

达梦技术手册

DM8 作业系统使用手册

Service manual of DM8_Job_System

前言

概述

本文档主要介绍 DM 提供的作业系统，并通过创建作业、调度作业、监控作业等操作来展现作业是如何管理任务的。

读者对象

本文档主要适用于 DM 数据库的：

- 开发工程师
- 测试工程师
- 技术支持工程师
- 数据库管理员

通用约定

在本文档中可能出现下列标志，它们所代表的含义如下：

表 0.1 标志含义

标志	说明
 警告：	表示可能导致系统损坏、数据丢失或不可预知的结果。
 注意：	表示可能导致性能降低、服务不可用。
 小窍门：	可以帮助您解决某个问题或节省您的时间。
 说明：	表示正文的附加信息，是对正文的强调和补充。

在本文档中可能出现下列格式，它们所代表的含义如下：

表 0.2 格式含义

格式	说明
宋体	表示正文。
黑体	标题、警告、注意、小窍门、说明等内容均采用黑体。
Courier new	表示代码或者屏幕显示内容。
粗体	表示命令行中的关键字（命令中保持不变、必须照输的部分）或者正文中强调的内容。
<>	语法符号中，表示一个语法对象。
::=	语法符号中，表示定义符，用来定义一个语法对象。定义符左边为语法对象，右边为相应的语法描述。
	语法符号中，表示或者符，限定的语法选项在实际语句中只能出现一个。
{ }	语法符号中，大括号内的语法选项在实际的语句中可以出现 0..N 次 (N 为大于 0 的自然数)，但是大括号本身不能出现在语句中。
[]	语法符号中，中括号内的语法选项在实际的语句中可以出现 0..1 次，但是中括号本身不能出现在语句中。
关键字	关键字在 DM_SQL 语言中具有特殊意义，在 SQL 语法描述中，关键字以大写形式出现。但在实际书写 SQL 语句时，关键字既可以大写也可以小写。

访问相关文档

如果您安装了 DM 数据库，可在安装目录的“\doc”子目录中找到 DM 数据库的各种手册与技术丛书。

您也可以通过访问我们的网站 www.dameng.com 阅读或下载 DM 的各种相关文档。

联系我们

如果您有任何疑问或是想了解达梦数据库的最新动态消息，请联系我们：

网址：www.dameng.com

技术服务电话：400-991-6599

技术服务邮箱：dmttech@dameng.com

目录

1 功能简介	1
1.1 操作员	1
1.2 作业	1
1.3 警报	2
1.4 调度	2
1.5 作业权限	2
2 创建作业环境	3
2.1 系统表的定义	3
2.1.1 SYSJOBS.....	3
2.1.2 SYSJOBSTEPS	5
2.1.3 SYSJOBSCHEDULES	7
2.1.4 SYSJOBHISTORIES	10
2.1.5 SYSJOBHISTORIES2.....	12
2.1.6 SYSSTEPHISTORIES2.....	13
2.1.7 SYSOPERATORS	14
2.1.8 SYSALERTS	15
2.1.9 SYSALERTNOTIFICATIONS	17
2.1.10 SYSALERTHISTORIES.....	18
2.1.11 SYSMAILINFO	20
2.2 管理系统表	21
2.2.1 通过系统过程创建和删除	21
2.2.2 通过图形化客户端创建和删除	21
3 操作员	23
3.1 通过系统过程实现	23
3.1.1 创建操作员	23
3.1.2 修改操作员	24
3.1.3 删除操作员	25
3.2 通过图形化客户端实现	25
3.1.1 创建操作员	26
3.1.2 修改操作员	27
3.1.3 设置过滤、清除过滤	28
3.1.4 删除操作员	29
4 作业	30
4.1 通过系统过程实现	30
4.1.1 创建、修改和删除作业	30
4.1.2 配置作业	33
4.1.3 查看、清除作业日志记录	45
4.2 通过图形化客户端实现	46

