17. Data Replication

A Replicação de Dados, no sentido mais amplo do termo, refere-se ao processo de representar objetos em mais de um banco de dados.

Há apenas alguns anos, os dados corporativos residiam em um local central. Departamentos remotos acessavam as informações de que precisavam estabelecendo conexões diretas com os sites centrais ou solicitando relatórios impressos do MIS central. No entanto, as conexões eram caras, não confiáveis e limitadas em número, enquanto os relatórios eram inflexíveis e demorados.

Os sistemas abertos trouxeram recursos de computação poderosos e acessíveis para todos os cantos de uma empresa. A capacidade de compartilhar informações corporativas de forma eficaz usando esses novos recursos tornou-se uma importante vantagem competitiva para as organizações. A questão que as empresas enfrentam hoje não é "Por que distribuir e compartilhar dados corporativos?", mas sim "Como distribuir informações de forma eficaz?" A replicação está rapidamente se tornando a arquitetura de escolha para a maioria das aplicações corporativas distribuídas.

17.1 Table Replication

What is Table Replication?

A replicação de tabelas cria uma cópia total ou parcial de uma tabela em um local de destino. Isso permite que usuários em locais remotos trabalhem com uma cópia local dos dados. A cópia local permanece sincronizada com os bancos de dados em outros locais. Dessa forma, cada banco de dados pode atender a solicitações de dados imediatamente e de forma mais eficiente, sem a necessidade de se conectar a outra máquina por meio de uma conexão de rede de longa distância mais lenta. Esse tipo de solicitação acontece com frequência entre a sede e as empresas ou filiais da área.

Por exemplo, após criar uma replicação da tabela A no servidor de banco de dados de Taipei para a tabela B no servidor de banco de dados de Tóquio, as modificações feitas na tabela A serão replicadas para a tabela B. Os clientes em Tóquio poderão então acessar o banco de dados de Tóquio em vez de se conectar ao banco de dados de Taipei para adquirir os mesmos dados.

As tabelas de destino também podem residir em bancos de dados no mesmo servidor. Essa situação pode ocorrer quando dois bancos de dados, projetados para funções diferentes, compartilham dados, ou quando se compartilham dados entre bancos de

dados de diferentes tipos (por exemplo, Oracle, Sybase). As tabelas que estão recebendo dados de outro banco de dados por meio de replicação são tabelas de destino, em vez de tabelas remotas, mesmo que as tabelas de destino possam residir em um banco de dados remoto.

Differences between Database and Table Replic

A principal diferença entre a replicação de banco de dados e a replicação de tabelas é que o objeto de dados replicado é diferente: um é o banco de dados inteiro; o outro é uma tabela. Os usuários podem escolher entre um deles, dependendo de suas necessidades. Se você optar pela replicação de banco de dados, uma vez que a unidade a ser replicada é o banco de dados inteiro, o banco de dados de destino (ou escravo) será somente leitura.

Two Types of Table Replication

Existem dois tipos de replicação de tabelas. Um é a replicação síncrona. "Síncrona" significa que a modificação no site de destino aparece imediatamente; a tabela de destino é modificada ao mesmo tempo que a tabela local. O DBMaker usa um "compromisso de duas fases" e gatilhos para realizar a replicação de tabelas síncronas. Assim, após estabelecer uma replicação, qualquer atualização na tabela de origem se tornará uma ação de DDB (banco de dados distribuído). Isso afetará o comportamento do banco de dados local. Se o servidor do banco de dados de destino estiver inacessível, as atualizações no banco de dados local falharão.

O outro tipo de replicação de tabela é a replicação assíncrona. As modificações no site de destino são atrasadas. O atraso entre os bancos de dados de origem e destino depende de um cronograma definido pelo usuário. A replicação de tabela assíncrona armazena alterações na tabela local e modifica a tabela de destino com base em um cronograma. Neste tipo de replicação, os dois bancos de dados de um par de replicação são independentes e podem funcionar normalmente, mesmo que a rede não esteja disponível.

Term Definitions

SOURCE TABLE

A tabela no banco de dados de origem da qual os dados são replicados.

DESTINATION TABLE

A tabela no banco de dados de destino para a qual os dados são replicados.

PUBLICATION

Um conjunto de dados na tabela de origem que está disponível para replicação.

SUBSCRIPTION

O conjunto de dados na tabela de destino que recebe uma publicação.

FRAGMENT

Também chamado de partição horizontal, um fragmento é a replicação de um determinado intervalo de dados.

PROJECTION

As colunas selecionadas de uma tabela base escolhidas para replicação.

REPLICATION DOMAIN

Um fragmento de replicação (partição horizontal) e uma projeção (partição vertical) são chamados de domínio de replicação. É o intervalo de dados de uma tabela a ser replicado.

Há um problema ao replicar uma mudança de domínio. Isso pode ocorrer ao replicar uma instrução UPDATE.

Exemplo:

```
dmSQL> CREATE REPLICATION rp_case1 WITH PRIMARY AS tb_example WHERE
number > 0
REPLICATE TO db2:tb_example;
dmSQL> CREATE REPLICATION rp_case2 WITH PRIMARY AS tb_example WHERE
number < 0
REPLICATE TO db2:tb_example;
dmSQL> UPDATE tb_example SET number=-7 WHERE number=7;
```

O domínio de replicação **rp_case1** é número > 0, e o domínio de replicação **rp_case2** é número < 0. Ao replicar, não se está apenas replicando a instrução **UPDATE**. Os tuplos atualizados mudam de domínio de replicação de **rp_case1** para **rp_case2**. Subsequentemente, a replicação realiza uma instrução **DELETE** para **rp_case1** (DELETE número = -7) e realiza uma instrução **INSERT** para **rp_case2** (INSERT número = 7).

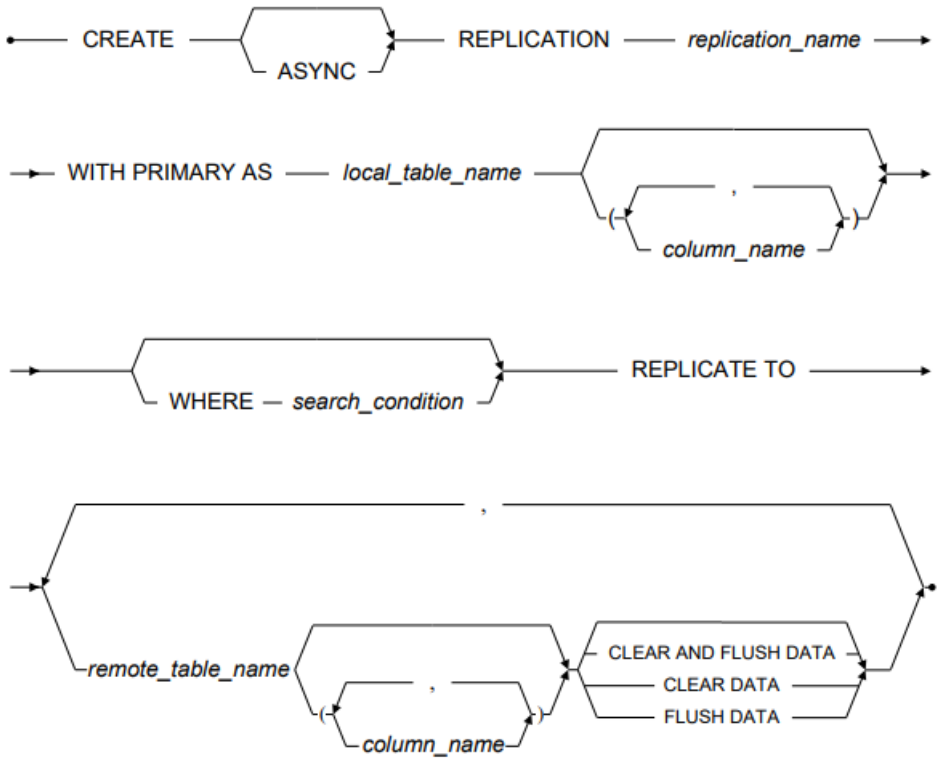
DATA INITIALIZATION

Ao criar repetições, os usuários podem especificar como inicializar os dados em ambos os lados automaticamente. Após criar uma replicação de tabela, qualquer modificação (inserção, exclusão, atualização) na tabela de origem afetará as tabelas de destino. O DBMaker oferece 4 opções:

- **Limpar dados** — durante a replicação da tabela, exclui todos os dados da tabela de destino.
- **Flush dados** — insere todos os dados que atendem ao fragmento da tabela de origem nas tabelas de destino.
- **Limpar e flush dados** — limpar e, em seguida, flush de dados.
- **Não fazer nada** — manter as tabelas de destino.

Creating Table Replication

O seguinte diagrama de sintaxe descreve tanto a replicação de tabela assíncrona quanto a replicação de tabela síncrona.



Exemplo:

Suponha que exista uma tabela TB1 no banco de dados DB30A e uma tabela TB2 no banco de dados DB30B. Queremos replicar os dados de TB1 para TB2 e não queremos modificar os dados atuais de TB2.

```
dmSQL> CREATE REPLICATION rp_example WITH PRIMARY AS TB1
 REPLICATE TO DB30B:TB2;
```

No exemplo acima, usamos um nome de sessão de banco de dados para especificar o banco de dados de destino. Alternativamente, poderia ser utilizado um nome de link de banco de dados.

