace SYSTEM será acessado no processo de gerenciamento das operações DML.
- Este modo de gerenciamento pode causar problemas de performance, uma vez que todos os usuários e aplicações farão uso do tablespace SYSTEM para o gerenciamento das extensões.

Tablespaces

Gerenciamento de Tablespaces

- **Quando o gerenciamento é realizado localmente, o Oracle mantém um bitmap em cada arquivo de dados do tablespace para monitorar a quantidade de espaço livre.**

- **Apenas as quotas são gerenciadas no dicionário de dados, desta forma melhorando a performance.**

A partir do Oracle 9i, se o tablespace SYSTEM for gerenciado localmente todos os outros tablespaces, exceto os que sejam somente de leitura (read only), devem obrigatoriamente ser gerenciados localmente.

Blocos de Dados

- Os blocos são as menores unidades de armazenamento do banco de dados Oracle.
- O tamanho do bloco é determinado através de um número específico de bytes e corresponde normalmente a um múltiplo do tamanho do bloco do sistema operacional.
- O parâmetro DB_BLOCK_SIZE especifica o tamanho do bloco de dados dos tablespaces SYSTEM, SYSAUX e outros tablespaces temporários.
- Outros quatro tamanhos podem ser utilizados para os blocos de outros tablespaces.

Extensões

- **Uma extensão é formada por um ou mais blocos de dados.**
- **Sempre que um objeto do banco de dados é expandido, o espaço adicionado ao objeto é alocado como uma extensão.**

Segmentos

- Um segmento é composto por um grupo de extensões que abrange um objeto do banco de dados (tabelas, índices, etc.).
- Esta será, portanto, a menor unidade de dados que um usuário de banco de dados utilizará na prática. Em um banco de dados Oracle há quatro tipos de segmentos:
  - Segmentos de dados;
  - Segmentos de índices;
  - Segmentos temporários;
  - Segmentos de undo.

Segmentos

Segmentos de dados

- Cada tabela é formada por um segmento de dados que são formados, por sua vez, por uma ou mais extensões.

- Tabelas particionadas ou clusterizadas podem ser formadas por mais de uma extensão. (Detalhes adicionais sobre tabelas serão apresentados na próxima aula.)

Segmentos

Segmentos de índices

- Cada índice criado no Oracle é armazenado em seu próprio segmento de índice.
- No entanto, assim como ocorre com as tabelas particionadas, cada partição de índice é armazenada em seu próprio segmento.

Segmentos

Segmentos temporários

- Quando determinadas operações precisam de espaço além daquele encontrado na memória do servidor, o Oracle aloca um segmento temporário.
- Operações de classificação de dados são exemplos de operações que normalmente precisam de espaço em disco.
- Os segmentos temporários têm o mesmo "tempo de vida" das instruções SQL que geraram a demanda por espaço em disco.

Segmentos

Segmentos de undo

- Sempre que uma transação altera os dados, uma versão dos dados anterior à atualização é gravada em um segmento de undo. Quando uma transação é iniciada o Oracle atribui a ela um segmento de undo. Este tipo de segmento é necessário para fornecer uma visão consistente de leitura para outros usuários que estejam acessando os dados que têm sua origem nos objetos que estão sendo alterados. Se a transação for revertida (ocorrer um rollback) também será necessário reconstruir os dados conforme estes se apresentavam antes do início da transação.
- Até o Oracle 9i estes segmentos eram chamados de segmentos de rollback. A partir desta versão foi oferecido o recurso denominado Automatic Undo Management que tornou automático o gerenciamento deste tipo de segmento e que, portanto, recebeu esta nova denominação.
- O Oracle 11g cria apenas um segmento de rollback na tablespace SYSTEM no momento que o banco de dados é criado, porém este segmento não é utilizado para as transações normais dos usuários.

Administração de Banco de Dados

Aula 6

Prof. Marcos Alexandruk

Aula 6

Estruturas lógicas do Oracle:

Tabelas

- **tabelas relacionais**
- **tabelas temporárias**
- **tabelas organizadas por índices**
- **tabelas de objetos**
- **tabelas externas**
- **tabelas clusterizadas**
- **tabelas particionadas**

Tabelas

- As tabelas são as unidades básicas de armazenamento do banco de dados Oracle. Tabelas são estruturas bidimensionais compostas por linhas (tuplas) e colunas (campos).
- O banco dados Oracle apresenta diferentes tipos de tabelas para atender a necessidades ou demandas específicas:
  - tabelas relacionais
  - tabelas temporárias
  - tabelas organizadas por índices
  - tabelas de objetos
  - tabelas externas
  - tabelas clusterizadas
  - tabelas particionadas

Tabelas

Tabelas relacionais

- As tabelas relacionais são as mais comuns em um banco de dados.
- Tabelas relacionais ou organizadas por heap apresentam linhas que não são organizadas em nenhuma ordem específica.
- É possível especificar no comando CREATE TABLE a cláusula ORGANIZATION HEAP, porém, como este é o padrão, torna-se desnecessário incluir esta cláusula.

Tabelas

Tabelas temporárias

- As tabelas temporárias, disponíveis a partir do Oracle 8i, são temporárias somente com relação aos dados que armazenam, pois sua estrutura, isto é, as denominações das colunas e seus respectivos tipos de dados e tamanhos, ficam gravadas no banco de dados até que ocorra sua alteração através de um comando ALTER TABLE ou sua eliminação através de um comando DROP TABLE.
- Qualquer usuário com permissões suficientes pode acessar as tabelas temporárias, utilizar instruções SELECT ou comandos DML (INSERT, UPDATE e DELETE), no entanto, cada usuário poderá ver apenas os seus dados. Quando utilizar o comando TRUNCATE TABLE, apenas os seus dados serão removidos da tabela.
- Há também duas cláusulas que controlam o modo como os dados serão removidos da tabela temporária. A cláusula ON COMMIT DELETE ROWS remove as linhas da tabela quando é emitido um comando COMMIT ou ROLLBACK e a cláusula ON COMMIT PRESERVE ROWS mantém as linhas na tabela mesmo após um comando COMMIT ou ROLLBACK, porém, ao final da sessão todos os dados do usuário serão removidos da tabela temporária.

Tabelas

Tabelas organizadas por índice

- A criação de um índice diminui o tempo necessário para localização de uma linha específica na tabela. Uma tabela organizada por índice (IOT - Index Organized Table) armazena as linhas de uma tabela em um índice de árvore B (B-tree). Cada nó contém uma coluna indexada juntamente com uma ou mais colunas não indexadas. Uma tabela organizada por índice deve, portanto, ter sempre uma chave primária.

- Uma das principais vantagens deste tipo de tabela é que apenas uma estrutura de armazenamento precisa ser atualizada e não duas como seria o caso se fosse criada uma tabela relacional e separadamente um índice.

Tabelas

Tabelas de objetos

- As linhas das tabelas de objetos correspondem aos objetos propriamente ditos ou a instanciações de definições de tipos.
- As linhas em uma tabela de objetos podem ser referenciadas através da chave primaria, se esta existir, ou através do OID (Object Identifier) que é exclusivo para cada objeto.

Tabelas

Tabelas externas

- As tabelas externas, disponíveis a partir do Oracle 9i, permitem que o usuário acesse uma fonte externa de dados, um arquivo de texto, por exemplo, como se fosse uma tabela. O conteúdo é armazenado externamente, apenas os metadados para a tabela são armazenados no dicionário de dados.
- Não é possível criar índices nem executar operações de inserção, atualização ou exclusão de dados em tabelas externas.

Tabelas

Tabelas clusterizadas

- As tabelas clusterizadas podem melhorar o desempenho quando duas ou mais tabelas são frequentemente acessadas juntas. Reduzem também a quantidade de espaço necessário para armazenar as colunas que duas tabelas têm em comum.
- O valor da chave é armazenado em um índice de cluster (cluster index) que, assim como em um índice tradicional, aprimora as consultas às tabelas clusterizadas quando acessadas pelo valor da chave do cluster.

Tabelas

Tabelas particionadas

- O particionamento permite decompor tabelas muito grandes e índices em partes menores melhorando assim o seu gerenciamento.
- Estas partes menores são denominadas partições. Cada partição é um objeto independente com seu próprio nome e, opcionalmente, suas próprias características de armazenamento.
- Portanto, se uma partição estiver em um volume de disco corrompido, as outras partições ainda estarão disponíveis para consultas enquanto o volume corrompido for reparado.
- O particionamento é transparente para a aplicação, isto é, não é necessário especificar explicitamente a partição quando se utiliza, por exemplo, a linguagem SQL para uma consulta ao banco de dados.

Constraints

- **Constraints ou restrições são regras que podem ser definidas em uma ou mais colunas de tabelas para implementar regras de negócio. Há vários tipos diferentes de constraints:**
  - **Primary Key**
  - **Foreign Key**
  - **Not Null**
  - **Unique**
  - **Check**

Constraints

Primary Key

- A constraint Primary Key (Chave Primária) é composta por uma coluna ou grupo de colunas e permite identificar uma única linha (tupla) da tabela.
- A coluna ou colunas que compõem a Primary Key não podem ter valores NULL.
- Cada tabela pode ter uma única constraint deste tipo.

Constraints

Foreign Key

- A constraint Foreign Key utiliza os valores de uma determinada coluna para estabelecer o relacionamento entre duas tabelas.
- Os valores desta coluna devem corresponder a valores encontrados na Primary Key da tabela-pai.
- Além disso, uma constraint Foreign Key pode ser autorreferencial, neste caso, seus valores deverão corresponder aos valores encontrados na Primary Key que estará localizada na mesma tabela.

Constraints

Not Null

- A constraint Not Null indica que o conteúdo de uma coluna não poderá ser nulo ou vazio.
- Nulo é diferente de 0 (zero). Nulo representa ausência de valor (o valor não existe).

Constraints

Unique

- A constraint Unique garante que não haverá nenhum valor repetido em uma coluna ou grupo de colunas.

- Uma coluna com restrição do tipo Unique poderá receber também uma restrição Not Null. Caso isso não ocorra, qualquer número de linhas poderá conter "valores" NULL, desde que as outras linhas desta mesma coluna tenham valores únicos.

Constraints

Check

- A constraint Check define um domínio de valores para o conteúdo da coluna.
- É possível estabelecer mais de uma constraint Check para a mesma coluna. Todas elas devem ser avaliadas como TRUE (verdadeiro) para permitir que um valor seja inserido na coluna.

Administração de Banco de Dados

Aula 7

Prof. Marcos Alexandruk

Aula 7

Índices (Indexes)

- Índices únicos
- Índices não únicos
- Índices de chave invertida
- Índices baseados em funções
- Índices de bitmap

Visões (Views)

- Visões regulares
- Visões materializadas
- Visões de objeto

Índices

- Índices (Indexes) permitem acesso mais rápido a determinadas linhas de uma tabela quando um pequeno subconjunto de linhas for selecionado.
- Portanto, os índices armazenam os valores das colunas que estão sendo indexadas juntamente com o RowID físico da respectiva linha, exceto no caso das tabelas organizadas por índice, que utilizam a Primary Key como um RowID lógico.
- O Oracle dispõe de vários tipos de índices, específicos para cada tipo de tabela, método de acesso ou ambiente de aplicação.
  - Índices únicos
  - Índices não únicos
  - Índices de chave invertida
  - Índices baseados em funções
  - Índices de bitmap

Índices

Índices únicos (exclusivos)

- Os índices únicos são os mais comuns do tipo árvore B e são utilizados principalmente para impor a constraint Primary Key de uma tabela. Este tipo de índice garante que não existirão valores duplicados na coluna ou nas colunas indexadas.

```
CREATE TABLE nome_da_tabela (
nome_coluna_1 tipo_de_dado(tamanho),
...
CONSTRAINT nome_da_constraint PRIMARY KEY(nome_da_coluna_1)
);
```

Índices

Índices não únicos (não exclusivos)

- Os índices não únicos, apesar de não impor exclusividade de valores, aceleram o acesso aos dados quando a consulta for realizada utilizando-se como parâmetro a coluna ou as colunas indexadas. Pode-se, por exemplo, criar um índice deste tipo na coluna NOME de uma tabela denominada CLIENTE para localizar mais rapidamente os dados de um cliente a partir de seu nome.

```
CREATE INDEX nome_do_indice ON nome_da_tabela(nome_da_coluna);
```

- Para renomear um índice:

```
ALTER INDEX nome_do_indice RENAME TO novo_nome_do_indice;
```

Índices

Índices de chave invertida

- Todos os bytes no valor de chave são invertidos nos índices de chave invertida. Por exemplo, para o número de pedido 1234 o índice de chave invertida armazenará como 4321.
- Um de seus principais objetivos é evitar disputas em sistemas multiusuários.
- Quando muitos usuários estiverem inserindo simultaneamente linhas com chaves baseadas em valores que aumentam sequencialmente, todas as suas inserções de índice são concentradas na extremidade mais alta do índice.
- Optando-se por índices de chave invertida, as inserções de chave de índice consecutiva são distribuídas por todo o intervalo do índice.

```
CREATE INDEX nome_do_indice ON nome_da_tabela(nome_da_coluna)
REVERSE;
```

Índices

Índices baseados em funções

- Um índice baseado em função calcula o valor de uma função ou expressão envolvendo uma ou mais colunas e o armazena no índice.
- O índice baseado em função pode ser uma árvore B ou um índice de bitmap (ver tópico seguinte).
- A função usada para construir o índice pode ser uma expressão aritmética, uma expressão que contém uma função SQL, uma função definida pelo usuário utilizando PL/SQL, etc.

```
CREATE INDEX nome_do_indice ON nome_da_tabela(expressão|função);
```

Índices

Índices de bitmap

- Os índices de mapas de bits estão disponíveis apenas na versão Enterprise Edition do Oracle.
- Um índice de bitmap armazena os RowIDs associados ao valor da chave como um bitmap.
- O comprimento da string de bits é igual ao número de linhas que está sendo indexada.
- Este tipo de índice é especialmente recomendável quando:
  - O número de valores distintos na coluna é baixo;
  - O número de linhas na tabela é alto;
  - A coluna é utilizada em operações de álgebra booleana.

```
CREATE BITMAP INDEX nome_do_indice
ON nome_da_tabela(nome_da_coluna);
```

Índices

Índices de bitmap

- **Exemplo: Índice de bitmap associado à coluna CAMPUS da tabela ALUNO.**

ALUNO		
RA	**NOME**	**CAMPUS**
111222333	ANTONIO ALVES	VILA MARIA
222333444	BEATRIZ BERNARDES	MEMORIAL
333444555	CLAUDIO CARVALHO	VERGUEIRO
444555666	DANIELA DAMASCENO	SANTO AMARO
555666777	ERNESTO ELISEU	MEMORIAL
666777888	FLAVIA FERNANDES	VILA MARIA
777888999	HIGINO HIPOLITO	SANTO AMARO
888999000	ISABEL INACIO	VERGUEIRO
999000111	JOSUE JORGEANO	MEMORIAL

ÍNDICE DE BITMAP NA COLUNA CAMPUS	
CAMPUS	**BITMAP**
VILA MARIA	100001000
MEMORIAL	010010001
VERGUEIRO	001000010
SANTO AMARO	000100100

O bitmap correspondente ao campus MEMORIAL é 010010001, pois este valor aparece na 2ª, 5ª e 9ª linha da tabela ALUNO.

