19. Performance Tuning

O DBMaker é um sistema de banco de dados altamente ajustável. A otimização do DBMaker aumentará seu nível de desempenho para atender às necessidades individuais. Este capítulo apresenta os objetivos e métodos utilizados no processo de ajuste para demonstrar como diagnosticar o desempenho de um sistema.

19.1 The Tuning Process

Antes de ajustar o DBMaker, você deve definir metas para melhorar o desempenho. Tenha em mente que algumas metas podem entrar em conflito. É necessário decidir quais das metas conflitantes são mais importantes.

A seguir, estão alguns dos objetivos para ajustar o DBMaker:

- Melhorar o desempenho das instruções SQL
- Melhorar o desempenho das aplicações de banco de dados
- Melhorar o desempenho do processamento concorrente
- Otimizar a utilização de recursos

Após determinar as metas, você estará pronto para começar a ajustar o DBMaker. Comece realizando os seguintes passos:

- Monitorar o desempenho do banco de dados
- Ajustar I/O
- Ajustar a alocação de memória
- Ajustar o processamento concorrente
- Monitorar o desempenho do banco de dados e comparar com as estatísticas anteriores

Os métodos utilizados para realizar o ajuste em cada um desses passos podem ter uma influência negativa sobre outros passos. Seguir a ordem mostrada acima pode reduzir essa influência. Após realizar todos os passos de ajuste, monitore o desempenho do DBMaker para verificar se o melhor desempenho geral foi alcançado.

Antes de ajustar o DBMaker, certifique-se de que as instruções SQL estão escritas de forma eficiente e que as aplicações de banco de dados empregam um bom design. Instruções SQL ineficientes ou aplicações mal projetadas podem ter uma influência negativa no desempenho do banco de dados que o ajuste não pode melhorar. Para

escrever instruções e aplicações eficientes, consulte o **Referência de Comandos e Funções SQL** e o **Guia do Programador ODBC**.

19.2 Monitoring a Database

Esta seção mostra como monitorar informações sobre o status de um banco de dados, incluindo status de recursos, status de operações, status de conexões e status de concorrência. Esta seção também mostra como encerrar uma conexão.

NOTA: O usuário deve ter autoridade de DBA para fazer login.

The Monitor Tables

O DBMaker armazena o status do banco de dados em quatro tabelas de catálogo do sistema: SYSINFO, SYSUSER, SYSLOCK e SYSWAIT.

A tabela SYSINFO contém valores do sistema do banco de dados, incluindo o tamanho total do DCCA, o tamanho disponível do DCCA, o número máximo de transações e o número de buffers de página. Também inclui estatísticas sobre ações do sistema, como o número de transações ativas, o número de transações iniciadas, o número de solicitações de bloqueio e semáforo, o número de I/O de disco físico, o número de I/O de registros de journal, entre outros. Use esta tabela para monitorar o status do sistema do banco de dados e para obter informações que ajudem a ajustar o banco de dados.

A tabela SYSUSER contém informações de conexão, incluindo ID da conexão, nome do usuário, nome de login, endereço IP de login e o número de operações DML que foram executadas. Use esta tabela para monitorar quais usuários estão utilizando o banco de dados.

A tabela SYSLOCK contém informações sobre objetos bloqueados, como o ID do objeto bloqueado, status de bloqueio, granularidade do bloqueio, ID da conexão que está bloqueando este objeto, entre outros. Use esta tabela para monitorar quais objetos estão sendo bloqueados por qual conexão e quais usuários estão bloqueando quais objetos.

A tabela SYSWAIT contém informações sobre o status de espera das conexões, incluindo o ID da conexão que está esperando e o ID da conexão pela qual está aguardando. Use esta tabela para monitorar o status de concorrência das conexões. Quando uma conexão está aguardando recursos que estão bloqueados por uma conexão ociosa ou morta, você pode determinar qual conexão está bloqueando esses

objetos. Em seguida, você pode encerrar a conexão ociosa ou morta para liberar os recursos.

Navegue por essas quatro tabelas de catálogo do sistema da mesma forma que se navega em tabelas comuns.

Exemplo:

Comando SQL SELECT usado para navegar na tabela SYSUSER:

```
dmSQL> SELECT * FROM SYSUSER;
```

Consulte a Referência do Catálogo do Sistema para mais informações sobre essas quatro tabelas de catálogo do sistema.

Killing Connections

Uma conexão deve ser encerrada quando estiver mantendo recursos e estiver ociosa por um longo período, ou quando os recursos forem urgentemente necessários. Além disso, todas as conexões ativas devem ser encerradas antes de desligar um banco de dados. Antes de encerrar uma conexão, navegue na tabela SYSUSER para determinar seu ID de conexão.

Exemplo 1:

Para encerrar a conexão de Eddie, primeiro recupere o ID da conexão:

```
dmSQL> SELECT CONNECTION_ID FROM SYSUSER WHERE USER_NAME = 'Eddie';
CONNECTION_ID
=============
352501
```

Exemplo 2:

Em seguida, para encerrar a conexão de Eddie, use:

```
dmSQL> KILL CONNECTION 352501;
```

19.3 Tuning I/O

A I/O de disco requer mais tempo no DBMaker.

Para evitar gargalos de I/O de disco, realize o seguinte:

- Determine as partições de dados
- Determine as partições de arquivos de journal
- Separe os arquivos de journal e os arquivos de dados em discos diferentes
- Use dispositivos brutos
- Pre-aloque espaço em um tablespace de autoexpansão
- Ative o daemon de I/O e checkpoint

Determining Data Partitions

Você pode usar tablespaces para particionar dados em vez de armazenar todos os dados juntos. Se os tablespaces forem usados corretamente, o DBMaker terá um desempenho melhor ao realizar funções de gerenciamento de espaço ou varreduras completas de tabelas. Tabelas pequenas que contêm dados de uma natureza semelhante podem ser agrupadas em um único tablespace, mas tabelas muito grandes devem ser colocadas em seu próprio tablespace.

Você pode melhorar a velocidade na I/O de disco utilizando striping de disco. Striping é a prática de separar setores de disco consecutivos para que eles se espalhem por vários discos. Isso pode ser usado para dividir os dados em uma grande tabela entre vários discos. Isso ajuda a evitar a contenção de disco que pode ocorrer quando muitos processos tentam acessar os mesmos arquivos simultaneamente.