Table Replication Rules

- Os esquemas das tabelas de origem e destino devem existir. Isso significa que o DBMaker não cria tabelas ao realizar a replicação de tabelas.
- Os nomes de replicação para uma tabela devem ser únicos.
- O nome do assinante para uma replicação deve ser único, utilizando a sintaxe <link_name|database_session_name> + <table_owner_name> + <table_name>.
- As colunas projetadas para cada tabela devem conter colunas de chave primária.
- As colunas de chave primária devem ser incluídas juntamente com as colunas de fragmento.
- Apenas o proprietário da tabela de origem ou um usuário com autoridade DBA ou superior tem o privilégio de criar, excluir ou alterar replicações.
- Se não existir um identificador de coluna na tabela de destino, os nomes das colunas de destino devem ser os mesmos que os nomes das tabelas base.
- O número de colunas de chave primária deve ser igual na tabela base e nas tabelas de destino.

Exemplo 1:

A seguinte instrução criará uma publicação que replicará a tabela tb_salary do banco de dados local para a tabela usr1.tb_salaryA no db1. Sem especificar os nomes das colunas, todas as colunas na tabela tb_salary devem existir em tb_salaryA e os tipos de coluna devem ser compatíveis. Se a tabela tb_salary contiver 3 colunas: id, nome e basepay, ela replicará id, nome e basepay da tabela tb_salary para as colunas id, nome e basepay na tabela usr1.tb_salaryA.

```
dmSQL> CREATE REPLICATION rp_salaryA WITH
 PRIMARY AS tb_salary
 REPLICATE TO db1:usr1.tb_salaryA;
```

Exemplo 2:

A seguinte instrução criará uma publicação usada para replicar tuplas onde id > 100, com as colunas (id, nome) para (Aid, Aname) em tb_salaryA no db1, e (id, nome) em tb_salaryB no db2. Após a emissão deste comando, todos os dados onde id > 100 na tabela tb_salary serão replicados para tb_salaryB no db2 e para Aid e Aname na tabela tb_salaryA no db1:

```
dmSQL> CREATE REPLICATION rp_salaryAB WITH
PRIMARY AS tb_salary (id,name) WHERE id > 100 ,
REPLICATE TO db1:tb_salaryA (Aid,Aname),
 db2:tb_salaryB flush data;
```

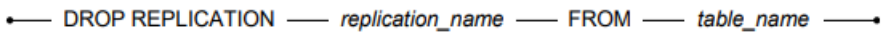
Exemplo 3:

A primeira função deste comando é excluir todos os dados da tabela tb_salaryA no db1 e, em seguida, criar uma publicação usada para replicar os registros onde id > 100, com apenas as colunas (id, nome) para (Aid, Aname) na tabela tb_salaryA no db1:

```
dmSQL> CREATE REPLICATION rp_salaryAClear WITH
PRIMARY AS tb_salary (id,name) WHERE id > 100 ,
REPLICATE TO
db1:tb_salaryA (Aid, Aname) clear data;
```

Drop Replication

Este comando excluirá uma replicação de uma tabela de origem.



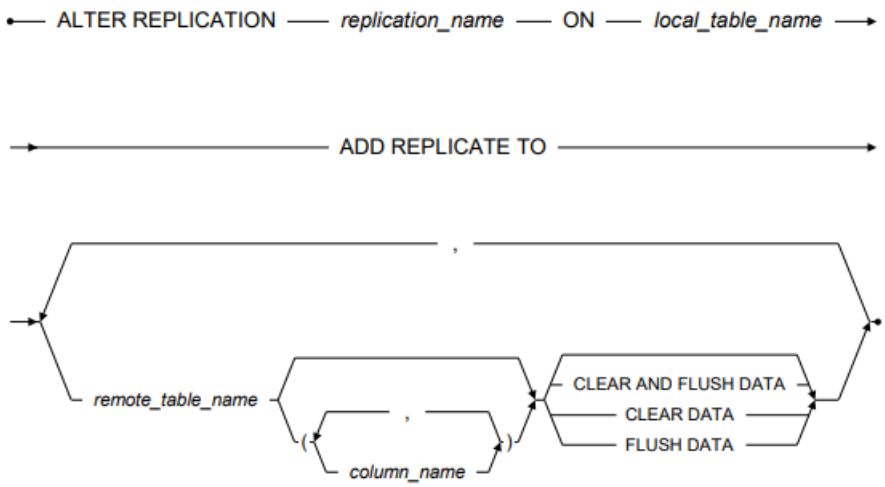
Exemplo:

Para excluir a replicação rp_salaryA da tabela tb_salary:

```
dmSQL> DROP REPLICATION rp_salaryA FROM tb_salary;
```

Alter Replication

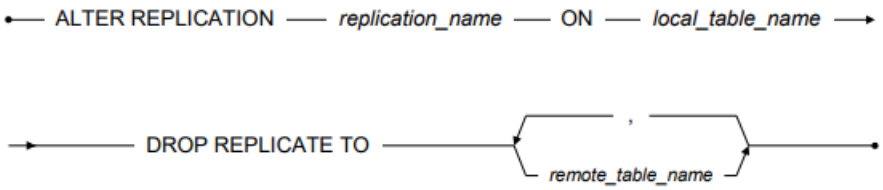
Os usuários podem adicionar tabelas de destino ou excluir tabelas de destino de uma replicação de tabela existente. Os seguintes diagramas de sintaxe e exemplos demonstram como fazer isso.



Exemplo 1:

O primeiro comando cria uma replicação para replicar a tabela tb_salary no banco de dados local para a tabela tb_salaryX no banco de dados dbX. O segundo comando adiciona 2 assinantes, tb_salaryA no db1 e tb_salaryB no db4, à replicação de rp_AlterRp na tabela tb_salary.

```
dmSQL> CREATE REPLICATION rp_AlterRp WITH PRIMARY AS tb_salary
 REPLICATE TO dbX:tb_salaryX;
dmSQL> ALTER REPLICATION rp_AlterRp ON tb_salary
 ADD REPLICATE TO db1: tb_salaryA,
 db4: tb_salaryB (Bid, Bname) clear data;
```



Exemplo 2:

Para excluir o assinante tb_salaryA no db1 da replicação rp_AlterRp da tabela tb_salary:

```
dmSQL> ALTER REPLICATION rp_AlterRp ON tb_salary
 DROP REPLICATE TO db1: tb_salaryA;
```

17.2 Synchronous Table Replication

O compromisso de duas fases permite a sincronização de dados distribuídos. Uma transação será aceita apenas se todos os sites distribuídos interconectados concordarem. Um mecanismo de "aperto de mão" através da rede permite que os sites distribuídos coordenem sua aceitação para cada transação. Portanto, o uso de replicação de tabela síncrona garante que os dados estarão sincronizados sempre que uma atualização ocorrer.

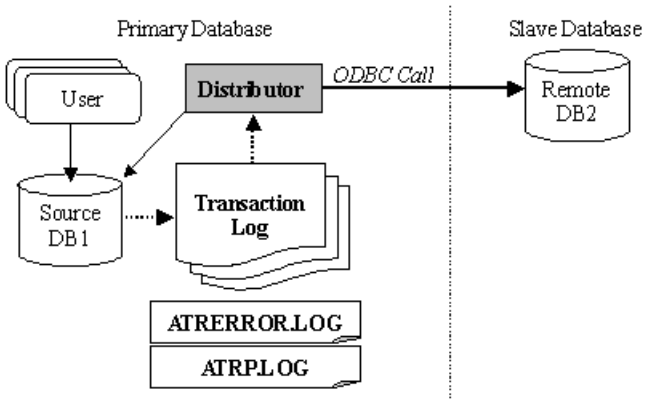
Synchronous Table Replication Setup

O DBMaker utiliza compromissos de duas fases para realizar a replicação de tabelas síncronas. Os bancos de dados de origem e destino devem estar no modo de banco de dados distribuído (DDB). Portanto, o primeiro passo é iniciar os bancos de dados no modo DDB (DD_DDBMd = 1) e adicionar sessões de banco de dados ao arquivo dmconfig.ini. Consulte o Capítulo 16, Bancos de Dados Distribuídos, para mais informações.

Após a criação de uma replicação de tabela, qualquer modificação (inserção, exclusão, atualizações) na tabela de origem afetará as tabelas de destino.

17.3 Asynchronous Table Replication

A replicação de tabela síncrona modifica a tabela de destino ao mesmo tempo em que modifica a tabela local, enquanto a replicação de tabela assíncrona armazena as alterações no lado local e modifica a tabela de destino com base em um cronograma. O DBMaker utiliza um arquivo conhecido como log de replicação para armazenar as alterações nas tabelas locais. As modificações nas tabelas locais são armazenadas nos logs de replicação e são replicadas na tabela de destino de acordo com um cronograma predefinido. O uso de logs de replicação permite que o DBMaker trate a transação local e a transação de destino de forma independente, permitindo que você atualize as tabelas locais normalmente, mesmo que a conexão remota não esteja disponível. Isso permite que as replicações de tabela assíncronas tolerem falhas na rede e no banco de dados de destino, já que o DBMaker continuará tentando até que as falhas sejam corrigidas.



A replicação de tabela assíncrona utiliza um sistema de log de replicação e o servidor distribuidor para gerenciar a replicação de dados. Os logs de replicação não são arquivos de diário do DBMaker. O nível do log de replicação é superior ao do diário e é exclusivo para a replicação de tabelas. O conteúdo de um diário é a modificação física de dados, enquanto o log de replicação consiste em comandos que são aplicados às tabelas de destino.

Quando o banco de dados de origem está em execução, o DBMaker registra a modificação das tabelas de origem nos arquivos de log de replicação. Quando o servidor distribuidor é ativado, ele irá refazer todas as alterações feitas nas tabelas de origem nas tabelas de destino de acordo com o log de replicação.

