

A arquitetura do ORACLE

Visão Geral

O conhecimento da arquitetura interna do ORACLE é de extrema importância para a compreensão das técnicas de otimização do produto. Basicamente, os seus mecanismos de execução são as estruturas de memória e os processos executados em background. Todas as vezes que um banco é inicializado, uma SGA é alocada e os processos são inicializados. A combinação das estruturas de memória na SGA e dos processos em background é chamada de instância ORACLE. Algumas arquiteturas de hardware permitem múltiplos computadores compartilharem os mesmos dados, softwares ou periféricos. Com a opção Parallel Server do ORACLE, podemos tirar proveito dessa característica através da execução de múltiplas instâncias que compartilham um único banco de dados. Assim, os usuários de diversas máquinas podem acessar o mesmo banco de dados com uma melhoria na performance.

SGA A SGA é um grupo de buffers de memória compartilhados que são destinados pelo ORACLE para uma instância. Basicamente é formada pelas estruturas identificadas por shared pool, database buffer cache e redo log buffer cache. Entretanto, em algumas configurações do ORACLE podem existir outras estruturas.

Processos em Background Os processos em background executam tarefas distintas assincronicamente em benefício a todos os usuários de um banco de dados. Não existe uma relação direta entre os processos em background e os processos dos usuários conectados a uma instância ORACLE. Apesar de poderem existir outros em uma instância, o que depende da configuração do ORACLE utilizada, os processos mais conhecidos são o PMON, SMON, DBWR, LGWR, RECO, LCK, CKPT e o ARCH.

Geralmente um banco de dados está associado a somente uma instância. Entretanto, como vimos, em algumas configurações do ORACLE, um banco de dados pode estar associado a mais de uma instância. Assim, precisamos diferenciar os dois conceitos: um banco de dados é formado pelos arquivos fisicamente armazenados em disco enquanto que uma instância é formada pelas estruturas e processos em memória. O banco de dados é permanente, enquanto que uma instância é volátil. Naturalmente, para acessarmos um banco de dados é necessário que uma instância seja inicializada e associada a ele.

Estruturas de Memória

As estruturas de memória são criadas pelo ORACLE e usadas para completar diversas tarefas. Por exemplo, elas são usadas para guardar o código de um programa que está sendo executado e os dados que podem ser compartilhados pelos usuários.

SGA e PGA.

As principais estruturas são a SGA (System Global Area ou Área Global do Sistema) e a PGA (Program Global Area ou Área Global de Programa).

A PGA é o buffer de memória que contém dados e algumas informações de controle de uma sessão de um usuário. A PGA é criada e alocada quando um novo processo é inicializado no servidor. As suas informações dependem da configuração do ORACLE. Assim, existe uma área de memória PGA para cada usuário que está executando seus trabalhos no ORACLE. Dentro da PGA existem três estruturas: uma contendo um espaço para a pilha (para armazenar as variáveis e matrizes), outra contendo dados sobre a sessão do usuário e uma terceira com as informações dos cursores usados. A PGA não é compartilhada entre os usuários; ela é única para cada sessão.

A SGA é uma região de memória compartilhada por todos os usuários e alocada pelo ORACLE. Contém os dados e as informações de controle de uma instância. Ela é alocada quando uma nova instância é inicializada e liberada quando a mesma é finalizada. Os dados na SGA são compartilhados pelos usuários que estiverem conectados ao banco de dados e, para otimizar a performance, as entradas na SGA devem ser as maiores possíveis para guardar a maior quantidade de dados e minimizar o I/O em disco, uma das causas críticas que tornam um banco de dados lento. As informações na SGA estão organizadas em diversos tipos de estruturas de memória, incluindo o buffer do banco de dados e o buffer para recuperação do banco, por exemplo. As estruturas têm tamanho fixo e são criadas durante a inicialização da instância. O grupo de buffers do banco de dados em uma instância são chamados de database buffer cache. Esses buffers podem conter os dados modificados que ainda não foram escritos em disco, para os arquivos de dados apropriados. Desse modo o I/O é minimizado e há uma melhora significativa da performance.

Essa estrutura é compartilhada entre todos os usuários conectados a um banco de dados e os blocos de dados que são armazenados no database buffer cache têm seus tamanhos determinados pelo parâmetro DB_BLOCK_SIZE. O número de blocos em memória é determinado pelo parâmetro DB_BLOCK_BUFFERS.

O conteúdo do database buffer cache é organizado em duas listas: a lista de blocos alterados e a lista dos blocos menos recentemente utilizados (LRU - Least Recently Used). Essa segunda lista contém os blocos livres, aqueles que estão em uso e os blocos alterados. Quando um processo servidor precisa ler dados de um bloco do disco para o database buffer cache, ele pesquisa a LRU para localizar um bloco livre e, quando encontrar um bloco alterado, movimenta-o para a lista de blocos alterados. Esse processo termina quando

um bloco livre é localizado ou quando um número específico de blocos são pesquisados sem encontrar um bloco livre.

Durante uma operação de SELECT, o ORACLE requer que os blocos que contêm a informação desejada esteja em memória. Assim, a lista LRU é pesquisada e, se os blocos não estiverem em memória, o produto efetua as leituras físicas necessárias. Caso o bloco esteja em memória, são efetuadas leituras lógicas. Lembremo-nos de que nenhuma tabela pode ocupar menos de dois blocos de dados: um bloco para o cabeçalho e pelo menos outro bloco de dados.

