7. Large Object Management

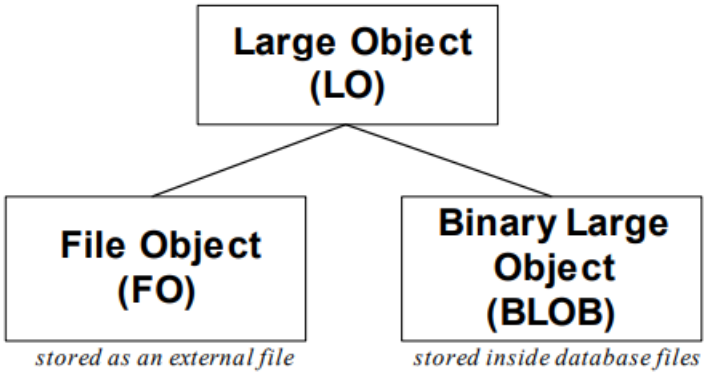
Um Objeto Grande (LO, do inglês Large Object) é qualquer dado de tamanho variável, como texto de documento, imagens, som ou vídeo. O DBMaker possui alta flexibilidade ao lidar com objetos grandes e oferece um excelente mecanismo para dados não estruturados.

O DBMaker não limita a quantidade de LOs (Objetos Grandes) que podem estar em uma tabela, e não há um limite de tamanho total para colunas LO. Isso significa que a capacidade de cada coluna LO pode chegar a 2 GB. O DBMaker pode usar extensões da linguagem SQL para acessar diretamente Objetos Grandes, eliminando a necessidade dos usuários aprenderem qualquer sintaxe especial. Todo o acesso às colunas LO é transparente nas instruções SQL, o que facilita o aprendizado do uso de objetos grandes. Além disso, os usuários podem inserir ou extrair dados LO de e para um arquivo usando comandos SQL ou a interface ODBC. Um LO é sempre gravado em disco como uma única unidade. No entanto, os usuários podem ler toda a parte de um LO. As instruções SELECT, UPDATE, INSERT e DELETE são permitidas com LOs. Itens LO (Objetos Grandes) só podem ser usados em expressões booleanas se os usuários quiserem testá-los para valores NULL. O DBMaker também fornece a função MATCH para uso com LOs para realizar pesquisas com correspondência de padrões. A função MATCH é semelhante à função LIKE, exceto que ela só funciona em colunas LO e não permite o uso de caracteres curinga (wildcards).

O DBMaker não permite a realização de operações aritméticas ou expressões de string em itens LO (Objetos Grandes). Itens LO também não podem ser usados das seguintes maneiras:

- Com funções de agregação
- Com predicados IN, ANY, EXIST ou LIKE
- Com a cláusula GROUP BY
- Com a cláusula ORDER BY

Existem dois tipos de LOs (Objetos Grandes): Objetos Grandes Binários (BLOBs), que são armazenados em arquivos do banco de dados, e Objetos de Arquivo (FOs), que são armazenados como arquivos externos no sistema de arquivos do host.



Um BLOB, armazenado em arquivos de banco de dados, só pode ser acessado através do DBMaker e insiste na integridade de dados fornecida pelo DBMaker, como controles de transação, registro e recuperação. Um BLOB só pode ser compartilhado entre tuplas na mesma tabela durante a atualização de registros. No entanto, um FO pode ser compartilhado entre tabelas em um banco de dados. Além disso, quando os dados precisam ser compartilhados por outros aplicativos não relacionados ao banco de dados, usar FOs será mais flexível.

7.1 Managing BLOBs

Existem dois tipos de itens BLOB: LONG VARCHAR (ou CLOB) e LONG VARBINARY. Dados do tipo LONG VARCHAR podem consistir em qualquer tipo de dado textual, como memorandos, texto longo, arquivos de origem HTML ou arquivos de origem de programa. O tipo de dado LONG VARBINARY pode armazenar qualquer tipo de dado binário, incluindo imagens, sons, planilhas e módulos de programa.

Um BLOB pode ser armazenado em um arquivo de dados ou em um arquivo BLOB específico, dependendo do seu tamanho. Embora o formato de um arquivo de dados seja fixo, o formato dos arquivos BLOB no banco de dados deve ser customizado para obter melhor desempenho e utilização do disco.

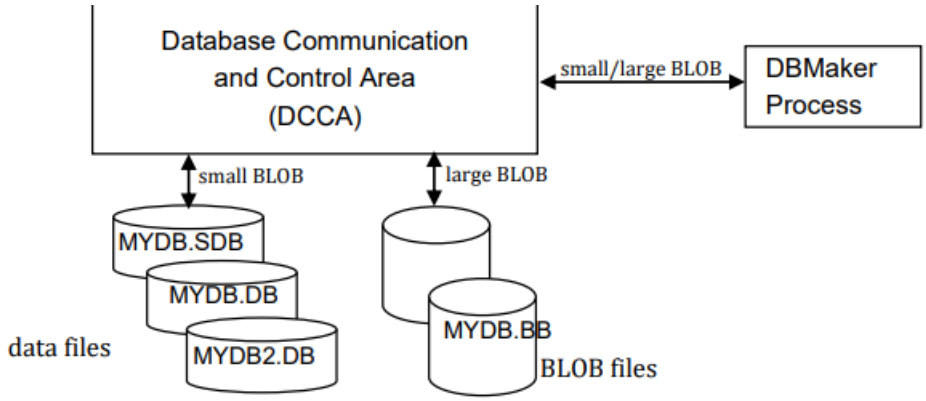
A escolha do registro de BLOB é opcional porque ocupa uma grande quantidade de espaço no journal e pode prejudicar o desempenho. Para economizar espaço de registro e melhorar o desempenho, o journal de BLOB pode ser desativado. No entanto, se o registro de BLOB for desativado, o DBMaker não garantirá que o conteúdo do BLOB esteja correto após a restauração do banco de dados a partir de um backup. Se o journal de BLOB estiver ativado, certifique-se de que o arquivo de journal tenha espaço suficiente para acomodar os dados BLOB.

Customizing BLOB Space

O DBMaker decide automaticamente onde armazenar dados BLOB. Se o tamanho de uma coluna LONG VARCHAR ou LONG VARBINARY for pequeno e o comprimento total de uma tupla não ultrapassar o limite para o tamanho máximo da tupla, o DBMaker colocará os dados BLOB em uma coluna junto com os outros dados no banco de dados. Isso aumenta a eficiência porque os dados BLOB também são obtidos quando o DBMaker busca uma tupla.

Quando o comprimento total de uma tupla de dados excede a limitação do tamanho máximo da tupla, o DBMaker armazena os dados BLOB separadamente. Nessa situação, obter os dados BLOB (chamado de BLOB indireto) requer duas operações de disco: uma para buscar a tupla de dados e outra para buscar os dados BLOB

De acordo com seu tamanho, um BLOB indireto pode ser armazenado em um arquivo DATA ou em um arquivo BLOB no mesmo tablespace da tabela. Os dados em uma coluna BLOB indireta são armazenados em um arquivo de dados quando seu tamanho é igual ou menor que 16.2402 bytes (considerando um tamanho de página de 16 KB), caso contrário, eles são armazenados em um arquivo BLOB.