Normalmente, o servidor distribuidor utiliza chamadas de função ODBC para se comunicar com os servidores de banco de dados de destino, permitindo a replicação de tabelas para servidores de banco de dados heterogêneos, como Oracle, SQL Server, Informix, etc. A replicação de tabela heterogênea será abordada mais adiante neste capítulo. A replicação assíncrona de tabela expressa é outro tipo de replicação de tabela assíncrona que não utiliza chamadas de função ODBC. Isso também será abordado mais adiante neste capítulo.

Existem três etapas principais para construir uma replicação de tabela assíncrona:

1. Habilitar a replicação de tabela assíncrona
2. Criar um cronograma para o banco de dados de destino
3. Com base no cronograma, criar a replicação de tabela assíncrona

Enabling Asynchronous Table Replication

O servidor distribuidor reside no banco de dados de origem. O distribuidor conecta-se periodicamente aos bancos de dados de destino e realiza a replicação das tabelas.

A palavra-chave DB_AtrMd no arquivo dmconfig.ini no banco de dados de origem especifica se o servidor distribuidor deve ser iniciado para um banco de dados. Se

você não iniciar o servidor distribuidor, o banco de dados não poderá ser a origem para replicações de tabela assíncronas.

A palavra-chave RP_LgDir no dmconfig.ini do banco de dados de origem especifica o diretório onde o DBMaker colocará os arquivos de log de replicação para a replicação de tabela assíncrona. Os arquivos de log de replicação são binários e os usuários não devem removê-los manualmente. O diretório padrão de RP_LgDir é o subdiretório chamado /TRPLOG no diretório home do banco de dados.

Ao criar replicações de tabela com verificação de esquema, o modo de banco de dados distribuído no banco de dados de origem e no banco de dados de destino deve ser habilitado (DD_DDBMd = 1). Se o cronograma estiver definido como NO CHECK, o DBMaker não verificará o esquema e o modo de banco de dados distribuído pode ser definido como DESATIVADO (DD_DDBMd = 0). Consulte a seção a seguir para mais informações sobre como criar cronogramas de replicação.

Exemplo:

Para replicar uma tabela do banco de dados SRCDB para o banco de dados de destino DESTDB, adicione as seguintes linhas ao arquivo dmconfig.ini no servidor do banco de dados de origem:

```
[SRCDB]
DB_DbDir = /disk1/DBMaker/src
DB_UsrBb = /disk1/DBMaker/src/SRCDB.BB 3
DB_UsrDb = /disk1/DBMaker/src/SRCDB.DB 150
RP_LgDir = /disk1/DBMaker/src/trplog
DB_AtrMd = 1
DD_DDBMd = 1
DB_SvAdr = srcpc
DB_PtNum = 22222
[DESTDB]
DB_SvAdr = destpc
DB_PtNum = 33333
```

O arquivo dmconfig.ini no banco de dados de destino:

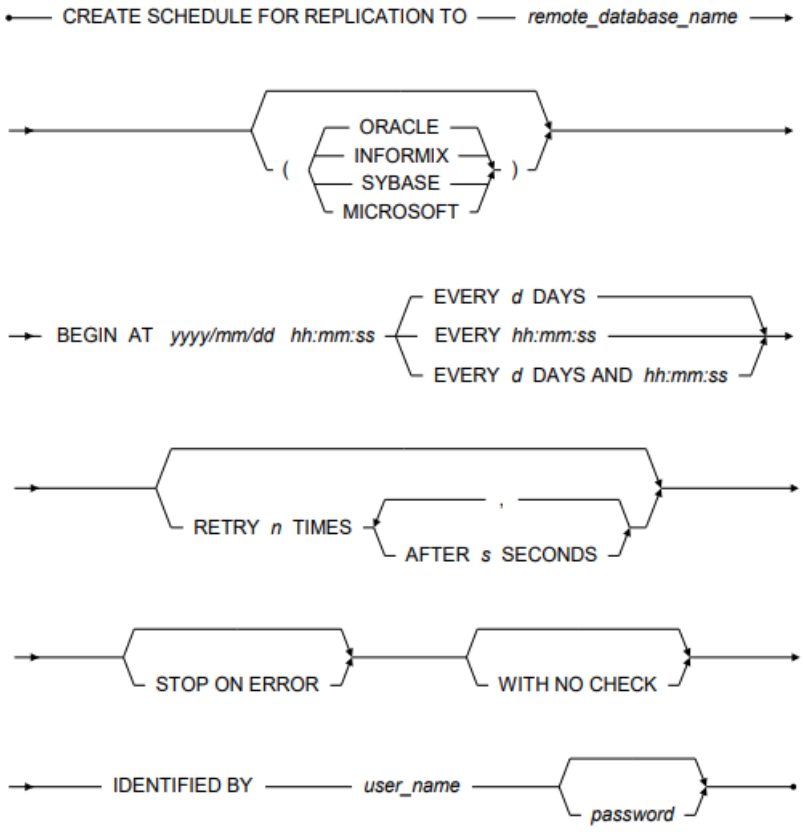
```
[SRCDB]
DB_SvAdr = srcpc
DB_PtNum = 22222
[DESTDB]
DB_DbDir = /disk3/DBMaker/dest
DB_UsrBb = /disk3/DBMaker/dest/DESTDB.BB 3
```

```
DB_UsrDb = /disk3/DBMaker/dest/DESTDB.DB 150
DD_DDBMd = 1
DB_SvAdr = destpc
DB_PtNum = 33333
```

Devido à forma como o servidor distribuidor utiliza o gerenciador de drivers ODBC para realizar a replicação de tabela assíncrona, o nome da fonte de dados ODBC (DSN) do banco de dados de destino deve ser configurado se o banco de dados de origem estiver sendo executado no ambiente Microsoft Windows.

Schedule (Creating and Dropping)

Um cronograma deve ser definido, por um usuário com privilégio de DBA ou superior, antes de criar a replicação assíncrona para uma ou mais tabelas de destino. Um cronograma define o horário de início, o período e a conta e a senha usadas para se conectar ao banco de dados de destino. O usuário pode criar vários cronogramas para diferentes bancos de dados de destino no mesmo banco de dados de origem, mas não pode criar mais de um cronograma para o mesmo banco de dados de destino.



Exemplo 1:

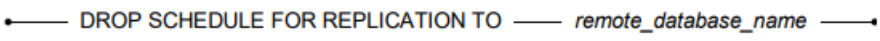
Para criar um cronograma para o banco de dados DESTDB:

```
dmSQL > CREATE SCHEDULE FOR REPLICATION TO destdb
 BEGIN AT 2000/1/1 00:00:00
 EVERY 12:00:00
 IDENTIFIED BY User Password;
```

A partir de 1º de janeiro de 2000, o servidor distribuidor será ativado a cada 12 horas para realizar uma replicação assíncrona. O log de mensagens ATRP.LOG do distribuidor, localizado no diretório home do banco de dados, registrará o horário de início e o status do servidor distribuidor.

A palavra-chave **IDENTIFIED BY** especifica a conta de destino usada pelo servidor distribuidor para se conectar ao banco de dados de destino e executar a replicação da tabela. A conta deve ter privilégios para inserir, excluir e atualizar nas tabelas de destino.

Se o cronograma não for necessário e não houver replicação associada a ele, os usuários que possuem privilégio de DBA no banco de dados de origem podem excluí-lo.



Exemplo 2:

Para excluir uma schedule:

```
dmSQL> DROP SCHEDULE FOR REPLICATION TO destdb;
```

Creating Asynchronous Table Replication

Todas as repetições de tabela assíncronas que dependem do mesmo cronograma serão realizadas ao mesmo tempo. O mecanismo de replicação de tabela assíncrona suporta todos os tipos de dados, incluindo LONG VARCHAR, LONG VARBINARY e FILE.

A ação para criar uma replicação de tabela assíncrona é semelhante à criação de uma replicação de tabela síncrona. Adicione uma palavra-chave (ASYNC, CREATE ASYNC, comando REPLICATION) para criar a replicação de tabela com base em um cronograma para um banco de dados de destino.

Exemplo 1:

Para criar uma replicação chamada rp_AsyRP para o banco de dados SRCDB, com base no cronograma para o banco de dados de destino DESTDB:

```
dmSQL> CREATE ASYNC REPLICATION rp_AsyRP
 WITH PRIMARY AS tb_salary
 REPLICATE TO destdb:tb_salary;
 CLEAR AND FLUSH DATA;
```

Os usuários podem especificar CLEAR DATA, FLUSH DATA ou CLEAR AND FLUSH DATA para realizar a inicialização de dados para um site de destino. Ao criar a replicação, o DBMaker pode usar um link de banco de dados para se conectar ao banco de dados de destino para verificação e inicialização. A ação é realizada através da conta do usuário atual ou da conta de destino, utilizando o banco de dados de destino DESTDB ou o nome do link do banco de dados.

Após a replicação de tabela assíncrona ser criada, a tarefa de replicação é transferida para o servidor distribuidor. A conta usada para se conectar ao banco de dados de destino será alterada para aquela especificada pela opção IDENTIFIED definida na instrução CREATE SCHEDULE.

Qualquer transação resultante da replicação de tabela assíncrona será registrada no log de mensagens do distribuidor ATRP.LOG, localizado no diretório home do banco de dados fonte. O log de mensagens do distribuidor é um arquivo de texto puro e registra a inicialização e as ações do servidor distribuidor.

Exemplo 2:

Conteúdo típico de um arquivo ATRP.LOG:

```
2000/02/09 10:02:30 : start up
2000/02/09 10:02:33 : replicate transactions before 2000/02/09
 10:02:29 (log:1.856152) to DESTDB
```

NO CASCADE OPTION

As palavras-chave NO CASCADE são opcionais. Elas funcionam apenas quando o tipo de replicação é assíncrono. A palavra-chave especifica a replicação em cascata. Comandos fluem na maioria das organizações do nível mais alto para os níveis inferiores; por exemplo, replicar dados de A para B e, em seguida, de B para C. Este é um tipo típico de replicação em cascata. Um modelo típico sem cascata replica

dados para B e B replica dados para A. Se o seu modelo de dados funciona assim, você pode ativar a opção NO CASCADE. O padrão é a configuração CASCADE. A opção NO CASCADE faz com que a replicação da tabela ocorra apenas em um site.