O redo log buffer cache da SGA armazena todas as alterações feitas em um banco de dados em memória. Todas as entradas redo log neste buffer são escritas nos arquivos redo log, que são usados para a recuperação do banco de dados, se necessário.

A shared pool é uma porção de memória compartilhada que contém as áreas chamadas shared SQL, estruturas de memória compartilhadas que contêm os comandos SQL que estão sendo executados pelos múltiplos usuários conectados a um banco de dados. Essas áreas compartilhadas shared SQL contêm informações como o texto e a forma interpretada dos comandos SQL, a fase de análise dos comandos SQL e seus planos de execução, informações do dicionário de dados e de geradores de números seqüenciais. Uma única área shared SQL pode ser compartilhada por diversas aplicações que usam o mesmo comando definido na área compartilhada de comandos SQL, deixando assim mais área em memória disponível para os outros usuários e melhorando a performance de execução de um comando, já que o plano de execução já está definido e o ORACLE não precisa defini-lo novamente.

A shared pool contém ainda o data dictionary cache, com as informações do dicionário de dados, e o sequence cache, com as informações dos geradores de números seqüenciais. Um cursor é um nome ou ponteiro para a memória associada a um comando específico. Muitas aplicações ORACLE tiram proveito dos cursores.

Processos.

Os processos podem ser vistos como programas que trabalham em memória (em background) e executam outras tarefas específicas para o ORACLE. Um processo é uma forma de controle ou um *mecanismo no sistema operacional que pode executar uma série de passos e normalmente tem sua área particular de memória. Alguns sistemas operacionais usam o termo job ou tarefa.

Existem dois tipos gerais de processos: os processos dos usuários e os processos do próprio ORACLE.

Um processo de usuário é criado e mantido para executar o código da aplicação (por exemplo um programa Pro*C) ou uma ferramenta ORACLE (por exemplo o SQL*Plus). Os processos dos usuários também gerenciam a comunicação com os processos do servidor ORACLE através do program interface.

Os processos ORACLE são chamados por outros processos para executar algumas funções específicas. O produto cria os processos servidores (server process) para controlar as requisições dos processos dos usuários conectados a um banco de dados. Assim, os processos servidores são incumbidos de comunicar-se com os processos dos usuários e interagir com o ORACLE para acessar seus recursos.

Por exemplo, se um usuário pesquisa alguns dados que não estejam no database buffer cache da SGA, o processo servidor lê os dados apropriados dos blocos de dados dos arquivos e os coloca na SGA, para uso dos usuários. Dependendo da configuração do ORACLE, um processo servidor pode ser compartilhado por diversos usuários.

Todos os comandos SQL são processados pelos processos servidores que se utilizam de três fases para o processamento: análise, execução e busca dos dados. O plano de cada comando é armazenado na SGA, nas áreas que contêm comandos SQL a serem compartilhados entre os usuários.

O ORACLE cria um conjunto de processos que rodam em background para cada instância. Esses processos executam diversas tarefas. São eles: DBWR, LGWR, CKPT, SMON, PMON, ARCH, RECO, Dnnn e LCKn.

O processo database writer (DBWR) escreve os blocos modificados do database buffer cache para os arquivos de dados físicos. O DBWR não precisa escrever os dados a cada comando COMMIT, pois é otimizado para minimizar o I/O. Geralmente o DBWR escreve os dados para o disco se muitos dados são lidos para o database buffer cache na SGA e não existe espaço livre para esses novos dados. Os dados menos recentemente usados são escritos para os arquivos de dados em primeiro lugar.

O processo log writer (LGWR) escreve todas as entradas de redo log para o disco. Os dados de redo log são armazenados em memória no redo log buffer cache, na SGA. No momento em que uma transação for efetivada com o comando COMMIT e o redo log buffer estiver preenchido, o LGWR escreve as entradas de redo log nos arquivos redo log apropriados.

A um tempo específico, todos os dados do database buffer cache modificados são escritos em disco pelo processo DBWR; este evento é chamado de checkpoint. O processo checkpoint é responsável para informar ao processo DBWR o momento de gravar os dados em disco. O DBWR também atualiza os arquivos de controle do banco de dados para indicar o mais recente checkpoint. O processo CKPT é opcional; se ele não estiver presente, o LGWR assume sua responsabilidade.

O processo system monitor (SMON) efetua a recuperação da instância em caso de falhas, durante a sua inicialização. Em um sistema com múltiplas instâncias (como na configuração Oracle Parallel Server, por exemplo), o processo SMON de uma instância também pode executar a recuperação de outras instâncias que podem ter falhado. Ele também limpa os segmentos temporários que não estão sendo usados, liberando memória, e recupera qualquer transação pendente no caso de uma falha em arquivos físicos ou mesmo no disco. O processo de recuperação dessas transações é executado pelo processo SMON quando a tablespace afetada volta a ficar disponível.

O process monitor (PMON) executa a recuperação do processo de um usuário quando esse processo falha. Limpa a área de memória e libera os recursos que o processo do usuário estava usando. O PMON também verifica o processo despachante (dispatcher) e os processos servidores (server processes) e os reinicializa se tiver acontecido qualquer falha.