Arquivos de dados contêm páginas, e arquivos BLOB contêm quadros. Existem duas principais diferenças entre páginas e quadros:

- Existem quatro valores para escolher o tamanho da página: 4 KB, 8 KB, 16 KB ou 32 KB. O tamanho da página é definido usando a palavra-chave DB_PgSiz ao criar bancos de dados, enquanto o tamanho de um quadro pode ser personalizado.
- Uma página pode conter mais de uma tupla, mas um quadro só pode conter um único BLOB.

O tamanho do quadro de um arquivo BLOB pode ser personalizado antes da criação do banco de dados para melhorar o desempenho e a utilização do disco. Para personalizar o tamanho do quadro, especifique o valor em kilobytes da palavra-chave

de configuração DB_BfrSz no arquivo dmconfig.ini. O valor padrão de DB_BfrSz é 32. Consulte a Seção 4.2, Criando um Banco de Dados, para obter mais informações sobre os parâmetros de configuração que devem ser definidos antes da criação do banco de dados

² Este valor é de 3952 bytes em um tamanho de página de 4 KB, 8048 bytes em um tamanho de página de 8 KB, 16240 bytes em um tamanho de página de 16 KB e 32624 bytes em um tamanho de página de 32 KB, respectivamente.

Exemplo 1:

Para especificar o tamanho do quadro BLOB adicionando uma linha ao arquivo dmconfig.ini:

```
DB_BfrSz = 16 ; BLOB frame size = 16K bytes
```

O intervalo válido de DB_BfrSz é de 8 a 256.

O tamanho do quadro de todos os arquivos BLOB em um banco de dados é o mesmo. Uma vez que um banco de dados é criado, o tamanho do quadro BLOB não pode ser alterado de sua configuração inicial. O DBMaker irá manter esse valor na tabela de informações do sistema do banco de dados. Quando o banco de dados é reiniciado, o DBMaker obtém o valor original da página de informações do sistema e ignora a palavra-chave DB_BfrSz no arquivo dmconfig.ini.

Exemplo 2:

Para consultar o tamanho do quadro na tabela do sistema SYSINFO:

```
dmSQL> SELECT INFO, VALUE FROM SYSINFO WHERE INFO =
'FRAME_SIZE';
 INFO VALUE
============================== ==============================
FRAME_SIZE 16384
1 rows selected
```

Determinar o tamanho do quadro é uma questão de equilíbrio entre utilização do disco e desempenho. Se BLOBs inteiros forem recuperados frequentemente, ajustar o tamanho do quadro para conter o BLOB inteiro resultará em melhor desempenho porque apenas um acesso ao disco será necessário. No entanto, pode haver grandes variações no tamanho dos dados BLOB. Se o tamanho do quadro for definido como grande o suficiente para conter o maior BLOB, ele poderá desperdiçar espaço em

disco, pois outros quadros que contêm BLOBs menores conterão espaço em disco não utilizado.

Alternativamente, se o tamanho do quadro for apenas o suficiente para conter os BLOBs menores, o desempenho será prejudicado ao buscar BLOBs maiores que estejam armazenados em vários quadros.

Cada quadro contém um cabeçalho para registrar informações do quadro. Se o tamanho do quadro for de 8 KB, por exemplo, o espaço ocupado pelo BLOB será menos de 8.192 bytes. Cerca de 1,8 KB é reservado para armazenar informações (como onde estão outros quadros) para cada item BLOB, então o espaço utilizável no primeiro quadro de um BLOB é bem menor que o tamanho do quadro inteiro. Portanto, se o tamanho real de um BLOB for 8.192 bytes, ele ocupará dois quadros: os primeiros 6.2 KB do BLOB serão armazenados no primeiro quadro e os bytes restantes do BLOB serão armazenados no segundo quadro.

Um grupo contém uma página BE e N blocos PE NBE3, a página BE é uma página de dados de TAMANHO_PÁGINA_KB KB. Então, cada bloco PE contém NPE4 + 1 quadros. O primeiro quadro é uma página PE de TAMANHO_PÁGINA5 KB. Os quadros NPE restantes são para dados, e seu tamanho é determinado por DB_BfrSz.

[3]NBE especifica que o número de blocos PE é controlado por uma página BE. O valor é 2004 para tamanho de página de 4 KB, 2026 para tamanho de página de 8 KB, 2716 para tamanho de página de 16 KB e 2723 para tamanho de página de 32 KB, respectivamente.

NPE especifica que o número de páginas é controlado por uma página PE. O valor é 165 para tamanho de página de 4 KB, 333 para tamanho de página de 8 KB, 671 para tamanho de página de 16 KB e 1347 para tamanho de página de 32 KB, respectivamente

O TAMANHO_PÁGINA é definido pela palavra-chave DB_PgSiz no arquivo `dmconfig.ini`

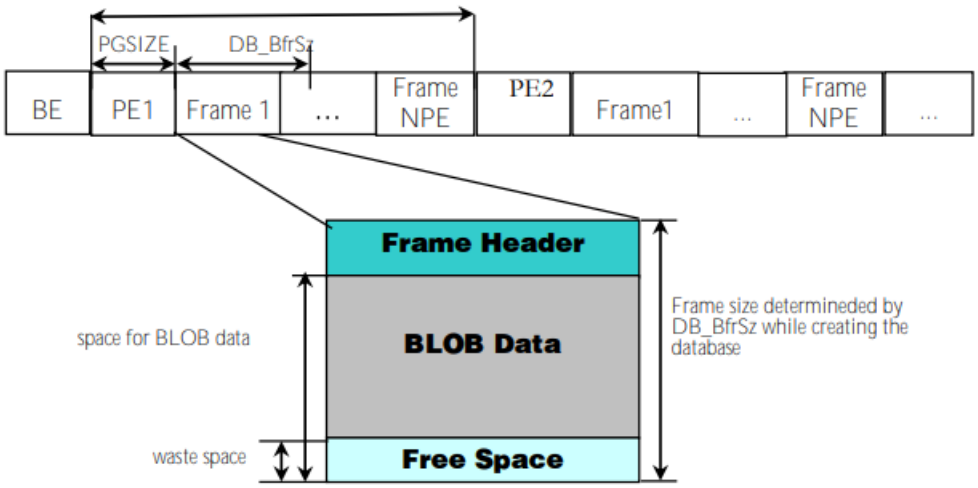


Figure 7-3: the structure of a BLOB file

Usuários podem calcular o tamanho de um arquivo BLOB de acordo com o número total de quadros, páginas PE, páginas BE e quadros de dados.

A fórmula a seguir mostra como calcular aproximadamente o tamanho de um BLOB em KB:

Número de páginas BE: [total de quadros / 6766856], onde [] significa arredondar para o inteiro superior mais próximo.