Exemplo:

A replicação Rp1 é de DB1:t1 para DB2:t2, e a replicação Rp2 é de DB2:t2 para DB3:t3. Se Rp1 tiver a opção NO CASCADE ativada, as alterações de DB1:t1 serão replicadas para DB2:t3, mas DB2 deixará de replicar as mesmas alterações para DB3:t3. Se Rp1 tiver a opção CASCADE, as alterações de DB1:t1 serão replicadas para DB2:t2 e, em seguida, de DB2:t2 para DB3:t3.

Error Handling

No processo de replicação de dados, o distribuidor pode enfrentar cinco tipos de erros: aviso, conexão, dados, instrução e transação.

WARNING

Por exemplo, se um tipo de dado CHAR(10) for replicado para uma coluna do tipo CHAR(5), haverá um aviso de truncamento de dados. O servidor distribuidor ignorará esse tipo de erro.

CONNECTION ERRORS

Se o servidor distribuidor não conseguir se conectar ao servidor de banco de dados de destino, ele abandonará a programação e aguardará até a próxima vez. Todos os trabalhos serão mantidos até então.

DATA ERRORS

Como a replicação assíncrona de tabelas é fracamente acoplada, é possível que outra pessoa atualize os dados de destino primeiro. Por exemplo, o servidor distribuidor deseja inserir um registro no banco de dados de destino, mas ele já existe. Outra situação pode ser que o servidor distribuidor deseja excluir um registro, mas ele não existe. Os usuários podem usar a opção STOP ON ERROR para fazer o servidor distribuidor parar quando ocorrerem esses erros. O comportamento padrão do distribuidor é ignorar esse tipo de erro.

NOTA: *O erro de dados mencionado acima inclui um erro de violação de integridade e um erro de linha afetada. O erro de violação de integridade chama a função SQLError() e retorna o estado de erro '23000'. O erro de linha afetada chama a função SQLRowCount(), cujo resultado não é um. Para mais informações sobre funções ODBC, consulte o Guia do Programador ODBC.*

STATEMENT ERRORS

Quando o servidor distribuidor enfrenta erros de tempo de espera de bloqueio ao executar uma instrução, ele precisa aguardar e tentar novamente usando a opção RETRY <n> TIMES. A opção AFTER <s> SECONDS especifica quanto tempo esperar antes da próxima tentativa.

TRANSACTION ERRORS

Por exemplo, um deadlock causa o rollback de transações pertencentes a erros de transação. Se o servidor distribuidor enfrentar erros que fazem o rollback de transações, ele tentará novamente uma vez para cada transação completa. Se o resultado ainda falhar, o distribuidor deixará essas ações para a próxima programação.

Os usuários podem verificar um arquivo de texto chamado ATRERROR.LOG para registros de erros que ocorreram durante a replicação. Este arquivo está localizado no diretório principal do banco de dados.

Schedule (Suspending and Resuming)

Se replicarmos dados para um banco de dados localizado em Tóquio e soubermos que o banco de dados será desligado em feriados tradicionais, podemos suspender a programação até que o banco de dados esteja pronto.

Usuários com privilégio de DBA podem suspender e retomar programações. Após uma programação ser suspensa, o servidor distribuidor parará de tentar se conectar para realizar as replicações.

Exemplo 1:

Para suspender a programação para o banco de dados de destino DESTDB:

```
dmSQL> SUSPEND SCHEDULE FOR REPLICATION TO destdb;
```

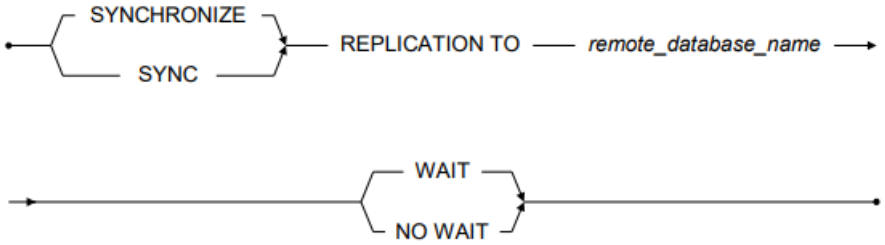
Exemplo 2:

Para reiniciar a programação para o banco de dados de destino DESTDB:

```
dmSQL> RESUME SCHEDULE FOR REPLICATION TO destdb;
```

Synchronizing a Replication

Os usuários às vezes precisarão de dados sincronizados; para isso, o DBMaker fornece programações sincronizadas. A programação sincronizada força o servidor distribuidor a realizar mudanças locais no banco de dados especificado imediatamente. Os usuários não precisam esperar até que a programação ative o servidor distribuidor.



Exemplo 1:

Para sincronizar a programação com o banco de dados de destino DESTDB:

```
dmSQL> SYNC REPLICATION TO destdb WAIT;
```

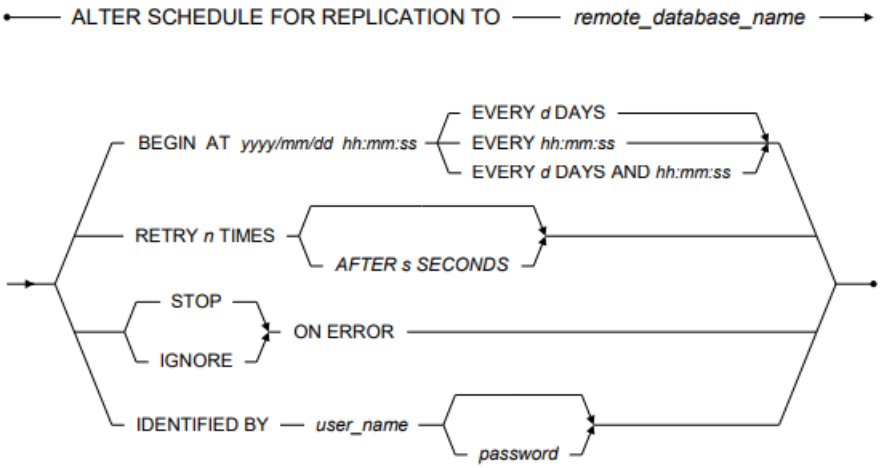
A opção padrão WAIT faz com que o servidor distribuidor aguarde até que todas as alterações tenham sido feitas. O comando retorna apenas após a conclusão da replicação. A opção NO WAIT instrui o servidor distribuidor a realizar seu trabalho imediatamente, e o comando SYNC retornará imediatamente.

Exemplo 2:

```
dmSQL> SYNC REPLICATION TO destdb NO WAIT;
```

Altering Schedule

Após criar uma programação, os usuários com privilégio de DBA podem alterar os atributos de uma programação, incluindo o intervalo de ativação do distribuidor, a conta usada para se conectar ao banco de dados de destino, a opção RETRY e a opção STOP/IGNORE ON ERROR.



Exemplo 1:

Para alterar a programação para replicações no banco de dados de destino DESTDB, adicionando a opção IGNORE ON ERROR:

```
dmSQL> ALTER SCHEDULE FOR REPLICATION TO destdb IGNORE ON ERROR;
```

Exemplo 2:

```
dmSQL> ALTER SCHEDULE FOR REPLICATION TO destdb IDENTIFIED BY User2
Password2;
dmSQL> ALTER SCHEDULE FOR REPLICATION TO destdb STOP ON ERROR;
dmSQL> ALTER SCHEDULE FOR REPLICATION TO destdb RETRY 5 TIMES AFTER
3 SECONDS;
```

Heterogeneous Asynchronous Table Replication

O DBMaker não apenas permite a replicação assíncrona de tabelas para outros bancos de dados DBMaker, mas também para bancos de dados Oracle, Informix, Sybase e Microsoft SQL Server. Esse tipo de replicação permite que o DBMaker coexista com outros bancos de dados em um ambiente heterogêneo, sendo conhecido como replicação de tabelas heterogêneas.

O DBMaker precisa pré-processar os dados replicados antes de enviá-los para um banco de dados de destino de terceiros; os usuários devem especificar o tipo de SGBD para o qual estão replicando ao criar uma programação em um ambiente heterogêneo, usando as palavras-chave ORACLE, INFORMIX, SYBASE e MICROSOFT.

Devido à forma como o DBMaker utiliza o ODBC Driver Manager para realizar a replicação assíncrona de tabelas, o servidor DBMaker deve estar localizado em um computador que execute Windows, e a definição do nome do banco de dados de destino não pode incluir um nome de link. Os bancos de dados de destino de terceiros podem estar localizados em plataformas Windows, UNIX ou Linux.

Ao criar uma programação para replicação de tabelas heterogêneas, utilize as palavras-chave WITH NO CHECK para impedir que o DBMaker realize a verificação de esquema. O usuário que cria a replicação deve assumir a responsabilidade pela verificação do esquema e garantir que as colunas e tipos de dados na tabela de destino sejam compatíveis com as colunas e tipos de dados na tabela local.