Índices

Index Organized Tables (IOTs)

- Tabelas organizadas por índice permitem que os dados do índice e da tabela sejam armazenados juntos.
- Portanto, haverá uma redução significativa no espaço em disco, porque as colunas indexadas não são armazenadas duas vezes (uma vez na tabela e outra no índice).
- As IOTs devem ser utilizadas principalmente em tabelas em que o método principal de acesso é a chave primária.

```
CREATE TABLE nome_da_tabela (
nome_coluna_1 tipo_de_dado(tamanho),
...
CONSTRAINT nome_da_constraint PRIMARY KEY(nome_da_coluna_1))
ORGANIZATION INDEX;
```

Índices

Exercícios

Criar um exemplo para cada um dos itens abaixo:

- **Índices únicos;**
- **Índices não únicos;**
- **Índices de chave invertida;**
- **Índices baseados em funções;**
- **Índices de bitmap;**
- **Tabela organizada por índice.**

Visões

- Uma visão (view) é uma representação lógica de uma ou mais tabelas.
- Uma visão deriva seus dados de tabelas denominadas tabelas base. (As tabelas base, na prática, podem ser tabelas ou outras visões.)
- As consultas em visões são realizadas da mesma maneira que as consultas em tabelas: basta incluir o nome da visão na cláusula WHERE.
- É possível realizar (com algumas restrições) operações DML (INSERT, UPDATE E DELETE) nas tabelas base através das visões
- Todas as operações executadas em uma visão afetam as tabelas base.
- As visões são também conhecidas como consultas armazenadas.
- O Oracle dispõe de vários tipos de visões. Exemplos:
  - Visões regulares
  - Visões materializadas

Visões

Visões regulares

- Uma visão regular armazena apenas sua definição ou consulta no dicionário de dados, não alocando, portanto, espaço em um segmento para armazenamento dos dados.
- Visões regulares podem seu utilizadas para ocultar a complexidade de determinadas consultas ao banco de dados ou para impor segurança.

Visões

Visões regulares

- **Exemplo: Visão FUNCIONARIO_VIEW criada a partir da tabela FUNCIONARIO.**

FUNCIONARIO			
MATRICULA	**NOME**	**DEPARTAMENTO**	**SALARIO**
1001	ANTONIO ALVES	ENGENHARIA	5300
1002	BEATRIZ BERNARDES	MARKETING	4800
1003	CLAUDIO CARCALHO	JURIDICO	5100

```
CREATE VIEW FUNCIONARIO_VIEW AS
SELECT MATRICULA, NOME, DEPARTAMENTO
FROM FUNCIONARIO;
```

FUNCIONARIO_VIEW		
MATRICULA	**NOME**	**DEPARTAMENTO**
1001	ANTONIO ALVES	ENGENHARIA
1002	BEATRIZ BERNARDES	MARKETING
1003	CLAUDIO CARCALHO	JURIDICO

Visões ajudam a melhorar a segurança no acesso aos dados.

Pode-se conceder acesso somente às visões e impedir a realização de consultas diretamente nas tabelas.

Visões regulares: READ ONLY

- Pode-se também criar uma visão com restrição **READ ONLY** (apenas leitura).

FUNCIONARIO			
MATRICULA	**NOME**	**DEPARTAMENTO**	**SALARIO**
1001	ANTONIO ALVES	ENGENHARIA	5300
1002	BEATRIZ BERNARDES	MARKETING	4800
1003	CLAUDIO CARCALHO	JURIDICO	5100

```
CREATE VIEW FUNCIONARIO_VIEW AS
SELECT MATRICULA, NOME, DEPARTAMENTO
FROM FUNCIONARIO
WITH READ ONLY CONSTRAINT FUNC_VIEW_READ_ONLY;
```

FUNCIONARIO_VIEW		
MATRICULA	**NOME**	**DEPARTAMENTO**
1001	ANTONIO ALVES	ENGENHARIA
1002	BEATRIZ BERNARDES	MARKETING
1003	CLAUDIO CARCALHO	JURIDICO

Visões regulares: UTILIZANDO APELIDOS PARA COLUNAS

FUNCIONARIO			
MATRICULA	NOME	DEPARTAMENTO	SALARIO
1001	ANTONIO ALVES	ENGENHARIA	5300
1002	BEATRIZ BERNARDES	MARKETING	4800
1003	CLAUDIO CARCALHO	JURIDICO	5100

```
CREATE VIEW FUNCIONARIO_VIEW (MAT_FUNC, NOME_FUNC, DEPT_FUNC) AS
SELECT MATRICULA, NOME, DEPARTAMENTO
FROM FUNCIONARIO;
```

ou:

```
CREATE VIEW FUNCIONARIO_VIEW AS
SELECT
MATRICULA AS MAT_FUNC,
NOME AS NOME_FUNC,
DEPARTAMENTO AS DEPT_FUNC
FROM FUNCIONARIO;
```

Visões

- **Visões facilitam a realização de consultas complexas baseadas em duas ou mais tabelas.**

```
--------------------    -----------------
TABELA: CLIENTE         TABELA: PEDIDO
--------------------    -----------------
CODCLI   NOME      UF   NR  VALOR  CODCLI
--------------------    -----------------
1001     FULANO    SP   1   4800   1002
1002     BELTRANO  RJ   2   3600   1003
1003     CICRANO   SP   3   5500   1001
```

```
SELECT C.UF, SUM(P.VALOR)
FROM CLIENTE C
INNER JOIN PEDIDO P
ON C.CODCLI = P.CODCLI
GROUP BY C.UF;
```

```
-------------------------------
VISAO: CLIENTE_PEDIDO_VIEW
-------------------------------
CODCLI   NOME      UF   NR  VALOR
-------------------------------
1001     FULANO    SP   1   4800
1002     BELTRANO  RJ   2   3600
1003     CICRANO   SP   3   5500
```

```
SELECT UF, SUM(VALOR)
FROM CLIENTE_PEDIDO_VIEW
GROUP BY C.UF;
```

Visões

- **Inserir ou atualizar dados em visões criadas a partir de duas ou mais tabelas base:**

```
--------------------    -----------------
TABELA: CLIENTE         TABELA: PEDIDO
--------------------    -----------------
CODCLI   NOME      UF   NR  VALOR  CODCLI
--------------------    -----------------
1001     FULANO    SP   1   4800   1002
1002     BELTRANO  RJ   2   3600   1003
1003     CICRANO   SP   3   5500   1001

--------------------------------
VISAO: CLIENTE_PEDIDO_VIEW
--------------------------------
CODCLI   NOME      UF   NR  VALOR
--------------------------------
1001     FULANO    SP   1   4800
1002     BELTRANO  RJ   2   3600
1003     CICRANO   SP   3   5500
```

```
CREATE VIEW CLIENTE_PEDIDO_VIEW AS
SELECT C.CODCLI, C.NOME, C.UF,
P.NR, P.VALOR
FROM CLIENTE C
INNER JOIN PEDIDO P
ON C.CODCLI = P.CODCLI;
```

Visões

- **Criar um trigger instead of para inserir dados nas tabelas CLIENTE e PEDIDO através da visão CLIENTE_PEDIDO_VIEW:**

```
CREATE OR REPLACE TRIGGER CLI_PED_INSERT
INSTEAD OF INSERT ON CLIENTE_PEDIDO_VIEW
REFERENCING NEW AS N
FOR EACH ROW
DECLARE
   rowcnt number;
BEGIN
   SELECT COUNT(*) INTO rowcnt FROM CLIENTE WHERE CODCLI = :N.CODCLI;
    IF rowcnt = 0  THEN
       INSERT INTO CLIENTE (CODCLI,NOME,UF) VALUES (:N.CODCLI, :N.NOME, :N.UF);
   ELSE
      UPDATE CLIENTE SET CLIENTE.NOME = :N.NOME, CLIENTE.UF = :N.UF
         WHERE CLIENTE.CODCLI = :N.CODCLI;
    END IF;
   SELECT COUNT(*) INTO rowcnt FROM PEDIDO WHERE NR = :N.NR;
   IF rowcnt = 0  THEN
       INSERT INTO PEDIDO (NR,VALOR,CODCLI) VALUES (:N.NR, :N.VALOR, :N.CODCLI);
    ELSE
      UPDATE PEDIDO SET PEDIDO.VALOR = :N.VALOR, PEDIDO.CODCLI = :N.CODCLI
         WHERE PEDIDO.NR = :N.NR;
   END IF;
END;
/
```

Visões

Materialized Views (Visões Materializadas)

- Uma visão materializada armazena sua definição ou consulta no dicionário de dados, porém, diferentemente de das visões regulares, aloca espaço em um segmento para armazenamento dos dados. Este tipo de visão pode, por exemplo, replicar uma cópia somente leitura da tabela base para outro banco de dados.
- Visões materializadas utilizam um log de visão materializada associado às tabelas base para realizar atualizações incrementais. Caso não se utilize este recurso, será preciso realizar uma atualização completa quando for necessária a atualização dos dados na visão materializada.

Materialized Views - Exemplos:

Criação de uma tabela denominada ALUNO:

```
---------------
TABELA: ALUNO
---------------
RA       NOME
---------------
1001    ANONIO
1002    BEATRIZ
1003    CLAUDIO
```

Criar um MATERIALIZED VIEW LOG (log de visão materializada):

```
CREATE MATERIALIZED VIEW LOG ON ALUNO;
```

Os logs de visões materializadas são usados para sincronização entre a tabela base (master table) e a visão.
Antes da criação da visão materializada, a master table deve ser associada a um materialized view log.

Materialized Views - Exemplo 1:

CRIAÇÃO DE UMA VISÃO MATERIALIZADA QUE SERÁ ATUALIZADA **MANUALMENTE**:

```
CREATE MATERIALIZED VIEW ALUNO_VIEW_1
-- popular a view no momento de sua criação
BUILD IMMEDIATE
-- selecionar os dados da tabela base
AS SELECT * FROM ALUNO;
```

CONSULTAR A VIEW ALUNO_VIEW_1:

```
SELECT * FROM ALUNO_VIEW_1;
```

INSERIR UMA NOVA LINHA NA TABELA ALUNO:

```
INSERT INTO ALUNO (RA, NOME) VALUES (4,'DANIELA');
COMMIT;
```

UTILIZAR A PROCEDURES DBMS_MVIEW.REFRESH PARA ATUALIZAR A VIEW:

```
CALL DBMS_MVIEW.REFRESH ('ALUNO_VIEW_1','F');
-- F: FAST (INCREMENTAL) | C: COMPLETE (COMPLETA) | ?: FORCE (INCREMENTAL
-- SE POSSÍVEL, CASO NÃO SEJA, REALIZA UMA ATUALIZAÇÃO COMPLETA)
```

CONSULTAR A VIEW ALUNO_VIEW_1:

```
SELECT * FROM ALUNO_VIEW_1;
```

Materialized Views - Exemplo 2:

CRIAÇÃO DE UMA VISÃO MATERIALIZADA QUE SERÁ ATUALIZADA **AUTOMATICAMENTE**:

```
CREATE MATERIALIZED VIEW ALUNO_VIEW_2
-- popular a view no momento de sua criação
BUILD IMMEDIATE
-- atualizar de forma incremental a cada 1 segundo
REFRESH FORCE START WITH SYSDATE NEXT SYSDATE + 1/86400
-- selecionar os dados da tabela base
AS SELECT * FROM ALUNO;
```

CONSULTAR A VIEW ALUNO_VIEW_2:

```
SELECT * FROM ALUNO_VIEW_1;
```

INSERIR UMA NOVA LINHA NA TABELA ALUNO:

```
INSERT INTO ALUNO (RA, NOME) VALUES (5,'ERNESTO');
COMMIT;
```

CONSULTAR A VIEW ALUNO_VIEW_2:

```
SELECT * FROM ALUNO_VIEW_2;
```

Visões

Visões de objeto

- Uma visão de objeto é uma tabela virtual de objeto. Cada linha na visão é um objeto, que é uma instância de um tipo de objeto. Um tipo de objeto é um tipo de dados definido pelo usuário.
- As visões de objeto permitem que as aplicações orientadas a objetos vejam os dados como uma coleção de objetos que têm atributos e métodos, facilitando a migração de um ambiente puramente relacional para um ambiente orientado a objetos.
- É possível utilizar as visões de objeto para recuperar, atualizar, inserir e excluir dados relacionais como se estes fossem armazenados como um tipo de objeto. Pode-se também definir visões com colunas que são tipos de dados de objetos, tais como objetos e coleções (tabelas aninhadas e varrays).