Número de páginas PE: [(total de quadros - número de páginas BE) / NPE + 1]

Tamanho do arquivo BLOB (aproximadamente em KB): (Número de páginas BE + Número de páginas PE) × TAMANHO_PÁGINA KB + (total de quadros - Número de páginas BE - Número de páginas PE) × DB_BfrSz KB

Por exemplo, se o tamanho da página for de 8 KB e o tamanho do quadro BLOB for de 32 KB, o tamanho de um arquivo BLOB com 3 quadros é

Number of BE pages = [3 / 676685] = 1

Number of PE pages = [(3 - 1) / 333 + 1] = 1

Esse valor depende do tamanho da página, sendo 332665 para um tamanho de página de 4 KB, 676685 para um tamanho de página de 8 KB, 1825153 para um tamanho de página de 16 KB e 3670605 para um tamanho de página de 32 KB.

BLOB file size = (1 + 1) × 8 + (3 - 1 - 1) × 32 = 48 KB

DB_BbFil especifica o nome do arquivo BLOB do sistema na tablespace do sistema, SYSTABLESPACE. Os usuários não podem especificar o tamanho do arquivo BLOB

do sistema. O nome de arquivo padrão para o arquivo BLOB do sistema é o nome do banco de dados concatenado com '.SBB'.

DB_UsrBb especifica o nome do arquivo BLOB padrão do usuário na tablespace padrão, DEFTABLESPACE, e também define o seu tamanho.

Para obter mais detalhes sobre como adicionar novos arquivos BLOB a um tablespace de usuário existente, consulte a subseçãoexpandindo uniformemente um tablespace com autoextensão

Existem três maneiras de expandir um tablespace com autoextensão:

- **Expansão sempre do primeiro arquivo:** Essa abordagem pode obter bom desempenho, mas nem todos os arquivos do tablespace serão expandidos uniformemente.
- **Expansão sempre do menor arquivo:** Essa opção mantém todos os arquivos do tablespace com expansão uniforme, mas o desempenho pode ser ruim porque todas as linhas de uma tabela podem ficar espalhadas por todos os arquivos.
- **Expansão híbrida:** Primeiro expande o menor arquivo e, em seguida, continua expandindo esse arquivo, sem mudar para o segundo menor arquivo até que o tamanho do arquivo atual seja maior que a soma do tamanho do segundo menor arquivo e do valor de DB_ExtHd. Dessa forma, é possível levar em consideração tanto o desempenho quanto o balanceamento do tamanho do arquivo.

Em todos os métodos citados acima, o DBMaker passará para o próximo arquivo para expansão se o arquivo selecionado não puder ser expandido devido a disco cheio, limitação do sistema de arquivos ou limitação de armazenamento do DBMaker. Se o próximo arquivo também não puder ser expandido pelo mesmo motivo, o DBMaker passará para o próximo arquivo até que todos os arquivos sejam expandidos. Consulte DB_ExtHd para obter mais informações.

Todas essas funções podem ser realizadas com a JConfiguration Tool ou por meio de uma combinação de comandos SQL e modificações no arquivo dmconfig.ini. O exemplo a seguir mostra como expandir um tablespace com autoextensão editando o arquivo dmconfig.ini e usando comandos SQL.

Defina a estratégia de expansão de um tablespace autoextensível usando a palavra-chave DB_ExtNp e DB_ExtHd ou chamando SETSYSTEMOPTION('EXTHD','novoValor'). O DBMaker expandirá automaticamente o tablespace com base na estratégia definida pelo usuário.

No arquivo de configuração dmconfig.ini, especifique que o tamanho para expansão do tablespace pelo DBMaker seja de 30 páginas e o valor limite para expansão repetida de um arquivo seja de 100 páginas.

```
DB_ExtNp=30
DB_ExtHd=100
```

Adicione os cinco arquivos seguintes modificando o arquivo dmconfig.ini:

```
D1=/home/dbmaker/testdb/D1 10
B1=/home/dbmaker/testdb/B1 30
D2=/home/dbmaker/testdb/D2 50
B2=/home/dbmaker/testdb/B2 70
D3=/home/dbmaker/testdb/D3 100
```

Digite os seguintes comandos no prompt do dmSQL para criar o tablespace autoextensível TS.

```
dmSQL> CREATE AUTOEXTEND TABLESPACE TS DATAFILE D1 TYPE=DATA, B1
TYPE=BLOB, D2
TYPE=DATA, B2 TYPE=BLOB, D3 TYPE=DATA;
```

Crie a tabela tb_t1 com a coluna c1 char(5000) no tablespace TS e insira várias linhas na tabela tb_t1. O tablespace TS será expandido automaticamente. De acordo com a nova regra, os arquivos serão expandidos da seguinte maneira:

```
1st, extend the smallest file D1, add 30 pages. Now, D1=40, D2=50,
D3=100.
2nd, extend D1, add 30 pages. Now, D1=70, D2=50, D3=100.
3rd, extend D1, add 30 pages. Now, D1=100, D2=50, D3=100.
4th, extend D1, add 30 pages. Now, D1=130, D2=50, D3=100.
5th, extend D1, add 30 pages. Now, D1=160, D2=50, D3=100.
6th, extend D2, because D1 > D2+EXTHD. Add 30 pages, D1=160, D2=80,
D3=100.
7th, extend D2, until D2 > D3(the smallest)+EXTHD, then extend D3.
```

NOTA: *Os arquivos B1 e B2 não serão expandidos, pois não há campo BLOB na tabela tb_t1.*

Exemplo 2:

Durante a execução, os usuários podem chamar o procedimento armazenado do sistema SETSYSTEMOPTION para alterar a opção de sistema EXTHD.

```
dmSQL> CALL SETSYSTEMOPTION('EXTHD','1000'); // changing EXTHD to
10000                                                          pages
```

Durante a execução, os usuários podem chamar o procedimento armazenado do sistema GETSYSTEMOPTION para mostrar o valor de EXTHD.

```
dmSQL> CALL GETSYSTEMOPTION('EXTHD',?); //reporting the current
value of EXTHD
```

Adicionando Arquivos aos Espaços de Tabelas na Seção 5.3

Generating BLOBs

Uma coluna BLOB é igual às outras colunas, exceto que seu tipo de dado é LONG VARCHAR ou LONG VARBINARY.

Exemplo:

Para criar duas colunas BLOB chamadas note e photo

```
dmSQL> CREATE TABLE tb_staff (id INTEGER, note LONG VARCHAR, photo
LONG
VARBINARY);
```

Inserir dados BLOB do arquivo ab.txt e do arquivo de imagem img001.gif usando variáveis de host:

```
dmSQL>        INSERT        INTO        tb_staff        VALUES(2,?,?);
dmSQL/Val>                 &ab.txt,                &img001.gif(2,4);
dmSQL/Val>                                                      END;
```

A coluna LONG VARBINARY resultante é representada em formato hexadecimal. Os seguintes resultados serão retornados ao navegar pela tabela:

```
dmSQL> SELECT * FROM tb_staff;
id note photo
====== =========== ================
 2 <script lan ffd8ffe000104a464
```

DBMaker also supports fetching BLOB data into a user-specified file. For more information on how to insert and fetch BLOB data, refer to the JDBA Tool User's Guide and the ODBC Programmer's Guide.

Updating BLOBs

Um item BLOB sempre é escrito no disco como um todo. Portanto, ao atualizar uma coluna BLOB, o DBMaker irá descartar o item BLOB original e então inserir os novos dados como um novo item BLOB.