Determining Journal File Partitions

O DBMaker oferece a flexibilidade de usar um ou mais arquivos de journal. Um único arquivo de journal é mais fácil de gerenciar, mas usar múltiplos arquivos de journal também tem algumas vantagens. Se você executar o DBMaker em modo de backup e usar o servidor de backup para realizar backups incrementais, utilizar múltiplos arquivos de journal pode melhorar o desempenho dos backups incrementais. Apenas arquivos de journal completos são copiados. Além disso, distribuir vários arquivos de journal em diferentes discos pode aumentar o desempenho da I/O de disco.

Você pode determinar o tamanho dos arquivos de journal examinando as necessidades das transações. No entanto, se você executar o DBMaker em modo de backup e realizar backups de acordo com o status de preenchimento do journal, o tamanho do journal também afetará o intervalo de tempo do backup. Um arquivo de journal maior aumenta o intervalo entre os backups.

Separating Journal Files and Data Files

Separar os arquivos de journal e os arquivos de dados em discos diferentes aumentará o desempenho da I/O de disco, permitindo que os arquivos sejam acessados simultaneamente em algum grau. Se os discos tiverem velocidades de I/O diferentes, considere quais arquivos colocar nos discos mais rápidos. Em geral, se você executar aplicações de processamento de transações on-line (OLTP) com frequência, coloque os arquivos de journal nos discos mais rápidos. No entanto, se você executar aplicações que realizam consultas longas, como um sistema de suporte à decisão, coloque os arquivos de dados nos discos mais rápidos.

Using Raw Devices

Se você executar o DBMaker em um sistema UNIX, construa arquivos de dispositivos brutos para armazenar os dados do DBMaker e os arquivos de journal. Como o DBMaker possui um bom mecanismo de buffer, é muito mais rápido ler/gravar em um dispositivo bruto do que em um arquivo UNIX. Para mais informações sobre como criar um dispositivo bruto, consulte o manual do sistema operacional ou entre em contato com seu administrador de sistema. A única desvantagem de usar dispositivos brutos é que o DBMaker não pode estender tablespaces neles automaticamente; portanto, é necessário mais planejamento ao usar arquivos de dispositivos brutos.

Pre-Allocating Autoextend Tablespaces

O DBMaker suporta tablespaces de autoexpansão para simplificar o gerenciamento de tablespaces. No entanto, se você conseguir estimar o tamanho necessário de um tablespace, é melhor definir o tamanho ao criar o tablespace. Isso melhora o desempenho, pois a extensão de páginas leva muito tempo. Você pode estender as páginas de um arquivo posteriormente usando o comando alter file. Prealocar o tamanho de um tablespace também pode evitar erros de disco cheio quando o DBMaker tenta estender um tablespace que já ocupa todo o espaço disponível no disco.

I/O and Checkpoint Daemons

I/O DAEMON

O DBMaker possui um daemon de I/O que periodicamente grava páginas sujas dos buffers de página menos utilizados no disco. Isso ajuda a reduzir a sobrecarga incorrida ao trocar páginas de dados para os buffers de página e aumenta o desempenho. Um parâmetro de configuração no arquivo dmconfig.ini controla o daemon de I/O.

DB_IOSvr — habilita e desabilita o daemon de I/O. Definir esta palavra-chave para um valor de 1 habilita o daemon de I/O, enquanto defini-la para um valor de 0 desabilita o daemon de I/O.

Exemplo:

Um trecho típico do arquivo dmconfig.ini:

```
[MYDB]
…
DB_IOSvr                                =                                1
```

O banco de dados MYDB possui 400 (DB_NBufs) buffers de página no DCCA. A cada 10 minutos, o daemon de I/O realiza os seguintes passos:

- Escanear os buffers de página menos utilizados
- Coletar as páginas sujas durante o processamento da varredura
- Gravar essas páginas sujas coletadas no disco

CHECKPOINT DAEMON

O DBMaker possui um daemon de checkpoint (baseado no daemon de I/O) que periodicamente realiza um checkpoint. Isso ajuda a reduzir o tempo gasto aguardando um checkpoint que ocorre durante um comando, quando um journal está cheio ou ao iniciar ou desligar um banco de dados. O daemon de checkpoint é, na verdade, uma subfunção do daemon de I/O, que pode realizar I/O isoladamente, checkpoints isoladamente ou ambos simultaneamente.

Para ativar o daemon de checkpoint, ative o daemon de I/O usando a palavra-chave **DB_IoSvr**. Se o daemon de I/O estiver ativado, ele automaticamente realizará um checkpoint a cada 10 minutos após o banco de dados ser iniciado com sucesso.

Na verdade, os daemons de I/O e de checkpoint utilizarão alguns recursos de I/O. Após iniciar o servidor de banco de dados, as mensagens de erro geradas pelos daemons de I/O e de checkpoint são escritas no arquivo **DMERROR.LOG**, enquanto as mensagens de aviso são registradas no **DMEVENT.LOG**.

19.4 Tuning Memory Allocation

O DBMaker armazena informações temporariamente em buffers de memória e permanentemente no disco. Como leva muito menos tempo recuperar dados da memória do que do disco, o desempenho aumentará se os dados puderem ser obtidos

dos buffers de memória. O tamanho de cada uma das estruturas de memória do DBMaker afetará o desempenho de um banco de dados. No entanto, o desempenho se torna um problema apenas se não houver memória suficiente.

Esta seção se concentra em ajustar o uso da memória para um banco de dados. Inclui informações sobre como calcular o tamanho necessário do DCCA e como monitorar e alocar memória suficiente para os buffers de página, buffers de journal e área de controle do sistema.

Aqui está a tradução para o português, respeitando a gramática e a concordância verbal:

Para alcançar o melhor desempenho, siga os passos na ordem apresentada:

1. Ajuste o sistema operacional
2. Ajuste o tamanho da memória do DCCA
3. Ajuste os buffers de página
4. Ajuste os buffers de journal
5. Ajuste a SCA

A necessidade de memória do DBMaker varia de acordo com as aplicações em uso; ajuste a alocação de memória após o ajuste dos programas de aplicação e das instruções SQL.

Tuning an Operating System

O sistema operacional deve ser ajustado para reduzir a troca de memória e garantir que o sistema funcione de maneira suave e eficiente.

A troca de memória entre a memória física e o arquivo de memória virtual no disco leva um tempo significativo. É importante ter memória física suficiente para os processos em execução. Meça o status de um sistema operacional com as utilitários do sistema operacional. Uma taxa de troca de páginas extremamente alta indica que a quantidade de memória física em um sistema não é grande o suficiente. Se esse for o caso, remova quaisquer processos desnecessários ou adicione mais memória física ao sistema.