Administração de Banco de Dados

Aula 8

Prof. Marcos Alexandruk

Aula 8

Gerenciamento de Instância

Instâncias

Instância

- Uma instância é a combinação dos processos de segundo plano e das estruturas de memória (SGA).
- A instância deve ser iniciada para que seja possível acessar os dados do banco de dados. Toda vez que uma instância é iniciada, uma SGA (System Global Area) é alocada e os processos de segundo plano do Oracle são iniciados.
- Instância e banco de dados são entidades separadas em um servidor Oracle. Portanto, elas podem existir independentemente uma da outra.
- Uma instância é definida pelo arquivo de parâmetros de instância INIT.ORA.

Instâncias

Arquivo de parâmetro de instância INIT.ORA

- O arquivo INIT.ORA apresenta alguns parâmetros que devem ser destacados:
  - DB_NAME: Este parâmetro nomeia o banco de dados ao qual a instância está associada.
  - DB_BLOCK_SIZE: Determina o tamanho dos blocos no cache de buffer do banco de dados.
  - CONTROL_FILES: Apresenta o local onde são encontrados os arquivos de controle, essenciais ao funcionamento do banco de dados.

Instâncias

- **O processo de inicialização é realizado em etapas: primeiro a instância é criada na memória do servidor, depois é ativada uma conexão com o banco de dados montando-o e, finalmente o banco de dados é aberto para uso.**

- **Um banco de dados Oracle pode estar em um dos seguintes estados:**

  - **SHUTDOWN: Todos os arquivos estão fechados e a instância não existe.**
  - **NOMUNT: A instância é criada na memória (a SGA é criada e os processos de background são iniciados). Não foi feita ainda nenhuma conexão com o banco de dados. É possível inclusive que o banco de dados não tenha sido criado.**
    Caso o arquivo de controle esteja corrompido ou estiver faltando uma cópia multiplexada, não será possível montar o banco de dados. Porém, talvez seja possível reparar estes arquivos e, em seguida, passar ao modo MOUNT.
  - **MOUNT: A instância localiza e lê o arquivo de controle do banco de dados.**
    Neste modo é possível reparar eventuais danos em arquivos de dados ou em arquivos de redo log e, em seguida passar ao modo OPEN.
  - **OPEN: Todos os arquivos do banco de dados são localizados, abertos e o banco de dados torna-se disponível para os usuários.**

Instâncias

SHUTDOWN

- As seguintes opções podem ser utilizadas no comando SHUTDOWN, todas elas requerem uma conexão SYSDBA ou SYSOPER:
  - NORMAL: Não permitirá nenhuma nova conexão de usuário, porém as conexões atuais continuarão até que todos os usuários façam logoff, somente então é que o banco de dados fará shutdown.
  - TRANSACTIONAL: Não permitirá nenhuma nova conexão de usuário. As sessões que não estiverem em uma transação serão terminadas. As sessões que estiverem em transação serão terminadas assim que concluírem a transação.
  - IMMEDIATE: Não permitirá nenhuma nova conexão. Todas as sessões serão terminadas. As transações ativas sofrerão rollback e o banco de dados fará shutdown.
  - ABORT: Equivale praticamente a um corte de energia. A instância termina imediatamente. Não há sequer a tentativa de terminar adequadamente as transações em andamento. Embora este tipo de shutdown não danifique o banco de dados, não é aconselhável fazer um backup do banco após um abort.

Instâncias

SHUTDOWN E STARTUP

- Exemplos de comando utilizados:

Exemplo 1:

```
SHUTDOWN IMMEDIATE;
STARTUP NOMOUNT;
ALTER DATABASE MOUNT;
ALTER DATABASE OPEN;
```

Exemplo 2:

```
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
---
ALTER DATABASE OPEN;
```

Instâncias

Informações importantes sobre a instância são obtidas na visão V$INSTANCE.

```
DESC V$INSTANCE

Nome                      Tipo
------------------------- -------------
INSTANCE_NUMBER           NUMBER
INSTANCE_NAME             VARCHAR2(16)
HOST_NAME                 VARCHAR2(64)
VERSION                   VARCHAR2(17)
STARTUP_TIME              DATE
STATUS                    VARCHAR2(12)
PARALLEL                  VARCHAR2(3)
THREAD#                   NUMBER
ARCHIVER                  VARCHAR2(7)
LOG_SWITCH_WAIT           VARCHAR2(15)
LOGINS                    VARCHAR2(10)
SHUTDOWN_PENDING          VARCHAR2(3)
DATABASE_STATUS           VARCHAR2(17)
INSTANCE_ROLE             VARCHAR2(18)
ACTIVE_STATE              VARCHAR2(9)
BLOCKED                   VARCHAR2(3)
```

Instâncias

INSTANCE_NUMBER

- Em uma instalação RAC (Real Application Clusters) indica o número da instância desse nó no cluster.

```
SELECT INSTANCE_NUMBER FROM V$INSTANCE;

INSTANCE_NUMBER
---------------
              1
```

- Em um ambiente de Real Application Clusters, várias instâncias podem ser associadas um único serviço de banco de dados.

Instâncias

INSTANCE_NAME

- Nome único da instância dentro do cluster.

```
SELECT INSTANCE_NAME FROM V$INSTANCE;

INSTANCE_NAME
----------------
orcl
```

Instâncias

HOST_NAME

- Nome do servidor de banco de dados.

```
SELECT HOST_NAME FROM V$INSTANCE;

HOST_NAME
----------
HP-DESKTOP
```

Instâncias

VERSION

- Versão do Oracle Database.

```
SELECT VERSION FROM V$INSTANCE;

VERSION
----------
11.2.0.1.0
```

Instâncias

STARTUP_TIME

- Data de inicialização da instância.

```
SELECT STARTUP_TIME FROM V$INSTANCE;

STARTUP
--------
18/03/13
```

Instâncias

STATUS

- Estado atual da instância.

```
SELECT STATUS FROM V$INSTANCE;

STATUS
------
OPEN
```

STATUS DA INSTÂNCIA:

- **STARTED**: Após STARTUP NOMOUNT
- **MOUNTED**: Após STARTUP MOUNT ou ALTER DATABASE CLOSE
- **OPEN**: Após STARTUP ou ALTER DATABASE OPEN
- **OPEN MIGRATE**: Após ALTER DATABASE OPEN UPGRADE | DOWNGRADE

Instâncias

PARALLEL

- Indica se a instância está montada no modo de banco de dados em cluster (YES) ou não (NO).

```
SELECT PARALLEL FROM V$INSTANCE;

PARALLEL
--------
NO
```

Instâncias

THREAD

- Thread de redo aberto pela instância.

```
SELECT THREAD# FROM V$INSTANCE;

THREAD#
-------
      1
```

- Thread é um parâmetro do RAC (Real Application Cluster) que especifica o número da thread de redo usado por uma instância.
- Quando você cria um banco de dados, Oracle cria e habilita o thread 1 como um thread público (pode ser usado por qualquer instância).
- Para criar e habilitar os threads subsequentes utiliza-se a cláusula ADD LOGFILE THREAD e a cláusula ENABLE THREAD na instrução ALTER DATABASE.
- O número de threads é limitado pelo parâmetro MAXINSTANCES especificado na instrução CREATE DATABASE.

Instâncias

ARCHIVER

- Status do arquivamento automático (automatic archiving).

```
SELECT ARCHIVER FROM V$INSTANCE;

ARCHIVE
-------
STARTED
```

STATUS DO ARQUIVAMENTO AUTOMÁTICO:

- **STOPPED**: Status apresentado no modo NOARCHIVELOG.
- **STARTED** : Status apresentado no modo ARCHIVELOG.
- **FAILED**: Não conseguiu arquivar um log na última vez, mas tentará fazê-lo novamente dentro de cinco minutos.

Instâncias

LOG_SWITCH_WAIT

- **Evento que a alteração de log está esperando:**

```
SELECT LOG_SWITCH_WAIT FROM V$INSTANCE;

LOG_SWITCH_WAIT
---------------
```

- ***Ocorre um LOG_SWITCH quando a Oracle para a gravação para o arquivo de log atual, fecha, abre o arquivo seguinte de log e começa a escrever no novo arquivo de log.***

  EVENTOS:

  - **ARCHIVE LOG**
  - **CLEAR LOG**
  - **CHECKPOINT**
  - **NULL: (há espaço no atual arquivo de redo log online)**

Instâncias

LOGINS

- Indica se a instância está no modo irrestrito (ALLOWED), permitindo logins por todos os usuários autorizados, ou no modo restrito (RESTRICTED), permitindo logins de banco de dados apenas por administradores.

```
SELECT LOGINS FROM V$INSTANCE;

LOGINS
-------
ALLOWED
```

Instâncias

SHUTDOWN_PENDING

- Indica se há algum shutdown (desligamento) pendente (YES) ou não (NO).

```
SELECT SHUTDOWN_PENDING FROM V$INSTANCE;

SHUTDOWN_PENDING
---------------
NO
```

Instâncias

DATABASE_STATUS

- Status do database (banco de dados).

```
SELECT DATABASE_STATUS FROM V$INSTANCE;

DATABASE_STATUS
---------------
ACTIVE
```

STATUS DO DATABASE:

- ACTIVE
- SUSPEDED
- INSTANCE RECOVERY

Instâncias

INSTANCE_ROLE

- **Indica se é uma instância ativa (PRIMARY_INSTANCE), uma instância secundária inativa (SECONDARY_INSTANCE) ou se a instância foi iniciada, mas não montada (UNKNOWN).**

```
SELECT INSTANCE_ROLE FROM V$INSTANCE;

INSTANCE_ROLE
----------------
PRIMARY_INSTANCE
```

- ***A implantação da instância primária/secundária permite configurar um sistema de alta disponibilidade de dois nós para o Oracle RAC (Real Application Clusters). Uma instância designada como instância primária em um nó aceita as conexões de usuários, enquanto uma instância designada como secundária no outro nó aceita conexões quando o nó primário falha, ou quando especificamente selecionado por meio do parâmetro INSTANCE_ROLE na entrada CONNECT_DATA do arquivo tnsnames.ora.***

Instâncias

ACTIVE_STATE

- Estado do banco em relação ao comando ALTER SYSTEM QUIESCE RESTRICTED.

```
SELECT ACTIVE_STATE FROM V$INSTANCE;

ACTIVE_STATE
------------
NORMAL
```

STATUS DO DATABASE:

- **NORMAL**: banco de dados está em um estado normal.
- **QUIESCING**: ALTER SYSTEM QUIESCE RESTRICTED foi emitido: não serão processadas novas transações de usuário, consultas ou instruções PL/SQL nesta instância. As instruções emitidas antes não são afetadas.
- **QUIESCED:** ALTER SYSTEM QUIESCE RESTRICTED foi emitido: não serão processadas transações de usuário, consultas ou instruções PL/SQL nesta instância. As instruções emitidas após não são processadas.

Instâncias

ALTER SYSTEM QUIESCE RESTRICTED;

- Os comandos **ALTER SYSTEM QUIESCE RESTRICTED** e **ALTER SYSTEM UNQUIESCE** permitem, respectivamente, **desativar** e **ativar** o banco de dados para realizar atividades de manutenção sem precisar desligar o banco.

```
ALTER SYSTEM QUIESCE RESTRICTED;
SELECT ACTIVE_STATE FROM V$INSTANCE;

ACTIVE_STATE
------------
QUIESCED

ALTER SYSTEM UNQUIESCE;
SELECT ACTIVE_STATE FROM V$INSTANCE;

ACTIVE_STATE
------------
NORMAL
```

Instâncias

BLOCKED

- Indica se todos os serviços estão bloqueados (YES) ou não (NO).

```
SELECT BLOCKED FROM V$INSTANCE;

BLOCKED
-------
NO
```

Administração de Banco de Dados

Aula 9

Prof. Marcos Alexandruk

Aula 9

Gerenciamento da estrutura física:

- arquivos
- memória
- processos de segundo plano

Estrutura Física: arquivos

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.

Estrutura Física: arquivos

Data Files

- Armazenam os dados (tabelas), os índices e as áreas temporárias e de rollback (undo) e estão sempre ligados a um tablespace.