Exemplo:

Para atualizar o conteúdo de uma coluna BLOB usando o comando UPDATE:

```
dmSQL> UPDATE tb_staff SET note = 'Hello !' WHERE id > 0;
dmSQL> SELECT * FROM tb_staff;
id note photo
===== =========== ================
 1 Hello ! 31323334353637
 2 Hello ! 33343536
```

Do ponto de vista do usuário, deve haver um BLOB para cada tupla. No entanto, para economizar espaço em disco, o DBMaker cria apenas uma única cópia dos dados BLOB compartilhados por todas as tuplas com um ID maior que zero. O DBMaker mantém um contador interno para registrar quantas tuplas referenciam um BLOB. Ao atualizar uma coluna BLOB para uma tupla que se refere ao BLOB compartilhado, o DBMaker gera um novo item BLOB e decrementa o contador do BLOB compartilhado em um. Isso impede que quaisquer alterações feitas na coluna BLOB afetem outras tuplas. No DBMaker, isso é conhecido como acoplamento fraco. Isso torna a utilização de disco mais eficiente, mas um item BLOB só pode ser compartilhado por tuplas que estão na mesma tabela. Se um BLOB não estiver vinculado a tuplas, o DBMaker o elimina automaticamente.

Predicate Operations on BLOB Columns

Objetos BLOB só podem ser usados em expressões CONTAIN, MATCH ou booleanas ao testar por valores NULL.

Exemplo 1:

Para buscar todos os dados da tabela tb_staff da coluna note que não sejam NULL, você pode usar a seguinte consulta SQL:

```
dmSQL> SELECT * FROM tb_staff WHERE note IS NOT NULL;
```

No DBMaker, o padrão de correspondência para BLOBs é fornecido pelas funções CONTAIN e MATCH, que são semelhantes à função LIKE, exceto que caracteres curinga não são suportados. A diferença entre CONTAIN e MATCH é que o primeiro realiza uma correspondência parcial de palavras, enquanto o último realiza uma correspondência completa de palavras. Por exemplo, 'This is a character.' CONTAIN 'char' e 'This is a character.' MATCH 'character', mas 'This is a character.' NOT MATCH 'char'.

Exemplo 2:

Para encontrar todos os registros na coluna note da tabela tb_staff que contenham a frase 'Database Administrator', você pode usar a seguinte consulta SQL:

```
dmSQL> SELECT * FROM tb_staff WHERE note MATCH 'Database
Administrator';
```

7.2 Managing File Objects

Cada coluna de objeto de arquivo (FO) faz referência a arquivos externos. O uso de FOs é benéfico quando os dados também são utilizados por outras aplicações, pois o arquivo pode ser acessado diretamente. A maioria das ferramentas multimídia atuais só consegue processar dados multimídia quando estes estão armazenados como um arquivo completo do tipo necessário. Dados multimídia armazenados em BLOBs ou arquivos de dados devem ser obtidos pelo usuário a partir do DBMaker e redirecionados para um arquivo que possa ser processado pela ferramenta apropriada. No entanto, se os dados BLOB forem armazenados como um FO, o usuário pode simplesmente obter o nome do arquivo do DBMaker e passá-lo para a ferramenta multimídia apropriada.

Existem dois tipos de FOs: FO do sistema e FO do usuário. Objetos de arquivo do sistema são criados quando um usuário insere dados no lado do cliente e o DBMaker os passa e os armazena em um arquivo externo especificado pelo parâmetro de configuração DB_FoDir. FOs do sistema são criados pelo DBMaker e podem ser reconhecidos pela extensão do nome do arquivo de origem do cliente. Objetos de arquivo do usuário são arquivos externos simplesmente vinculados a uma coluna. Um objeto de arquivo do usuário pode ser um arquivo em qualquer dispositivo acessível pelo DBMaker através do sistema operacional do servidor.

A diferença principal entre objetos de arquivo do sistema e do usuário é que o DBMaker gera automaticamente um FO do sistema. Isso significa que um FO do sistema será excluído quando nenhuma coluna o referenciar. Portanto, com FOs do sistema, os usuários podem deixar o gerenciamento de armazenamento para o DBMaker. Outra vantagem de usar FOs do sistema é que os dados são duplicados para o lado do servidor, permitindo que os usuários gerenciem os dados a partir do servidor. As funcionalidades de backup e restauração do DBMaker também protegem FOs do sistema.

Um FO do usuário não será excluído quando não houver mais referências a ele. A principal vantagem dos FOs do usuário é que o DBMaker pode vincular uma coluna a um arquivo existente diretamente. Não é necessário duplicar dados, como um arquivo em um CD-ROM. Isso conserva espaço em disco e facilita o compartilhamento de um arquivo entre vários registros. No entanto, se um arquivo for excluído fora do DBMaker, todas as colunas que fazem referência a este arquivo podem se tornar inválidas. Um arquivo vinculado como FO do usuário deve ter suas permissões de leitura abertas.

Os arquivos FO do usuário devem ser acessíveis a partir do banco de dados. Eles podem estar espalhados em vários diretórios no lado do servidor. Em vez de especificar um diretório FO do usuário, os usuários precisam configurar a palavra-chave DB_UsrFO como 1 no arquivo dmconfig.ini para habilitar o uso de objetos de arquivo do usuário. Por padrão, os FOs do usuário estão desabilitados.

Os usuários podem obter o nome do arquivo e o tamanho do arquivo de um FO usando as funções embutidas filename() e filelen().

Customizing the System File Object Path

O DBMaker gera uma série de subdiretórios de objetos de arquivo para armazenar objetos de arquivo do sistema. Esses subdiretórios estão localizados no diretório de objetos de arquivo, que é especificado pela palavra-chave DB_FoDir no arquivo dmconfig.ini. Quando um subdiretório de objeto de arquivo atinge um valor limite de

objetos de arquivo, um novo subdiretório é criado. O valor limite é especificado pela palavra-chave DB_FoSub no arquivo dmconfig.ini e pode variar de 100 a 10.000.

Os nomes dos objetos de arquivo seguem o formato ZZxxxxxx.ext, onde xxxxxx é um número serial de base 36 de seis dígitos, e ext é a extensão de arquivo do objeto. A extensão do arquivo depende do comando SET EXTNAME. Consulte as informações sobre Nomes de Extensão de Objetos de Arquivo do Sistema para mais detalhes.

Os subdiretórios seguem uma convenção de nomenclatura baseada no nome do primeiro objeto de arquivo no subdiretório, no formato SUBxxxxxx, onde xxxxxx é o número de base 36 de seis dígitos do primeiro objeto de arquivo a ser inserido no diretório.

Embora um diretório FO possa ser compartilhado por mais de um banco de dados para simplificar o gerenciamento de FO, não é recomendado porque isso pode se tornar inconveniente ao fazer o backup de um banco de dados. O caminho do objeto de arquivo pode ser alterado antes do início do banco de dados modificando o parâmetro de configuração, ou durante a execução.

SETTING THE FO PATH OFFLINE

Os usuários devem especificar onde colocar os FOs do sistema configurando o valor de DB_FoDir no arquivo dmconfig.ini. O valor de DB_FoDir deve ser o caminho completo de um diretório existente. O DBMaker deve possuir permissão de escrita nesse diretório.

