13. Schedules

As organizações têm muitas tarefas, e lidar manualmente com cada uma pode ser assustador. Para ajudar os usuários a simplificar essas tarefas de gerenciamento, além de oferecer um conjunto rico de funcionalidades para necessidades complexas de agendamento, o DBMaker oferece capacidades avançadas de agendamento de tarefas. Um agendamento indica quando uma tarefa deve ser executada. Um agendamento inclui uma hora de início que especifica a data e a hora em que o agendamento começa, uma hora de término que indica a data e a hora em que o agendamento expira, bem como uma tabela de horários que indica quando um agendamento será executado.

O agendamento permite que os usuários controlem quando e onde várias tarefas acontecem no ambiente de banco de dados. Essas tarefas podem ser demoradas e complicadas, então, usar o agendamento pode ajudar os usuários a melhorar o gerenciamento e o planejamento dessas tarefas. Além disso, ao garantir que muitas tarefas rotineiras do banco de dados ocorram sem intervenção manual, é possível reduzir os custos operacionais, implementar rotinas mais confiáveis, minimizar erros humanos e diminuir o tempo necessário para o sistema.

O DBMaker fornece capacidades avançadas de agendamento de tarefas com um conjunto de procedimentos armazenados:

1. iniciar e parar o DMSCHSVR:START_DMSCHSVR, STOP_DMSCHSVR

2. procedimentos armazenados sobre tarefas:TASK_CREATE, TASK_ALTER, TASK_DROP

3. procedimentos armazenados sobre agendamentos:SCHEDULE_CREATE, SCHEDULE_ALTER, SCHEDULE_DROP, SCHEDULE_RELOAD, SCHEDULE_ENABLE, SCHEDULE_DISABLE, SCHELOG_CLEANPara mais detalhes sobre esses procedimentos, consulte o SQL Command and Function Reference. Um usuário com privilégio de RESOURCE pode criar/alterar/excluir uma tarefa/agendamento e ativar/desativar um agendamento para si mesmo; um usuário com privilégio de DBA ou superior pode criar/alterar/excluir uma tarefa/agendamento, ativar/desativar um agendamento de todos os usuários, iniciar/parar o dmschsvr e recarregar todos os agendamentos. Além disso, os usuários criam um agendamento apenas com base em sua própria tarefa.

13.1 Dmschsvr

Dmschsvr é um servidor de agendamento para gerenciar todos os agendamentos no DBMaker. O daemon dmschsvr é utilizado principalmente para executar uma instrução SQL, um procedimento ou um programa executável, todos projetados pelos usuários com antecedência. Os usuários podem criar agendamentos para tarefas rotineiras, e o dmschsvr executará automaticamente essas tarefas, reduzindo significativamente a manutenção de rotina. O dmschsvr executa um trabalho periodicamente com base nas informações armazenadas nas tabelas do sistema SYSSCHEDULE e SYSTASK.

1. SYSSCHEDULE é usada para armazenar agendamentos no banco de dados. As instruções SCHEDULE_CREATE, SCHEDULE_ALTER, SCHEDULE_DROP, SCHEDULE_ENABLE e SCHEDULE_DISABLE são usadas, respectivamente, para criar, alterar, excluir, habilitar e desabilitar um agendamento.

2. SYSTASK é usada para armazenar tarefas no banco de dados. As instruções TASK_CREATE, TASK_ALTER e TASK_DROP são usadas, respectivamente, para criar, alterar e excluir uma tarefa. Após a chamada desses procedimentos, as informações armazenadas em SYSSCHEDULE e SYSTASK serão alteradas.

O dmschsvr utiliza um usuário especial para se conectar a um banco de dados através do ODBC quando está em execução, e então escaneia todas as informações de agendamento armazenadas em SYSSCHEDULE para encontrar os agendamentos habilitados, e finalmente lê as informações do agendamento para o sistema. De acordo com essas informações, o dmschsvr calcula a tarefa que tem o menor intervalo entre o horário atual e seu horário de execução, e então insere os dados dessa tarefa em uma fila. Quando o horário de execução chega, essa tarefa é executada. Se os dados de SYSSCHEDULE mudarem, o dmschsvr recarregará automaticamente todos os agendamentos. Os usuários podem iniciar o dmschsvr através do comando dmschsvr –d db_name, definindo o valor de DB_SchSv para 1, ou chamando o procedimento armazenado start_dmschsvr. Após a chamada do procedimento armazenado stop_dmschsvr, o dmschsvr será interrompido dentro de um minuto, e os agendamentos não serão executados se o dmschsvr estiver interrompido.