4.2.1 新建作业	46
4.2.2 设置、清除过滤	56
4.2.3 查看、清除作业历史信息	58
4.2.4 修改作业	59
4.2.5 删除作业	60
5 警报	61
5.1 通过系统过程实现	61
5.1.1 创建、删除和修改警报	61
5.1.2 为警报关联操作员	64
5.1.3 清除警告日志记录	66
5.2 通过图形化客户端实现	66
5.2.1 常规	66
5.2.2 通知信息	69
5.2.3 DDL.....	70
6 监控作业	71
6.1 配置监控服务管理员	71
6.1.1 通过系统过程实现	71
6.1.2 通过图形化客户端实现	74
6.2 开启监控服务	76
6.2.1 通过图形化客户端启动	76
6.2.2 通过命令行工具启动	77
7 一个典型示例	79
7.1 配置作业管理	79
7.2 查看监控结果	80

1 功能简介

在管理员的工作中，有许多日常工作都是固定不变的。例如，定期备份数据库，定期生成数据统计报表等等。这些工作既单调又费时，如果这些重复任务能够自动化完成，那就可以节省大量的时间。

DM 的作业系统为用户提供了创建作业，并对作业进行调度执行以完成相应管理任务的功能。可以让这些重复的数据库任务自动完成，实现日常工作自动化。作业系统大致包含作业、警报和操作员三部分。用户需要为作业配置步骤和调度。还可以创建警报，当发生警报时，将警报信息通知操作员，以便操作员能够及时做出响应。

用户通过作业可以实现对数据库的操作，并将作业执行结果以通知的形式反馈到操作员。通过为作业创建灵活的调度方案可以满足在不同时刻运行作业的要求。用户还可以定义警报响应，以便当服务器发生特定的事件时通知操作员或者执行预定义的作业。

为了更好地理解作业与调度，下面介绍一些相关的概念：

1.1 操作员

操作员是负责维护 DM 服务器运行实例的个人。在有些企业中，操作员由单独一个人担任。在那些拥有很多服务器的大型企业中，操作员由多人共同担任。在预期的警报（或事件）发生时，可以通过电子邮件或网络发送的方式将警报（或事件）的内容通知到操作员。

1.2 作业

作业是由 DM 代理程序按顺序执行的一系列指定的操作。作业可以执行更广泛的活动，包括运行 DMPL/SQL 脚本、定期备份数据库、对数据库数据进行检查等。可以创建作业来执行经常重复和可调度的任务，作业按照一个或多个调度的安排在服务器上执行。作业也可以由一个或多个警报触发执行，并且作业可产生警报以通知用户作业的状态（成功或者失败）。每个作业由一个或多个作业步骤组成，作业步骤是作业对一个数据库或者一个服务器执行的动作。每个作业必须至少有一个作业步骤。

1.3 警报

警报是系统中发生的某种事件，如发生了特定的数据库操作，或出错信号，或者是作业的启动、执行完毕等事件。警报主要用于通知指定的操作员，以便其迅速了解系统中发生的状况。可以为警报定义产生的条件，还可以定义当警报产生时系统采取的动作，如通知一个或多个操作员执行某个特定的作业等。

1.4 调度

调度是用户定义的一个时间安排，在给定的时刻到来时，系统会启动相关的作业，按作业定义的步骤依次执行。调度可以是一次性的，也可以是周期性的。

1.5 作业权限

通常作业的管理是由 DBA 来维护，普通用户没有操作作业的权限，为了让普通用户可以创建、配置和调度作业，需要赋予普通用户管理作业权限：ADMIN JOB。

例如，授权 ADMIN JOB 给用户 NORMAL_USER。

```
GRANT ADMIN JOB TO NORMAL_USER;
```

默认 DBA 拥有全部的作业权限；ADMIN JOB 权限可以添加、配置、调度和删除作业等，但没有作业环境初始化 SP_INIT_JOB_SYS(1)和作业环境销毁 SP_INIT_JOB_SYS(0)的权限。