Exemplo 1:

Para se conectar a um banco de dados Oracle com um nome de fonte de dados ODBC de orcdb, o usuário orcuser com a senha mypassword deve inserir:

```
dmSQL> CREATE SCHEDULE FOR REPLICATION TO orcdb (ORACLE)
 BEGIN AT 2000/01/01 00:00:00 EVERY 2 DAYS
 WITH NO CHECK
IDENTIFIED BY orcuser mypassword;
```

As palavras-chave CLEAR DATA, FLUSH DATA ou CLEAR AND FLUSH DATA não podem ser usadas ao criar uma replicação de tabelas heterogêneas. Os dados no banco de dados de destino de terceiros devem ser manualmente excluídos ou inseridos para colocar a tabela em seu estado inicial antes que a replicação comece. As programações de replicação heterogênea usam a mesma sintaxe que as programações homogêneas.

Exemplo 2:

O seguinte mostra a replicação heterogênea da tabela tb_salary:

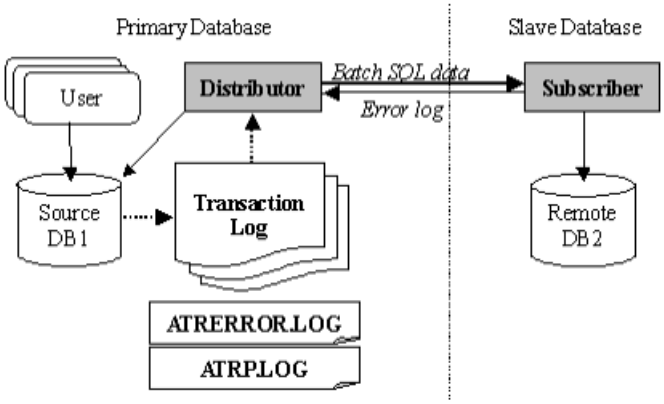
```
dmSQL> CREATE ASYNC REPLICATION rp_HeteroRp
 WITH PRIMARY AS tb_salary
 REPLICATE TO orcdb:orcuser.tb_salary;
```

Express Asynchronous Table Replication

A replicação assíncrona de tabelas utiliza chamadas de função ODBC para se comunicar com bancos de dados de destino, o que pode causar um desempenho ruim em um ambiente de WAN. Para alcançar um desempenho melhor em uma WAN, o

DBMaker fornece outro mecanismo chamado replicação assíncrona de tabelas expressa. O DBMaker agrupa comandos em um pacote para percorrer a rede.

Como outros sistemas de gerenciamento de banco de dados não suportam esse protocolo, as replicações assíncronas de tabelas expressas não podem funcionar com replicações heterogêneas. Além disso, não é suportada a opção STOP ON ERROR ao criar programações expressas.



Há um servidor distribuidor no banco de dados de origem e um daemon assinante no banco de dados de destino. Eles colaboram para realizar o trabalho da replicação assíncrona de tabelas expressa. O distribuidor não aplica diretamente as alterações da tabela de origem na tabela de destino por meio de chamadas ODBC. Em vez disso, ele apenas empacota os comandos SQL e os dados relacionados aplicados na tabela de origem e envia o pacote para o daemon assinante no banco de dados de destino. Após receber o pacote no banco de dados de destino, o daemon assinante aplica os comandos nas tabelas de destino.

Express Replication Setup

Para construir uma replicação expressa:

1. Ative o servidor distribuidor no banco de dados de origem e o daemon assinante nos bancos de dados de destino.
2. Crie uma programação expressa para os bancos de dados de destino.
3. Com base na programação, crie a(s) replicação(ões) assíncronas de tabelas.

ENABLING SUBSCRIBER DAEMON

Para iniciar o daemon assinante, a palavra-chave DB_EtrPt no arquivo dmconfig.ini para o banco de dados de destino (assinante) deve ser configurada. Ela especifica o

número da porta do canal de comunicação entre o servidor distribuidor e o daemon assinante.

Exemplo:

Para replicar uma tabela do banco de dados de origem SRCDB para o banco de dados de destino DESTDB por meio de replicação expressa, o daemon assinante deve ser iniciado no banco de dados de destino.

Um exemplo de arquivo dmconfig.ini em um banco de dados de destino é o seguinte:

```
[SRCDB]
DB_SvAdr = srcpc ; tell the target database where the source
DB_PtNum = 22222 ; database is
[DESTDB]
DB_DbDir = /disk3/DBMaker/dest
DB_UsrBb = /disk3/DBMaker/dest/DESTDB.BB 3
DB_UsrDb = /disk3/DBMaker/dest/DESTDB.DB 150
DD_DDBMd = 1
DB_SvAdr = destpc
DB_PtNum = 33333
DB_EtrPt = 44444 ; port number used by Subscriber Daemon
```

No banco de dados de origem, a palavra-chave DB_AtrMd é utilizada para iniciar o servidor distribuidor. Não é necessário informar ao distribuidor qual número de porta o daemon assinante utiliza para o banco de dados de destino.

Um exemplo de arquivo dmconfig.ini do banco de dados de origem é o seguinte:

```
[SRCDB]
DB_DbDir = /disk1/DBMaker/src
DB_UsrBb = /disk1/DBMaker/src/SRCDB.BB 3
DB_UsrDb = /disk1/DBMaker/src/SRCDB.DB 150
RP_LgDir = /disk1/DBMaker/src/trplog
DB_AtrMd = 1
DD_DDBMd = 1
DB_SvAdr = srcpc
DB_PtNum = 22222
[DESTDB]
DB_SvAdr = destpc
DB_PtNum = 33333
```

SCHEDULE FOR EXPRESS ASYNCHRONOUS TABLE REPLICATION

A replicação assíncrona de tabelas expressas utiliza a opção EXPRESS especificada no comando CREATE SCHEDULE.

Exemplo:

Para construir uma programação para replicação assíncrona de tabelas expressas:

```
dmSQL> CREATE SCHEDULE FOR EXPRESS REPLICATION TO destdb
 BEGIN AT 2000/1/1 00:00:00
 EVERY 12:00:00
 IDENTIFIED BY User Password;
```

CREATING EXPRESS ASYNCHRONOUS TABLE REPLICATION

Os passos são os mesmos que os para criar replicações assíncronas de tabelas. Consulte a seção Criando Replicação Assíncrona de Tabelas ou o Referência de Comandos e Funções SQL para mais informações.

17.4 Database Replication

A maioria das empresas ou organizações precisava armazenar todos os dados em um único servidor de arquivos ou banco de dados em sua sede, e todos os terminais precisavam se conectar diretamente ao servidor. Sob essa arquitetura, o sistema de aplicação poderia funcionar de forma suave se todos os terminais estivessem no mesmo prédio ou área. No entanto, o desempenho era muito mais lento se uma filial remota quisesse acessar o banco de dados, devido à velocidade de transmissão e à largura de banda da rede.

Para compartilhar dados e aumentar a velocidade de acesso, o DBMaker fornece replicação de banco de dados. A replicação de banco de dados irá replicar o banco de dados primário para o banco de dados escravo durante um período de tempo definido. Em outras palavras, quando houver alterações no banco de dados primário, como dados recém-adicionados, modificados ou excluídos, o DBMaker replicará automaticamente os dados alterados para o banco de dados escravo.

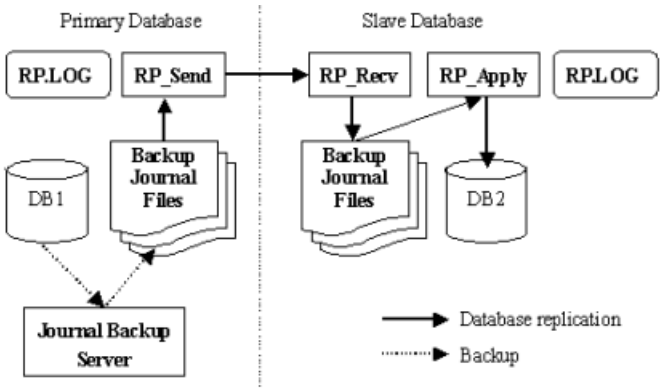
Existem três vantagens em usar a replicação de banco de dados. Primeiro, o desempenho dos sistemas de aplicação aumentará, pois eles poderão acessar dados diretamente do lado local. Segundo, o banco de dados de destino terá as mesmas modificações de dados que o local; assim, o objetivo de compartilhamento de dados

pode ser alcançado de forma eficiente. Terceiro, se não for possível conectar ao banco de dados local para uma aplicação, ainda será possível se conectar ao banco de dados remoto e continuar a operação.

Embora existam vantagens em usar a replicação de dados, mais armazenamento para dados e mais processos para gerenciar a replicação são necessários.

Database Replication Basics

Nesta seção, utilizaremos a Figura 17-11 para explicar o fluxo da replicação de banco de dados. A replicação de banco de dados funcionará com o servidor de backup de diário; se você não estiver familiarizado com o backup e a restauração de banco de dados, consulte o Capítulo 15, Recuperação de Banco de Dados, Backup e Restauração.



A replicação de banco de dados é realizada com 4 servidores: servidor de backup de diário, servidor RP_SEND, servidor RP_RECV e servidor RP_APPLY. O servidor de backup de diário e o servidor RP_SEND são iniciados a partir do banco de dados primário. Os servidores RP_RECV e RP_APPLY são iniciados a partir do banco de dados escravo.

O primeiro passo da replicação de banco de dados é fazer com que o banco de dados escravo seja igual ao primário. Todas as mudanças subsequentes, inserções de dados, exclusões, criação de esquemas etc., devem ser feitas primeiro no banco de dados primário. Em seguida, o servidor de backup de diário em execução no banco de dados primário grava regularmente as alterações no diário de backup.

O servidor RP_SEND envia periodicamente o diário de backup do banco de dados primário para o servidor RP_RECV no escravo. O RP_RECV notifica o RP_APPLY para aplicar o diário de backup recebido ao banco de dados escravo.

O banco de dados escravo é somente leitura para todos os usuários; apenas o RP_APPLY pode atualizar o banco de dados escravo.