Os usuários podem iniciar o **dmschsvr** através do comando **dmschsvr –d db_name,** definindo o valor de **DB_SchSv** para 1, ou chamando start_dmschsvr('taskNum', 'logPath'). Para mais detalhes sobre START_DMSCHSVR, consulte a referência de Comandos e Funções SQL. Além disso, o comando **dmschsvr –d** possui outros parâmetros: -n, que especifica o número máximo de tarefas a serem executadas ao mesmo tempo; e -p, que especifica o diretório onde os logs de agendamento serão

salvos. No entanto, os usuários só podem parar o dmschsvr chamando stop_**dmschsvr**, e o **dmschsvr** será interrompido dentro de um minuto após o usuário chamar stop_dmschsvr.

Após a inicialização, o **dmschsvr** gerará um arquivo de log diariamente, que será armazenado no diretório **DB_DBDIR**, e o formato desse arquivo de log será <db_name><_><date>.log. Por exemplo, o arquivo de log criado em 04/03/2016 será nomeado como DBname_20160304.log. Esses arquivos de log registram principalmente o status de operação do **dmschsvr**, agendamentos e tarefas, incluindo o horário de início, horário de término e informações anômalas. Consultando esses arquivos de log, um usuário pode monitorar se o dmschsvr está funcionando normalmente e depurar seus agendamentos e tarefas até que possam ser executados corretamente. Além disso, um usuário pode usar **DB_SchLgDir** e **DB_SchLgLev** para definir separadamente o diretório de armazenamento e o nível dos arquivos de log do dmschsvr. Para detalhes sobre **DB_SchLgDir** e **DB_SchLgLev**, consulte o Apêndice de Palavras-Chave no arquivo dmconfig.ini. Com o aumento dos arquivos de log, um usuário pode excluir automaticamente alguns arquivos de log antigos, ou também pode chamar SCHELOG_CLEAN() para excluir arquivos de log anteriores aos gerados mais recentemente por dias especificados. Para detalhes sobre SCHELOG_CLEAN(), consulte o Capítulo 5 - System-Stored Procedures in the SQL Command and Function Reference.

Os exemplos de **DMSCHSVR** abaixo requerem privilégios de DBA ou superiores:

1. Chamar procedimento armazenado para iniciar e parar o DMSCHSVR

Use o seguinte comando para iniciar o dmschsvr:

```
dmSQL> CALL START_DMSCHSVR('taskNum' ,'logPath');
```

taskNum: o número de tarefas que podem ser executadas ao mesmo tempo. O intervalo válido é de 1 a 50, e o valor padrão é 30.

logPath: o diretório onde o arquivo de log será armazenado. O diretório padrão é o diretório do banco de dados DB_DBDIR.

Use o seguinte comando para parar o dmschsvr.

```
dmSQL> CALL STOP_DMSCHSVR;
```

Exemplo:

Inicie e pare o dmschsvr.

```
dmSQL> CALL START_DMSCHSVR ("","); //use default values to start
dmschsvr
dmSQL> CALL STOP_DMSCHSVR; //stop dmschsvr
```

1. Iniciar e parar o DMSCHSVR configurando o dmconfig.ini

```
DB_SchSv = 1; start DMSCHSVR when the database starts
DB_TskNo= 30; at most 30 tasks can be executed at the same time
DB_SchLgDir = logPath;
DB_SchLgLev = 0;
```

DB_SchSv=<0, 1> — Especifica se o serviço dmschsvr está habilitado (0 desativado, 1 ativado).

DB_TskNo=<1~50> — Indica quantas tarefas podem ser ativadas ao mesmo tempo.

DB_SchLgDir=<nome_do_arquivo> — Especifica o diretório de armazenamento onde os arquivos de log do dmschsvr serão salvos.

DB_SchLgLev=<0~4> — Especifica o nível de log do dmschsvr, onde diferentes níveis armazenam diferentes informações:

- 0: Status de operação do dmschsvr.
- 1: Mensagem de erro das tarefas e status de operação do dmschsvr.
- 2: Avisos e mensagens de erro das tarefas, além do status de operação do dmschsvr.
- 3: Todas as informações das tarefas, agendamentos e status de operação do dmschsvr.
- 4: Informações de cálculo do dmschsvr, permitindo ao usuário monitorar se o dmschsvr está funcionando normalmente.