O processo archiver (ARCH) copia os arquivos redo log para fita ou mesmo outro disco, no momento em que um deles torna-se completo. Esse processo geralmente está presente quando o banco de dados está sendo utilizado no modo ARCHIVELOG. Os arquivos redo log nada têm a ver com auditoria. Eles são usados somente para a recuperação de um banco de dados.

O processo recoverer (RECO) é usado para resolver transações distribuídas pendentes causadas por uma falha na rede em um sistema de bancos de dados distribuídos. A certos intervalos de tempo, o processo RECO do banco de dados local tenta conectar-se ao banco de dados remoto para automaticamente completar e efetivar a transação (COMMIT) ou descartar (ROLLBACK) a porção local de uma transação pendente em um sistema distribuído.

Os processos em background dispatchers (Dnnn) são opcionais e estão presentes somente quando a configuração do Oracle Multi-thread Server é usada. Pelo menos um processo dispatcher é criado para cada protocolo de comunicação em uso (D000, D0001, ..., Dnnn). Cada processo dispatcher é responsável pelo direcionamento das requisições dos processos dos usuários conectados ao banco de dados para o processo servidor disponível e pelo retorno da resposta de volta para o processo do usuário apropriado.

Por sua vez, os processos lock (LCKn) são usados para controlar o lock entre instâncias em uma configuração Parallel Server.

Program interface.

O program interface é o mecanismo pelo qual um processo do usuário se comunica com o processo servidor. Serve como um método de comunicação padrão entre a porção cliente de uma aplicação ou uma ferramenta e o próprio servidor ORACLE.

O program interface age como um mecanismo de comunicação, através da formatação dos dados requisitados, trafegando esses dados, verificando e retornando possíveis erros. Também executa conversões de dados, particularmente entre diferentes tipos de computadores ou tipos de dados usados pelos usuários.

Se o usuário e os processos servidores estão em diferentes computadores de uma rede ou se o processo dispatcher estiver sendo usado para conectar processos de usuários e processos do servidor, então o program interface inclui um software de comunicação chamado SQL*Net, que faz a comunicação e transferência de dados entre computadores.

Como o ORACLE trabalha

Conhecendo os processos e estruturas de memória, fica bastante fácil para que entendamos o modo como ORACLE trabalha:

1. Consideremos que uma instância esteja sendo executada em um computador (servidor de um banco de dados).
2. Um computador usado para executar uma aplicação (porção cliente ou front end) executa uma aplicação de um usuário. Essa aplicação cliente tenta estabelecer uma conexão com o servidor usando o driver apropriado do SQL*Net.
3. O servidor está executando o driver apropriado do SQL*Net e detecta a requisição de conexão da aplicação cliente e cria um processo servidor dedicado ao usuário.
4. O usuário executa um comando SQL e efetiva a transação com o comando COMMIT.
5. O processo servidor recebe o comando e verifica se as áreas shared SQL, armazenadas na shared pool area, contêm um comando idêntico ao emitido pelo usuário. Se localiza uma área shared SQL com um comando idêntico, o processo servidor verifica os privilégios de acesso do usuário aos dados requisitados e o plano de execução definido é usado para buscar os dados solicitados. Se o comando emitido pelo usuário não estiver presente nessa área, uma nova estrutura para o comando é alocada e então ele pode ser analisado e processado.
6. O processo servidor recupera qualquer valor armazenado nos arquivos de dados ou os busca da memória, se lá estiverem, no database buffer cache.
7. O processo servidor modifica os dados na SGA. O processo DBWR escreve os dados modificados em disco, quando necessário. No momento do comando COMMIT, o processo LGWR escreve imediatamente os registros das transações no arquivo redo log que estiver sendo usado no momento.

Se a transação for bem sucedida, o processo servidor manda uma mensagem através da rede para a aplicação. Se não for bem sucedida, uma mensagem de erro é então emitida.

Acesso aos Dados

Antes que os dados possam ser acessados, um processo servidor criado para um determinado usuário conectado ao ORACLE traz os blocos dos arquivos fisicamente armazenados nos discos para dentro do database buffer cache. Cada comando SQL é armazenado na estrutura de memória shared pool e são compartilhados entre todos os usuários conectados a uma instância. Em certo momento, os blocos de dados modificados pelos comandos dos usuários que se encontram no database buffer cache são escritos novamente para os arquivos de dados. Isso é feito pelo processo em background DBWR.

Portanto, toda manipulação dos dados dá-se na memória principal, ou seja, na SGA. É por isso que os dados precisam ser trazidos do disco para a memória antes de serem manipulados.

Usamos dois termos para referenciar o acesso aos dados: cache miss e cache hit. O termo cache miss é usado para identificar as vezes que um processo experimenta acessar uma informação e o bloco que a contém precisa ser lido do disco. O termo cache hit é usado para identificar as vezes que um processo encontra uma informação na memória. Assim, um acesso através de um cache hit é mais rápido do que através de um cache miss.