Exemplo 1:

Para criar FOs do sistema no diretório /disk1/usr/fo, adicione a seguinte linha ao arquivo dmconfig.ini:

```
DB_FoDir = /disk1/usr/fo
```

Exemplo 2:

Para definir DB_UsrFo = 1 e habilitar objetos de usuário, adicione a seguinte linha ao arquivo dmconfig.ini:

```
DB_UsrFo = 1 ; enable USER file objects
```

SETTING THE FO PATH ONLINE

O DBMaker fornece um procedimento de sistema para os usuários modificarem o diretório de objetos de arquivo do sistema enquanto o banco de dados está em execução. Essa operação altera as configurações dos seguintes 3 itens para o novo valor:

- Diretório de FO em tempo de execução: após a alteração, todos os novos objetos de arquivo do sistema serão armazenados no novo diretório de FO.
- DB_FoDir: na próxima vez que o banco de dados for reiniciado, ele usará o novo diretório de FO.
- Alias $DB_FoDir: o alias padrão de FO, que corresponde à configuração da palavra-chave DB_FoDir no arquivo dmconfig.ini.

Exemplo 1: Para alterar o diretório de objetos de arquivo (FO) para um novo diretório, por exemplo, */home/DBMaker/mydb/fo*, execute o seguinte comando:

```
dmSQL> CALL SETSYSTEMOPTION('fodir', '/home/DBMaker/mydb/fo');
```

Exemplo 2:

O seguinte comando retorna a configuração atual do diretório de objetos de arquivo (FO).

```
dmSQL> CALL GETSYSTEMOPTION('fodir', ?);
OPTION_VALUE: /home/DBMaker/mydb/fo
```

Generating File Objects

Para gerar objetos de arquivo no DBMaker, são necessários vários passos. Primeiro, uma coluna do tipo FILE deve ser criada em uma tabela. Tanto objetos de arquivo do sistema quanto do usuário podem ser inseridos em uma coluna do tipo FO. Para criar uma coluna FO, defina o tipo da coluna como FILE ao criar a tabela.

Exemplo 1:

Para criar uma tabela chamada tb_person com uma coluna de objeto de arquivo chamada photo:

```
dmSQL> CREATE TABLE tb_person (name CHAR(10), photo FILE);
```

Exemplo 2:

Se o objeto de arquivo (FO) a ser inserido estiver no servidor, o DBMaker irá vincular a coluna FO ao arquivo existente e gerar um FO de usuário. Se o FO estiver no lado do cliente, o DBMaker criará um FO de sistema copiando o arquivo do lado do cliente para o diretório de FO no lado do servidor. Para inserir dados do FO:

```
dmSQL> INSERT INTO tb_person VALUES
('cathy','/disk1/image/cathy.bmp')
 2>; // stored as a USER FO
dmSQL> INSERT INTO tb_person VALUES ('jeff',?);
dmSQL/Val> &jeff.gif; // stored as a SYSTEM FO
dmSQL/Val> END;
```

Exemplo 3:

Existem três variedades de métodos de busca para uma coluna FO: conteúdo, nome do arquivo e tamanho do arquivo. Para buscar um arquivo FO chamado cathy.bmp:

```
dmSQL> SELECT photo, FILENAME(photo), FILELEN(photo) FROM tb_person
;
 photo filename(photo) filelen(photo)
========= ========================== ===============
012034451 /disk1/image/cathy.bmp 21100
349045821 /disk1/usr/fo/ZZ000000.hmp 12034
```

Para obter mais informações sobre manipulação de colunas FO, consulte o Guia do Usuário da Ferramenta JDBA e o Guia do Programador ODBC.

System File Object Extension Names

Os usuários podem definir o nome da extensão do objeto de arquivo do sistema usando o comando SET EXTNAME.

Exemplo 1:

Para definir o nome da extensão do objeto de arquivo do sistema <extension_name>:

```
SET EXTNAME TO <extension_name>
```

Existem dois tipos de <extension_name>:

- Uma cadeia de caracteres com menos de sete caracteres, como 'bmp', 'avi' e 'jpg'.
- Usando a opção SOURCE, o nome da extensão é definido igual ao do arquivo de origem do cliente.

Exemplo 2:

Para usar o comando SET EXTNAME:

```
dmSQL> CREATE TABLE tb_example (c1 INT, f1 FILE);
dmSQL> INSERT INTO tb_example (c1, f1) VALUES (?, ?);
dmSQL/Val> SET EXTNAME TO FOB;
dmSQL/Val> 1, &readme.txt; //extension name : '.FOB'
1 rows inserted
dmSQL/Val> SET EXTNAME TO doc;
dmSQL/Val> 2, &readme.txt; //extension name : '.doc'
1 rows inserted
dmSQL/Val> SET EXTNAME TO SOURCE;
dmSQL/Val> 3, &readme.txt; //extension name : '.txt'
dmSQL/Val> END;
dmSQL> SELECT FILENAME(f1) FROM tb_example;
 c1 FILENAME (f1)
========== ==============================
 1 /usr1/fo/ZZ000001.FOB
 2 /usr1/fo/ZZ000002.doc
 3 /usr1/fo/ZZ000003.txt
3 rows selected
```

Updating File Objects

Para atualizar o conteúdo de uma coluna FO, use o comando SQL UPDATE. O DBMaker substitui a coluna FO por um novo arquivo. Assim como na inserção de FOs, uma coluna FO pode ser atualizada para um novo FO de SISTEMA ou vinculada a um FO de USUÁRIO.

Exemplo:

Para vincular a coluna de foto a /disk2/image/common.bmp:

```
dmSQL> UPDATE tb_person SET photo = '/disk2/image/common.bmp' WHERE
name =
'cathy';
```

Alternativamente, os usuários podem inserir novos dados de um arquivo no lado do cliente. Para mais informações, consulte o Guia do Usuário da Ferramenta JDBA e o Guia do Programador ODBC.

Se o resultado da operação UPDATE contiver mais de uma tupla, apenas um arquivo é criado. Este arquivo é compartilhado entre as tuplas para economizar espaço em disco. O DBMaker mantém um contador interno para registrar quantas tuplas fazem referência ao arquivo. Além disso, se um usuário modificar o conteúdo do arquivo através de um programa de aplicação externo, todas as tuplas reconhecerão a modificação.

Quando nenhuma tupla mantiver links para um FO de sistema após operações UPDATE ou DELETE, o DBMaker deleta automaticamente o arquivo após a transação ser confirmada. No entanto, o DBMaker nunca remove nenhum FO de USUÁRIO, mesmo quando nenhuma tupla está fazendo referência a ele, porque este arquivo não foi gerado pelo DBMaker.