Use a linha de comando para iniciar o DMSCHSVR

```
dmschsvr –d database_name [–n max_running_tasks_number] [–p
schedule_log_path] [–h]
```

Exemplo: Nota: O dmschsvr só pode ser desligado pelo procedimento armazenado STOP_DMSCHSVR ou configurando DB_SCHSV=0.

```
dmschsvr –d dbsample5
```

13.2 Creating Tasks

A tarefa é uma ação que o agendamento executará; cada agendamento deve se referir a uma tarefa, e uma tarefa pode ser usada em vários agendamentos. Os capítulos a seguir irão apresentar procedimentos armazenados relacionados a tarefas:

TASK_CREATE, TASK_ALTER, TASK_DROP

Use o seguinte comando para criar uma tarefa.

```
dmSQL> CALL TASK_CREATE('task_name','task_type','action');
```

task_name: o nome da tarefa a ser criada. Pode conter de 1 a 128 letras, números e underscores, mas o primeiro caractere não pode ser um número.

task_type: o tipo de tarefa a ser criada. Existem três tipos de tarefas:

1. SQL_STATEMENT (SQL): a tarefa é uma instrução SQL.
2. STORE_PROCEDURE (SP): a tarefa é um procedimento armazenado.
3. EXECUTABLE (EXEC): a tarefa é um programa executável (.exe, .bat, etc.).

action: as ações que a tarefa executará regularmente. Deve corresponder ao tipo de tarefa a ser criada, e seu comprimento máximo é de 2K bytes.

Exemplo 1: Criar uma tarefa chamada insert_t1 para inserir valores na tabela t1:

```
dmSQL> CALL TASK_CREATE('insert_t1','SQL_STATEMENT','INSERT INTO t1
VALUES(1,2)');
```

Exemplo 2: Criar uma tarefa chamada call_sp1 para executar o procedimento armazenado SP1:

```
dmSQL> CALL TASK_CREATE('call_sp1',' STORE_PROCEDURE','SP1');
```

Exemplo 3: Criar uma tarefa chamada call_exe1 para executar o programa executável EXEC1:

```
dmSQL> CALL TASK_CREATE('call_exe1','EXECUTABLE','EXEC1');
```

13.3 Altering Tasks

TASK_ALTER é usado para alterar tarefas criadas. Os usuários podem modificar qualquer coisa, exceto o task_name. O **shedule** que se refere à tarefa alterada usará as novas configurações na próxima execução.

Use o seguinte comando para alterar uma tarefa.

```
dmSQL> CALL TASK_ALTER ('task_name','task_type','action');
```

Nota: Para obter informações sobre os parâmetros, consulte a seção 13.2 Criando Tarefas.

Exemplo:

Alterar a tarefa insert_t1, mudando os valores de entrada (1,2) para (5,6):

```
dmSQL> CALL TASK_ALTER('insert_t1','SQL_STATEMENT','INSERT INTO t1
VALUES(5,6)');
```

13.4 Dropping Tasks

TASK_DROP é usado para excluir tarefas que não estão mais em uso. Use o seguinte comando para excluir uma tarefa:

```
dmSQL> CALL TASK_DROP('task_name');
```

Exemplo:

Excluir a tarefa insert_t1:

```
dmSQL> CALL TASK_DROP('insert_t1');
```

NOTA: Tarefas com diferentes privilégios:

- Usuários com privilégio RESOURCE podem criar, alterar e excluir suas próprias tarefas.
- Usuários com privilégio DBA ou superior podem criar, alterar e excluir tarefas de qualquer usuário.

13.5 Creating Schedules

SCHEDULE_CREATE é usado para criar agendamentos.

Use o seguinte comando para criar um agendamento:

```
dmSQL> CALL SCHEDULE_CREATE('schedule_name','task_name','timetable',
'start_time','end_time');
```

schedule_name: o nome do agendamento a ser criado. Pode conter de 1 a 128 letras, números e underscores, mas o primeiro caractere não pode ser um número.

task_name: o nome da tarefa envolvida no schedule.

timetable: o cronograma de execução da tarefa. É composto por cinco campos em sequência: <minuto> <hora> <dia do mês> <mês> <dia da semana>, e os cinco campos devem ser separados por um espaço.

- <minute>: define em qual minuto da hora o schedule será executado. O intervalo válido é de 0 a 59.
- <hour>: define em qual hora do dia o schedule será executado. O intervalo válido é de 0 a 23.
- <day of month>: define em qual dia do mês o agendamento será executado. O intervalo válido é de 1 a 31.
- <month>: define em qual mês do ano o schedule será executado. O intervalo válido é de 1 a 12.
- <day of week>: define em qual dia da semana o schedule será executado. O intervalo válido é de 0 a 7, onde 0 e 7 significam domingo.