Essa é a forma básica em que se processa o acesso aos dados, usando como exemplo um comando SQL para a atualização de informações em uma tabela:

```
SQL> UPDATE emp
```

```
2 SET sal = sal * 1.1
```

```
3 WHERE ename = 'SCOTT';
```

1. O usuário emite um comando UPDATE, para atualizar a coluna SAL da linha identificada pela coluna ENAME='SCOTT' de uma tabela hipotética chamada EMP.
2. O comando emitido pelo usuário é analisado e armazenado na SGA, na estrutura shared pool.
3. O processo servidor, criado quando o usuário faz a sua conexão com o ORACLE, efetua as leituras físicas necessárias e traz os blocos de dados armazenados nos arquivos de dados para dentro da SGA, na estrutura database buffer cache.
4. Em seguida o ORACLE aplica a alteração definida no comando UPDATE nos blocos de dados que possuem a linha identificada por ENAME='SCOTT'.
5. Sob certas condições, o processo em background DBWR escreve os blocos de dados alterados de volta para o arquivo de dados físico apropriado. Esse processo em background é o responsável por essa tarefa. Ele simplesmente libera área de memória do database buffer cache, já que a área dessa estrutura é limitada.

Processos de usuários e processos servidores.

Um processo de um usuário é criado quando o usuário executa uma aplicação, ou seja, quando cria uma conexão com uma instância. Nesse momento, o ORACLE cria um processo servidor dedicado que é usado para executar as requisições do processo do usuário ao qual se associa.

Portanto, um processo servidor comunica-se com um processo de um usuário, ou seja, sempre vai ser requisitado para executar qualquer comando. Entretanto, em algumas configurações do ORACLE, um processo servidor pode ser compartilhado por diversos processos de usuários, isto é, não vai ser utilizado para a conexão direta com qualquer processo de usuário; na verdade essa conexão dá-se com a utilização de outros processos. Portanto, nem sempre é verdade que um processo servidor deve estar dedicado a um processo de um usuário.

Basicamente, as funções de um processo servidor são:

1. Analisar e executar os comandos SQL.
2. Verificar se os blocos de dados encontram-se na estrutura database buffer cache.
3. Ler os blocos de dados dos arquivos físicos no disco e levá-los para dentro do database buffer cache, na SGA. Essa operação somente é feita se os blocos de dados a serem utilizados não se encontrarem em memória.
4. Retornar resultados dos comandos SQL para os processos dos usuários que os emitiram.

Em um terminal dedicado em arquitetura multi-usuário, os processos dos usuários permanecem no servidor, assim como os processos servidores criados pelo ORACLE . Em arquitetura cliente-servidor os processos dos usuários permanecem na porção cliente, enquanto os processos servidores criados pelo ORACLE permanecem no servidor. Entretanto, para o ORACLE a forma de acesso independe da arquitetura utilizada, pois as estruturas na SGA, os processos e os próprios arquivos físicos são basicamente os mesmos.

A estrutura shared pool e seus buffers.

A estrutura de memória compartilhada chamada shared pool contém informações usadas para executar os comandos SQL. É formada pelos buffers denominados shared SQL, data dictionary cache e sequence cache.

Os buffers identificados como shared SQL areas contêm o seguinte:

1. O texto dos comandos SQL e PL/SQL.
2. A forma analisada dos comandos SQL e PL/SQL.

3. O plano de execução para os comandos SQL e PL/SQL.

O compartilhamento dos planos de execução dos diversos comandos nas áreas de comandos SQL compartilhados melhoram o uso da memória, uma vez que as definições dos comandos podem ser compartilhadas entre as diversas aplicações.

A memória também é dinamicamente ajustada de acordo com o conjunto de comandos SQL que são executados e, como a fase de parse ou análise é resumida, o tempo de execução de um comando pode diminuir consideravelmente.

Por sua vez os buffers identificados como data dictionary cache contêm:

1. Linhas com as informações do dicionário de dados.

Finalmente, os buffers identificados como sequence cache contêm:

1. Informações sobre os geradores de números seqüenciais usados pelos usuários.

Database buffer cache.

A estrutura de memória compartilhada chamada database buffer cache contém cópias dos blocos de dados que são lidos do disco pelos processos servidores. Os buffers são compartilhados por todos os usuários conectados a uma instância ORACLE.

O tamanho dos blocos de dados é determinado pelo parâmetro `DB_BLOCK_SIZE`, especificado no momento da sua criação e não pode ser alterado a menos que o banco seja novamente recriado.

O número de blocos lógicos em memória é determinado pelo parâmetro `DB_BLOCK_BUFFERS`. Esses dois parâmetros configuram o tamanho do database buffer cache.

Ele é organizado em duas listas: a dirty list e a least recently used list (LRU). A dirty list é uma lista que contém os blocos alterados que ainda não foram escritos em disco.

A LRU é uma lista que contém blocos do ORACLE que foram alterados pelos comandos dos usuários mas ainda não foram gravados em disco. Contém ainda blocos livres e blocos em uso.

Assim, quando um processo servidor precisa ler um bloco de dados do disco para a memória, ele:

1. Pesquisa nas listas LRU e dirty list pelo bloco de dados desejado.
2. Caso esse bloco de dados não seja localizado, o processo servidor pesquisa a lista LRU em busca de um bloco livre.