Database Replication Setup

Para configurar manualmente a replicação de banco de dados:

1. Duplique o banco de dados primário para o banco de dados escravo.
2. Configure o servidor de backup de diário e o servidor RP_SEND no banco de dados primário.
3. Configure o servidor RP_RECV e o servidor RP_APPLY no banco de dados escravo.
4. Inicie os bancos de dados primário e escravo.
5. Verifique o log de replicação (RP.LOG) e o log de erros (DMERROR.LOG).

FULL BACKUP

Como mencionado anteriormente, o primeiro passo da replicação de banco de dados é tornar o banco de dados escravo idêntico ao primário. É importante observar que a ordenação de bytes das máquinas do banco de dados primário e escravo deve ser a mesma. Por exemplo, se o banco de dados primário estiver sendo executado em uma máquina x86, a máquina do banco de dados escravo deve ser compatível em termos de ordenação de bytes com a x86; a ordenação de bytes do Sun Sparc é diferente da da x86, portanto, o banco de dados escravo não pode ser executado em uma máquina baseada em Sparc ou vice-versa.

Para duplicar o banco de dados primário:

1. Desligue o banco de dados primário.
2. Faça uma cópia dos arquivos de dados, arquivos BLOB e arquivos de diário do banco de dados primário.
3. Mova a cópia dos arquivos para a máquina que executará o banco de dados escravo.
4. Modifique o arquivo de configuração dmconfig.ini do banco de dados escravo para refletir as alterações correspondentes no diretório dos arquivos.

Os passos 1 e 2, combinados, constituem um backup completo offline. Consulte a Seção de Backups Completos Offline para obter informações mais detalhadas. Aqui, ilustraremos como realizar um backup completo offline manualmente. Nesta seção, todos os exemplos assumem um processador x86 e o banco de dados primário em execução no Linux ou FreeBSD.

Exemplo:

Para copiar os arquivos do banco de dados primário, consulte a tabela de sistema SYSFILE para obter os nomes lógicos:

```
dmSQL> SELECT * FROM SYSFILE;
```

Um trecho do arquivo dmconfig.ini para visualizar os nomes físicos dos arquivos:

```
[MYDB] ;; Primary Database Configuration, CPU : x86, OS: FreeBSD
DB_DbDir = /home/dbmaker/mydb
DB_UsrBb = /home/dbmaker/mydb/MYDB.BB 3
DB_UsrDb = /home/dbmaker/mydb/MYDB.DB 150
FILE1 = /home/dbmaker/mydb/data/FILE1.DB 50
FILE2 = /home/dbmaker/mydb/data/FILE2.DB 50
DB_JnFil = JN1.JNL JN2.JNL
...
```

Após desligar o banco de dados primário, copie os arquivos do banco de dados primário para a máquina que executará o banco de dados escravo. Aqui, a máquina escrava é um x86 rodando plataformas Windows. No exemplo anterior, os arquivos a serem copiados incluem os arquivos de sistema MYDB.SDB e MYDB.SBB, os arquivos de usuário padrão MYDB.DB e MYDB.BB, os arquivos de diário JN1.JNL e JN2.JNL, e os arquivos definidos pelo usuário FILE1.DB e FILE2.DB.

Em seguida, copie a seção do banco de dados primário no arquivo de configuração dmconfig.ini para o dmconfig.ini no computador que serve o banco de dados escravo. Depois, modifique o dmconfig.ini do banco de dados escravo para garantir que os nomes físicos dos arquivos correspondam logicamente, uma vez que os caminhos reais e as convenções de nomes de caminho podem ser diferentes para máquinas distintas.

Um trecho do dmconfig.ini do banco de dados escravo pode parecer assim:

```
[MYDB] ;; Slave Database Configuration, CPU : x86, OS: MS Windows NT
DB_DbDir = d:\\dbmaker\\db\\mydb
DB_UsrBb = d:\\dbmaker\\db\\MYDB.BB 3
DB_UsrDb = d:\\dbmaker\\db\\MYDB.DB 150
FILE1 = d:\\dbmaker\\db\\mydb\\FILE1.DB 50
FILE2 = d:\\dbmaker\\db\\mydb\\FILE2.DB 50
DB_JnFil = JN1.JNL JN2.JNL
...
```

O DBMaker suporta até 256 bancos de dados escravos; se houver mais de um banco de dados escravo, o administrador do banco de dados terá que repetir os passos acima para cada banco de dados escravo.

SETUP PRIMARY DATABASE'S JOURNAL BACKUP SERVER

Todas as alterações em um banco de dados são registradas nos arquivos de jornal. Portanto, o DBMaker utiliza os arquivos de backup de jornal como a fonte de dados para replicação. Após o servidor de backup de jornal realizar um backup incremental, o servidor RP_SEND envia os arquivos de backup de jornal para o banco de dados escravo. Todas as transações registradas nos arquivos de jornal são, então, confirmadas para garantir que todas as alterações no banco de dados primário também ocorram no lado escravo.

Exemplo: Apenas o banco de dados primário precisa iniciar o servidor de backup de jornal. As configurações do arquivo dmconfig.ini no servidor do banco de dados primário especificam o diretório de backup e o cronograma:

```
DB_BMode = 1 ; Database start up in BACKUP-DATA mode
DB_BkSvr = 1 ; Start up journal backup server
DB_BkDir = /home/dbmaker/mydb/bkdir ; Directory of backup journal
files
DB_BkTim = 00/01/01 00:00:00 ; The initial backup time
DB_BkItv = 0-12:00:00 ; Perform journal backup evey 12 hours
```

O DB_BMode pode ser configurado como modo BACKUP-DATA (1) ou BACKUP-DATA-AND-BLOB (2). No entanto, mesmo que o DB_BMode esteja no modo BACKUP-DATA-AND-BLOB, a opção BACKUP BLOB ON ainda deve ser configurada para todos os tablespaces que precisam ser replicados; caso contrário, os dados BLOB não serão replicados para o banco de dados escravo.

DATA TRANSMITTED AND RECEIVED

No processo de replicação de banco de dados, há três servidores residentes envolvidos: RP_SEND, RP_RECV e RP_APPLY (conforme ilustrado na Figura 17-11). O RP_SEND está localizado no banco de dados primário e é responsável por enviar arquivos de backup de journal para o lado escravo. O RP_RECV e o RP_APPLY estão no lado do banco de dados escravo. O RP_RECV recebe arquivos de backup de journal do banco de dados primário, enquanto o RP_APPLY executa as alterações de acordo com os arquivos de journal.

O RP_SEND é iniciado e encerrado automaticamente sempre que o banco de dados primário é iniciado ou desligado. O RP_RECV e o RP_APPLY também começam e terminam simultaneamente com o banco de dados escravo.

Depois que o servidor de backup de journal completa um backup incremental, o RP_SEND enviará todos os arquivos de backup que ainda não foram enviados para o banco de dados escravo, a partir do diretório de backup (DB_BkDir). Enquanto isso, o RP_RECV no banco de dados escravo receberá todos os arquivos de backup transferidos do banco de dados primário e os colocará sob o diretório de backup (DB_BkDir) no banco de dados escravo. Após os arquivos terem sido recebidos, o RP_APPLY executará as alterações registradas nos arquivos de backup de journal no banco de dados escravo, e o fluxo de replicação de banco de dados será concluído.

SETUP RP_SEND SERVER ON THE PRIMARY SIDE

RP_SEND e RP_RECV são responsáveis pela transferência e recepção do arquivo de registro de backup, respectivamente. Assim, o RP_SEND deve conhecer o endereço IP e o número da porta da máquina onde o RP_RECV está localizado. Observe que o número da porta é especificamente para a comunicação entre o RP_SEND e o RP_RECV para replicação de dados, e deve ser diferente daquele utilizado pelo servidor de banco de dados. O RP_SEND utiliza a palavra-chave RP_SlAdr no dmconfig.ini do banco de dados primário para determinar para onde enviar os dados de replicação.

Sintaxe para RP_SlAdr:

```
RP_SlAdr = {address[:port number]}
```

O número da porta padrão é 23001.

Exemplo 1:

O banco de dados primário pode suportar até 256 bancos de dados escravos. Use uma vírgula ou um espaço para separar as informações dos bancos de dados escravos. O exemplo a seguir utiliza três bancos de dados escravos: 192.168.9.222 (com o número da porta 5100), Mars (com o número da porta 5101) e Scorpio (com o número da porta padrão 23001).

```
RP_SlAdr = 192.168.9.222:5100, Mars:5101, Scorpi
```

O horário inicial e o intervalo de tempo para o RP_SEND executar a replicação do banco de dados podem ser definidos porque o destino da replicação de dados é

conhecido. Após estabelecer a programação, o RP_SEND executa periodicamente a replicação de dados.

Exemplo 2:

No arquivo de configuração, RP_BTime e RP_Iterv são utilizados para especificar a programação da replicação de dados. RP_BTime é o horário de início do envio dos arquivos de registro de backup para o lado escravo. O formato de RP_BTime é <ano/mês/dia hora:minuto>. Se não for fornecido, o sistema definirá RP_BTime como o horário de início do banco de dados primário. O RP_Iterv especifica o intervalo de tempo em que o RP_SEND é ativado. Seu formato é

dia-hora:minuto:segundo. O valor padrão é um dia. O intervalo válido varia de 0 a 24.855.