O DBMaker fornece algumas palavras-chave para o cronograma para representar configurações de tempo comumente usadas:

- @minute: executa a cada minuto (primeiro segundo de cada minuto), equivalente a * * * *.
- @hourly: executa a cada hora (primeiro minuto de cada hora), equivalente a (0 * * * *).
- @midnight: executa todos os dias (primeiro minuto do dia), equivalente a (0 0 * * *).

- @daily: executa todos os dias, equivalente a @midnight.
- @weekly: executa toda semana (primeiro minuto de segunda-feira), equivalente a (0 0 1 * *).
- @monthly: executa todo mês (primeiro minuto do 1º dia do mês), equivalente a (0 0 1 * *).
- @once: significa que o agendamento será executado apenas uma vez, e o horário de execução é baseado em start_time.
- @once m n: significa que o agendamento será executado apenas uma vez; se falhar, será executado novamente após n minutos, no máximo m vezes. Intervalos válidos para m e n: 1 < m < 525600; 1 < n < 1440.

start_time: a data e hora em que o agendamento começa; o formato é aaaa-mm-dd hh:mm:ss. Após a versão 5.4.4 do DBMaker, se o start_time estiver vazio, o valor padrão será now().

end_time: a data e hora em que o agendamento expira; o formato é aaaa-mm-dd hh:mm:ss. Como a unidade mínima de tempo do daemon de agendamento é em minutos, o end_time deve ser pelo menos um minuto após o start_time. Note que, normalmente, os usuários devem definir o end_time, mas se o valor do cronograma for definido como @once ou @once m n, é permitido que os usuários não definam o end_time. Nessa situação, o sistema considerará automaticamente o tempo um minuto após o start_time e (m*n+1) minutos após como os possíveis tempos de término. Após a versão 5.4.4 do DBMaker, se o end_time estiver vazio, o valor padrão será ilimitado. No entanto, a coluna end_time em sysschedule exibirá '2038/01/19 11:14:07'.

Os usuários podem usar os seguintes caracteres coringa:

- **Asterisco (*)**

É válido especificar um * para representar todos os valores possíveis para uma posição. Por exemplo, um * na 2ª posição é o mesmo que especificar todos os valores possíveis para a hora.

- **Vírgula (,)**

É válido especificar vários valores separados por vírgulas. Por exemplo, se um usuário deseja que um comando seja executado a cada 10 minutos, ele pode especificar 0,10,20,30,40,50 para os minutos.

- Traço (-) É válido especificar o intervalo de um valor com um -, por exemplo, um usuário pode especificar 0-12 para a hora, representando todas as horas da manhã.

- Barra (/) É válido especificar um intervalo regular com uma /, por exemplo, um usuário pode especificar */3 para o minuto, representando a cada 3 minutos.

Exemplo 1:

Crie uma schedules chamada insert_into_t1 para a tarefa insert_t1.

```
dmSQL> CALL SCHEDULE CREATE('insert_into_t1','insert t1','10 0,1 * * *','2012-12-12 12:00:00','2015-12-12 12:00:00'); // The task 'insert t1' will run at 0:10 and 1:10 every day from 2012-12-12 12:00 to 2015-12-12 12:00).
```

Exemplo 2: Crie uma schedules chamada call_sp1 para a tarefa SP1

```
dmSQL> CALL SCHEDULE_CREATE('call_sp1','SP1','30 * * * 1,7','2012-12-12 12:00:00','2015-12-12 12:00:00'); // The task 'SP1' will run every 30 minutes on Monday and Sunday from 2012-12-12 12:00 to 2015-12-12 12:00.
```

Exemplo 3:

Crie uma schedules chamada call_exe1 para a tarefa EXE1.

```
dmSQL> CALL SCHEDULE_CREATE('call_exe1','EXE1','@ONCE','NOW()',''); // The task 'EXE1' will run once at next minute from current time.
```

Uma nova schedules criada estará desabilitada e precisará ser habilitada para que seja executada.

Use o seguinte comando para habilitar a schedules.

```
dmSQL> CALL SCHEDULE_ENABLE('schedule_name');
```

Consulte "Habilitando e Desabilitando Programações" para mais informações.

13.6 Altering Schedules

SCHEDULE_ALTER é usado para alterar a schedules criada. Tudo, exceto o nome da schedules (schedule_name), pode ser alterado. Após a alteração, o dmschsvr usará a nova configuração de schedules na próxima execução.