3. Em seguida, o processo servidor move os blocos alterados encontrados na lista LRU para a dirty list, ou seja, movimenta-os para a lista de blocos alterados ainda não gravados nos arquivos de dados, de acordo com a localização de cada um deles, durante o processo de pesquisa de um bloco livre.

4. Finalmente, o processo servidor efetua uma cópia do bloco de dados do disco para um bloco livre.

5. Esse procedimento termina quando o processo servidor localiza um bloco livre ou se um número específico de blocos forem pesquisados sem encontrar um único bloco livre.

Se nenhum bloco foi encontrado, o ORACLE deve gravar os blocos alterados da dirty list para os arquivos em disco, para liberar espaço em memória para os novos blocos de dados que precisam ser manipulados pelos comandos dos usuários.

Operação envolvendo o comando SELECT.

Para uma operação que envolve o comando SELECT é preciso que os blocos de dados que contêm as linhas a serem retornadas, de acordo com o critério de pesquisa, estejam em memória, no database buffer cache.

São executados os seguintes passos:

1. A lista LRU é pesquisada para que os blocos de dados necessários sejam encontrados.
2. Caso não se encontrem em memória, o processo do servidor executa as leituras físicas necessárias e traz os blocos para a memória.
3. Em seguida são feitas leituras lógicas em memória.

Nenhuma tabela ocupa menos de dois blocos de dados. Portanto, quando uma certa informação armazenada em uma tabela é requerida na memória, pelo menos dois blocos de dados são necessários: um bloco de cabeçalho e outro bloco com os dados.

Segmentos de rollback.

Um segmento de rollback é uma porção de um banco de dados que registra as ações das transações dos usuários nos dados para que possam ser desfeitas sob certas circunstâncias; é um objeto usado para gravar os dados alterados pelos processos dos usuários. Cada banco de dados deve possuir pelo menos um deles.

Um segmento de rollback é usado para permitir a consistência da leitura, recuperar um comando quando ocorre o dead-lock, recuperar uma transação até uma certa marca identificada por um SAVEPOINT, recuperar uma transação terminada por uma falha de processo de um usuário e desfazer todas as transações pendentes durante a recuperação de uma instância.

Cada transação deve ser assinalada a um segmento de rollback. Isso pode ser feito automaticamente baseado em alguns critérios que o ORACLE possui, como pode ser feito manualmente pelos usuários através do comando:

```
SQL> ALTER SYSTEM USE ROLLBACK SEGMENT rbs_<numero>;
```

Onde:

RBS_<numero> Nome do segmento de rollback.

Operação envolvendo o comando UPDATE.

Todas as operações de atualização de dados em um banco de dados envolvem os segmentos de rollback para permitir a consistência da leitura, a recuperação das informações e permitir que uma transação ou um comando sejam desconsiderados ou desfeitos.

São executados os seguintes passos:

1. Os blocos de dados da tabela a ser alterada, com as linhas que sofrerão as alterações, são trazidos para a memória.
2. Os blocos de um segmento de rollback são alocados na mesma estrutura database buffer cache. Nesse momento, o ORACLE aloca automaticamente um segmento de rollback disponível ou algum especificado pelo comando ALTER SYSTEM USE ROLLBACK SEGMENT.
3. São feitos locks exclusivos nas linhas modificadas.
4. Os dados antigos são gravados em um bloco do segmento de rollback acionado anteriormente. Nele são armazenados também a identificação da transação do usuário que executou o comando UPDATE, o endereço da coluna com a especificação do bloco de dados acionado, a identificação do arquivo físico e o número da linha e da coluna a serem alteradas em seguida.
5. As alterações são aplicadas nas linhas da tabela em cada um dos blocos de dados que as armazenam.

Caso o mesmo usuário que tenha executado um comando UPDATE pesquisar a tabela atualizada, ele enxergará sua alteração. Os outros usuários não a enxergarão, isto é, lerão apenas o valor antigo armazenado no segmento de rollback. Dessa forma mantém-se a consistência de leitura. Naturalmente, quando o usuário que executou o comando UPDATE efetivar as alterações com o comando COMMIT, todos os outros usuários passarão a enxergar as alterações feitas, exceto se algum outro estiver executando uma operação em andamento com o comando SELECT.

Consistência de leitura.

Durante todo o processamento de um comando SQL, o ORACLE mantém uma consistência dos dados de uma tabela de acordo com o momento em que o comando for inicializado.

Para o comando SELECT, o ORACLE marca o momento da sua execução como o instante a partir do qual a consistência de leitura é mantida.

A partir deste momento, quaisquer alterações feitas em uma tabela por outros usuários não são enxergadas pelo usuário que emitiu o comando SELECT, até que os outros usuários que atualizaram a tabela terminem suas transações, com os comandos COMMIT ou ROLLBACK.

Todas as alterações feitas são mantidas em segmentos de rollback alocados pelo ORACLE ou pelos próprios usuários. Para quem estiver lendo a tabela o ORACLE lê os valores antigos no segmento de rollback apropriado, e não nos blocos de dados alterados.

A seguir apresentamos o funcionamento desse mecanismo:

10 h 00 min SQL> UPDATE EMP ...;

Às dez horas o usuário A executa o comando UPDATE mas não efetiva as alterações.