Tuning DCCA Memory

A Área de Comunicação e Controle do Banco de Dados (DCCA) é um grupo de memória compartilhada alocada pelos servidores DBMaker. Sempre que o DBMaker é iniciado, ele aloca e inicializa a DCCA.

O modelo cliente/servidor UNIX do DBMaker aloca a DCCA do pool de memória compartilhada do UNIX. Certifique-se de que o tamanho da DCCA seja menor do que o tamanho máximo de memória compartilhada permitido pelo sistema operacional. Se o tamanho solicitado para a DCCA for maior do que o limite do sistema operacional, consulte o manual de administração do sistema operacional para obter informações sobre como aumentar o tamanho máximo da memória compartilhada.

O tamanho da memória compartilhada do DBMaker é de 231 páginas (231 × TAMANHO DA PÁGINA em bytes) em plataformas de 64 bits. A memória compartilhada em plataformas de 32 bits é de 2 GB. Isso inclui o tamanho do buffer de página, o tamanho do buffer de journal e o tamanho da SCA.

Para o Linux de 64 bits, o tamanho shmmax deve ser configurado ao solicitar memória compartilhada além de 231. Observe que é necessário ter RAM suficiente. Edite o arquivo /etc/sysctl.conf para definir kernel.shmmax = n, onde n especifica o tamanho máximo da memória compartilhada.

Exemplo:

```
kernel.shmmax = 8405194752.
```

CONFIGURING THE DCCA

A DCCA contém os blocos de controle de comunicação de processo, blocos de controle de concorrência e os buffers de cache para páginas de dados, blocos de journal e catálogos. O DBMaker mantém os blocos de controle de concorrência e o status de comunicação de cada processo do DBMaker na DCCA. Cada processo do DBMaker acessa os mesmos dados no disco por meio dos buffers de cache na DCCA.

Definir os parâmetros apropriados no dmconfig.ini antes de iniciar o banco de dados configura o tamanho de cada um dos componentes da DCCA.

Exemplo 1:

Uma configuração de exemplo para a DCCA no arquivo dmconfig.ini:

```
DB_NBuFs = 200
DB_NjnlB = 50
DB_ScaSz = 50
```

DB_NBufs especifica o número de buffers de páginas de dados (8.192 bytes por buffer se você definir o tamanho da página para 8 KB), DB_NJnlB especifica o número de buffers de blocos de journal (8.192 bytes por buffer se você definir o tamanho da página para 8 KB) e DB_ScaSz especifica o tamanho da SCA em páginas (8.192 bytes por página se você definir o tamanho da página para 8 KB). O DBMaker lê esses parâmetros da DCCA apenas ao iniciar um banco de dados. Para ajustar os parâmetros, termine o banco de dados, modifique os valores no arquivo dmconfig.ini e reinicie o banco de dados. Para mais informações sobre como definir esses parâmetros, consulte Palavras-chave em dmconfig.ini.

Exemplo 2:

Se o tamanho da página for 8 KB, para calcular o tamanho total da DCCA:

```
size of DCCA = (200 + 50 + 50) * 8 KB
                            = 2400 KB
```

ALLOCATING SUFFICIENT DCCA PHYSICAL MEMORY

A DCCA é o recurso mais frequentemente acessado pelos processos do DBMaker. É importante garantir que haja memória física suficiente para evitar que o sistema operacional troque a DCCA para o disco com muita frequência, pois isso pode degradar seriamente o desempenho do banco de dados. Meça a taxa de troca de páginas usando utilitários do sistema operacional.

Exemplo:

Para determinar o tamanho da memória alocada para a DCCA a partir da tabela do sistema SYSINFO:

```
dmSQL> SELECT INFO, VALUE FROM SYSINFO WHERE INFO = 'DCCA_SIZE'
 OR INFO = 'FREE_DCCA_SIZE';
        INFO                              VALUE
============================ ==============================
DCCA_SIZE                                1228800
FREE_DCCA_SIZE                           189024
```

DCCA_SIZE — o tamanho da memória, em bytes, da DCCA.

FREE_DCCA_SIZE — o tamanho, em bytes, da memória livre restante na DCCA.

A memória livre na DCCA é reservada para uso por blocos de controle dinâmicos, como blocos de controle de bloqueio.

Geralmente, um número maior de buffers é melhor para o desempenho do sistema. No entanto, se a DCCA for grande demais para caber na memória física, o desempenho do sistema irá degradar. Portanto, é importante alocar memória suficiente para a DCCA, mas ainda garantir que a DCCA caiba na memória física.

Tuning Page Buffer Cache

O DBMaker utiliza o pool de memória compartilhada para o cache de buffers de páginas de dados. O cache de buffers permite que o DBMaker acelere o acesso a dados e o controle de concorrência. O DBMaker configura automaticamente o número de buffers de página por padrão. Definir a palavra-chave DB_Nbufs no arquivo dmconfig.ini como zero permite que o DBMaker defina automaticamente o número de buffers de página. O DBMaker pode ajustar dinamicamente o número de buffers de página em sistemas que permitem que o DBMaker detecte o uso da memória física. O número não será inferior a 500 páginas no Windows 95/98, ou não será inferior a 2.000 páginas para outras plataformas do Windows ou UNIX. Se o DBMaker não conseguir detectar o uso da memória física do sistema, ele alocará a quantidade mínima.

Ajustar o tamanho dos buffers de página terá o maior efeito no desempenho. As próximas seções mostram como monitorar o desempenho do cache de buffers e calcular as taxas de acerto de buffers.

 Para melhorar o desempenho do cache de buffers:

1. Atualize as estatísticas dos objetos de esquema.
2. Defina NOCACHE em tabelas grandes.
3. Reorganize os dados em índices mal agrupados.
4. Aumente os buffers de cache.
5. Reduza o efeito dos checkpoints.

MONITORING PAGE BUFFER CACHE PERFORMANCE

O DBMaker coloca as estatísticas de acesso ao cache de buffers na tabela de sistema SYSINFO.