2 创建作业环境

要进行作业管理，需要先创建作业环境，即创建一些系统表来存储作业相关的对象、历史记录等信息。

这些系统表有 SYSJOBS、SYSJOBSTEPS、SYSJOBSCHEDULES、SYSMAILINFO、SYSJOBHISTORIES2、SYSSTEPHISTORIES2、SYSALERTHISTORIES、SYSOPERATORS、SYSALERTS、SYSALERTNOTIFICATIONS 共十张，均位于 SYSJOB 模式下。下面一一详细介绍。

2.1 系统表的定义

2.1.1 SYSJOBS

SYSJOBS 表存储用户定义的作业信息。每一个作业对应此表中的一条记录。每一条记录都有一个自增 ID，用来唯一表示这个作业，同时这个作业还具有一个聚集关键字 NAME，这意味着作业不可以同名。

语法如下：

```
CREATE TABLE SYSJOB.SYSJOBS (  
  
    ID                INT IDENTITY(1,1),  
  
    NAME              VARCHAR ,  
  
    ENABLE            INT ,  
  
    USERNAME          VARCHAR ,  
  
    CREATE TIME       VARCHAR ,  
  
    MODIFY TIME       VARCHAR ,  
  
    ENABLE_EMAIL      INT ,  
  
    EMAIL_OPERID      INT ,  
  
    EMAIL_TYPE        INT ,  
  
    ENABLE_NETSEND    INT ,  
  
    NETSEND_OPERID    INT ,  
  
    NETSEND_TYPE      INT ,
```

```

VALID                CHAR,

DESCRIBE            VARCHAR,

CLUSTER PRIMARY KEY(NAME)

);

```

参数详解

- ID
作业 ID 号。一个作业只有唯一一个 ID 号。
- NAME
作业名称。
- ENABLE
表示该作业是否被启用。1 是；0 否。
- USERNAME:
作业的创建者名称。也就是 SYSMAILINFO 表中的 LOGIN_NAME 登录名。
- CREATETIME
作业的创建时间，由系统时间指定。
- MODIFYTIME
表示作业最后一次被修改的时间，由系统时间指定。
- ENABLE_EMAIL、EMAIL_OPERID、EMAIL_TYPE
ENABLE_EMAIL: 表示作业是否开启邮件系统。1 是；0 否。如果开启，那么该作业相关的一些日志会通过邮件通知操作员；不开启就不会发送邮件。
EMAIL_OPERID: 指定操作员的 ID 号。如果开启了邮件通知功能，邮件会发送给该 ID 号的操作员。
EMAIL_TYPE: 表示如果在开启了邮件发送之后，在什么情况下发送邮件。情况分为三种：0、1、2。0 表示在作业执行成功后发送；1 表示在作业执行失败后发送；2 表示在作业执行完成后发送。
- ENABLE_NETSEND、NETSEND_OPERID、NETSEND_TYPE
ENABLE_NETSEND: 表示作业是否开启网络发送。1 是；0 否。如果开启，那么这个作业相关的一些日志会通过网络发送通知操作员；如果不开启就不会通知。
NETSEND_OPERID: 如果开启了网络信息通知功能，则这个列指定通过网络发送来通知

哪一个操作员，这里记录的是操作员 ID 号。

NETSEND_TYPE: 表示如果在开启了网络发送之后，在什么情况下发送网络信息。这个情况也有三种，和上面的 EMAIL_TYPE 是完全一样的。

- VALID

表示作业是否完整。Y 是；N 否。没有配置的作业是不完整的。如果不完整，那么这个作业需要重新配置，使它的状态为完整。

- DESCRIBE

作业的描述信息。

2.1.2 SYSJOBSTEPS

SYSJOBSTEPS 存储作业包括的所有步骤信息。每一行存储了某个作业的某个步骤的所有属性。这个表的聚集关键字为 JOBID 和步骤名，意味着在一个指定的作业下，不能有两个同名的步骤。