10 h 01 min SQL> SELECT ... FROM emp;

Às dez horas e um minuto o usuário B pesquisa a tabela EMP. Ele não enxerga as alterações feitas pelo usuário A. Do segmento de rollback que registrou a alteração do usuário A é trazido o valor antigo às alterações, ocorrendo a consistência da leitura.

10 h 02 min SQL> COMMIT;

Às dez horas e dois minutos o usuário A efetiva sua transação. Como não existe nenhum processo de leitura em andamento e não foi utilizado comando SET TRANSACTION READ ONLY, os segmentos de rollback alocados são liberados.

10 h 03 min SQL> SELECT ... FROM emp;

Finalmente, às dez horas e três minutos o usuário B passa a enxergar as alterações feitas na tabela EMP pelo comando UPDATE do usuário A, pois a transação foi terminada e efetivada com o comando COMMIT.

Processo DBWR.

O processo Database Writer (DBWR) gerencia o database buffer cache para que os processos dos usuários sempre localizem blocos livres para o processamento de seus comandos.

Ele escreve todos os buffers alterados para os arquivos de dados, usando o algoritmo LRU para manter os blocos mais utilizados em memória.

O DBWR adia ao máximo a escrita dos blocos alterados para a otimização do I/O em disco, que é uma das principais causas para a queda da performance de um banco de dados.

O processo DBWR escreve os blocos alterados para o disco quando:

1. A dirty list ultrapassar um certo limite. Essa lista é usada no database buffer cache e contém os buffers alterados.
2. Um processo pesquisar um número específico de buffers na LRU sem encontrar um bloco livre.
3. Ocorrer o time-out, ou seja, quando um certo tempo limite for ultrapassado. Esse tempo limite geralmente é de três segundos.
4. Ocorrer um checkpoint.

Configuração multi-threaded

O ORACLE pode ser configurado em três diferentes formas para variar o número dos processos de usuários que podem estar conectados em cada processo do servidor.

Dedicated Server Um processo servidor dedicado manuseia as requisições emitidas por um único usuário.

Esse processo servidor é criado quando ocorre a conexão de um usuário com o ORACLE.

Multi-Threaded Server A configuração Multi-Threaded Server do ORACLE permite que diversos processos de usuários conectados a uma instância possam compartilhar um conjunto de processos servidores disponíveis.

Esses processos servidores são fornecidos pelo ORACLE quando o usuário requisita um comando.

Combined User/Server Process Nesta configuração os códigos de uma aplicação e do ORACLE são combinados em uma única tarefa.

Essa configuração é disponível em alguns sistemas operacionais, como o VMS.

Com a utilização apropriada dessas configurações, podemos eventualmente melhorar o desempenho do banco de dados. Por isso, nessa sessão discutiremos a arquitetura multi-threaded, suas vantagens e a configuração do ambiente.

Quando devemos usar?

O uso do multi-threaded tem diversas vantagens em relação às outras configurações. Com ele podemos reduzir o número de processos em execução na instância e, dessa forma, conseguimos aumentar o número de possíveis usuários. O número de processos desocupados pode ser drasticamente diminuído e temos uma sensível melhora no uso da memória.

Somente em algumas situações especiais devemos usar a configuração de servidores dedicados. Para a execução de procedimentos em lote, com uma grande quantidade de comandos SQL e para nos conectarmos como INTERNAL (para fazermos o STARTUP, SHUTDOWN ou a recuperação do banco de dados, por exemplo), devemos usar os servidores dedicados. Também devemos fazê-lo em algumas situações incomuns envolvendo os dead-locks no ambiente multi-threaded.

A arquitetura multi-threaded.

A primeira é caracterizada pela conexão dos usuários. Durante uma tentativa de conexão, um processo chamado LISTENER (que faz parte do SQL*Net versão 2) percebe a requisição e determina se o processo do usuário pode ou não usar um processo servidor compartilhado. Caso seja permitido, o LISTENER informa ao processo do usuário o endereço de um processo chamado despachante, ao qual permanecerá conectado enquanto durar a sua sessão. Quando o usuário requisita uma conexão dedicada, o LISTENER cria um processo servidor dedicado e o associa ao usuário. Essa facilidade somente é possível com a versão 2 do SQL*Net. As versões anteriores não suportam a facilidade do multi-threaded, ou seja, elas aceitam tão somente as conexões a processos servidores dedicados.

A segunda fase é caracterizada pela emissão dos comandos SQL por parte dos usuários. Quando um deles emite qualquer comando, essa requisição é recebida pelo processo despachante ao qual o usuário está conectado. Por sua vez, o despachante coloca a requisição em uma fila de requisições, ou fila de entrada, que se encontra na SGA. O primeiro processo servidor compartilhado que estiver disponível obtém a requisição na fila de entrada e o processa. Ao término do processamento, o processo servidor coloca a resposta em uma fila de respostas, única para o despachante ao qual o usuário estiver conectado. Finalmente, esse despachante retorna a resposta ao usuário original.

A fila de entrada, que recebe todas as requisições dos usuários, é única na instância e é compartilhada por todos os despachantes. Essa fila é do tipo FIFO, ou seja, primeiro-que-entra-primeiro-que-sai (first-in-first-out). As filas de respostas são usadas para conter todas as respostas dos comandos SQL executados pelos processos servidores compartilhados. Cada um dos despachantes possui a sua própria fila de respostas.