NOTA: *Esses valores devem ser definidos antes de iniciar o banco de dados.*

RP_ReTry define quantas vezes tentar novamente caso a conexão de rede falhe. O valor de RP_Clear determina se os arquivos de registro de backup devem ser limpos após serem enviados para o escravo. Configurar RP_Clear como 1 limpa os arquivos de registro de backup. O valor padrão é 0. Se os arquivos de registro de backup forem usados apenas para replicação de banco de dados, limpar esses arquivos pode recuperar um espaço de armazenamento. No entanto, se ocorrer uma falha de hardware, os dados dos arquivos de registro de backup não poderão ser recuperados e não poderão ser restaurados. Nesse caso, um backup completo do banco de dados escravo deve ser feito para restaurar o banco de dados primário. Portanto, se os arquivos de registro de backup também forem usados para o backup do banco de dados primário, RP_Clear deve ser definido como 0. RP_BkFoM determina se os objetos de arquivo do sistema, funções definidas pelo usuário e procedimentos armazenados devem ser replicados. O RP_CRCChk determina se a verificação CRC deve ser realizada durante a replicação de arquivos.

SETUP RP_RECV AND RP_APPLY SERVERS ON A SLAVE

Para iniciar os servidores RP_RECV e RP_APPLY, o modo de inicialização DB_SMode do banco de dados escravo deve ser definido como 5.

Exemplo 1:

Um banco de dados escravo pode receber dados de replicação apenas de um banco de dados primário; portanto, é necessário usar RP_Primy para especificar o nome ou endereço da máquina do banco de dados primário.

```
RP_Primy = FreeBSD ; Receive replication data from machine 'FreeBSD'
```

Exemplo 2:

O servidor RP_RECV utiliza um número de porta diferente do DB_PtNum para receber dados do RP_SEND. Por exemplo, suponha que haja um banco de dados escravo localizado na máquina NTPC, com a configuração da palavra-chave RP_PtNum.

```
RP_PtNum = 5100 ; Port number for RP_SEND and RP_RECV connection
DB_PtNum = 3333 ; Port number for user database access
```

Exemplo 3:

O banco de dados primário usa a palavra-chave RP_SlAdr para recuperar o nome ou endereço da máquina do banco de dados escravo e o número da porta.

```
RP_SlAdr = NTPC:5100 ; Slave-side machine name and address
```

Além disso, o banco de dados escravo deve definir o diretório de backup DB_BkDir para armazenar os arquivos de registro de backup recebidos do banco de dados primário. Após o RP_APPLY aplicar as alterações ao banco de dados escravo com os arquivos de registro de backup, o DBMaker os removerá automaticamente.

SLAVE DATABASE IS READ-ONLY

Os dados no banco de dados escravo devem ser os mesmos do primário. Ele não pode aceitar definições de dados (DDL, como Create Table e Alter Table) e dados atualizados (como INSERT, UPDATE e DELETE). Assim, podemos afirmar que o banco de dados escravo é somente leitura.

Durante o processo de replicação de banco de dados, o banco de dados escravo utiliza os arquivos de registro de backup recebidos do banco de dados primário para restaurar os dados. O sistema não bloqueará os dados no banco de dados primário durante o processo de restauração. Portanto, as consultas ao banco de dados escravo são uma forma de leitura suja. Em outras palavras, em um determinado momento, a mesma consulta nos bancos de dados primário e escravo pode retornar valores diferentes porque o RP_APPLY está restaurando dados.

START THE PRIMARY AND SLAVE DATABASES

O modo de inicialização do banco de dados primário é diferente do escravo. Se você deseja definir um banco de dados como primário, deve configurar o modo de inicialização para o modo de banco de dados primário. Por outro lado, se você deseja definir um banco de dados como escravo, deve configurar o modo de inicialização para o modo de banco de dados escravo. Usando o DB_SMode localizado no arquivo dmconfig.ini, é possível especificar todas as opções de configuração. O modo de inicialização DB_SMode para o modo de banco de dados primário é 4. O DB_SMode para o modo de banco de dados escravo é 5.

Você pode iniciar os bancos de dados primário e escravo separadamente, sem uma ordem específica.

A seguir, um resumo de todas as palavras-chave dmconfig.ini relacionadas à replicação mencionadas nesta seção.

Exemplo:

No seguinte arquivo dmconfig.ini, o nome do banco de dados primário é FreeBSD, e o banco de dados escravo é NTPC, utilizando as configurações mínimas para o banco de dados primário.

```
[MYDB] ;; Primary Config, CPU : x86, OS : FreeBSD, Name : FreeBSD
;; Database related settings
DB_DbDir = /home/dbmaker/mydb
DB_UsrBb = /home/dbmaker/mydb/MYDB.BB 3
DB_UsrDb = /home/dbmaker/mydb/MYDB.DB 150
FILE1 = /home/dbmaker/mydb/data/FILE1.DB 50
FILE2 = /home/dbmaker/mydb/data/FILE2.DB 50
DB_JnFil = JN1.JNL JN2.JNL
DB_SvAdr = FreeBSD ; Machine name of primary database
DB_PtNum = 3333 ; Port number of primary database
;; journal backup server related settings
DB_BMode = 1 ; Database start up in BACKUP-DATA mode
DB_BkSvr = 1 ; Start up journal backup server
DB_BkDir = /home/dbmaker/mydb/bkdir ; Directory of backup journal
files
DB_BkTim = 00/01/01 00:00:00 ; The initial backup time
DB_BkItv = 0-12:00:00 ; Perform journal backup every 12 hours
;; RP_SEND server related settings
DB_SMode = 4 ; Start as primary database
 ; and also start up RP_SEND server
```

```
RP_BTime = 00/01/01 01:00:00 ; Initial replication time
RP_Iterv = 1-00:00:00 ; Replicate everyday
RP_SlAdr = NTPC:5100 ; Replicate to NTPC with port no. 5100
RP_ReTry = 3 ; retry 3 timesif network connection fails
RP_Clear = 1 ; Clear backup journal files after sending
RP_BkFoM = 1 ; Replicate system FO, UDF and SP
RP_CRCChk = 1; Do the CRC check when there's file of the same name
on both side
```

A seguir estão as configurações mínimas para o banco de dados escravo:

```
[MYDB];; Slave Config., CPU : x86, OS : MS Windows NT, Name : NTPC
;; Database related settings
DB_DbDir = d:\\dbmaker\\db\\mydb
DB_UsrBb = d:\\dbmaker\\db\\MYDB.BB 3
DB_UsrDb = d:\\dbmaker\\db\\MYDB.DB 150
FILE1 = d:\\dbmaker\\db\\mydb\\FILE1.DB 50
FILE2 = d:\\dbmaker\\db\\mydb\\FILE2.DB 50
DB_JnFil = JN1.JNL JN2.JNL
DB_SvAdr = NTPC
DB_PtNum = 3333
;; RP_RECV and RP_APPLY servers related settings
DB_SMode = 5 ; Start as slave database,
 ; also start up RP_RECV and RP_APPLY servers
RP_Primy = FreeBSD ; Receive replication data only from this machine
RP_PtNum = 5100 ; Port number for RP_SEND and RP_RECV connection
DB_BkDir = e:\\mydb\\bkdir ; Directory of temporary backup journal
files
```

ADD OR DELETE SLAVE DATABASE IN DMSQL

Se você deseja adicionar ou excluir um banco de dados escravo online, pode usar o procedimento armazenado do sistema setSystemOption()/setSystemOptionW() no dmsql para adicionar ou excluir um banco de dados escravo. A diferença entre setSystemOption e setSystemOptionW é se deve ou não gravar no dmconfig.ini. O endereço do escravo alterado estará ativo na próxima replicação.

NOTA: *Esta operação só pode ser utilizada no banco de dados mestre (DB_SMODE=4).*

A sintaxe para adicionar o endereço do escravo é a seguinte:

```
call SETSYSTEMOPTION('ADDSLAVR’,slave database address);
call SETSYSTEMOPTION (‘SLADRADD’,slave database address);
```

Exemplo:

adicionar um endereço de escravo host2:20000

```
call SETSYSTEMOPTION (‘ADDSLAVE’,’host2:20000’)
```

Sintaxe para excluir endereço de escravo:

```
call SETSYSTEMOPTION (‘DELSLAVE, slave database address);
call SETSYSTEMOPTION (‘SLADRDEL’,slave database address)
```

Exemplo:

excluir banco de dados de escravo host2:20000

```
call SETSYSTEMOPTIONW (‘DELSLAVE’, ’host2:20000’)
```

EXECUTE DATABASE REPLICATION IMMEDIATEL

Mencionamos que o servidor residente para replicação de banco de dados detectará se houve alterações nos dados e replicará os dados automaticamente.

Exemplo:

Use a ferramenta dmSQL para inserir um comando SQL que faça o DBMaker executar a replicação do banco de dados imediatamente.

```
dmSQL> SET FLUSH;
```

Use este comando quando precisar sincronizar os dados entre um banco de dados primário e um escravo.

Quando você executar este comando, o servidor de backup de diário executará imediatamente o backup incremental e fará o backup dos arquivos de diário atuais.

Em seguida, três servidores residentes (RP_SEND, RP_RECV e RP_APPLY) seguirão o procedimento e replicarão os dados.