语法如下：

```
CREATE TABLE SYSJOB.SYSJOBSTEPS (
    ID                INT IDENTITY(1,1),
    NAME              VARCHAR,
    JOBID             INT,
    SEQNO             INT,
    TYPE              INT,
    DBNAME            VARCHAR,
    COMMAND           VARCHAR(1800),
    SUCC_ACTION       INT,
    FAIL_ACTION       INT,
    RETRY_ATTEMPTS    INT,
    RETRY_INTERVAL    INT,
    OUTPUT_FILE_PATH  VARCHAR(256),
    APPEND_FLAG       INT,
    CLUSTER PRIMARY KEY(JOBID, NAME)
```

);

参数详解

- ID

步骤 ID 号。一个步骤唯一对应一个 ID 号。步骤 ID 号用来表示步骤的唯一性。

- NAME

步骤名称。必须是有效的标识符，同时不能是 DM 关键字。同一个作业不能有同名的步骤名称。

- JOBID

作业 ID 号。表示步骤属于 JOBID 号指定的作业，创建时这个作业必须存在才能创建成功。

- SEQNO

步骤在作业中的序列号。

- TYPE

步骤的类型。取值 0、1、2、3、4、5 和 6。说明如下：

0 表示执行一段 SQL 语句或者是语句块。

1 表示执行基于 V1.0 版本的备份还原（没有 WITHOUT LOG 和 PARALLEL 选项）。

2 表示重组数据库。

3 表示更新数据库的统计信息。

4 表示执行 DTS（数据迁移）。

5 表示执行基于 V1.0 版本的备份还原（有 WITHOUT LOG 和 PARALLEL 选项）。

6 表示执行基于 V2.0 版本的备份还原。

- DBNAME

表示步骤所属的数据库名（实例名），这是系统在创建一个步骤时指定的，也就是当前运行的数据库实例名。

- COMMAND

该列的值与步骤类型 TYPE 有关。在不同步骤类型下，该列的值就是步骤在运行时所执行的不同语句。

当 TYPE=0 时，这个列的值就是用户指定的要执行的 SQL 语句或者语句块，如果要指定多条语句，在语句之间必须用分号隔开。不支持多条 DDL 语句一起执行，否则在执行时可能

会报出不可预知的错误信息。

当 TYPE 是 1、2、3、4、5 或 6 时，要执行的语句就是由系统内部根据不同类型生成的不同语句或者过程，生成时会用到上面的参数 DBNAME。

- SUCC_ACTION

指定步骤执行成功后，下一步该做什么事。取值 0、1 或 3。说明如下：

0 表示执行下一步。

1 表示报告执行成功。

3 表示返回第一个步骤继续执行。

- FAIL_ACTION

指定步骤执行失败后，下一步该做什么事。取值 0、2 或 3。说明如下：

0 表示执行下一步。

2 表示报告“执行失败”。

3 表示返回第一个步骤继续执行。

- RETRY_ATTEMPTS

表示当步骤执行失败后，需要重试的次数。取值范围 0~100。

- RETRY_INTERVAL

表示在每两次步骤执行重试之间的间隔时间。取值范围 0~10。

- OUTPUT_FILE_PATH

表示步骤执行时输出文件的路径。这个路径必须是有效的。

- APPEND_FLAG

输出文件的追写方式。如果指定输出文件，那么这个参数表示在写入文件时是否从文件末尾开始追写。1 是；0 否。如果是 0，那么从文件指针当前指向的位置开始追写。

2.1.3 SYSJOBSCHEDULES

一个作业可以有多个调度，调度用来指定一个作业的执行情况，可以指定作业的执行方式及时间范围。SYSJOBSCHEDULES 表存储作业的调度信息，聚集关键字为 JOBID 及调度名，意味着对于一个指定的作业，不能具有同名的调度。