O conteúdo da PGA e da SGA diferencia-se quando implementamos o uso dos processos servidores dedicados e compartilhados. A alocação de memória sem o multi-threaded, ou seja, na configuração convencional (dedicada), difere-se da multi-threaded por que, nessa, parte do conteúdo da PGA passa a residir na SGA; somente encontra-se originalmente na

PGA um espaço para a pilha, que contém as variáveis usadas por um usuário. As informações sobre as sessões dos usuários, que inclui dados sobre a segurança e o uso dos recursos do ORACLE, assim como as informações sobre o estado dos cursores, passam a residir na SGA. Essa alteração na PGA e na SGA é totalmente transparente para os usuários. Podemos especificar o montante de memória na SGA a ser alocada para cada usuário através dos profiles, que controlam o uso dos recursos banco de dados.

A configuração do multi-threaded é relativamente simples. Devemos inicialmente instalar e configurar o SQL*Net Versão 2. Sem a versão 2 desse produto ficamos impedidos de usar essa configuração.

Nesse documento não abordaremos toda a configuração, entretanto apresentaremos, em seguida, os passos básicos para configurarmos uma máquina servidora de banco de dados:

Passo 1: Configurar e criar o processo LISTENER.

O LISTENER é o processo que controla as conexões às instâncias. Podemos ter vários processos rodando em uma mesma máquina; entretanto apenas um já é o suficiente, pois podemos configurá-lo para suportar diversas instâncias e diferenciados protocolos. Os tipos de conexões são determinados pelos protocolos usados pelos processos despachantes.

Existe um arquivo especial, denominado LISTENER.ORA, que usamos para a configuração do LISTENER. Geralmente ele encontra-se no diretório \$ORACLE_HOME/NETWORK/ADMIN. Em alguns sistemas UNIX, esse diretório default pode ser o /etc.

Entretanto, podemos especificar qualquer diretório que desejarmos; para isso configuramos a variável de ambiente chamada TNS_ADMIN com o nome do diretório desejado.

Nesse arquivo texto (LISTENER.ORA) inserimos todas as informações sobre a configuração do LISTENER. Abaixo, apresentamos um modelo:

```
#####  
# Exemplo do arquivo listener.ora  
#####  
LISTENER =  
(ADDRESS_LIST =  
(ADDRESS =  
(PROTOCOL=TCP)  
(HOST=192.1.1.10)  
(PORT=1525)  
)  
)  
STARTUP_WAIT_TIME_LISTENER=0  
CONNECT_TIMEOUT_LISTENER=10  
LOG_FILE_LISTENER=listener.log  
SID_LIST_LISTENER=
```

```
(SID_LIST=
(SID_DESC=
(SID_NAME=sid1)
(ORACLE_HOME=/usr/oracle)
)
)
TRACE_LEVEL_LISTENER=0
Para criarmos o processo, usamos o utilitário LSNRCTL:
$ lsnrctl start
```

Passo 2: Configurar os descritores de conexão.

Os descritores de conexão são usados para facilitar a conexão dos usuários. Eles são armazenados no arquivo TNSNAMES.ORA, que fica nos mesmos diretórios onde podemos encontrar o LISTENER.ORA. A seguir temos um exemplo:

```
#####
# Exemplo do arquivo tnsnames.ora
#####
sid1mts =
(DESCRIPTION=
(AADDRESS_LIST=
(AADDRESS=
(PROTOCOL=TCP)
(HOST=192.1.1.10)
(PORT=1525)
)
)
(CONNECT_DATA=
(SID=sid1)
)
)
sid1dedic =
(DESCRIPTION=
(AADDRESS_LIST=
(AADDRESS=
(PROTOCOL=TCP)
(HOST=192.1.1.10)
(PORT=1525)
)
)
(CONNECT_DATA=
(SID=sid1)
(SERVER=DEDICATED)
)
)
```

Passo 3: Configurar e criar a instância.

Devemos fechar o banco e encerrar a instância para em seguida colocarmos o banco no ar, usando o arquivo de inicialização com os seguintes parâmetros:

```
MTS_DISPATCHERS="tcp,3", "ipc,2"  
MTS_MAX_DISPATCHERS=10  
MTS_SERVERS=4  
MTS_MAX_SERVERS=14  
MTS_SERVICE=sid1  
MTS_LISTENER_ADDRESS="(ADDRESS=(PROTOCOL=TCP)(HOST=gp.com)(PORT=1525))"
```

Registro das Transações

O ORACLE registra todas as alterações feitas em um banco de dados na estrutura redo log buffer cache.

Um processo em background denominado LGWR escreve as informações desses buffers para o disco, sob certas circunstâncias.

Um outro processo em background conhecido como ARCH pode ser opcionalmente utilizado para armazenar as informações sobre as alterações feitas nos dados em outro dispositivo, sempre que um arquivo redo log for preenchido.

Somente um arquivo redo log é utilizado por vez, entretanto em um banco de dados podem existir diversos arquivos de redo log. O seu número mínimo é de dois grupos, cada um deles podendo conter um ou mais arquivos.

Redo log buffer cache.