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

```
SELECT NAME, STATUS FROM V$DATAFILE;

SELECT * FROM V$DBFILE;
```

Estrutura Física: arquivos

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;
```

Estrutura Física: arquivos

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. 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$LOG
  - V$LOGFILE

```
SELECT MEMBER, TYPE FROM V$LOGFILE;
```

Estrutura Física: memória

SGA (System Global Area)

- A área de memória alocada para uma instância Oracle é denominada SGA (System Global Area). Além disso, uma área de memória privativa denominada PGA (Program Global Area) é alocada para cada processo no servidor.

- Para obter-se informações importantes sobre a SGA pode-se utilizar uma das três consultas apresentadas a seguir:

```
SHOW SGA

SELECT * FROM V$SGA;

SELECT * FROM V$SGAINFO;
```

Estrutura Física: memória

- O Oracle 11g apresenta um novo parâmetro denominado MEMORY_TARGET que tem por objetivo equilibrar a memória disponível entre a SGA e a PGA. Este parâmetro especifica a memória utilizável de todo o sistema Oracle. O banco de dados ajusta a memória conforme o valor MEMORY_TARGET, reduzindo ou ampliando a SGA e a PGA conforme necessário.
- O parâmetro MEMORY_TARGET está especificado no arquivo texto INIT.ORA ou no arquivo binário SPFILE.
- Caso seja omitido o parâmetro MEMORY_MAX_TARGET e seja incluído um valor para MEMORY_TARGET, o banco de dados automaticamente define MEMORY_MAX_TARGET para o valor de MEMORY_TARGET.
- Por outro lado, caso seja omitida a linha para MEMORY_TARGET e seja incluído um valor para o MEMORY_MAX_TARGET, o parâmetro MEMORY_TARGET padrão será 0 (zero).
- Após a inicialização, pode-se alterar dinamicamente MEMORY_TARGET para um valor diferente de zero, desde que não exceda o valor de MEMORY_MAX_TARGET.

Estrutura Física: memória

- Para verificar o valor do parâmetro MEMORY_TARGET deve-se utilizar a seguinte consulta:

```
SHOW PARAMETER MEMORY_TARGET
```

- Para verificar o valor do parâmetro MEMORY_MAX_TARGET deve-se utilizar a seguinte consulta:

```
SHOW PARAMETER MEMORY_MAX_TARGET
```

- Para alterar o valor do parâmetro MEMORY_TARGET (até o limite, conforme apresentado no parâmetro MEMORY_MAX_SIZE) deve-se utilizar o seguinte comando:

```
ALTER SYSTEM SET MEMORY_TARGET = 250M;
```

Estrutura Física: processos

Os principais **processos de segundo plano** são os seguintes:

- **SMON – System Monitor:** executa a recuperação de falhas aplicando as entradas dos arquivos de Redo Log online quando ocorrem erros causados por queda de energia ou falha da CPU.
- **PMON – Process Monitor:** limpa o Cache de Buffer de Dados e outros recursos utilizados pelo usuário quando ocorre a falha de um processo de usuário ou se uma conexão de usuário falhar.
- **DBWn – Database Writer:** grava blocos de dados do Cache de Buffer nos arquivos (em disco).
- **LGWR – Log Writer:** responsável pelo gerenciamento do Buffer de Redo Log.
- **ARCn – Achiver Process:** copia o conteúdo de Redo Log Files para arquivos localizados em discos rígidos ou fitas.
- **CKPT – Checkpoint Process:** 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 são registradas no banco de dados de forma permanente.
- **RECO – Recoverer Process:** usado na configuração de um banco de dados distribuído para resolver automaticamente falhas envolvendo transações distribuídas.

Estrutura Física: processos

- Para obter-se uma lista completa com os nomes de todos os processos de segundo plano disponíveis no servidor Oracle utiliza-se a seguinte consulta:

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

Administração de Banco de Dados

Aula 10

Prof. Marcos Alexandruk

Aula 10

Gerenciamento da estrutura lógica:

Tablespaces

Tablespaces

Tablespaces

- Um **tablespace** é um container de armazenamento lógico para segmentos.
- **Segmentos** são objetos de banco de dados, como tabelas e índices que consomem espaço de armazenamento.
- A nível físico, um tablespace armazena dados em um ou mais arquivos de dados ou arquivos temporários.
- Um banco de dados Oracle 11g deve ter os tablespaces **SYSTEM** e **SYSAUX**.

Tablespaces

Tablespaces

- O banco de dados Oracle apresenta três tipos principais de tablespaces:
  - **Permanente:** contém segmentos que persistem além da duração de uma transação ou sessão;
  - **Undo:** armazenam valores anteriores a uma inclusão, atualização ou exclusão. Um banco de dados pode ter mais de um tablespace de undo, mas apenas um pode estar ativo em um dado momento.
  - **Temporário:** contém dados transitórios que só existem enquanto durar a sessão, alocando, por exemplo, espaço para concluir uma classificação de dados que não cabe na memória.

Tablespaces

Consultar nomes dos tablespaces

- Para consultar os nomes dos tablespaces presentes em um determinado banco de dados deve-se utilizar o seguinte comando:

```
SELECT TABLESPACE_NAME FROM DBA_TABLESPACES;

TABLESPACE_NAME
------------------------------
SYSTEM
SYSAUX
UNDOTBS1
TEMP
USERS
EXAMPLE
```

Tablespaces

Tablespaces

- **SYSTEM:** tablespace obrigatório e o mais crítico porque contém o dicionário de dados, onde são armazenadas todas as informações necessárias para o próprio gerenciamento do banco de dados. Se por algum motivo ele se tornar indisponível, a instância do Oracle abortará. O tablespace SYSTEM nunca pode ser colocado offline, ao contrário de um tablespace comum como, por exemplo, o tablespace USERS.
- **SYSAUX:** tablespace auxiliar disponível a partir do Oracle 10g. Foi criado para aliviar o tablespace SYSTEM de segmentos associados a algumas aplicações do próprio banco de dados e segmentos relacionados ao funcionamento do Oracle Enterprise Manager. Como resultado, alguns gargalos de I/O associados ao tablespace SYSTEM foram reduzidos ou eliminados. Portanto, não é recomendável que o tablespace SYSAUX seja colocado no modo offline.
- **UNDOTBS1:** armazena valores anteriores a uma inclusão, atualização ou exclusão.
- **TEMP:** contém dados temporários que só existem enquanto durar a sessão.
- **USERS:** é o tablespace padrão para os usuários. Se algum usuário criar um objeto, tal como uma tabela ou um índice, sem especificar o tablespace, o Oracle irá criá-lo no tablespace USERS, isso se não for definido outro tablespace como padrão para o usuário em questão.
- **EXAMPLE:** este tablespace apresenta exemplos de segmentos e estruturas de dados do Oracle. O tablespace EXAMPLE deve ser descartado em um ambiente de produção.

Tablespaces

Tipos de gerenciamento de tablespaces

- Os tablespaces podem ser gerenciados por dicionário ou localmente.
- A partir do Oracle 9i, se o tablespace SYSTEM for gerenciado localmente todos os outros tablespaces, exceto os que sejam somente de leitura (read only), devem obrigatoriamente ser gerenciados localmente.
- Para verificar o tipo de gerenciamento de extensões do tablespace SYSTEM deve-se utilizar o seguinte comando:

```
SELECT TABLESPACE_NAME, EXTENT_MANAGEMENT FROM DBA_TABLESPACES
WHERE TABLESPACE_NAME = 'SYSTEM';
```

Tablespaces

Criando um tablespace permanente gerenciado localmente

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

```
CREATE TABLESPACE TESTE1
DATAFILE 'C:\DADOS1.DBF' SIZE 10M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 256K;
```

- A cláusula **UNIFORM SIZE** indica que todas as extensões (extents) terão o mesmo tamanho. O tamanho (SIZE) default é 1MB.

- Para consultar os nomes dos tablespaces utilize o comando:

```
SELECT TABLESPACE_NAME FROM DBA_TABLESPACES;
```

Tablespaces

Criando um tablespace permanente gerenciado localmente

- Se a cláusula UNIFORM SIZE for omitida, será assumido o valor default: AUTOALLOCATE, opção em que a alocação das extensões será gerenciada pelo Oracle.

- O exemplo a seguir apresenta a criação do tablespace TESTE2 com a opção AUTOALLOCATE:

```
CREATE TABLESPACE TESTE2
DATAFILE 'C:\DADOS2.DBF' SIZE 10M
EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

- Para consultar os nomes dos tablespaces utilize novamente o comando:

```
SELECT TABLESPACE_NAME FROM DBA_TABLESPACES;
```

Tablespaces

Criando tablespaces de UNDO e TEMPORÁRIOS

- **Para criar um TABLESPACE de UNDO utilize o seguinte comando:**

```
CREATE UNDO TABLESPACE UNDOTBS2
DATAFILE 'C:\UNDOTBS02.ORA' SIZE 10M
EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

- **Para criar um TABLESPACE de TEMPORÁRIO utilize o seguinte comando:**

```
CREATE TEMPORARY TABLESPACE TEMP2
TEMPFILE 'C:\TEMP2.ORA' SIZE 10M
EXTENT MANAGEMENT LOCAL;
```

Tablespaces

Alterando um tablespace

- **Para incluir mais um arquivo de dados (datafile) em um tablespace criado anteriormente utilize o seguinte comando:**

```
ALTER TABLESPACE TESTE2
ADD DATAFILE 'C:\DADOS3.DBF' SIZE 10M;
```

Tablespaces

Eliminando um tablespace

- Para eliminar um tablespace utilize o seguinte comando:

```
DROP TABLESPACE TESTE1
INCLUDING CONTENTS AND DATAFILES CASCADE CONSTRAINTS;
```

- **INCLUDING CONTENTS:** indica que o tablespace deverá ser eliminado mesmo que contenha segmentos. No entanto, a operação fracassará se houverem segmentos de undo ou temporários ativos.
- **AND DATAFILES:** utilizado com INCLUDING CONTENTS, força a eliminação física dos arquivos de dados (datafiles) que compõem o tablespace.
- **CASCADE CONSTRAINTS:** Elimina constraints relacionadas a tabelas do tablespace que está sendo eliminado que estejam em outro tablespace.

Datafiles

Datafiles

- Cada arquivo de dados (datafile) do Oracle corresponde a um arquivo físico do sistema operacional.
- Conforme apresentado na aula 02, um tablespace pode ser composto por vários arquivos de dados, porém um arquivo de dados é membro de somente um tablespace.
- Para alterar o tamanho de um datafile deve-se utilizar o comando ALTER DATABASE. O exemplo apresentado a seguir altera o tamanho do arquivo DADOS1.DBF para 20 MB:

```
ALTER DATABASE
DATAFILE 'C:\DADOS2.DBF' RESIZE 20M;
```

Segmentos

- Um segmento é composto por um grupo de extensões e abrange um objeto do banco de dados (tabela, índice, etc.).
- O exemplo a seguir apresenta a criação de uma tabela no tablespace USERS. Observe que foi alocado um segmento que recebeu o mesmo nome da tabela (coluna SEGMENT_NAME).

```
CREATE TABLE CLIENTE (
CODIGO NUMBER(4),
NOME VARCHAR2(30))
TABLESPACE USERS;

SELECT TABLE_NAME, TABLESPACE_NAME FROM USER_TABLES
WHERE TABLE_NAME = 'CLIENTE';

TABLE_NAME           TABLESPACE_NAME
-------------------- ------------------
CLIENTE              USERS

SELECT SEGMENT_NAME, SEGMENT_TYPE, TABLESPACE_NAME FROM USER_SEGMENTS
WHERE TABLESPACE_NAME = 'USERS';

SEGMENT_NAME         SEGMENT_TYPE         TABLESPACE_NAME
-------------------- -------------------- ------------------
CLIENTE              TABLE                USERS
```

Extensões

Extensões

- As extensões são formadas por um ou mais blocos de dados. Quando um objeto do banco de dados é expandido são alocadas mais extensões.
- Para verificar o tipo de gerenciamento de extensões em cada tablespace deve-se utilizar o seguinte comando:

```
SELECT TABLESPACE_NAME, EXTENT_MANAGEMENT FROM DBA_TABLESPACES;

TABLESPACE_NAME                EXTENT_MAN
------------------------------ ----------
SYSTEM                         LOCAL
SYSAUX                         LOCAL
UNDOTBS1                       LOCAL
TEMP                           LOCAL
USERS                          LOCAL
EXAMPLE                        LOCAL
```

Blocos

Blocos

- Os blocos são as menores estruturas de armazenamento no banco de dados Oracle. Um bloco de dados pode ser constituído de um ou mais blocos do sistema operacional.
- Para verificar o tamanho dos blocos em cada tablespace deve-se utilizar o seguinte comando:

```
SELECT TABLESPACE_NAME, BLOCK_SIZE FROM DBA_TABLESPACES;

TABLESPACE_NAME                BLOCK_SIZE
------------------------------ ----------
SYSTEM                               8192
SYSAUX                               8192
UNDOTBS1                             8192
TEMP                                 8192
USERS                                8192
EXAMPLE                              8192
```

Administração de Banco de Dados

Aula 11

Prof. Marcos Alexandruk

Aula 11

Gerenciamento de usuários (schemas)

Gerenciamento de usuários

Autenticação de usuários

- Os usuários podem ser autenticados no banco de dados através de três métodos diferentes:
  - através do banco de dados;
  - através do sistema operacional;
  - através da rede.

Gerenciamento de usuários

Criação de usuários

- Quando um usuário cria um novo objeto no banco de dados (uma tabela, por exemplo) este objeto fará parte de um schema (esquema) que tem o mesmo nome do usuário.

- Um usuário de banco de dados somente poderá ser criado pelo DBA ou por outro usuário com o privilégio de sistema CREATE USER. Além de informar o nome e a senha (provisória), é possível determinar também quais tablespaces estarão disponíveis e até mesmo quanto espaço de armazenamento o novo usuário poderá utilizar em cada tablespace.

Gerenciamento de usuários

Criando um usuário

- Para criar um novo usuário deve-se utilizar o comando CREATE USER, conforme o exemplo a seguir:

```
CREATE USER FULANO IDENTIFIED BY ABC123
ACCOUNT UNLOCK
PASSWORD EXPIRE
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP;
```

`ACCOUNT UNLOCK:` A conta estará desbloqueada (default).

`PASSWORD EXPIRE:` Usuário deverá alterar a senha no primeiro login.

`DEFAULT TABLESPACE:` Tablespace padrão do usuário.

`TEMPORARY TABLESPACE:` Tablespace temporária para este usuário.

Gerenciamento de usuários

Alterando um usuário

- Pode-se alterar os privilégios e outros atributos dos usuários do banco de dados com o comando ALTER USER.

- O exemplo a seguir estabelece uma cota de 100 MB no tablespace USERS para o usuário anteriormente criado:

```
ALTER USER FULANO
QUOTA 100M ON USERS;
```

Gerenciamento de usuários

Concedendo privilégios a um usuário

- Para conceder privilégios a um usuário deve-se utilizar a instrução **GRANT**.

- O exemplo a seguir apresenta a concessão do privilégio **CONNECT** ao novo usuário que permitirá a ele conectar-se ao banco de dados:

```
GRANT CONNECT TO FULANO;
```

Gerenciamento de usuários

Eliminando um usuário

- Para eliminar usuários do banco de dados utiliza-se o comando **DROP USER**.
- O exemplo a seguir elimina o usuário anteriormente criado e seus objetos (tabelas, visões, etc.):

```
DROP USER FULANO CASCADE;
```

Gerenciamento de usuários

Apresentando os usuários

- Os nomes dos usuários do banco de dados podem ser obtidos através da visão DBA_USERS. O comando a seguir apresenta os nomes dos usuários:

```
SELECT USERNAME FROM DBA_USERS;
```

- Para consultar o status de cada usuário (conta), isto é, para verificar se a conta está ou não bloqueada, utiliza-se:

```
SELECT USERNAME, ACCOUNT_STATUS FROM DBA_USERS;
```

`OPEN:` A conta está disponível para uso;

`LOCKED:` A conta foi bloqueada pelo DBA;

`EXPIRED:` A senha expirou. O usuário deverá redefini-la;

`EXPIRED & LOCKED:` A conta foi bloqueada e a senha expirou.

Gerenciamento de usuários

Conectando-se como outro usuário

- Há casos em que o DBA precisará conectar-se como outro usuário para resolver determinados problemas.

EXERCÍCIO

- Utilize o roteiro apresentado na aula 11 e observe como isso pode ser feito.

Gerenciamento de usuários

1. O DBA (conectado como SYSTEM) obtém a senha conforme gerada por um algoritmo HASH:

```
SELECT NAME, PASSWORD FROM SYS.USER$ WHERE NAME = 'FULANO';

NAME                           PASSWORD
------------------------------ ----------------
FULANO                         22DD56A22A50F1B8
```

2. O DBA copia a senha em um arquivo texto.

3. O DBA (conectado como SYTEM) altera temporariamente a senha do usuário:

```
ALTER USER FULANO IDENTIFIED BY SENHA_TEMP;
```

4. O DBA conecta-se ao banco utilizando a senha temporária:

```
CONNECT FULANO/SENHA_TEMP;
```

5. O DBA realiza as operações necessárias na conta do usuário:

6. O DBA (conectado como SYSTEM) 'repõe' a senha original do usuário:

```
ALTER USER FULANO IDENTIFIED BY VALUES '22DD56A22A50F1B8';.
```

Administração de Banco de Dados

Aula 12

Prof. Marcos Alexandruk

Aula 12

Perfis de usuários (profiles)

Perfis de usuários

Gerenciando perfis de usuários

- Recursos como CPU, espaço em disco e outros são limitados. Portanto, para controlar quanto destes recursos os usuários poderão utilizar, o Oracle oferece um conjunto identificado de limites de recursos denominado perfil de usuário.

- Os perfis de usuários podem ser utilizados também para controlar a criação, reutilização e validação de senhas.

Perfis de usuários

Criando um perfil de usuário

- Um novo perfil de usuário deve ser criado com o comando CREATE PROFILE.
- O exemplo a seguir apresenta a criação de um PROFILE denominado LIMITE_LOGIN para limitar o número de vezes consecutivas que um login pode falhar antes de bloquear a senha do usuário.

```
CREATE PROFILE LIMITE_LOGIN
LIMIT FAILED_LOGIN_ATTEMPTS 3;
```

- O próximo exemplo apresenta a criação de um PROFILE denominado LIMITE_CONEXAO para limitar o tempo de conexão a um usuário em meia hora (trinta minutos).

```
CREATE PROFILE LIMITE_CONEXAO
LIMIT CONNECT_TIME 30;
```

Perfis de usuários

IMPORTANTE:

- **Para que os limites relacionados aos recursos de sistema possam ser controlados através de perfis de usuários deve-se realizar a seguinte alteração:**

```
ALTER SYSTEM SET resource_limit = TRUE;
```

Perfis de usuários

Criando um perfil de usuário

- Quando não for especificado nenhum perfil para um novo usuário do banco de dados o Oracle aplica a este o perfil DEFAULT.

- Para conhecer quais são os limites do perfil DEFAULT deve-se utilizar a seguinte consulta ao dicionário de dados:

```
SELECT * FROM DBA_PROFILES
WHERE PROFILE = 'DEFAULT';
```

Perfis de usuários

Controle de recursos

- **A consulta anterior apresenta os parâmetros relacionados aos recursos que poderão ser utilizados pelos usuários.**

PARÂMETRO	DESCRIÇÃO
SESSIONS_PER_USER	Número de sessões simultâneas que um usuário poderá abrir.
CPU_PER_SESSION	Tempo máximo de CPU por sessão em centésimos de segundo.
CPU_PER_CALL	Tempo máximo de CPU para análise de uma instrução, execução ou operação de busca em centésimos de segundo.
LOGICAL_READS_PER_SESSION	Número total de leituras de blocos por sessão.
LOGICAL_READS_PER_CALL	Número máximo de leituras de blocos para análise de uma instrução, execução ou operação de busca.
IDLE_TIME	Tempo máximo de inatividade contínua de uma sessão em minutos.
CONNECT_TIME	Tempo máximo de conexão de um usuário em minutos.
PRIVATE_SGA	Quantidade máxima de memória que uma sessão pode alocar no pool compartilhado (pode ser especificado em bytes, kilobytes ou megabytes).
COMPOSITE_LIMIT	Unidades de serviço composta pela soma ponderada de PRIVATE_SGA, CONNECT_TIME, CPU_PER_SESSION, LOGICAL_READS_PER_SESSION.

Perfis de usuários

Controle de senhas dos usuários

- **A consulta anterior apresenta também os parâmetros relacionados à administração de senhas de usuários.**

PARÂMETRO	DESCRIÇÃO
FAILED_LOGIN_ATTEMPTS	Número de tentativas mal sucedidas antes de bloquear a conta.
PASSWORD_LIFE_TIME	Número de dias que a senha poderá ser utilizada antes de precisar ser alterada.
PASSWORD_REUSE_TIME	Número de dias que o usuário deverá aguardar até que possa reutilizar a mesma senha. (Utilizado com o parâmetro PASSWORD_REUSE_MAX).
PASSWORD_REUSE_MAX	Número de alterações de senhas até que a mesma senha possa ser utilizada novamente. (Utilizado com o parâmetro PASSWORD_REUSE_TIME).
PASSWORD_VERIFY_FUNCTION	Script PL/SQL com uma rotina para verificação da senha. Se for especificado NULL, nenhuma verificação será realizada.
PASSWORD_LOCK_TIME	Número de dias que a conta ficará bloqueada caso exceda o limite de tentativas mal sucedidas especificado em FAILED_LOGIN_ATTEMPTS. Após este período a conta será automaticamente desbloqueada.
PASSWORD_GRACE_TIME	Número de dias depois que a senha expirada deverá ser alterada. Caso não seja alterada dentro deste período a conta expirará e a senha deverá ser alterada para que o usuário possa utilizar novamente a conta.

Perfis de usuários

Alterando um perfil de usuário

- Um perfil de usuário pode ser alterado com o comando ALTER PROFILE.

- O exemplo a seguir altera o parâmetro LIMITE_LOGIN do perfil criado anteriormente.

```
ALTER PROFILE LIMITE_LOGIN
LIMIT FAILED_LOGIN_ATTEMPTS 5;
```

- O próximo exemplo altera o tempo de conexão de um usuário para uma hora (sessenta minutos).

```
ALTER PROFILE LIMITE_CONEXAO
LIMIT CONNECT_TIME 60;
```

Perfis de usuários

Associando um usuário a um perfil

- **Para associar um usuário a um determinado perfil:**

```
-- Ao criar o usuário:
CREATE USER FULANO IDENTIFIED BY 'ABC123' PROFILE NOME_PERFIL;

-- Para usuário criado anteriormente:
ALTER USER FULANO PROFILE NOME_PERFIL;
```

- **Para consultar a que perfil um usuário está relacionado:**

```
SELECT USERNAME, PROFILE FROM DBA_USERS
WHERE USERNAME = 'FULANO';
```

Administração de Banco de Dados

Aula 13

Prof. Marcos Alexandruk

Aula 13

Privilégios de sistema

Privilégios de sistema

- **Privilégios de sistema envolvem os direitos de realizar determinadas ações sobre objetos do banco de dados e alteração de parâmetros de sistema, dentre outros.**
- **A versão 11.2 do Oracle Database oferece 208 privilégios de sistema que são apresentados na tabela de dicionário de dados SYSTEM_PRIVILEGE_MAP.**

Privilégios de sistema

```
SELECT NAME FROM SYSTEM_PRIVILEGE_MAP;

NAME
--------------------
ALTER SYSTEM
AUDIT SYSTEM
CREATE SESSION
CREATE TABLESPACE
ALTER TABLESPACE
DROP TABLESPACE
CREATE USER
ALTER USER
DROP USER
CREATE TABLE
CREATE ANY TABLE
ALTER ANY TABLE
DROP ANY TABLE
...
```

```
...
CREATE INDEX
CREATE ANY INDEX
ALTER ANY INDEX
DROP ANY INDEX
SYSDBA
SYSOPER
CREATE VIEW
ALTER DATABASE
CREATE PROCEDURE
CREATE ANY PROCEDURE
...
```

Privilégios de sistema

PRIVILÉGIO DE SISTEMA	DESCRIÇÃO
SYSDBA | SYSOPER	Criar uma nova entrada no arquivo externo de senhas, inicializar ou interromper uma instância, criar, alterar e recuperar um banco de dados, etc.
ALTER SYSTEM	Alterar os parâmetros no arquivo SPFILE. Emitir instruções ALTER SYSTEM.
AUDIT SYSTEM	Fazer auditoria no banco de dados.
CREATE SESSION	Conectar com o banco de dados.
ALTER DATABASE	Alterar o estado de um banco de dados. Exemplo: alterar o estado de MOUNT para OPEN.
CREATE TABLESPACE	Criar novos tablespaces.
ALTER TABLESPACE	Alterar parâmetros dos tablespaces.
DROP TABLESPACE	Eliminar tablespaces.
CREATE USER	Criar novos usuários.
ALTER USER	Alterar privilégios e outros dados dos usuários.
DROP USER	Eliminar usuários.
CREATE TABLE	Criar novas tabelas em seu próprio esquema.
CREATE ANY TABLE	Criar novas tabelas em qualquer esquema.
ALTER ANY TABLE	Alterar as tabelas em qualquer esquema.
DROP ANY TABLE	Eliminar tabelas em qualquer esquema.
CREATE PROCEDURE	Criar uma procedure, função ou pacote em seu próprio esquema.
CREATE ANY PROCEDURE	Criar uma procedure, função ou pacote em qualquer esquema.
...	...

Privilégios de sistema

Concedendo privilégios de sistema

- Privilégios são concedidos aos usuários utilizando-se o comando GRANT. O exemplo a seguir apresenta o comando para conceder ao usuário FULANO os privilégios necessários para criar tabelas e visões em seu próprio esquema:

```
GRANT CREATE TABLE, CREATE VIEW TO FULANO;
```

- Privilégios também podem ser concedidos a todos os usuários do banco de dados. Neste caso, os privilégios devem ser concedidos a um grupo especial denominado PUBLIC:

```
GRANT CREATE TABLE TO PUBLIC;
```

Privilégios de sistema

Concedendo privilégios de sistema

- Pode-se também conceder privilégios a determinado usuário e permitir que ele, por sua vez, também conceda para outros os mesmos privilégios que está recebendo. Para que isto seja possível, deve-se acrescentar a opção WITH ADMIN OPTION ao comando GRANT:

```
GRANT CREATE TABLE TO FULANO WITH ADMIN OPTION;
```

- Se a permissão do usuário FULANO para conceder seus privilégios a outros for revogada, os usuários aos quais ele concedeu os privilégios continuarão a retê-los (não serão revogados).

Privilégios de sistema

Revogando privilégios de sistema

- Os privilégios concedidos aos usuários podem ser revogados. Utiliza-se para isso o comando REVOKE. O exemplo a seguir apresenta o comando para revogar o privilégio CREATE VIEW concedido anteriormente ao usuário FULANO.