语法如下：

```
CREATE TABLE SYSJOB.SYSJOBSCHEDULES (
```

```

ID                INT IDENTITY(1,1),
NAME              VARCHAR,
JOBID             INT,
ENABLE            INT,
TYPE              INT,
FREQ_INTERVAL     INT,
FREQ_SUB_INTERVAL INT,
FREQ_MINUTE_INTERVAL INT,
STARTTIME         VARCHAR,
ENDTIME           VARCHAR,
DURING_START_DATE VARCHAR,
DURING_END_DATE   VARCHAR,
SCHNAME           VARCHAR,
TRIGNAME          VARCHAR,
VALID             CHAR,
DESCRIBE          VARCHAR(500),
CLUSTER PRIMARY KEY(JOBID, NAME)
);

```

参数详解

- ID
调度 ID 号。一个调度唯一对应一个 ID 号。调度 ID 号用来表示调度的唯一性。
- NAME
调度的名称。必须是有效的标识符，同时不能是 DM 关键字。一个作业不能创建两个同名的调度。
- JOBID
作业 ID 号。表示该调度属于 JOBID 号指定的作业。一个作业可以有多个调度。
- ENABLE
表示该调度是否启用。1 是；0 否。
- TYPE

调度的类型。表示指定的作业按什么类型来执行。取值 0、1、2、3、4、5、6、7、8。

说明如下：

- 0 表示只执行一次。
- 1 表示按天的频率来执行。
- 2 表示按周的频率来执行。
- 3 表示在一个月中的某一天执行。
- 4 表示在一个月中的第一周第几天执行。
- 5 表示在一个月中的第二周的第几天执行。
- 6 表示在一个月中的第三周的第几天执行。
- 7 表示在一个月中的第四周的第几天执行。
- 8 表示在一个月中的最后一周的第几天执行。

● **FREQ_INTERVAL**

这个列的可选值与上面的不同调度类型有关。表示执行的频率。说明如下：

当 TYPE=0 时，这个值无效，系统不会做检查。

当 TYPE=1 时，这个列表示每隔几天执行，值的有效范围为 1 到 100。

当 TYPE=2 时，这个列表示的是每隔几个星期执行，值的范围没有具体的限制。

当 TYPE=3 时，表示每几个月中的某一天执行，值的范围没有具体的限制。

当 TYPE 为 4、5、6、7 或 8 时，都表示每几个月的某一周执行，值的范围也没有具体的限制。

● **FREQ_SUB_INTERVAL**

这个列的可选值与上面的不同调度类型有关。表示执行的频率，在 FREQ_INTERVAL 基础上，继续指定更为精细的频率。说明如下：

当 TYPE=0 或 1 时，这个值无效，系统不会做检查。

当 TYPE=2 时，表示的是某一个星期的星期几执行，可以同时选中七天中的任意几天。取值范围 1-127。具体如何取值，请用户参考如下规则。因为每周有七天，所以 DM 用七位二进制来表示选中的日子。从最低位开始算起，依次表示周日、周一...周五、周六。选中周几，就将该位置 1，否则 0。例如，选中周二和周六，7 位二进制就是 1000100，转化成十进制就是 68，所以 FREQ_SUB_INTERVAL 就取值 68。

当 TYPE=3 时，表示将在一个月中的第几天执行，此时这个列的有效值范围为 1 到 31。

当 TYPE 为 4、5、6、7 或 8 时，都表示将在某一周内第几天执行，有效值为 1 到 7，

分别表示从周一到周日。

- **FREQ_MINUTE_INTERVAL**
表示一天内每隔多少分钟执行一次。取值范围 1~1440。
- **STARTTIME**
定义作业调度的起始时间。
- **ENDTIME**
定义作业调度结束时间。
- **DURING_START_DATE**
指定作业被调度的有效日期范围的起始日期，必须是有效的日期字符串，不可以为空。
- **DURING_END_DATE**
指定作业被调度的有效日期范围的结束日期，可以为空。如果不为空，必须是有效的日期字符串，同时必须是在 **DURING_START_DATE** 日期之后。
- **SCHNAME**
表示调度对应的驱动触发器所属的数据库模式名。
- **TRIGNAME**
表示驱动触发器的触发器名。
- **VALID**
表示调度是否合法。Y 是；N 否。如果是 N，说明这个调度的配置是没有完成的，需要重新配置完成后才能起作用。
- **DESCRIBE**
表示调度的注释信息，最大值为 500 个字节。