Use o seguinte comando para alterar uma programação.

```
dmSQL> CALL SCHEDULE_ALTER('schedule_name','task_name','timetable',
'starttime','endtime');
```

Exemplo: Altere a schedules **insert_into_t1**. Neste exemplo, altere o plano de execução de "10 0,1 * * *" para "20 2,3 * * *". Para mais informações sobre a schedules **insert_into_t1**, consulte o exemplo no Capítulo 13.2, Creating Tasks.

```
dmSQL> CALL SCHEDULE_ALTER('insert_into_t1', 'insert_t1', '20 2,3 * * *', '2012-
12-12 12:00:00', '2015-12-12 12:00:00'); // The task 'insert_t1' will run at 2:20
and 3:20 every day from 2012-12-12 12:00 to 2015-12-12 12:00).
```

13.7 Dropping Schedule

SCHEDULE_DROP é usado para excluir a schedules que não está mais em uso.

Use o seguinte comando para excluir uma schedules.

```
dmSQL> CALL SCHEDULE_DROP('schedule_name');
```

Exemplo: Exclua a schedules **insert_into_t1**. Para mais informações sobre a schedules **insert_into_t1**, consulte o exemplo no Capítulo 13.2, Creating Schedules

```
dmSQL> CALL SCHEDULE_DROP('insert_into_t1');
```

13.8 Enabling and Disabling Schedules

SCHEDULE_ENABLE é usado para habilitar schedules.

Use o seguinte comando para habilitar uma schedules.

```
dmSQL> CALL SCHEDULE_ENABLE('schedule_name');
```

Uma nova schedules criada estará desabilitada e precisará ser habilitada para que seja executada. Somente quando o cronograma for definido como @ONCE, a schedules criada será habilitada.

Se o cronograma for definido como @ONCE, após a execução, a schedules será automaticamente desabilitada.

Exemplo: Habilite a schedules **'insert_into_t1'**.

```
dmSQL> CALL SCHEDULE_ENABLE(' insert_into_t1');
```

SCHEDULE_DISABLE é usado para desabilitar programações.

Use o seguinte comando para desabilitar uma schedules.

```
dmSQL> CALL SCHEDULE_DISABLE('schedule_name');
```

Exemplo: Desabilite a schedules **'insert_into_t1'**.

```
dmSQL> CALL SCHEDULE_DISABLE('schedule_name');
```

13.9 Reloading Schedules

SCHEDULE_RELOAD é usado para recarregar as schedules habilitadas. O dmschsvr recarregará automaticamente todas as schedules se houver uma programação criada ou alterada.

Use o seguinte comando para recarregar as schedules.

```
dmSQL> CALL SCHEDULE_RELOAD;
```

Exemplo:

```
dmSQL> CALL SCHEDULE_RELOAD;
```

Use o seguinte comando para recarregar todas as schedules habilitadas.

```
dmSQL> CALL SCHEDULE_RELOAD;
```

Recarregue todas as schedules habilitadas no sistema.

```
dmSQL> CALL SCHEDULE_RELOAD;
```

13.10 Cleaning Schedule Log

SCHELOG_CLEAN é usado para limpar o log de schedules antigas.

Use o seguinte comando para limpar o log de schedules.

```
dmSQL> CALL SCHELOG_CLEAN(reserve_day);
```

reserve_day: significa quantos dias do log de programação manter antes do mais recente. Suponha que o log de programação mais recente do usuário foi criado em 20/01. Se **reserve_day=10**, significa que o log de schedules antes de 10/01 será excluído. Se definido como 0, significa limpar todo o log de schedules. O intervalo válido é de 0 a 7300 dias. Exemplo:

Suponha que o log de schedules mais recente do usuário foi criado há 10 dias. O seguinte comando é usado para limpar o log de schedules 10 dias antes do mais recente. Isso significa que o log de schedules anterior a 20 dias será excluído.

```
dmSQL> CALL SCHELOG_CLEAN(10);
```

NOTA: Schedules *com diferentes privilégios*

Usuários com privilégio de **RESOURCE** podem criar, alterar, excluir, habilitar e desabilitar suas próprias programações.

Usuários com privilégio de **DBA** ou superior podem criar, alterar, excluir, habilitar e desabilitar as schedules de todos os usuários, iniciar, parar o dmschsvr e recarregar schedules. Além disso, os usuários só podem criar schedules referentes às suas próprias tarefas.