Exemplo:

Para obter os valores do cache de buffers, use as seguintes instruções SQL:

```
dmSQL> SELECT INFO, VALUE FROM SYSINFO WHERE INFO = 'NUM_PAGE_BUF';
            INFO                              VALUE
============================== ==============================
NUM_PAGE_BUF 4000
1 rows selected
dmSQL> SELECT INFO, VALUE FROM SYSINFO WHERE INFO =
'NUM_PHYSICAL_READ'
 2> OR INFO = 'NUM_LOGICAL_READ'
 3> OR INFO = 'NUM_PHYSICAL_WRITE'
 4> OR INFO = 'NUM_LOGICAL_WRITE';
              INFO                              VALUE
============================== ==============================
NUM_PHYSICAL_READ                                64
NUM_PHYSICAL_WRITE                               1
NUM_LOGICAL_READ                                 509
NUM_LOGICAL_WRITE                                0
4 rows selected
```

NUM_PAGE_BUF — número de páginas usadas para o cache de buffer de dados NUM_PHYSICAL_READ — número de páginas lidas do disco NUM_LOGICAL_READ — número de páginas lidas do cache de buffer NUM_PHYSICAL_WRITE — número de páginas escritas no disco NUM_LOGICAL_WRITE — número de páginas escritas no cache de buffer

Calcule a taxa de acertos de leitura/escrita do buffer de páginas com as seguintes fórmulas:

$$\text{read hit ratio} = 1 - \left(\frac{\text{NUM\_PHYSICAL\_READ}}{\text{NUM\_LOGICAL\_READ}}\right)$$

$$\text{write hit ratio} = 1 - \left(\frac{\text{NUM\_PHYSICAL\_WRITE}}{\text{NUM\_LOGICAL\_WRITE}}\right)$$

Para calcular a taxa de acertos de leitura/escrita do buffer usando o exemplo acima, siga as fórmulas:

$$\text{read hit ratio} = 1 - (\frac{13207}{331595})$$
$$= 0.960$$
$$= 96.0\%$$
$$\text{write hit ratio} = 1 - (\frac{7361}{127423})$$
$$= 0.942$$
$$= 94.2\%$$

Com base na razão de acertos de leitura/escrita, determine como melhorar o desempenho do cache de buffer. Se a razão de acertos for muito baixa, ajuste o DBMaker com os métodos descritos nas subseções seguintes.

Se a razão de acertos for sempre alta, por exemplo, maior que 99%, o cache provavelmente é grande o suficiente para armazenar todas as páginas mais frequentemente usadas. Nesse caso, tente reduzir o tamanho do cache para reservar memória para as aplicações. Para garantir um bom desempenho, monitore o desempenho do cache antes e depois de fazer as modificações.

STATISTICS VALUES ARE OUTDATED

Se a razão de acertos de leitura/escrita for muito baixa, pode ser que os valores das estatísticas dos objetos do esquema (tabelas, índices, colunas) estejam desatualizados. Estatísticas erradas podem fazer com que o otimizador do DBMaker use um plano ineficiente para a instrução SQL. Se os usuários tiverem inserido grandes quantidades de dados no banco de dados após a última atualização dos valores das estatísticas, atualize os valores novamente.

Exemplo 1:

Para atualizar os valores das estatísticas de todos os objetos do esquema:

```
dmSQL> UPDATE STATISTICS;
```

Se um banco de dados for extremamente grande, levará muito tempo para atualizar os valores estatísticos de todos os objetos do esquema. Um método alternativo é atualizar as estatísticas de objetos específicos do esquema que foram modificados desde a última atualização e definir a taxa de amostragem.

Exemplo 2:

Para atualizar objetos específicos do esquema:

```
dmSQL> UPDATE STATISTICS tabel1, table2, user1.table3 SAMPLE = 30;
```

Após atualizar com sucesso os valores estatísticos dos objetos do esquema, monitore o desempenho do cache de buffer de páginas com o método especificado em Monitoramento do Desempenho do Cache de Buffer de Páginas.

SWAP OUT CACHE

O DBMaker determina quais buffers de página trocar com base na regra do Menos Recentemente Usado (LRU). Isso mantém as páginas mais frequentemente acessadas nos buffers de página e troca as páginas que são usadas com menos frequência. No entanto, se uma tabela grande for consultada, todos os buffers de página podem ser trocados apenas para realizar uma varredura na tabela. Por exemplo, em um banco de dados com 200 buffers de página, se uma tabela com 250 páginas for consultada, o DBMaker pode ler todas as 250 páginas nos buffers de página e descartar as 200 páginas mais frequentemente usadas. No pior cenário, o DBMaker deve ler 200 páginas do disco ao acessar outros dados após uma varredura completa da tabela. No entanto, se o modo de cache da tabela estiver definido como NOCACHE, o DBMaker colocará as páginas recuperadas no final da cadeia LRU quando uma varredura completa da tabela for realizada. Portanto, 199 das 200 páginas mais frequentemente usadas ainda são mantidas no cache de buffer.

Normalmente, as tabelas com números de página que excedem os buffers de página devem ser definidas como NOCACHE. Tabelas que não são usadas com frequência ou com números de página próximos ao número de buffers de página também devem ser definidas como NOCACHE.

Exemplo 1:

Para determinar o número de páginas e o modo de cache para uma tabela:

```
dmSQL> SELECT TABLE_OWNER, TABLE_NAME, NUM_PAGE, CACHEMODE FROM
SYSTEM.SYSTABLE
WHERE TABLE_OWNER != 'SYSTEM';
TABLE_OWNER TABLE_NAME   NUM_PAGE  CACHEMODE
=========== =========== ========== =========
BOSS           salary        5         T
MIS            asset        45         T
MIS            department    3         T
MIS            employee     29         T
MIS            worktime     450        T
TRADE          customer     350        T
```

```
TRADE          inventory    167        T
TRADE          order        112        T
TRADE          transaction  1345       F
9 rows selected
```

- **NUM_PAGE** — o número de páginas em uma tabela
- **CACHEMODE** — modo de cache da varredura completa da tabela; 'T' significa que a varredura da tabela é armazenável em cache, e 'F' significa que a varredura da tabela não é armazenável em cache.

No exemplo acima, a tabela **TRADE.transaction** já está definida como NOCACHE. As outras tabelas ainda são armazenáveis em cache. Se houver 200 buffers de página, as tabelas **MIS.worktime** e **TRADE.customer** devem ser definidas como NOCACHE, e as tabelas **TRADE.order** e **TRADE.inventory** devem ser definidas como NOCACHE se forem raramente usadas.

Exemplo 2:

Para definir o modo de cache de uma tabela como NOCACHE:

```
dmSQL> ALTER TABLE MIS.worktime SET NOCACHE ON;
```

Se não houver índices válidos para uma tabela ou se o predicado em uma consulta referenciar colunas não indexadas, o DBMaker também pode realizar uma varredura completa da tabela. Para evitar isso, procure escrever instruções SQL da maneira mais eficiente possível e utilize colunas indexadas sempre que possível.