2.1.4 SYSJOBHISTORIES

SYSJOBHISTORIES 存储作业步骤的执行情况的日志。每当一个作业执行开始时都会向这个表中插入一条作业执行开始的记录，其 **STATUS** 为“JOB START”；作业完成时也会插入一条作业执行完成的记录，其 **STATUS** 为“JOB END”；每一个步骤执行开始时都会插入一条步骤开始执行的记录，其 **STATUS** 为“JOB STEP START”；如果为重试执行开始则为“JOB STEP RERTY START”；每一个步骤执行完成或者重试完成都会插入一条相应状态的记录。

这个表中的所有记录都是由作业在运行过程中系统自动插入的，不是由用户来操作的。

语法如下：

```
CREATE TABLE SYSJOB.SYSJOBHISTORIES (  
  
    ID                INT IDENTITY(1,1),  
  
    NAME              VARCHAR ,  
  
    STEPNAME         VARCHAR ,  
  
    STATUS            VARCHAR ,  
  
    ERRTYPE          INT ,  
  
    ERRCODE           INT ,  
  
    ERRINFO          VARCHAR(1024),  
  
    CUR_TIME         VARCHAR ,  
  
    RETRY_ATTEMPTS   INT ,  
  
    HAS_NOTIFIED     INT ,  
  
    CLUSTER PRIMARY KEY(ID)  
  
);
```

参数详解

- ID
表中每一行的唯一标识。
- NAME
表示某一条历史记录是由哪一个作业产生的，用作业名表示。
- STEPNAME
表示历史记录是由哪一个步骤产生的，用步骤名表示。
- STATUS
表示某一条历史记录是在作业或步骤的什么状态下产生的。
- ERRTYPE
这个列一般情况不用，现在都是 0。
- ERRCODE
表示在步骤执行错误后，产生的错误码。
- ERRINFO

表示在步骤执行错误后，系统产生的错误描述信息。

- CUR_TIME

表示产生这条历史记录的系统时间。

- RETRY_ATTEMPTS

表示这条历史记录是第几次重试时产生的，这个列记录其当前次数。

- HAS_NOTIFIED

表示这条历史记录是否已经（邮件及网络发送）通知用户。如果已通知则这个值为 1，未通知则为 0。

2.1.5 SYSJOBHISTORIES2

SYSJOBHISTORIES2 存储作业的执行情况的日志。当一个作业执行完成后，会向这个表中插入一条作业执行情况的记录。

这个表中的所有记录都是由作业在运行过程中系统自动插入的，不是由用户来操作的。

语法如下：

```
CREATE TABLE SYSJOB.SYSJOBHISTORIES2 (  
  
    EXEC_ID                INT UNIQUE,  
  
    NAME                   VARCHAR ,  
  
    START_TIME             VARCHAR ,  
  
    END_TIME               VARCHAR ,  
  
    ERRCODE                INT ,  
  
    ERRINFO                VARCHAR(1024) ,  
  
    HAS_NOTIFIED           INT ,  
  
    CLUSTER PRIMARY KEY(EXEC_ID, NAME));
```

参数详解

- EXEC_ID

作业执行的 ID 号。

- NAME

表示某一条历史记录是由哪一个作业产生，用作业名表示。

- **START_TIME**
作业开始的时间。
- **END_TIME**
作业结束的时间。
- **ERRCODE**
表示作业执行错误后，产生的错误码。
- **ERRINFO**
表示作业执行错误后，系统产生的错误描述信息。
- **HAS_NOTIFIED**
表示这条历史记录是否已经（邮件及网络发送）通知用户。如果已通知则这个值为 1，未通知则为 0。

2.1.6 SYSSTEPHISTORIES2

SYSSTEPHISTORIES2 存储作业步骤的执行情况的日志。每当一个作业步骤执行完成时都会向这个表中插入一条作业步骤执行情况的记录。如果为重试步骤，RETRY_ATTEMPTS 会记录重试的次数。