Renaming File Objects

Às vezes, discos cheios ou reorganizações no layout do disco tornam necessário alterar as posições ou nomes dos FOs. O DBMaker permite que os usuários usem o comando MOVE FILE OBJECT para alterar o nome ou caminho do FO. Antes de usar o comando MOVE FILE OBJECT, mova os arquivos para a nova localização usando o sistema operacional; o DBMaker garantirá que os novos arquivos existam antes de permitir a movimentação. Exemplo 1:

Para obter os nomes dos arquivos que serão movidos usando filename():

```
dmSQL> MOVE FILE OBJECT '/disk1/usr/fo/ZZ000000.FOB' TO
'/disk3/pub/photo1.bmp';
```

O DBMaker também permite aos usuários mover FOs de um diretório para outro. Note que o DBMaker permite o uso de apenas um caractere * para o nome do arquivo especificado no diretório de origem, mas não permite o uso de caracteres * no diretório de destino. O DBMaker não suporta mover arquivos recursivamente. Para mover todos os arquivos, excluindo subdiretórios, de um diretório para outro, especifique o diretório anterior adicionando os caracteres '/' ou '/*' ao final do diretório.

Exemplo 2:

Para mover ABC1.FOB, ABC2.FOB, ABC3.FOB e ABC4.FOB de /disk1/usr/fo para /disk3/pub, você pode usar o seguinte comando:

```
dmSQL> MOVE FILE OBJECT '/disk1/usr/fo/ ' TO '/disk3/pub/ ';
dmSQL> MOVE FILE OBJECT '/disk1/usr/fo/* ' TO '/disk3/pub/ ';
dmSQL> MOVE FILE OBJECT '/disk1/usr/fo/*.FOB ' TO '/disk3/pub/ ';
dmSQL> MOVE FILE OBJECT '/disk1/usr/fo/A* ' TO '/disk3/pub/ ';
```

Isso moverá todos os arquivos que começam com "ABC" e têm a extensão ".FOB" do diretório /disk1/usr/fo para o diretório /disk3/pub.

```
dmSQL> MOVE FILE OBJECT '/disk1/usr/fo/*1.FOB ' TO '/disk3/pub/ ';
dmSQL> MOVE FILE OBJECT '/disk1/usr/fo/A*1.FOB ' TO '/disk3/pub/ ';
```

Unloading System File Objects

O DBMaker permite que o administrador do banco de dados decida se deseja descarregar objetos de arquivo do sistema usando o comando UNLOAD FILEOBJ ON/OFF ou não. A configuração padrão para esta opção é ON. Todos os objetos de arquivo no servidor serão descarregados para o diretório de trabalho atual quando os usuários não especificarem esta opção. Se os usuários não quiserem descarregar todos os arquivos FO para o diretório de trabalho devido a problemas de espaço em disco, podem usar o comando SET UNLOAD FILEOBJ OFF para copiar os objetos de arquivo manualmente para um diretório de descarregamento.

Ao carregar esses arquivos descarregados para um novo banco de dados, todos os objetos de arquivo do sistema serão reordenados e terão seus nomes alterados. Se os usuários tiverem citado esses nomes de FO em suas aplicações e não desejarem reordenar esses objetos de arquivo do sistema ao migrar para um novo banco de dados, podem usar a opção SET UNLOAD FILEOBJ NAME antes de UNLOAD DB. O dmSQL descarregará o script como objeto de arquivo do cliente com caminho completo e nome de arquivo. O usuário deve garantir que DB_UsrFo = 1 antes de carregar o banco de dados, caso contrário, a carga falhará, e todos os objetos de arquivo devem existir no servidor com o mesmo caminho e nome de arquivo.

Retrieving the Length of File Objects

O comprimento de um objeto de arquivo (FO) pode ser recuperado usando as funções embutidas FILELENEX e FILELEN. Para mais informações sobre essas funções, consulte o SQL Command and Function Reference.

Predicate Operations on File Objects

Assim como com BLOBs, os usuários podem testar FOs para valores NULL e usar as funções CONTAIN e MATCH para realizar buscas por padrões. Além disso, o item FO pode ser usado em expressões aritméticas com a função incorporada FILELEN(), em expressões booleanas com a função incorporada FILEEXIST() e em expressões de string com a função incorporada FILENAME().

No caso de um arquivo ter sido removido ou renomeado no sistema operacional, use a função incorporada FILEEXIST() para testar quais arquivos existem. Um valor de 0 indica que o arquivo FO referenciado não existe, 1 indica que ele ainda existe, e NULL indica que a tupla é um valor NULL.

Exemplo 1:

Para selecionar tuplas com a extensão .gif da coluna foto:

```
dmSQL> SELECT * FROM tb_person WHERE FILENAME(photo) LIKE '%.gif';
```

Exemplo 2:

Para buscar tuplas com tamanho maior que 100KB da coluna foto:

```
dmSQL> SELECT * FROM tb_person WHERE FILELEN(photo) > 102400;
```

Exemplo 3:

Para buscar todas as tuplas para arquivos que existem:

```
dmSQL> SELECT * FROM tb_person WHERE FILEEXIST(photo)=1;
```

File Object UNC Names

Os nomes de arquivos da Convenção Universal de Nomenclatura (UNC) podem ser usados para o caminho do objeto de arquivo e os nomes de diretórios em ambientes Microsoft Windows. Isso facilita a especificação dos caminhos e nomes de diretórios quando um servidor DBMaker está sendo executado em uma plataforma Microsoft Windows. Diretórios em máquinas diferentes da máquina que hospeda o servidor também podem ser especificados.

Exemplo 1:

Para buscar tuplas com tamanho maior que 100KB na coluna foto:

```
dmSQL> SELECT * FROM tb_person WHERE FILELEN(photo) > 102400;
```

Exemplo 2:

Para buscar todas as tuplas para arquivos que existem:

```
dmSQL> SELECT * FROM tb_person WHERE FILEEXIST(photo)=1;
```

File Object Path Default Aliases

O DBMaker suporta dois nomes de alias para o caminho do objeto de arquivo: $DB_DbDir e $DB_FoDir. O alias do caminho do objeto de arquivo fornece um nome alternativo para representar o caminho real do objeto de arquivo. Os usuários podem inserir, atualizar e excluir objetos de arquivo usando o nome do caminho do alias. Os objetos de arquivo podem ser movidos mais facilmente para outro caminho de diretório. Os dois nomes de alias são definidos usando as palavras-chave DB_DbDir e DB_FoDir no arquivo dmconfig.ini para $DB_DbDir e $DB_FoDir, respectivamente.

Exemplo 1:

O alias do caminho do objeto de arquivo é definido para o caminho especificado pela palavra-chave DB_FoDir no arquivo dmconfig.ini:

```
…
DB_FoDir = "/usr1/tmp/employeedata/FO"
```

Exemplo 2:

Para usar o alias do caminho do objeto de arquivo para inserir valores na coluna do tipo FILE chamada photo:

```
dmSQL> CREATE TABLE tb_example (c1 INT, photo FILE);
dmSQL> INSERT INTO tb_example VALUES (2, '$DB_FoDir/photo471.jpg');
```

No exemplo acima, o arquivo photo471.jpg também poderia ter sido inserido usando o caminho completo do objeto de arquivo '/usr1/tmp/employeedata/FO/photo471.jpg'.