POOR CLUSTERING OF RECORDS

Ao recuperar muitos registros que devem ser ordenados por uma chave de índice, ou quando o predicado referencia uma coluna indexada, a aglomeração de índices se torna um fator importante que afeta o desempenho do cache de buffer.

Exemplo 1: Para selecionar todas as colunas da tabela **tb_customer** e ordená-la usando a chave primária **custid**:

```
dmSQL> SELECT * FROM tb_customer ORDER BY custid;
```

Suponha que haja 3.500 registros na tabela **tb_customer** distribuídos em 350 páginas, e que haja 200 buffers de página no sistema. Se os registros estiverem agrupados usando **custid** e o agrupamento for muito bom (organizado

sequencialmente em todas as páginas), o DBMaker só precisa ler 350 páginas do disco. Por outro lado, se o agrupamento for ruim (sem registros sequenciais na mesma página), o DBMaker pode ter que ler 3.500 páginas do disco no pior caso (cada registro precisa de uma leitura do disco)! Para determinar o estado de um cluster de índice, atualize primeiro as estatísticas da tabela.

Exemplo 2:

Para criar um índice chamado **custid_index** na coluna **custid** da tabela **tb_customer**:

```
dmSQL> SELECT CLSTR_COUNT FROM SYSTEM.SYSINDEX
 WHERE TABLE_OWNER = 'TRADE'
 AND TABLE_NAME = 'tb_customer'
 AND INDEX_NAME = 'custid_index';
```

Resultado:

```
CLSTR_COUNT
===========
 385
1 rows selected
```

CLSTR_COUNT — contagem de clusters, que representa o número de páginas de dados que serão recuperadas por uma varredura totalmente indexada com poucos buffers. O DBMaker realiza no máximo 385 leituras de página do disco quando a tabela completa de clientes é escaneada e os resultados são ordenados pela coluna **custid**.
Exemplo 3:

Para recuperar o número de páginas e registros:

```
dmSQL> SELECT NUM_PAGE,NUM_ROW FROM SYSTEM.SYSTABLE
 WHERE TABLE_OWNER = 'TRADE'
 AND TABLE_NAME = 'tb_customer';
```

Resultado:

```
Result:
NUM_PAGE NUM_ROW
=========== ===========
   350          4375
```

```
1 rows selected
```

NUM_PAGE — o número de páginas alocadas por uma tabela

NUM_ROW — o número de registros em uma tabela

Com **CLSTR_COUNT**, **NUM_PAGE** e **NUM_ROW**, estime o fator de agrupamento com a seguinte fórmula:

$$\text{clustering factor} = \frac{(\text{CLSTR\_COUNT} - \text{NUM\_PAGE})}{\text{NUM\_ROW}}$$

In the above example, the clustering factor is 0.8%.

$$\text{clustering factor} = \frac{(385 - 350)}{4375}$$
$$= 0.008$$
$$= 0.8\%$$

O fator de agrupamento estará entre 0 e 100%. Em casos onde **CLSTR_COUNT** é apenas um pouco menor que **NUM_PAGE**, pode ser tratado como zero. Se o fator de agrupamento for zero, isso significa que os dados estão totalmente agrupados para o índice. Se o fator de agrupamento for muito alto, por exemplo, maior que 20% (o que determina uma alta taxa depende do tamanho da tabela, tamanho médio dos registros, etc.), o índice apresenta um agrupamento ruim. Quando o DBMaker encontra um índice com agrupamento ruim, o otimizador do DBMaker pode optar por realizar uma varredura completa da tabela quando uma instrução SQL é executada, mesmo que uma varredura de índice pareça mais apropriada.

 Para melhorar o agrupamento ruim de um índice frequentemente usado:

1. Descarregue todos os dados da tabela (ordenados pelo índice)
2. Reorganize os dados descarregados por ordem
3. Exclua os índices da tabela
4. Delete todos os dados na tabela
5. Recarregue os dados na tabela
6. Recrie os índices na tabela

Após o recarregamento dos dados, o índice estará totalmente agrupado. Note, entretanto, que uma tabela pode ser agrupada apenas com um índice. Se uma tabela tiver muitos índices, mantenha o agrupamento no índice mais importante. Normalmente, o índice mais importante é a chave primária. Como descarregar e recarregar dados leva muito tempo e espaço de armazenamento, ajuste o

agrupamento de índices apenas nas tabelas que são muito grandes e frequentemente consultadas.

LOW DATA PAGE BUFFERS

Se os buffers de página de dados alocados não forem suficientes para o acesso ao banco de dados, adicione buffers de página ao DCCA.

Para modificar o número de buffers de página:

1. Finalize o servidor de banco de dados
2. Redefina **DB_NBufs** no arquivo **dmconfig.ini** para um valor maior
3. Reinicie o banco de dados

Após aumentar com sucesso os buffers de dados, execute o banco de dados por um período e, em seguida, monitore o desempenho do cache de buffer novamente. Se a razão de acertos do buffer tiver aumentado, a adição de páginas de buffer resultou em uma melhoria no desempenho. Se não, adicione mais páginas ao cache de buffer ou verifique outros motivos pelos quais o desempenho do sistema pode estar reduzido.

CHECKPOINTS OCCURRING TOO OFTEN

Se a razão de acertos de gravação for muito inferior à razão de acertos de leitura, os pontos de verificação (checkpoints) podem estar sendo processados com muita frequência.

Quando um ponto de verificação é processado, o DBMaker grava todos os buffers de página sujos no disco. Como os pontos de verificação exigem muito tempo de CPU, especifique o daemon de pontos de verificação para realizar um ponto de verificação em um cronograma regular. Outra vantagem de realizar pontos de verificação periodicamente é reduzir o tempo de recuperação necessário pelo DBMaker para iniciar um banco de dados após uma falha no sistema.

Exceto quando o daemon de pontos de verificação realiza pontos de verificação regularmente, o DBMaker executará pontos de verificação automaticamente quando o espaço livre do diário acabar no modo NÃO-BACKUP ou quando um backup incremental for realizado no modo BACKUP. Para aumentar o intervalo de tempo entre esses tipos de pontos de verificação, amplie o tamanho do diário.

Exemplo:

Para determinar quantos pontos de verificação foram processados:

```
dmSQL> SELECT INFO, VALUE FROM SYSINFO WHERE INFO =
'NUM_CHECKPOINT';
               INFO                             VALUE
============================== ==============================
           NUM_CHECKPOINT                          26
1 rows selected
```

RE-MONITOR THE BUFFER CACHE PERFORMANCE

Após ajustar um sistema com os métodos acima, monitore o desempenho do cache de buffer.