```
REVOKE CREATE VIEW FROM FULANO;
```

Privilégios de sistema

Visões de dicionário de dados de privilégios de sistema

- O dicionário de dados apresenta algumas visões que podem ser consultadas para obter-se os privilégios concedidos aos usuários do banco de dados.

- O DBA poderá, por exemplo, listar os privilégios de sistema atribuídos diretamente para o usuário SCOTT através da seguinte consulta ao dicionário de dados:

```
SELECT * FROM DBA_SYS_PRIVS WHERE GRANTEE = 'SCOTT';
```

Privilégios de sistema

Visões de dicionário de dados de privilégios de sistema

VISÃO	DESCRIÇÃO
DBA_SYS_PRIVS	Privilégios de sistema atribuídos a usuários e papéis.
SESSION_PRIVS	Privilégios de sistema concedidos ao usuário na sessão atual diretamente ou através de um papel.
ROLE_SYS_PRIVS	Privilégios de sistema concedidos ao usuário na sessão atual através de um papel.

Administração de Banco de Dados

Aula 14

Prof. Marcos Alexandruk

Aula 14

Privilégios de objetos

Privilégios de objetos

- Privilégio de objetos é o direito de realizar um tipo específico de ação sobre determinados objetos de um banco de dados (por exemplo: tabelas, visões e procedures) que não fazem parte do esquema do usuário.
- Como ocorre com os privilégios de sistema, para conceder privilégios sobre objetos de banco de dados utiliza-se o comando GRANT e para revogá-los utiliza-se o comando REVOKE.
- Os privilégios de objetos podem ser concedidos a todos os usuários do banco de dados através do grupo especial PUBLIC.
- Pode-se também conceder privilégios de objetos a determinado usuário e permitir que ele, por sua vez, também conceda-os para outros. Para que isto seja possível deve-se acrescentar a opção WITH GRANT OPTION ao comando GRANT.

```
GRANT SELECT ON NOME_TABELA TO FULANO WITH GRANT OPTION;
```

- Porém diferente do que acontece quando se concede permissões de sistema, se os privilégios de objetos do usuário forem revogados, serão também revogados os privilégios de todos os usuários da cadeia, isto é, que receberam seus privilégios através deste usuário (FULANO no exemplo acima).

Privilégios de objetos

Privilégios de tabela

- Os privilégios de tabelas podem ser concedidos para operações de DDL (Data Definition Language) e de DML (Data Manipulation Language).
  - As operações de DDL incluem adicionar, modificar ou descartar colunas das tabelas ou ainda criar índices nas tabelas.
  - As operações de DML incluem os comandos SELECT, INSERT, UPDATE e DELETE. É possível restringir os privilégios a determinadas colunas das tabelas.

```
GRANT UPDATE (ID, NOME) ON FULANO.FUNCIONARIO TO SCOTT;
```

- O privilégio concedido anteriormente poderá ser revogado a qualquer momento utilizando-se o comando REVOKE:

```
REVOKE UPDATE ON FULANO.FUNCIONARIO FROM SCOTT;
```

Privilégios de objetos

Revogando privilégios de sistema

- Os privilégios concedidos aos usuários podem ser revogados. Utiliza-se para isso o comando REVOKE. O exemplo a seguir apresenta o comando para revogar o privilégio CREATE VIEW concedido anteriormente ao usuário FULANO.

```
REVOKE CREATE VIEW FROM FULANO;
```

Privilégios de objetos

Visões de dicionário de dados de privilégios de tabela

- O dicionário de dados apresenta algumas visões que podem ser consultadas para obter-se os privilégios de objetos concedidos aos usuários do banco de dados.

- A consulta ao dicionário de dados a seguir apresenta os privilégios de tabela concedidos diretamente ao usuário SCOTT:

```
SELECT * FROM DBA_TAB_PRIVS WHERE GRANTEE = 'SCOTT';
```

Privilégios de objetos

Visões de dicionário de dados de privilégios de objetos

VISÃO	DESCRIÇÃO
DBA_TAB_PRIVS	Privilégios de tabelas concedidos a usuários e papéis.
DBA_COL_PRIVS	Privilégios de colunas concedidos ao usuário na sessão atual diretamente ou através de um papel.
SESSION_PRIVS	Privilégios de sistema concedidos ao usuário na sessão atual diretamente ou através de um papel.
ROLE_TAB_PRIVS	Privilégios de tabelas concedidos ao usuário na sessão atual através de um papel.

Administração de Banco de Dados

Aula 15

Prof. Marcos Alexandruk

Aula 15

Gerenciamento de papéis (roles)

Gerenciamento de papéis (roles)

- Papéis ou roles são grupos identificados de privilégios que podem incluir tanto privilégios de sistema como privilégios de objetos.
- A utilização de papéis facilita a administração dos privilégios concedidos aos usuários do banco de dados. Pois, em vez de conceder diversos privilégios individualmente aos usuários, é possível concedê-los a um papel e este, por sua vez, ser concedido aos usuários.
- Caso seja necessária alguma alteração, esta poderá ser feita no papel e, consequentemente, os privilégios de todos os usuários que utilizam este papel serão automaticamente alterados. Isto pode reduzir significativamente os números de comandos GRANT e REVOKE necessários para a administração dos privilégios dos usuários do banco de dados.

Gerenciamento de papéis (roles)

Papéis predefinidos

- A tabela a seguir apresenta alguns dos principais papéis predefinidos:

PAPEL	PRIVILÉGIO
CONNECT	ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW.
RESOURCE	CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE.
DBA	Todos os privilégios de sistema WITH ADMIN OPTION.
SELECT_CATALOG_ROLE	Privilégio SELECT nos objetos do dicionário de dados.
EXP_FULL_DATABASE	Privilégio para exportar todos os objetos do banco de dados.
IMP_FULL_DATABASE	Privilégio para importar todos os objetos do banco de dados.

Gerenciamento de papéis (roles)

Criando um papel

- Para criar um novo papel (role) deve-se utilizar o comando CREATE ROLE. Porém, é necessário possuir o privilégio de sistema CREATE ROLE que geralmente é concedido apenas aos administradores do banco de dados.

```
CREATE ROLE TESTE;
```

Descartando um papel

- Para descartar um papel deve-se utilizar o comando DROP ROLE, conforme o exemplo apresentado a seguir.

```
DROP ROLE TESTE;
```

Gerenciamento de papéis (roles)

Concedendo privilégios a um papel

- Privilégios são concedidos a um papel da mesma forma que seriam concedidos a um usuário do banco de dados, através do comando GRANT.

- Exemplo: conceder um privilégio de objeto na tabela FUNCIONARIOS ao papel GERENTE_RH. (Nota: o papel GERENTE_RH deve ser criado antes.)

```
GRANT SELECT ON FUNCIONARIOS TO GERENTE_RH;
```

- Exemplo: conceder um privilégio de sistema ao papel GERENTE_RH.

```
GRANT CREATE TRIGGER TO GERENTE_RH;
```

Gerenciamento de papéis (roles)

Atribuindo um papel a um usuário

- Para atribuir um papel a determinado usuário do banco de dados deve-se utilizar o comando GRANT. No exemplo a seguir observa-se a atribuição do papel GERENTE_RH ao usuário FULANO.

```
GRANT GERENTE_RH TO FULANO;
```

- Caso sejam concedidos outros privilégios ao papel GERENTE_RH estes serão imediatamente atribuídos ao usuário FULANO.

Gerenciamento de papéis (roles)

Atribuindo um papel a outro papel

- Papéis podem também ser atribuídos a outros papéis permitindo assim que o DBA tenha a sua disposição uma hierarquia de papéis.
- No exemplo a seguir, em vez de atribuir-se privilégios de objetos individuais ao papel TODOS_DEPT, preferiu-se atribuir os papéis MM_DEPT, RH_DEPT, FI_DEPT e SD_DEPT ao papel TODOS_DEPT.

```
GRANT MM_DEPT, RH_DEPT, FI_DEPT, SD_DEPT TO TODOS_DEPT;
```

- Portanto, o papel TODOS_DEPT poderia, por exemplo, ser atribuído ao presidente da empresa e este teria acesso às tabelas de todos os departamentos.

```
GRANT TODOS_DEPT TO USUARIO_PRESIDENTE;
```

- O papel TODOS_DEPT poderia ter também outros privilégios de sistema ou de objetos que não seriam atribuídos aos outros (MM_DEPT, RH_DEPT, FI_DEPT e SD_DEPT).

Gerenciamento de papéis (roles)

Revogando um papel

- Revoga-se um papel através do comando REVOKE:

```
REVOKE GERENTE_RH FROM FULANO;
```

- Caso outros papéis atribuídos ao usuário FULANO tiverem alguns dos privilégios concedidos ao papel GERENTE_RH, o usuário (FULANO) continuará a retê-los até que sejam explicitamente revogados.

Gerenciamento de papéis (roles)

Ativando um papel protegido por senha

- O DBA poderá atribuir uma senha a um papel, aumentando com esta medida a segurança.

```
CREATE ROLE TESTE_SENHA IDENTIFIED BY ABC123;
```

- O papel TESTE_SENHA deve ser concedido normalmente através do comando GRANT.

```
GRANT TESTE_SENHA TO FULANO;
```

- Quando o usuário FULANO conectar-se ao banco de dados deverá fornecer o nome do papel e a sua respectiva senha para que possa "receber" os privilégios.

```
SET ROLE TESTE_SENHA IDENTIFIED BY ABC123;
```

Gerenciamento de papéis (roles)

Visões de dicionário de dados referentes aos papéis

VISÃO	DESCRIÇÃO
DBA_ROLES	Apresenta todos os papéis e se eles requerem senha.
DBA_ROLE_PRIVS	Apresenta os papéis concedidos a outros usuários ou a outros papéis.
ROLE_ROLE_PRIVS	Apresenta os papéis concedidos a outros papéis.
ROLE_SYS_PRIVS	Apresenta os privilégios de sistema que foram concedidos aos papéis.
ROLE_TAB_PRIVS	Apresenta os privilégios de tabelas e colunas de tabelas que foram concedidos aos papéis.
SESSION_ROLES	Apresenta os papéis que estão em efeito na sessão atual.

Administração de Banco de Dados

Aula 16

Prof. Marcos Alexandruk

Aula 16

Gerenciamento de índices

Gerenciamento de índices

Índices

- **Índices são estruturas associadas a tabelas e clusters que permitem executar mais rapidamente consultas.**
- **Assim como o índice em um livro ajuda a localizar rapidamente as informações, um índice de banco de dados Oracle fornece um caminho de acesso mais rápido aos dados de tabela.**
- **Índices podem ser utilizados sem que seja necessário reescrever qualquer consulta.**

Índices

- O Oracle dispõe de vários tipos de índices, específicos para cada tipo de tabela, método de acesso ou ambiente de aplicação.
- Os principais tipos de índices são:
  - Índices únicos
  - Índices não únicos
  - Índices de chave invertida
  - Índices baseados em funções
  - Índices de bitmap

Gerenciamento de índices

Índices

- **Os índices são lógica e fisicamente independentes dos dados das tabelas associadas, formam estruturas independentes e, por isso, necessitam de espaço de armazenamento.**
- **Pode-se criar ou descartar um índice sem afetar as tabelas base, aplicativos de banco de dados ou outros índices.**

Diretrizes para criação de índices

- **Crie um índice se frequentemente for necessário recuperar menos de 15% das linhas de uma tabela grande (geralmente com milhares de linhas).**
- **Crie um índice para melhorar o desempenho em junções de várias tabelas. Neste caso deve-se indexar as colunas usadas para as junções.**
- **Tabelas pequenas não exigem índices. *(Se uma consulta estiver demorando muito tempo, deve-se verificar o tamanho da tabela. É possível que ela tenha crescido e seja necessário criar um ou a mais índices, conforme destacado no primeiro item.)***

Nota: Chaves primárias e únicas automaticamente já têm índices, mas pode ser necessário criar um índice em uma chave estrangeira.

Quantidade de índices por tabela

- **Uma tabela pode ter qualquer quantidade de índices. No entanto, quanto maior a quantidade de índices em uma tabela, maior será a sobrecarga para atualização dos índices.**
- **Quando linhas são inseridas ou excluídas da tabela ou quando uma coluna é atualizada, todos os índices que contêm a coluna devem ser atualizados.**
- **Há uma compensação entre a velocidade de recuperação de dados de e velocidade de atualização da tabela. Por exemplo, se uma tabela é principalmente para consultas, sem que ocorram muitas alterações, uma quantidade maior de índices pode ser útil; mas se uma tabela é atualizada com muita frequência, é preferível que tenha menos índices.**

Descartar índices

- Deve-se considerar a hipótese de descartar índices nas seguintes situações:
- Se o índice não acelerar as consultas. (A tabela pode ser muito pequena, ou pode haver muitas linhas na tabela, mas poucas entradas de índice.)
- As consultas realizadas através das aplicações não utilizam o índice.

- Para descartar um índice utiliza-se o comando **DROP INDEX**:

```
DROP INDEX NOME_DO_INDICE;
```

Gerenciamento de índices

Monitorar a frequência de utilização de um índice

- **O Oracle fornece um meio para acompanhamento da utilização de índices para determinar se eles estão sendo usados. Se um índice não estiver sendo usado, então ele pode ser descartado, eliminando a sobrecarga desnecessária por ele gerada.**
- **Para iniciar o monitoramento de um índice, deve-se utilizar a instrução a seguir:**

```
ALTER INDEX NOME_DO_INDICE MONITORING USAGE;
```

- **Quando for necessário parar o monitoramento, utiliza-se a seguinte instrução:**