FO and Applications

Os dados do tipo FILE são suportados apenas pelo DBMaker e não são definidos pelo ODBC. Ferramentas de desenvolvimento como Inprise/Borland Delphi ou Microsoft Visual Basic não reconhecem dados do tipo FILE como válidos. O parâmetro de configuração DB_FoTyp pode ser usado para especificar para qual tipo de dados os dados do tipo FILE serão mapeados. Para permitir que essas ferramentas acessem dados do tipo FILE, o DBMaker deve ser configurado para mapear internamente os dados do tipo FILE para LONG VARBINARY, definindo DB_FoTyp como 1. Se DB_FoTyp for 0, não haverá mapeamento e as ferramentas não reconhecerão dados do tipo FILE.

Exemplo:

Para configurar o banco de dados para mapear dados do tipo FILE para LONG VARBINARY:

```
DB_FoTyp = 1
```

7.3 Journal of Large Objects

O registro de transações envolvendo dados BLOB (LONG VARCHAR ou LONG VARBINARY) requer grandes quantidades de espaço em disco e resulta em diminuição de desempenho. O DBMaker permite que o administrador do banco de dados decida se um BLOB em um determinado tablespace será registrado ou não. O DBMaker não suporta o registro para objetos de arquivo (FILE).

Por padrão, o DBMaker não registra o conteúdo de dados BLOB. Durante o período entre o início e o desligamento do banco de dados, o DBMaker garante a consistência dos dados BLOB. Mesmo em caso de falha do sistema, os dados BLOB permanecem consistentes após a recuperação.

No entanto, ao restaurar um banco de dados a partir de backups incrementais, o DBMaker não garante a consistência dos dados BLOB. Dois passos devem ser tomados para garantir que os dados BLOB sejam registrados no registro. Primeiro, o parâmetro de configuração DB_BMode deve ser definido para registrar transações BLOB. Segundo, o tablespace que contém os dados BLOB a serem backupados deve ter sido criado com a opção BACKUP BLOB ON.

BLOB Journal Logging

- Definir o valor da palavra-chave DB_BMode no arquivo dmconfig.ini como 2 (modo BACKUP DATA AND BLOB).
- Adicionar arquivos BLOB a um tablespace que foi criado com a opção BACKUP BLOB ON.

SETTING THE DB_BMODE VALUE

A palavra-chave DB_BMode especifica o modo de backup de um banco de dados. Definir o valor como 0 habilita o modo SEM BACKUP, 1 habilita o modo BACKUP-DATA, e 2 habilita o modo BACKUP-DATA-AND-BLOB.

- NON-BACKUP (0) mode — não suporta backup incremental para tablespaces, incluindo sistemas ou definidos pelo usuário.
- BACKUP-DATA (1) mode — suporta backup incremental para o tablespace do sistema, dados em tablespaces definidos pelo usuário, mas não suporta BLOBs em tablespaces definidos pelo usuário.
- BACKUP-DATA (2) mode — suporta backup incremental para o tablespace do sistema, dados em tablespaces definidos pelo usuário, BLOBs em tablespaces definidos pelo usuário criados com a opção BACKUP BLOB ON, mas não suporta BLOBs em tablespaces definidos pelo usuário criados com a opção BACKUP BLOB OFF.

Para ativar o registro de jornal de BLOB, adicione uma linha ao arquivo dmconfig.ini:

```
DB_BMode = 2 ;log all data including BLOB
```

Para detalhes sobre o modo de backup do banco de dados, consulte o Capítulo 15, Backup, Recuperação e Restauração do banco de dados.

SETTING THE CREATE TABLESPACE BACKUP OPTION

O modo de backup para um tablespace individual é definido durante sua criação. A sintaxe para o comando CREATE TABLESPACE é a seguinte:

```
CREATE [AUTOEXTEND] TABLESPACE tablespace_name [backup_mode]
DATAFILE [tsfile , tsfile, ...];
```

onde:

```
backup_mode ::= BACKUP BLOB OFF | BACKUP BLOB ON
tsfile ::= file_name TYPE = DATA | file_name TYPE = BLOB
```

Os usuários podem colocar BLOBs importantes em tablespaces com a sinalização BACKUP BLOB ON. É uma boa ideia colocar BLOBs que não precisam ser copiados de segurança em tablespaces com a configuração BACKUP BLOB OFF para melhorar o desempenho do sistema. Os criadores do tablespace determinam o equilíbrio entre esses aspectos.

Os arquivos de dados e BLOB devem ser especificados no arquivo dmconfig.ini antes da criação do tablespace. Isso pode ser feito através da página "user files" na ferramenta JConfiguration. Consulte o Referencial da Ferramenta JConfiguration para instruções detalhadas sobre como criar arquivos de dados e BLOBs.

Exemplo 1:

Para criar o tablespace ts_reg com BACKUP BLOB DESLIGADO e ts_aut com BACKUP BLOB LIGADO:

```
dmSQL> CREATE TABLESPACE ts_reg BACKUP BLOB OFF
 2> DATAFILE f1 TYPE = DATA, f2 TYPE = BLOB;
dmSQL> CREATE TABLESPACE ts_aut BACKUP BLOB ON
 2> DATAFILE f3 TYPE = DATA, f4 TYPE = BLOB;
```

Exemplo 2:

Consulte a coluna BK_MODE da tabela SYSTABLESPACE para saber o modo de backup de cada tablespace. O valor 1 significa que BACKUP BLOB está DESLIGADO, enquanto 2 significa que está LIGADO. Consultar o modo de backup de um tablespace resultará no seguinte:

```
dmSQL> SELECT TS_NAME, BK_MODE FROM SYSTEM.SYSTABLESPACE;
 TS_NAME BK_MODE
============== =============
SYSTABLESPACE 2
DEFTABLESPACE 2
ts_reg 1
ts_aut 2
4 rows selected
```

Aqui está um resumo da interação dos modos de backup entre um banco de dados e seus tablespaces. 'Sim' indica que o tipo de tablespace em questão é realizado backup, 'Não' indica que não

DATABASE BACKUP MODE	**TABLESPACE BACKUP MODE**	**USER-DEFINED TABLESPACE (DATA)**	**USER-DEFINED TABLESPACE (BLOB)**	**SYSTEM TABLESPACE (DATA AND BLOB)**
NON BACKUP (**DB_BMode = 0**)	---	No	No	No
BACKUP DATA (**DB_BMode = 1**)	---	Yes	No	Yes
BACKUP DATA AND BLOB (**DB_BMode = 2**)	BACKUP BLOB OFF ---------- BACKUP BLOB ON	Yes ---------- Yes	No ---------- Yes	Yes ---------- Yes

Antes de definir o modo de backup, certifique-se de que o arquivo de log seja grande o suficiente para registrar todos os dados BLOB; caso contrário, uma mensagem de log cheio pode ser retornada. Para informações sobre como ajustar o tamanho do arquivo de log, consulte a subseção "Ajustando o Espaço do Log" no Capítulo 5. Para conceitos sobre arquivos de dados, arquivos BLOB e tablespaces, consulte o Capítulo 5, Arquitetura de Armazenamento. Para informações sobre o comando CREATE TABLESPACE, consulte o Referencial de Comandos e Funções SQL. Para informações sobre a tabela SYSTABLESPACE, consulte o Referencial de Catálogo do Sistema.