Para monitorar o desempenho do cache de buffer:

1. Execute o banco de dados por um período para garantir que as informações no banco de dados estejam em um estado estável.
2. Redefina os valores das estatísticas na tabela de sistema **SYSINFO** com o seguinte:

```
dmSQL>                SET                SYSINFO                CLEAR;
```

1. Execute o banco de dados por um período de tempo.
2. Obtenha o contador de leitura/gravação da tabela **SYSINFO** e verifique a razão de acertos.

Tuning Journal Buffers

Os buffers de diário armazenam os blocos de diário mais recentemente usados. Com um número suficiente de buffers de diário, o tempo necessário para gravar os blocos de diário no disco ao atualizar dados e para ler blocos de diário do disco ao reverter transações é reduzido.

Se você raramente executa uma longa transação que modifica (insere, exclui, atualiza) muitos registros, pode pular esta seção. Caso contrário, deve-se determinar se há buffers de diário suficientes para o sistema. O número ideal de buffers de diário é a soma dos blocos de diário necessários pelas transações de mais longa duração ao mesmo tempo.

Para estimar o número de buffers de diário, realize o seguinte:

1. Certifique-se de que haja apenas um usuário ativo no banco de dados.
2. Limpe os contadores na tabela **SYSINFO** com o seguinte comando:

```
dmSQL>                    SET                    SYSINFO                    CLEAR;
```

1. Execute a transação que atualizará o maior número de registros.
2. Execute a seguinte instrução SQL para determinar o número de blocos de diário utilizados:

```
dmSQL> SELECT INFO, VALUE FROM SYSINFO WHERE INFO =
'NUM_JNL_BLK_WRITE';
             INFO                              VALUE
============================== ==============================
NUM_JNL_BLK_WRITE                               626
1 rows selected
```

NOTA: ***NUM_JNL_BLK_WRITE*** *— os blocos utilizados nesta transação. O tamanho do bloco de diário utilizado neste exemplo é de 512 bytes. No exemplo acima, você precisaria de aproximadamente 21 páginas de buffer de diário (se definir 1 página = 8 KB).*

Outra medida que pode ser utilizada para determinar a utilização do buffer de diário é a taxa de descarga do buffer de diário. A taxa de descarga do buffer de diário é a porcentagem de buffers de diário descarregados no disco quando o DBMaker grava no diário. Se a taxa de descarga do buffer de diário for muito alta (por exemplo, superior a 50%), aumente o número de buffers de diário.

Exemplo:

Para calcular a taxa de descarga do buffer de diário:

```
dmSQL> SELECT INFO, VALUE FROM SYSINFO WHERE INFO =
'NUM_JNL_BLK_WRITE'
 2> OR INFO = 'NUM_JNL_FRC_WRITE';
             INFO                              VALUE
============================== ==============================
NUM_JNL_BLK_WRITE                             41438
NUM_JNL_FRC_WRITE                               159
2 rows selected
```

NUM_JNL_BLK_WRITE — número de blocos de diário gravados no buffer

NUM_JNL_FRC_WRITE — número de gravações forçadas dos buffers de diário no disco

Suponha que **DB_NJnlB** esteja definido para 50 páginas (ou seja, há 400 buffers de diário). No exemplo abaixo, a taxa de descarga do diário (0,65) é um pouco alta. Adicione buffers de diário para melhorar o desempenho do buffer de diário.

$$\text{journal flush rate} = \frac{(\text{NUM\_JNL\_BLK\_WRITE}/\text{NUM\_JNL\_FRC\_WRITE})}{(\text{DB\_NJNLB} \times 8)}$$
$$= \frac{(41438/159)}{(50 \times 8)}$$
$$= 0.65$$

Tuning the System Control Area (SCA)

Os buffers de cache e alguns blocos de controle, como informações de sessão e transação, têm um tamanho fixo e são pré-alocados do DCCA quando um banco de dados é iniciado. No entanto, alguns blocos de controle de concorrência são alocados dinamicamente do DCCA enquanto o banco de dados está em execução; seu tamanho é especificado por **DB_ScaSz**.

Se um aplicativo de banco de dados receber a mensagem de erro "database request shared memory exceeds database startup setting", isso significa que o DBMaker não pode alocar dinamicamente memória da área SCA. Normalmente, esse erro se deve a uma longa transação que utiliza muitos bloqueios. Se essa situação ocorrer com frequência, resolva-a com os métodos ilustrados abaixo.

AVOID LONG TRANSACTIONS

Uma longa transação ocupará muitos blocos de controle de bloqueio e blocos de diário. Se houver uma longa transação em andamento quando o erro acima ocorrer, analise se a transação pode ser dividida em várias transações menores.

AVOID EXCESSIVE LOCKS ON LARGE TABLES

Selecionar muitos registros de uma tabela grande usando uma varredura de índice requer muitos recursos de bloqueio. Para diminuir a quantidade de recursos de bloqueio usados pela transação, eleve o modo de bloqueio antes de realizar a varredura da tabela.

Por exemplo, se o modo de bloqueio padrão da tabela for por linha, eleve o modo de bloqueio padrão para página ou tabela. Embora isso reduza os recursos utilizados, também sacrificará a concorrência em certa medida.

INCREASE THE SCA SIZE

Se ambas as condições acima não ocorrerem, aumente o tamanho da SCA. Redefina o valor de **DB_ScaSz** no **dmconfig.ini** para um valor maior e, em seguida, reinicie o banco de dados.

Tuning the Catalog Cache

O DBMaker armazena o cache do catálogo na SCA. Se os objetos de esquema forem raramente modificados, ative o modo turbo do dicionário de dados definindo **DB_Turbo = 1** no arquivo **dmconfig.ini**. Quando o modo turbo está ativado, o DBMaker estenderá a vida útil do cache do catálogo. Isso pode melhorar o desempenho dos programas de processamento de transações on-line (OLTP).

19.5 Tuning Concurrent Processing

A contenção de recursos ocorre em um sistema de banco de dados multiusuário quando mais de um processo tenta acessar os mesmos recursos do banco de dados simultaneamente. Isso também pode levar a uma situação conhecida como deadlock, que ocorre quando dois ou mais processos esperam um pelo outro. A contenção de recursos faz com que os processos aguardem o acesso a um recurso do banco de dados, reduzindo o desempenho do sistema.