```
ALTER INDEX NOME_DO_INDICE NOMONITORING USAGE;
```

Gerenciamento de índices

Monitorar a frequência de utilização de um índice

- A visão **V$OBJECT_USAGE** pode ser consultada para o índice que está sendo monitorado para ver se o mesmo tem sido usado:
- `SELECT * FROM V$OBJECT_USAGE;`
- O modo de exibição contém uma coluna denominada **USED**, cujo valor é YES ou NO, dependendo se o índice tem sido usado dentro do período de tempo no qual está sendo monitorado.
- A exibição também contém os tempos de início e fim do período de acompanhamento e outra coluna para indicar se o monitoramento está ativo no momento.

Gerenciamento de índices

Monitorar o uso de espaço de um índice

- Cada vez que for especificado o uso de monitoramento, o modo de exibição da visão V$OBJECT_USAGE é redefinido para o índice especificado. As informações de uso anterior são descartadas e uma nova hora de início é gravada. Quando for especificado o uso de NOMONITORING, o monitoramento não é mais realizado, e o tempo final é registrado para o período de monitoramento.
- Até que a próxima instrução ALTER INDEX ... MONITORING USAGE seja emitida, os dados da visão não serão alterados.

Gerenciamento de índices

Monitorar o uso de espaço de um índice

- A porcentagem de uso de espaço do índice varia de acordo com a frequência com que as chaves de índice são inseridas, atualizadas ou excluídas.
- Deve-se monitorar a eficiência do uso do espaço em intervalos regulares analisando a estrutura do índice:

```
ANALYZE INDEX NOME_DO_INDICE VALIDATE STRUCTURE;
```

- A seguir, deve-se consultar a visão **INDEX_STATS**:

```
SELECT PCT_USED FROM INDEX_STATS
WHERE NAME = 'NOME_DO_INDICE';
```

- A coluna **PCT_USED** indica a porcentagem de espaço alocado para o índice. Quando o espaço não é mais utilizado eficientemente (PCT_USED < 70) é recomendável a reconstrução do índice:

```
ALTER INDEX NOME_DO_INDICE REBUILD;
```

Gerenciamento de índices

Monitorar o uso de espaço de um índice

- **É possível desenvolver um histórico de eficiência média de uso de espaço para um índice executando a seguinte sequência de operações várias vezes:**
  - **Análise das estatísticas;**
  - **Validação do índice;**
  - **Verificação do PCT_USED;**
  - **Eliminação e reconstrução do índice.**

Gerenciamento de índices

ATIVIDADE

```
-- CRIAR A TABELA CLIENTE:
CREATE TABLE CLIENTE (
CODIGO NUMBER(4),
NOME VARCHAR2(15),
CONSTRAINT CLIENTE_PK PRIMARY KEY(CODIGO));

-- TENTAR CRIAR UM ÍNDICE NA COLUNA CODIGO. O QUE ACONTECEU? POR QUE?

-- CRIAR UM ÍNDICE DENOMINADO CLIENTE_IDX NA COLUNA NOME.

-- CONSULTAR OS ÍNDICES DA TABELA CLIENTE.

-- INICIAR O MONITORAMENTO DO ÍNDICE CLIENTE_IDX.

-- CONSULTAR A VISÃO V$OBJECT_USAGE PARA VERIFICAR SE O USO ÍNDICE ESTÁ
-- SENDO MONITORADO E SE O ÍNDICE FOI UTILIZADO.

-- INSERIR AS SEGUINTES LINHAS NA TABELA CLIENTE:
INSERT INTO CLIENTE (CODIGO, NOME) VALUES(1001,'ANTONIO');
INSERT INTO CLIENTE (CODIGO, NOME) VALUES(1002,'BEATRIZ');
INSERT INTO CLIENTE (CODIGO, NOME) VALUES(1003,'CLAUDIO');
COMMIT;
```

Gerenciamento de índices

ATIVIDADE

```
-- REALIZAR A SEGUINTE CONSULTA:
SELECT CODIGO FROM CLIENTE WHERE NOME = 'BEATRIZ';

-- CONSULTAR A VISÃO V$OBJECT_USAGE PARA VERIFICAR SE O USO ÍNDICE ESTÁ
-- SENDO MONITORADO E SE O ÍNDICE FOI UTILIZADO.

-- MONITORAR A EFICIÊNCIA DO USO DO ESPAÇO DO ÍNDICE CLIENTE_IDX ANALISANDO
-- A SUA ESTRUTURA.

-- CONSULTAR A VISÃO INDEX_STATS PARA OBTER O VALOR DA COLUNA PCT_USED.

-- O QUE É RECOMENDÁVEL FAZER CASO O VALOR DE PCT_USED SEJA INFERIOR A 70?

-- PARAR O MONITORAMENTO DO ÍNDICE CLIENTE_IDX.

-- CONSULTAR A VISÃO V$OBJECT_USAGE PARA VERIFICAR SE O USO ÍNDICE ESTÁ
-- SENDO MONITORADO.

-- RECONSTRUIR O ÍNDICE CLIENTE_IDX.

-- ELIMINAR O ÍNDICE CLIENTE_IDX.
```

Administração de Banco de Dados

Aula 17

Prof. Marcos Alexandruk

Aula 17

Backup e recovery

Exportação e Importação

Exportação e Importação

O Oracle apresenta três métodos para fazer backup de um banco de dados:

- exportação/importação (backup lógico);
- backup off-line (backup físico);
- backup on-line (backup físico).

Nesta aula serão apresentados os detalhes necessários para realização do backup lógico de dados (exportação/importação).

No entanto, uma boa estratégia de backup para um banco de dados envolve tanto backups lógicos como físicos.

Exportação e Importação

O backup lógico envolve a leitura de um conjunto de registros do banco de dados e a gravação destes registros em um determinado arquivo. A leitura dos registros ocorre independentemente das suas localizações físicas.

Apesar de utilitários mais antigos como o Import e o Export ainda estarem disponíveis, é recomendável a utilização do utilitário Data Pump Export (disponível a partir da versão 10g) para realização do backup e do Data Pump Import para recuperação dos mesmos.

O Data Pump Export consulta o banco de dados, inclusive o dicionário de dados, e grava os registros em um arquivo padrão XML chamado "arquivo dump de exportação". Podem ser exportados para este arquivo todo o banco de dados, determinados usuários, tablespaces ou tabelas.

Exportação e Importação

Após a exportação dos dados através do Data Pump Export, estes poderão ser recuperados através do utilitário Data Pump Import. É possível recuperar todos os dados ou apenas uma parte dos dados exportados.

Caso seja importado todo o arquivo gerado a partir de uma exportação completa, então todos os objetos do banco (tablespaces, arquivos de dados e usuários) serão criados durante a importação.

Por outro lado, caso sejam importados apenas uma parte dos dados, os tablespaces, arquivos de dados e usuários, deverão ser devidamente configurados antes da importação.

Utilizando o DATA PUMP EXPORT

O exemplo que será apresentado tem como base a exportação dos objetos do usuário FULANO que deverá ser criado e deverá receber alguns privilégios, conforme segue:

```
CREATE USER FULANO IDENTIFIED BY ABC123;

GRANT CONNECT, RESOURCE TO FULANO;
```

Utilizando o DATA PUMP EXPORT

O próximo passo apresenta a criação de uma tabela simples denominada TAB1 para teste. A tabela será criada pelo usuário FULANO:

```
CREATE TABLE TAB1 (
COL1 NUMBER(2));
```

A seguir serão inseridas duas linhas na tabela TAB1:

```
INSERT INTO TAB1 VALUES (1);
INSERT INTO TAB1 VALUES (2);
COMMIT;
```

Utilizando o DATA PUMP EXPORT

Cria-se na sequência um diretório para armazenar os arquivos de dados e de controle que o Data Pump irá criar (na exportação) e ler (na importação).

É necessário que o usuário que emitirá o comando tenha o privilégio de sistema CREATE ANY DIRECTORY.

Portanto, o DBA criará um diretório denominado TESTE e concederá os privilégios de leitura e escrita ao usuário FULANO:

```
CREATE DIRECTORY TESTE AS 'C:\TESTE';

GRANT READ, WRITE ON DIRECTORY TESTE TO FULANO;
```

Utilizando o DATA PUMP EXPORT

O utilitário EXPDP serve como interface com o Data Dump. A tabela seguir apresenta alguns dos principais parâmetros utilizados com o EXPDP.

PARÂMETRO	DESCRIÇÃO
DIRECTORY	Especifica o diretório de destino para os arquivos de log e para os arquivos de dump.
DUMPFILE	Especifica os nomes dos arquivos de dump.
CONTENT	Especifica o que será exportado: DATA_ONLY, METADATA_ONLY ou ALL.

Utilizando o DATA PUMP EXPORT

Para utilizar o EXPDP deve-se digitar EXPDP em uma janela de prompt de comando do sistema operacional.

O exemplo abaixo apresenta a utilização do EXPDP pelo usuário FULANO.

Foi passado o valor METADATA_ONLY ao parâmetro CONTENT. Portanto, apenas a estrutura da tabela TAB1 (único objeto do usuário FULANO) será exportada.

Para exportação da estrutura e dos dados da tabela o valor deste parâmetro (CONTENT) deveria ser alterado para ALL.

```
C:\> EXPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=METADATA_ONLY
```

Será solicitado o nome do usuário e sua senha. Após fornecê-los o EXPDP passará à criação do arquivo de log e do arquivo dump no diretório TESTE.

Utilizando o DATA PUMP EXPORT

A etapa seguinte envolverá a importação dos dados com base na exportação que acaba de ocorrer.

Porém, antes de executar o utilitário de importação, será eliminada a tabela TAB1 criada no exemplo apresentado.

```
DROP TABLE TAB1;
```

Utilizando o DATA PUMP IMPORT

Deve-se utilizar o Data Pump Import para importação de arquivos exportados via Data Pump Export.

Para importação dos objetos do usuário FULANO, deve-se utilizar o comando:

```
C:\> IMPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=METADATA_ONLY
```

Voltando ao SQL Plus ao utilizar o comando DESCRIBE (ou sua abreviação DESC) pode-se observar que o objeto anteriormente excluído (a tabela TAB1) foi recuperado.

```
DESC TAB1
```

Exportação e Importação

No exemplo apresentado apenas a estrutura da tabela foi exportada.

Para que os dados também fossem exportados seria necessário passar o valor ALL ao parâmetro CONTENT.

Caso a tabela a ser importada já existir e a opção CONTENT receber o valor METADATA_ONLY esta será simplesmente ignorada.

Porém, se a tabela já existir e a opção CONTENT receber o valor DATA_ONLY os "novos" dados serão acrescentados aos dados já existentes na tabela. Para alterar isso, deve-se utilizar a opção TABLE_EXISTS_ACTION atribuindo-lhe um dos seguintes valores:

- **SKIP: Conserva apenas os valores que já se encontravam na tabela, nada será importado;**
- **APPEND: Acrescenta os "novos" valores aos que já se encontram na tabela;**
- **TRUNCATE: Corta (elimina) os valores da tabela e insere os dados contidos no arquivo de importação;**
- **REPLACE: Substitui toda a tabela (estrutura e dados). Para esta opção deve ser especificado o valor ALL ao parâmetro CONTENT na exportação e na importação.**

Exportação e Importação

TAB1
COL1
1
2

```
EXPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=ALL
```

TAB1
COL1
1
2
3

```
IMPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=ALL TABLE_EXISTS_ACTION=SKIP
```

TAB1
COL1
1
2
3

Exportação e Importação

TAB1
COL1
1
2

```
EXPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=ALL
```

TAB1
COL1
1
2
3

```
IMPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=ALL TABLE_EXISTS_ACTION=APPEND
```

TAB1
COL1
1
2
3
1
2

Exportação e Importação

TAB1
COL1
1
2

```
EXPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=ALL
```

TAB1
COL1
2
3

```
IMPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=ALL TABLE_EXISTS_ACTION=TRUNCATE
```

TAB1
COL1
1
2

Exportação e Importação

TAB1
COL1
1
2

```
EXPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=ALL
```

TAB1	
COL1	COL2
1	3
2	4

```
IMPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=ALL TABLE_EXISTS_ACTION=REPLACE
```

TAB1
COL1
1
2

Utilizando as opções EXCLUDE, INCLUDE e QUERY

É possível excluir ou incluir conjuntos de tabelas ou ainda apenas determinadas linhas de tabelas em uma exportação utilizando as opções EXCLUDE, INCLUDE e QUERY.

O exemplo a seguir exclui a tabela TAB1 do arquivo de exportação (todas as outras tabelas seriam exportadas, exceto TAB1):

```
C:\>EXPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=ALL
EXCLUDE=TABLE:\"=\'TAB1\'\"
```

Caso não seja especificado nenhum nome, todos os objetos do tipo especificado não serão incluídos na exportação.

O exemplo a seguir não importará nenhum índice:

```
C:\>EXPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=ALL
EXCLUDE=INDEX
```

Exportação e Importação

Utilizando as opções EXCLUDE, INCLUDE e QUERY

O próximo exemplo inclui apenas a TAB1 no arquivo de exportação (todas as outras tabelas não seriam exportadas):

```
C:\>EXPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=ALL
INCLUDE=TABLE:\"=\'TAB1\'\"
```

Pode-se exportar apenas determinadas linhas de uma tabela. O exemplo a seguir apresenta a exportação das linhas cujos valores da COL1 (da tabela TAB1) sejam menores que 5:

```
C:\>EXPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=ALL
INCLUDE=TABLE:\"=\'TAB1\'\" QUERY=TAB1:\"WHERE COL1 < 5\"
```

Para importação dos dados utiliza-se, por exemplo, o seguinte comando:

```
C:\>IMPDP DIRECTORY=TESTE DUMPFILE=DATA.DMP CONTENT=ALL
TABLE_EXISTS_ACTION=APPEND
```

Exportação e Importação

ATIVIDADE

```
-- [SYS] Criar um usuário denominado FULANO.