VERIFY REPLICATION LOG (RP.LOG) AND ERROR LOG(DMERROR.LOG)

No processo de replicação de banco de dados, se houver falhas de rede ou mensagens de erro, o sistema gerará um arquivo de log, DMERROR.LOG, no diretório do banco de dados atual. As entradas do log de erros seguem a seguinte sintaxe:

```
yy/mm/dd hh:mm:ss Daemon name:Error number:Error message
```

Exemplo 1:

A DMERROR.LOG:

```
97/12/31 11:40:59 - RP_SEND:rc = 1503, cannot connect to server 192.72.116.130
97/12/31 11:43:36 - RP_SEND:rc = 1503, cannot connect to server 192.72.116.130
97/12/31 11:45:00 - RP_SEND:rc = 1503, cannot connect to server 192.72.116.130
97/12/31 11:50:00 - RP_SEND:rc = 1503, cannot connect to server 192.72.116.130
97/12/31 11:50:45 - RP_SEND:rc = 1503, cannot connect to server 192.72.116.130
```

Tanto o banco de dados primário quanto o escravo podem gerar o DMERROR.LOG. Se a replicação for bem-sucedida, o sistema também gerará um arquivo de log chamado RP.LOG. O formato é:

No banco de dados primário:

```
RP_SEND:RPID    id    ~    id    sent    at    yy/mm/dd    hh:mm:ss
```

Na base de dados escrava:

```
RP_RECV:RPID id ~ id sent at yy/mm/dd hh:mm:ss
RP_APPLY:RPID id ~ id applied at yy/mm/dd hh:mm:ss
```

O log de replicação, RP.LOG, estará nos servidores de banco de dados primário e escravo. Cada arquivo de diário de backup tem um ID, o RPID. No formato acima, o RPID indica qual arquivo de diário está sendo processado – o RPID nas linhas RP_SEND indica qual está sendo enviado, nas linhas RP_RECV indica qual está sendo

recebido e nas linhas RP_APPLY indica qual está sendo restaurado. Normalmente, o RPID é o mesmo que o ID do arquivo de backup incremental.

Exemplo 2:

RP.LOG em uma base de dados primária:

```
RP_SEND : RPID 7 ~ 10 sent to 192.72.116.130 at 97/12/16 15:36:17
RP_SEND : RPID 11 ~ 11 sent to 192.72.116.130 at 97/12/16 15:59:42
RP_SEND : RPID 12 ~ 12 sent to 192.72.116.130 at 97/12/31 11:52:28
```

Exemplo 3:

RP.LOG em uma base de dados escrava:

```
RP_RECV : RPID 7 ~ 10 received at 97/12/16 15:35:53
RP_APPLY : RPID 7 ~ 10 applied at 97/12/16 15:35:55
RP_RECV : RPID 11 ~ 11 received at 97/12/16 15:59:18
RP_APPLY : RPID 11 ~ 11 applied at 97/12/16 15:59:18
RP_RECV : RPID 12 ~ 12 received at 97/12/31 11:52:01
RP_APPLY : RPID 12 ~ 12 applied at 97/12/31 11:52:02
```

RP.LOG é usado para registrar todas as ações realizadas por três servidores residentes, RP_SEND, RP_RECV e RP_APPLY. O arquivo está localizado no diretório da base de dados, DB_DbDir, em todas as bases de dados participantes da replicação. Os administradores de banco de dados devem monitorar esses arquivos periodicamente para garantir que a replicação esteja funcionando corretamente.

Por exemplo, se a conexão com um banco de dados remoto falhar constantemente, verifique se a rede está funcionando normalmente ou se o banco de dados remoto estava ocupado no momento da falha.

JServer Manager Environment Settings

Para replicar uma base de dados, você precisa estabelecer o ambiente inicial da base de dados. Use um editor de texto para modificar o arquivo dmconfig.ini diretamente, como mostrado na seção anterior. Nesta seção, o ambiente de replicação da base de dados é configurado usando o JServer Manager do DBMaker.

PRODUCE A FULL BACKUP

Primeiro, gere um backup completo offline da base de dados primária e, em seguida, copie o backup para a base de dados escrava. Para mais informações sobre backup completo offline, consulte a Seção 15.5, Backups Completos Offline.

PRIMARY DATABASE SETUP

Para permitir que o servidor de replicação da base de dados funcione, primeiro configure a base de dados para que sejam realizados backups incrementais. Consulte o Capítulo 15, Recuperação de Banco de Dados, Backup e Restauração, para mais informações. Em seguida, configure o ambiente para a base de dados primária.

Para configurar o ambiente da base de dados primária:

1. Inicie o aplicativo JServer Manager no servidor da base de dados primária.
2. Clique no botão Configurar na janela Iniciar Base de Dados.
3. Clique na aba Replicação na janela de configurações avançadas da Iniciar Base de Dados.
4. Para habilitar a replicação da base de dados: a) Insira o IP e os números de porta da base de dados escrava nos campos IP e número da porta da base de dados de destino. b) Insira uma data e hora nos campos de Hora de Início da replicação da base de dados. c) Insira um valor no campo Tentativas de repetição em caso de falha. d) Se desejar, habilite a opção Remover arquivos de registro de backup após a replicação. e) Insira o número de dias, horas, minutos e segundos entre cada replicação sucessiva da base de dados nos campos de Intervalo de Tempo para iniciar a replicação da base de dados.
5. Clique no botão Salvar.
6. Clique no botão Cancelar para retornar à janela Iniciar Base de Dados.

SLAVE DATABASE SETUP

Após copiar um backup completo da base de dados primária para a base de dados escrava, use o JServer Manager para configurar a configuração da base de dados escrava.

1. Inicie o aplicativo JServer Manager no servidor da base de dados escrava.
2. Clique no botão Configurar na janela Iniciar Base de Dados.
3. Clique na aba Replicação na janela de configurações avançadas da Iniciar Base de Dados.

4. Para habilitar a replicação da base de dados: a) Insira o endereço IP da Base de Dados Fonte. b) Insira um número de porta para RP_RECV no campo Número da porta do daemon de recebimento na base de dados de destino.
5. Clique no botão Salvar.
6. Clique no botão Cancelar para retornar à janela Iniciar Base de Dados.

Database Configuration File

Esta seção é um resumo das palavras-chave relacionadas à replicação de banco de dados usadas no arquivo dmconfig.ini.

PRIMARY DATABASE CONFIGURATION

Palavras-chave da base de dados primária para replicação de dados:

- **DB_SMode** — Configurar DB_SMode para 4 significa que esta base de dados será iniciada em modo primário.
- **RP_BTime** — hora de início para os arquivos de backup completo da base de dados primária serem enviados para a base de dados escrava. O formato é ano/mês/dia hora:minuto. O valor padrão é o horário de início da base de dados primária. Por exemplo: 97/12/31 12:00:00.
- **RP_Iterv** — intervalo de tempo para enviar os arquivos de registro de backup. O formato é dia-hora:minuto. Por exemplo, 1-12:00:00 significa enviar os arquivos de backup a cada um dia e meio. O intervalo válido de dias é de 0 a 24.855.
- **RP_ReTry** — número de tentativas para reconectar uma conexão de rede com falha.
- **RP_Clear** — limpa os arquivos de backup de registro após enviá-los. Um valor de 1 limpa os arquivos; um valor de 0 não os limpa. O valor padrão é 0. Se o valor for definido como 1, a base de dados primária não pode ser restaurada em caso de danos de hardware, a menos que a base de dados escrava seja usada para restaurá-la.
- **RP_SlAdr** — o endereço ou número da máquina e o número da porta da base de dados escrava. O DBMaker suporta de 1 a 8 bases de dados escravas para cada base de dados primária.

Sintaxe para RP_SlAdr:

```
RP_SlAdr = {Address[:Port Number]}
```

A base de dados primária precisa iniciar o servidor de backup de registro para realizar backups incrementais:

- **RP_BkFoM** — configurar como 1 significa replicar o FO do sistema, UDF e SP.
- **RP_CRCChk** — configurar como 1 significa realizar a verificação CRC no arquivo replicado.
- **DB_BMode** — 1 (BACKUP-DATA) ou 2 (BACKUP-DATA-AND-BLOB).
- **DB_BkSvr** — Defina DB_BkSvr como 1 para iniciar o servidor de backup de registro.
- **DB_BkDir** — diretório para armazenar arquivos de registro de backup.
- **DB_BkTim** — hora de início do servidor de backup de registro. O formato é <ano/mês/dia hora:minuto>. O valor padrão é a hora de início da base de dados primária.
- **DB_BkItv** — intervalo de tempo para realizar backups incrementais, e o formato é dia-hora:minuto:segundo. Por exemplo, 0-12:00:00 significa fazer backup a cada 12 horas.

SLAVE DATABASE CONFIGURATION

As palavras-chave da base de dados escrava para replicação de dados:

- **DB_SMode** — Configurar DB_SMode para 5 inicia a base de dados em modo escravo.
- **RP_Primy** — endereço ou nome da máquina da base de dados primária.
- **RP_PtNum** — número da porta utilizado pelo RP_RECV. O valor deve ser diferente de DB_PtNum e deve ser o mesmo que o número da porta especificado em RP_SlAdr da base de dados primária.
- **DB_BkDir** — diretório para armazenar arquivos temporários de registro de backup recebidos da base de dados primária. O diretório padrão é <diretório da base de dados>/backup.

Database Replication Limitations

Resumo das limitações da replicação de banco de dados:

- A ordem dos bytes deve ser a mesma nas máquinas primária e escrava.
- A base de dados escrava é somente leitura.
- Uma base de dados primária suporta até 256 bases de dados escravas.
- É possível restaurar a base de dados primária com a sequência de backups. No entanto, após a restauração da base de dados primária, a replicação não continuará. Se os usuários desejarem continuar a replicação, todas as bases de dados escravas devem ser substituídas pela nova base de dados primária; ou seja, os usuários devem copiar os arquivos da base de dados primária para substituir os arquivos de todas as bases de dados escravas.

- O servidor de replicação pode não limpar os arquivos de backup incremental devido ao backup completo, portanto, um grande número de arquivos pode existir no diretório BKDIR se o próximo backup completo não for realizado por um longo período.
- Para replicar dados BLOB, ou seja, colunas do tipo LONG VARCHAR ou LONG VARBINARY, configure DB_BMode para 2 (BACKUP-DATA-AND-BLOB) e lembre-se de definir a opção BACKUP BLOB ao criar tablespaces.