O DBMaker fornece os seguintes métodos para detectar e reduzir a contenção de recursos:

- Reduzir a contenção de bloqueios
- Limitar o número de processos
- Definir a afinidade de CPU
- Definir a prioridade de trabalho

Reducing Lock Contention

Ao acessar dados em um banco de dados, os processos do DBMaker bloqueiam automaticamente os objetos-alvo (registros, páginas, tabelas). Quando dois processos desejam bloquear o mesmo objeto, um deve esperar. Se mais de dois processos aguardarem que os outros processos liberem o bloqueio, ocorre um deadlock. Quando um deadlock ocorre, o DBMaker sacrificará a última transação que ajudou a causar o deadlock, revertendo-a. O deadlock reduz o desempenho do sistema. Monitore as estatísticas de bloqueio para evitar um deadlock no DBMaker.

Exemplo:

Para visualizar as estatísticas de deadlock:

```
dmSQL> SELECT INFO, VALUE FROM SYSINFO WHERE INFO =
'NUM_LOCK_REQUEST'
 2> OR INFO = 'NUM_DEADLOCK'
 3> OR INFO = 'NUM_STARTED_TRANX';
          INFO                    VALUE
======================   ====================
NUM_STARTED_TRANX                  33
NUM_LOCK_REQUEST                   173
NUM_DEADLOCK                        0
3 rows selected
```

NUM_LOCK_REQUEST — número de solicitações de bloqueio NUM_DEADLOCK — número de deadlocks NUM_STARTED_TRANX — número de transações emitidas

No exemplo acima, em média, uma transação está em deadlock a cada 51 (9287/181) transações, e uma transação solicita aproximadamente 83 (772967/9287) bloqueios.

Se a frequência de deadlock for alta, examine o design do esquema, as instruções SQL e os aplicativos. Definir o modo de bloqueio padrão da tabela mais baixo, como ROW lock, pode reduzir a contenção de bloqueio, mas exigirá mais recursos de bloqueio.

Outro método é usar o modo de leitura para consultar uma tabela em uma consulta longa, se os dados não precisarem permanecer consistentes após o ponto no tempo em que foram recuperados. Isso é útil ao visualizar os dados ou realizar cálculos usando os dados, enquanto não realiza nenhuma atualização. Ele fornece uma instantânea dos dados solicitados em um determinado ponto no tempo, mas com o benefício de aumentar a concorrência e consumir menos recursos de bloqueio, pois os bloqueios são liberados assim que os dados são lidos.

Limiting the Number of Processes

DBMaker permite até 4.800 conexões de sessão simultâneas a um servidor. Se os recursos do servidor (como memória, poder de CPU) não forem suficientes, limite o número máximo de conexões para evitar a contenção de recursos. O parâmetro de configuração **DB_MaxCo** afeta o número máximo de conexões no banco de dados.

Quando um banco de dados é criado, o arquivo de log é formatado para um número específico de conexões. O arquivo de log precisa preservar uma matriz de informações de transações para cada conexão. O número de conexões disponíveis,

de acordo com o arquivo de log, também é conhecido como número de conexões rígido. Este valor é determinado pelo valor de **DB_MaxCo** quando o banco de dados é criado. O número de conexões rígido tem um valor mínimo de 2, um valor máximo de 4.800 e deve ser um múltiplo de 40. Se **DB_MaxCo** for definido como um valor que não é um múltiplo de 40, então o número de conexões rígido é arredondado para cima para um valor que é um múltiplo de 40. O número de conexões rígido é uma limitação do arquivo de log, portanto, para alterá-lo, o usuário deve reconfigurar **DB_MaxCo** e iniciar em modo de log novo (**DB_SMode=2**).

A seguinte fórmula é usada para calcular o número de conexões rígidas:

Número de conexões rígidas = (DB_MaxCo + números de conexões reservadas + 40 - 1) / 40 * 40

Atualmente, o DBMaker possui 20 conexões reservadas para suportar conexões internas, como daemon, uso administrativo, etc.

O número de conexões rígidas não afeta diretamente o tamanho do **DCCA**. Isso é determinado por um valor conhecido como número de conexões suaves. Um número maior de conexões significa mais memória, tanto no servidor quanto no cliente, portanto, mesmo que o número de conexões rígidas de um banco de dados seja grande, os usuários podem iniciar o banco de dados com um número menor de conexões suaves para economizar memória. O número de conexões suaves é determinado pelo valor de **DB_MaxCo** quando o banco de dados é iniciado. O número de conexões suaves determina o número de conexões que o **DCCA** suportará e, consequentemente, o uso de memória do **DCCA**. O número de conexões suaves pode ser qualquer valor menor ou igual ao valor de conexões rígidas. Para alterar o número de conexões suaves, reinicie o banco de dados normalmente após alterar **DB_MaxCo**.

A seguinte fórmula é usada para calcular o número de conexões suaves:

Número de conexões suaves = DB_MaxCo + números de conexões reservadas.

Exemplo 1: No arquivo de configuração a seguir, o número de conexões rígidas para **DB1** é 240. Para o banco de dados **DB2**, é 1120.

```
[DB1]
DB_MaxCo = 50 ;; the hard connection number is 240
;; the soft connection number is 70
[DB2]
DB_MaxCo = 1100 ;; the hard connection number is 1120
;; the soft connection number is 1120
```

Exemplo 2: Após iniciar o banco de dados com sucesso, o novo número de conexões rígidas para **DB1** torna-se 320.

```
[DB1]
DB_SMode = 2 ;; startup with new journal file
DB_MaxCo = 280 ;; the new hard connection number is 320
```

Exemplo 3: Supondo que **DB2** já tenha sido criado, conforme no exemplo 1, a seguinte entrada no arquivo **dmconfig.ini** resultará em um número de conexões rígidas de 1120 e um número de conexões suaves de 40.

```
[DB2]
DB_SMode = 1 ;; normal start
DB_MaxCo = 20 ;; the new soft connection number is 40
```

Setting CPU Affinity

Para melhorar o desempenho do multitasking, o sistema operacional distribui processos e threads entre diferentes CPUs. Embora isso seja eficiente do ponto de vista do sistema operacional, essa atividade pode reduzir o desempenho do DBMaker sob cargas pesadas do sistema, já que o cache de cada processador é recarregado repetidamente com dados. Se cada processo for executado na CPU na qual foi executado da última vez, o DBMaker terá um desempenho melhor. Nesse momento, o usuário pode obter a afinidade do processo consultando a tabela sysuser e, em seguida, usar o procedimento armazenado do sistema SETAFFINITY para definir uma nova máscara de afinidade. Dessa forma, o processo será executado apenas em uma CPU específica.