-- [SYS] Conceder privilégios CONNECT e RESOURCE para o usuário FULANO.

-- [FULANO] Criar uma tabela denominada CLIENTE conforme segue:

CREATE TABLE CLIENTE (
CODCLI NUMBER(4),
NOMECLI VARCHAR2(10),
CONSTRAINT CLIENTE_PK PRIMARY KEY (CODCLI));

-- [FULANO] Incluir as seguintes linhas na tabela CLIENTE:

INSERT INTO CLIENTE VALUES (1001,'ANTONIO');
INSERT INTO CLIENTE VALUES (1002,'BEATRIZ');
INSERT INTO CLIENTE VALUES (1003,'CLAUDIO');
INSERT INTO CLIENTE VALUES (1004,'DANIELA');
COMMIT;

-- [SYS] Criar um diretório chamado TESTE

-- [SYS] Conceder privilégio de leitura e escrita no diretório TESTE ao
   usuário FULANO.
```

Exportação e Importação

ATIVIDADE

```
-- [FULANO] Exportar para um arquivo denominado CLIENTE_ESTRUTURA.DMP apenas
   a estrutura da tabela CLIENTE.

-- [FULANO] Exportar para um arquivo denominado CLIENTE_DADOS.DMP apenas os
   dados da tabela CLIENTE.

-- [FULANO] Exportar para um arquivo denominado CLIENTE_COMPLETO.DMP apenas
   a estrutura e os dados da tabela CLIENTE.

-- [FULANO] Eliminar a tabela CLIENTE.

-- [FULANO] Importar o arquivo CLIENTE_COMPLETO.DMP para o banco de dados.

-- [FULANO] Verificar se a estrutura e os dados da tabela CLIENTE foram
   importados.

-- [FULANO] Eliminar (novamente) a tabela CLIENTE.
```

Exportação e Importação

ATIVIDADE

```
-- [FULANO] Importar o arquivo CLIENTE_ESTRUTURA.DMP para o banco de dados.

-- [FULANO] Verificar se a estrutura da tabela CLIENTE foi importada.

-- [FULANO] Importar o arquivo CLIENTE_DADOS.DMP para o banco de dados.

-- [FULANO] Verificar se os dados da tabela CLIENTE foram importados.
```

Administração de Banco de Dados

Aula 18

Prof. Marcos Alexandruk

Aula 18

Backup físico: off-line

Backups Físicos

Os backups físicos podem ser realizados com utilitários dos principais sistemas operacionais:

- **Windows (copy ou winzip32);**
- **Unix (cp ou tar).**

O Oracle disponibiliza dois tipos de backups físicos:

- **Off-line (também chamado fechado ou a frio);**
- **On-line (também chamado aberto ou a quente).**

Backups Físicos

Opções disponíveis:

- **Integral ou parcial**
  - **Integral: conjunto inteiro de arquivos de dados e de controle;**
  - **Parcial: apenas alguns arquivos de dados e/ou de controle.**
- **Total ou incremental**
  - **Total: autocontido, utilizável por si próprio;**
  - **Incremental: apenas os blocos que foram alterados desde o último backup.** (Realizados somente através do RMAN)

Backups off-line

Requisitos:

O banco deverá ser "desligado" através de um destes tipos de shutdown:

- **shutdown normal;**
- **shutdown immediate;**
- **shutdown transactional.**

Não se deve realizar um backup off-line após um shutdown abort para que não ocorram inconsistências nos dados.

Caso seja necessário fazer um shutdown abort, para realização de um backup off-line consistente, deve-se reiniciar o banco de dados e realizar novamente um shutdown com uma das três opções acima.

Backups off-line

Deve-se fazer backup de todos os arquivos dos seguintes grupos:

- **Arquivos de dados;**
- **Arquivos de controle;**
- **Arquivos de logs de redo on-line;**
- **Arquivo init.ora e spfile (se utilizado).**

Backups off-line

1. Criar o diretório para arquivar os backups (Exemplo: C:\BACKUP)

2. Criar (no SQL*Plus) o shell script para executar o backup:

```
SPOOL C:\BACKUP\BACKUP_OFFLINE.BAT
SELECT 'COPY '||NAME||' C:\BACKUP' FROM V$DATAFILE
UNION ALL
SELECT 'COPY '||NAME||' C:\BACKUP' FROM V$CONTROLFILE
UNION ALL
SELECT 'COPY '||MEMBER||' C:\BACKUP' FROM V$LOGFILE;
CREATE PFILE='C:\BACKUP\SPFILE_BACKUP.ORA' FROM SPFILE;
SPOOL OFF
```

Para criar o script é preciso fazer login como sysdba (ou equivalente).

Backups off-line

Arquivo gerado:

```
SQL> SELECT 'COPY '||NAME||' C:\BACKUP' FROM V$DATAFILE
  2  UNION ALL
  3  SELECT 'COPY '||NAME||' C:\BACKUP' FROM V$CONTROLFILE
  4  UNION ALL
  5  SELECT 'COPY '||MEMBER||' C:\BACKUP' FROM V$LOGFILE;

'COPY'||NAME||'C:\BACKUP'
--------------------------------------------------------------------------------
COPY C:\APP\MASTER\ORADATA\ORCL\SYSTEM01.DBF C:\BACKUP
COPY C:\APP\MASTER\ORADATA\ORCL\SYSAUX01.DBF C:\BACKUP
COPY C:\APP\MASTER\ORADATA\ORCL\UNDOTBS01.DBF C:\BACKUP
COPY C:\APP\MASTER\ORADATA\ORCL\USERS01.DBF C:\BACKUP
COPY C:\APP\MASTER\ORADATA\ORCL\EXAMPLE01.DBF C:\BACKUP
COPY C:\APP\MASTER\ORADATA\ORCL\CONTROL01.CTL C:\BACKUP
COPY C:\APP\MASTER\FLASH_RECOVERY_AREA\ORCL\CONTROL02.CTL C:\BACKUP
COPY C:\APP\MASTER\ORADATA\ORCL\REDO03.LOG C:\BACKUP
COPY C:\APP\MASTER\ORADATA\ORCL\REDO02.LOG C:\BACKUP
COPY C:\APP\MASTER\ORADATA\ORCL\REDO01.LOG C:\BACKUP

10 linhas selecionadas.

SQL> CREATE PFILE='C:\BACKUP\SPFILE_BACKUP.ORA' FROM SPFILE;

Arquivo criado.

SQL> SPOOL OFF
```

Backups off-line

Para executar o backup é preciso realizar o shutdown no banco de dados (como sysdba):

```
SHUTDOWN IMMEDIATE
```

O backup é realizado através do sistema operacional com a 'execução' do script anteriormente criado (BACKUP_OFFLINE.BAT).

Administração de Banco de Dados

Aula 19

Prof. Marcos Alexandruk

Aula 19

Backup físico: on-line

Backups on-line

Backups on-line (também denominados abertos ou "a quente") são aqueles realizados enquanto o banco de dados Oracle encontra-se aberto.

Não é possível fazer um backup on-line no modo NOARCHIVELOG.

Neste modo todas as transações encerradas com COMMIT, mas que ainda não foram gravadas nos arquivos de dados ficam disponíveis nos arquivos de redo log online.

Por outro lado, quando o modo ARCHIVELOG está ativado, o processo em background ARCn (Archiver Process) faz uma cópia de cada arquivo de redo log antes de sobrescrevê-lo.

Backups on-line

VERIFICANDO O ARCHIVELOG MODE

Deve-se realizar a consulta a seguir para verificar se o archivelog mode está ativado:

```
SELECT LOG_MODE FROM V$DATABASE;
```

Se o archivelog mode estiver desativado a consulta retornará o seguinte:

```
LOG_MODE
------------
NOARCHIVELOG
```

Backups on-line

ATIVANDO O ARCHIVELOG MODE

Antes de ativar o archivelog mode é preciso parar o banco:

```
SHUTDOWN IMMEDIATE
```

A seguir, deve-se subir o banco para o estado mount:

```
STARTUP MOUNT;
```

Para alterar archivelog mode deve-se utilizar o seguinte comando:

```
ALTER DATABASE ARCHIVELOG;
```

O próximo passo será abrir o banco de dados:

```
ALTER DATABASE OPEN;
```

Backups on-line

BACKUP DOS ARQUIVOS DE CONTROLE

Alternativa 1:

```
ALTER DATABASE BACKUP CONTROLFILE TO '/BACKUP/CONTROL1.BKP';
```

Realiza uma cópia binária do arquivo de controle, isto é, uma cópia idêntica byte-a-byte do arquivo de controle.

Alternativa 2:

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE AS '/BACKUP/CONTROL2.BKP';
```

Cria um novo arquivo de controle, um arquivo ASCII, que poderá ser executado enquanto a instância estiver no modo NOMOUNT para criar um novo arquivo de controle com conteúdo idêntico ao original.

Backups on-line

BACKUP DOS ARQUIVOS DE UM TABLESPACE

Determinar quais são os arquivos associados ao tablespace (ex. SYSAUX):

```
SELECT FILE_NAME FROM DBA_DATA_FILES
WHERE TABLESPACE_NAME = 'SYSAUX';
```

Colocar o tablespace (ex. SYSAUX) no modo backup:

```
ALTER TABLESPACE SYSAUX BEGIN BACKUP;
```

Fazer o backup do(s) arquivo(s) de dados utilizando um utilitário do sistema operacional. (No caso do Microsoft Windows pode-se utilizar o utilitário copy.)

```
C:\>COPY C:\APP\MASTER\ORADATA\ORCL\SYSAUX01.DBF C:\BACKUP\SYSAUX01.DBF;
```

Encerrar o modo backup no tablespace SYSAUX:

```
ALTER TABLESPACE SYSAUX END BACKUP;
```

Administração de Banco de Dados

Aula 20

Prof. Marcos Alexandruk

Aula 20

RMAN (Recovery Manager)

RMAN (Recovery Manager)

O RMAN é muito mais do que um simples utilitário que pode ser utilizado do lado cliente para realizar backups.

Apresenta muitos recursos que não estão disponíveis em outros métodos de backup.

RMAN (Recovery Manager)

EFETUANDO BACKUP COM O RMAN

Para efetuar um backup com o RAMAN deve-se primeiramente verificar se o ARCHIVELOG está habilitado:

```
SELECT LOG_MODE FROM V$DATABASE;
```

Se o archivelog mode estiver desativado a consulta retornará o seguinte:

```
LOG_MODE
------------
NOARCHIVELOG
```

RMAN (Recovery Manager)

ATIVANDO O ARCHIVELOG MODE

Antes de ativar o archivelog mode é preciso parar o banco:

```
SHUTDOWN IMMEDIATE
```

A seguir, deve-se subir o banco para o estado mount:

```
STARTUP MOUNT;
```

Para alterar archivelog mode deve-se utilizar o seguinte comando:

```
ALTER DATABASE ARCHIVELOG;
```

O próximo passo será abrir o banco de dados:

```
ALTER DATABASE OPEN;
```

RMAN (Recovery Manager)

INICIANDO O RMAN

A seguir, deve-se entrar no prompt (do sistema operacional) e digitar o seguinte comando:

```
RMAN TARGET SYS/SENHA_DO_SYS
```

O comando anterior produzirá a seguinte saída:

```
Conectado ao banco de dados de destino: ORCL (DBID=1178846893)
```

RMAN (Recovery Manager)

BACKUP DOS ARQUIVOS DE DADOS

Criar o diretório para backup dos arquivos.

Configurar o diretório para backup:

```
RMAN> CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT 'C:\BACKUP\%U';
```

Realizar o backup dos arquivos de dados:

```
RMAN> BACKUP DATABASE PLUS ARCHIVELOG;
```

RMAN (Recovery Manager)

BACKUP DO ARQUIVO DE CONTROLE

Configurar o diretório para backup do arquivo de controle:

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE
DISK TO 'C:\BACKUP\%F';
```

Realizar o backup do arquivo de controle:

```
RMAN> BACKUP CURRENT CONTROLFILE;
```

É possível consultar os backups realizados através da seguinte consulta:

```
RMAN> LIST BACKUP;
```

RMAN (Recovery Manager)

RESTORE DO ARQUIVO DE CONTROLE

Entrar no utilitário RMAN:

```
RMAN TARGET SYS/SENHA_DO_SYS;
```

Iniciar o banco de dados no modo NOMOUNT:

```
RMAN> STARTUP FORCE NOMOUNT;
```

Fazer o restore do arquivo:

```
RMAN> RESTORE CONTROLFILE FROM 'C:\BACKUP\NOME_ARQ_CONTROLE';
```

Nota: O arquivo de controle foi feito por último, portanto, provavelmente ele deve ser o último arquivo do conjunto de backup.

RMAN (Recovery Manager)

RESTORE DOS ARQUIVOS DE DADOS

Montar a base com o arquivo de controle recuperado:

```
RMAN> STARTUP FORCE MOUNT;
```

Restaurar os arquivos de dados:

```
RMAN> RESTORE DATABASE;
```

Recuperar (recover) a base de dados:

```
RMAN> RECOVER DATABASE;
```

Abrir o banco de dados (com o parâmetro RESETLOGS que cria novos arquivos de REDOLOG):

```
RMAN> ALTER DATABASE OPEN RESETLOGS;
```

RMAN (Recovery Manager)

APAGAR ARQUIVOS DE BACKUP

Excluir os arquivos de backup:

```
RMAN> DELETE BACKUP;
```