File Object Journal Logging

O DBMaker não suporta o registro de log de FOs (File Objects). Ao fazer o backup do banco de dados, faça o backup de todos os FOs pertencentes ao banco de dados manualmente copiando-os para um diretório de backup. Alternativamente, o Backup Server pode ser usado para fazer o backup automático de objetos de arquivo para um diretório de backup. Para mais informações sobre como fazer backup de objetos de

arquivo, consulte a seção 15.6, Backup Server. Para determinar quais arquivos pertencem a um banco de dados, consulte a tabela SYSFILEOBJ.

Exemplo:

Para recuperar os nomes de arquivos de todos os FOs consultando a tabela SYSFILEOBJ:

```
dmSQL> SELECT FILE_NAME FROM SYSFILEOBJ;
```

Copie todos os FOs para o local de armazenamento de backup. Ao restaurar o banco de dados a partir de um backup, copie todos os FOs também. Se os caminhos ou nomes dos arquivos tiverem sido alterados, use o comando MOVE FILE OBJECT para atualizar os nomes dos arquivos na tabela SYSFILEOBJ.

7.4 Large Objects and SELECT INTO Command

O comando SELECT INTO seleciona dados e os insere em uma tabela especificada. Objetos de arquivo e BLOBs podem ser movidos de uma tabela para outra usando este comando. O comando SELECT INTO pode ser usado em um ambiente de banco de dados distribuído (DDB).

Em uma instrução SELECT INTO local-para-local, o DBMaker precisa duplicar os dados do BLOB ou aumentar o contador compartilhado do objeto de arquivo do sistema ou de um objeto de arquivo do usuário.

Em um ambiente DDB, o DBMaker copia os dados do BLOB de um site para outro, mas há muitas considerações para objetos de arquivo. O DBMaker fornece o modo de duplicação de objetos de arquivo distribuídos (comando SET DFO DUPMODE) para cuidar do processamento de objetos de arquivo em um ambiente DDB.

SET DFO DUPMODE

O DFO DUPMODE informa a um banco de dados se os objetos de arquivo devem ser copiados para o banco de dados de destino ou não. Existem dois modos para o DFO DUPMODE: modo NULL e modo COPY.

Exemplo:

A sintaxe para DFO DUPMODE nos modos NULL e COPY é a seguinte:

```
dmSQL> SET DFO DUPMODE NULL;
dmSQL> SET DFO DUPMODE COPY;
```

SET DFO DUPMODE NULL

Existem dois casos a serem considerados no modo DDB:

- As bases de dados de origem e destino são as mesmas, incluindo a base de dados local ou ambas as bases de dados remotas. Como são a mesma base de dados, o DBMaker apenas aumenta os contadores compartilhados dos objetos de arquivo.
- As bases de dados de origem e destino não são as mesmas. A coluna FILE do destino é definida como NULL. Assim, os objetos de arquivo na base de dados de origem não são enviados.

SET DFO DUPMODE COPY

Existem três situações a serem consideradas para objetos de arquivo:

- Para objetos de arquivo do usuário, o DBMaker passa apenas o nome do arquivo de origem para o banco de dados de destino. O usuário precisa copiar os arquivos para um local onde o banco de dados de destino possa acessá-los. Às vezes, o comando UPDATE ou o comando MOVE FILE OBJECT devem ser usados para alterar os nomes dos arquivos no banco de dados de destino se os novos diretórios não forem os mesmos no banco de dados de origem.
- Para objetos de arquivo do sistema entre dois bancos de dados diferentes, o DBMaker cria um novo objeto de arquivo do sistema no banco de dados de destino e copia o conteúdo do banco de dados de origem para ele.
- Para objetos de arquivo do sistema no mesmo banco de dados, local-para-local ou remoto-para-remoto, o DBMaker apenas incrementa os contadores compartilhados.

Limitations

O modo DFO DUPMODE não afeta um comando SELECT INTO usado em uma coluna BLOB (LONG VARCHAR e LONG VARBINARY). Os dados BLOB podem ser copiados usando o comando SELECT INTO, independentemente do DFO DUPMODE.

Em um ambiente DDB, se um comando SELECT INTO for usado em um objeto de arquivo do usuário e a opção do DFO DUPMODE estiver definida como COPY, o usuário deve estar ciente da localização do arquivo vinculado no banco de dados de

destino. O objeto de arquivo vinculado deve existir no mesmo caminho relativo no banco de dados de destino. Se não estiver, o usuário deve usar o sistema operacional para copiar o arquivo do banco de dados de origem para o banco de dados de destino e usar os comandos UPDATE ou MOVE FILE OBJECT para essas colunas se os caminhos dos arquivos dos bancos de dados de origem e destino forem diferentes.

Se o usuário não realizar as operações acima, uma mensagem de erro será retornada ao consultar o objeto de arquivo, porque o arquivo não existe no caminho completo ou o caminho do arquivo está incorreto.

Ao selecionar um objeto de arquivo do sistema de um banco de dados remoto para o banco de dados local, o DBMaker precisa manter um registro das informações compartilhadas. As informações são mantidas dentro de um comando SELECT INTO. Portanto, ainda há um problema de duplicação de arquivos, que desperdiça espaço. Além disso, selecionar objetos de arquivo do sistema em um banco de dados remoto cria arquivos duplicados.

A opção SET EXTNAME não afeta o resultado do comando SELECT INTO. Os nomes de extensão dos objetos de arquivo nos bancos de dados de origem e destino são os mesmos. Por exemplo, o nome do arquivo no banco de dados de origem é 'ZZ000001.BMP', e o nome do arquivo de destino no banco de dados de destino pode ser 'ZZXXXXXX.BMP'.

Exemplo:

DBMaker assume os dados de CHAR, VARCHAR ou BINARY como o nome do arquivo, portanto, os usuários devem garantir que **db2** possa acessar o arquivo ***/etc/hosts*** do ponto de vista de **db1**. Selecione a coluna CHAR na coluna FILE, onde a coluna **c2** na tabela **t2** no banco de dados **db2** é do tipo FILE

```
dmSQL> SELECT c1, '/etc/hosts' FROM db1:t1 INTO db2:t2(c1, c2);
```

Considerando a coluna do tipo FILE no banco de dados de destino, a tabela abaixo resume o efeito do comando SELECT INTO com diferentes tipos de dados de origem:

TYPE ON THE SOURCE DATABASE	**ENVIRONMENT**	**SET DFO DUPMODE**	**RESULT**

string expression CHAR VARCHAR BINARY	Ambiente não-DDB ou DDB.	...	Fonte: passa o nome do arquivo. Destino: insere novos objetos de arquivo do usuário.
FILE	A fonte e o destino são o mesmo banco de dados. --- A fonte e o banco de dados de destino não são os mesmos. ---	... --- NULL --- COPY	Aumenta o contador compartilhado dos objetos de arquivo. --- Destino: insere o valor NULL. --- A fonte é o objeto de arquivo do usuário: Fonte: passa o nome do arquivo. Destino: insere o novo objeto de arquivo do usuário. A fonte é o objeto de arquivo do sistema: Fonte: passa o conteúdo do objeto de arquivo. Destino: insere o novo objeto de arquivo do sistema.
LONG VARCHAR LONG VARBINARY Other	...	...	Not supported.