Diferentemente dos sistemas operacionais antigos, a afinidade de CPU é um recurso dos sistemas operacionais modernos, como Linux 2.6, Windows NT e Windows 98. Existem dois tipos de afinidade de CPU: uma é a afinidade suave, que o usuário não pode alterar, e a outra é a afinidade rígida, que o usuário pode modificar.

A afinidade de CPU é definida pela máscara de afinidade. A máscara de afinidade é um vetor de bits no qual cada bit representa um processador. O DBMaker define a máscara de afinidade como char(64), permitindo assim o uso de até 64 CPUs.

Exemplo 1:

Aqui estão os valores da máscara de afinidade para um sistema com 8 CPUs (omitindo os zeros contínuos na posição alta):

Decimal value	Binary bit mask	Allow run on CPU
1	'1'	0
3	'11'	0 and 1
7	'111'	0, 1, and 2
15	'1111'	0, 1, 2, and 3
31	'11111'	0, 1, 2, 3, and 4
63	'111111'	0, 1, 2, 3, 4, and 5
127	'1111111'	0, 1, 2, 3, 4, 5, and 6
255	'11111111'	0, 1, 2, 3, 4, 5, 6, and 7

Usando GETCPUNUMBER e SETAFFINITY, os usuários podem obter o estado atual do sistema e definir a afinidade da CPU de uma conexão sem reiniciar o DBMaker durante a execução.

O protótipo para GETCPUNUMBER é:

```
GETCPUNUMBER (INT CPU_NUMBER OUTPUT)
```

CPU_NUMBER: parâmetro de saída, o número de processadores lógicos na máquina.

O protótipo para SETAFFINITY é:

```
SETAFFINITY (INT CONNECTION_ID INPUT, CHAR(64) AFFINITY_MASK INPUT)
```

CONNECTION_ID: parâmetro de entrada, o ID das conexões ou servidores. O usuário pode obtê-lo com "select connection_id from sysuser" ou verificando o monitor do sistema. É o ID da thread no Windows e o ID do processo em sistemas Unix-like.

AFFINITY_MASK: parâmetro de entrada, máscara de afinidade da CPU. A máscara de afinidade válida é composta por '1' ou '0'. '1' significa que a CPU é válida para a conexão; '0' significa que a CPU é inválida para a conexão.

Exemplo 2:

Os usuários devem obter algumas informações do sistema antes de definir a afinidade da CPU, como o número de CPUs no servidor, o uso de CPU de cada conexão e a máscara de afinidade correta. Para obter o número de CPUs, deve-se chamar GETCPUNUMBER:

To get the number of CPU by calling GETCPUNUMBER:

```
dmSQL> CALL GETCPUNUMBER(?);
```

Para obter o uso de CPU de cada conexão e a máscara de afinidade correta:

```
dmSQL> SELECT connection_id, affinity_mask, priority_level, cpu_usage FROM
sysuser;
```

Para definir a afinidade da CPU para o sysuser e permitir que a conexão seja executada nas CPUs 0 e 1:

```
dmSQL> SELECT connection_id , user_name FROM sysuser;
 CONNECT* USER_NAME
========= ==============
30420     BACKUP_SERVER
30418         SYSADM

2 rows selected
dmSQL> CALL SETAFFINITY(30418,'11');
```

NOTA: *Apenas o sysadm pode chamar SETAFFINITY.*

Para obter a máscara de afinidade da CPU consultando o sysuser para uma conexão específica:

```
dmSQL> SELECT affinity_mask FROM sysuser WHERE connection_id = ?;
```

Setting Job Priority

A prioridade dos processadores e threads é uma característica básica em sistemas operacionais de multitarefa. O escalonador verificará a carga do sistema e mudará o nível de prioridade de um trabalho, se necessário. O DBMaker permite que o usuário defina a prioridade de uma conexão para melhorar o desempenho de todo o sistema. Por exemplo, se há um trabalho de longa duração que consome muito tempo de CPU, as outras conexões terão que esperar por muito tempo. Se os usuários reduzirem a prioridade do trabalho longo, as outras conexões terão mais tempo de CPU para serem executadas, melhorando assim o desempenho de todo o sistema. Os usuários podem definir novos valores para algumas conexões importantes para melhorar o desempenho dessas conexões, enquanto outras conexões podem ter um desempenho inferior.

Usando GETCPUNUMBER e SETPRIORITY, os usuários podem obter o estado atual do sistema e definir a prioridade de uma conexão sem reiniciar o DBMaker durante a execução. O protótipo para SETPRIORITY é:

SETPRIORITY (INT CONNECTION_ID INPUT,INT PRIORITY_LEVEL INPUT)

CONNECTION_ID: parâmetro de entrada, o ID das conexões ou servidores. O usuário pode obtê-lo com "select connection_id from sysuser" ou verificando o monitor do sistema. É o ID da thread no Windows e o ID do processo em sistemas Unix-like.

PRIORITY_LEVEL: parâmetro de entrada, existem cinco níveis, sendo que o nível de prioridade normal e padrão é três. Os níveis de prioridade válidos são '1', '2', '3', '4' e '5'. '1' significa prioridade mais baixa; '2' significa prioridade inferior; '3' significa prioridade normal; '4' significa prioridade superior; '5' significa prioridade mais alta.

NOTA: *Os usuários não podem definir uma prioridade mais alta no Linux, pois isso requer privilégios de root. Portanto, você só pode definir um nível mais baixo no Linux, mas não há limites no Windows.*

Exemplo:

O usuário deve obter algumas informações do sistema antes de definir a afinidade da CPU, como o número de CPUs no servidor, o uso de CPU de cada conexão e a prioridade. Para obter o número de CPUs, deve-se chamar GETCPUNUMBER:

```
dmSQL> CALL GETCPUNUMBER(?);
```

Para obter o uso de CPU de cada conexão e a prioridade:

```
dmSQL> SELECT connection_id, affinity_mask, priority_level,
cpu_usage FROM
sysuser;
```

Para definir o nível de prioridade para o sysuser:

```
dmSQL> SELECT connection_id, user_name FROM sysuser;
 CONNECT* USER_NAME
=============== ================
30420             BACKUP_SERVER
 30418                SYSADM
```

```
2 rows selected
dmSQL> CALL SETPRIORITY(30418,3);
```

NOTA: *Apenas o SYSADM pode chamar SETPRIORITY.*

Para obter o nível de prioridade consultando o sysuser para uma conexão específica:

```
dmSQL> SELECT priority_level FROM sysuser WHERE connection_id = ?;
```