O redo log buffer cache é uma estrutura de memória de uso circular que contém buffers ou conjuntos de blocos ORACLE com informações sobre todas as alterações feitas nos dados de um banco de dados. Essas informações são armazenadas sob a forma de entradas de redo log e são usadas para reconstruir as informações dos segmentos alterados, inclusive os segmentos de rollback.

As entradas de redo log armazenam todas as alterações feitas em um banco de dados dentro da estrutura redo log buffer cache. São usadas para reconstruir ou descartar as alterações feitas nos dados quando uma recuperação for necessária, ou seja, armazenam a before image e a after image. Esses termos são usados para identificarmos os dados antes e depois de uma alteração.

Em situações especiais, podemos desejar não registrar as alterações nos arquivos de log. Por exemplo, na criação de um índice ou de uma tabela e na carga de dados através do

SQL*Loader; nos comandos de criação de tabelas e índices podemos usar a cláusula UNRECOVERABLE.

O tamanho dessa estrutura é determinado pelo parâmetro LOG_BUFFER.

Comando UPDATE e o redo log buffer.

Como vimos, todas as alterações feitas nos dados são armazenadas como entradas de redo na estrutura redo log buffer cache.

Assim, a operação de UPDATE envolve realmente os seguintes passos:

1. Os blocos de dados da tabela a ser alterada com as linhas que sofrerão as alterações são trazidos para a memória, para dentro do database buffer cache.
2. Os blocos de um segmento de rollback são alocados na mesma estrutura. Nesse momento, o ORACLE aloca automaticamente um segmento de rollback disponível ou algum especificado pelo comando ALTER SYSTEM USE ROLLBACK SEGMENT.
3. São feitos locks exclusivos nas linhas a serem modificadas.
4. A imagem das informações antes e depois das modificações são acionadas para dentro do redo log buffer cache como entradas de redo log.
5. Os dados antigos são gravados em um bloco do segmento de rollback acionado anteriormente juntamente com a identificação da transação do usuário que executou o comando UPDATE, o endereço da coluna com a especificação do bloco de dados acionado, a identificação do arquivo físico e o número da linha e da coluna a serem alteradas em seguida.
6. As alterações são aplicadas nas linhas da tabela em cada um dos blocos de dados que as armazenam.

Processo LGWR.

O processo em background log writer (LGWR) escreve as entradas de redo log para o disco. Isso acontece quando:

1. Ocorre a efetivação de uma transação com o comando COMMIT.
2. A estrutura redo log buffer atinge aproximadamente 1/3 de seu tamanho.
3. O processo em background DBWR precisa limpar os blocos dos buffers para a ocorrência de um checkpoint.
4. Ocorre o time-out.

Em uma instância existe somente um único grupo de arquivos redo log sendo utilizado para a escrita das entradas de redo log, da memória para o disco, simultaneamente, assim como somente um processo LGWR ativo. Enquanto uma transação não for registrada em um arquivo redo log o COMMIT emitido não é confirmado. Uma transação pode fazer com que outras transações sejam também gravadas nos arquivos redo log (piggy-backed, brincadeira conhecida entre nós como cavalinho), quando são efetivadas simultaneamente.

Quando um grupo for preenchido, ocorre o log switch, ou seja, o próximo grupo disponível passa a ser utilizado. Caso o banco opere no modo ARCHIVELOG, podemos usar os parâmetros LOG_ARCHIVE_BUFFER_SIZE e LOG_ARCHIVE_BUFFERS para melhorarmos a gravação dos mesmos para outro dispositivo.

Operação envolvendo o comando COMMIT.

O comando COMMIT efetiva as alterações feitas nos dados por uma transação, tornando-as permanentes.

São executados os seguintes passos:

1. Um usuário emite o comando COMMIT para finalizar sua transação.
2. Um registro desse COMMIT é colocado no redo log buffer.
3. O processo em background LGWR grava as entradas de redo log dos buffers para o arquivo redo log correntemente em uso, se possível usando uma gravação multi-bloco.
4. O usuário é notificado de que a transação foi efetivada.
5. Os locks nos recursos são liberados assim como os blocos do segmento de rollback alocados para a transação do usuário.

Apesar de não fazerem parte do processo de COMMIT de uma transação, precisamos assinalar que os blocos de dados são marcados como alterados e os blocos do segmento de rollback são liberados ou marcados como reutilizáveis e, eventualmente, o processo em background DBWR pode escrever os blocos de dados alterados do database buffer cache para os arquivos físicos em disco.

O processo LGWR registra permanentemente todas as alterações nos dados feitas pelas transações dos usuários, enquanto o DBWR adia ao máximo a escrita dos blocos alterados nas transações para diminuir o I/O, ou seja, reduzir o tempo de processamento da gravação dos blocos de dados alterados que se encontram na estrutura database buffer cache para os arquivos físicos em disco, melhorando, assim, a performance. O I/O é um dos aspectos que causam os maiores problemas e deve ser melhorado.

Os comandos COMMIT simultâneos dos usuários fazem com que as entradas de redo log para suas transações sejam gravadas juntas em uma única operação de gravação física nos arquivos redo log. Além do mais, ocorre somente uma única operação de gravação de

entradas redo log por transação, pois essa gravação ocorre no momento do COMMIT e esse comando termina uma transação lógica. O tamanho de uma transação não afeta o tempo necessário para uma operação de COMMIT.