语法如下：

```
CREATE TABLE SYSJOB.SYSSTEPHISTORIES2 (  
  
EXEC_ID                INT,  
  
NAME                   VARCHAR,  
  
STEPNAME               VARCHAR,  
  
START_TIME             VARCHAR,  
  
END_TIME               VARCHAR,  
  
ERRTYPE                INT,  
  
ERRCODE                INT,  
  
ERRINFO                VARCHAR(1024),  
  
RETRY_ATTEMPTS         INT,  
  
CLUSTER PRIMARY KEY(EXEC_ID, NAME, STEPNAME, RETRY_ATTEMPTS));
```

参数详解

- EXEC_ID
作业执行的 ID 号。
- NAME
表示某一条历史记录是由哪一个作业产生，用作业名表示。
- STEPNAME
表示历史记录是由哪一个步骤产生的，用步骤名表示。
- START_TIME
作业步骤开始的时间。
- END_TIME
作业步骤结束的时间。
- ERRTYPE
这个列一般不用，现在都是 0。
- ERRCODE
表示作业步骤执行错误后，产生的错误码。
- ERRINFO
表示作业执行错误后，系统产生的错误描述信息。
- RETRY_ATTEMPTS
表示这条历史记录是第几次重试时产生的，这个列记录其当前重试次数。

2.1.7 SYSOPERATORS

SYSOPERATORS 存储作业管理系统中所有已定义操作员的信息，以 NAME 为聚集索引，意味着不能具有同名的操作员。

语法如下：

```
CREATE TABLE SYSJOB.SYSOPERATORS (  
    ID                INT IDENTITY(1,1),  
    NAME              VARCHAR,  
    ENABLE            INT,  
    EMAILADDR        VARCHAR,  
    NETSEND_IP       VARCHAR,
```

```
CLUSTER PRIMARY KEY(NAME)
);
```

参数详解

- ID
操作员 ID 号。
- NAME
操作员的名称。
- ENABLE
表示操作员是否被启用。1 是；0 否。
- EMAILADDR
表示操作员的 EMAIL 地址。
- NETSEND_IP
表示操作员所在的 IP 地址。

2.1.8 SYSALERTS

SYSALERTS 存储作业管理系统中所有已定义的警报信息，聚集索引为 NAME，意味着不能定义同名的警报。

语法如下：

```
CREATE TABLE SYSJOB.SYSALERTS (
    ID                INT IDENTITY(1,1),
    NAME              VARCHAR,
    ENABLE             INT,
    TYPE              INT,
    DBNAME            VARCHAR,
    ERRTYPE           INT,
    ERRCODE           INT,
    DELAYTIME         INT,
    ADDITION_TXT      VARCHAR,
    CLUSTER PRIMARY KEY(NAME)
```

);

参数详解

- ID

每一个警报都有一个唯一标识的 ID 号。

- NAME

警报名。

- ENABLE

警报是否开启。1 是；0 否。

- TYPE

警报的类型。取值 0 或 1。0 表示发生错误时警报；1 表示发生某些数据库事件时警报。

- DBNAME

警报所在的数据库实例名。

- ERRTYPE

触发警报类型：错误和数据库事件。与参数 TYPE 相关。

当 TYPE=0 时，ERRTYPE 表示触发警报的错误。取值 1~12。具体含义分别为：1 常规错误；2 启动错误；3 系统错误；4 服务器配置错误；5 分析阶段错误；6 权限错误；7 运行时错误；8 备份恢复错误；9 作业管理错误；10 数据库复制错误；11 其它错误；12 指定错误码。

当 TYPE=1 时，ERRTYPE 表示触发警报的数据库事件。取值 1~4。具体含义分别为：1 DDL 事件；2 授权回收事件；3 连接或断开数据库事件；4 数据库备份恢复事件。

- ERRCODE

指定错误码。错误码取值和参数 TYPE、ERRTYPE 相关。

当 TYPE=0、ERRTYPE 取 1~12 时，指定各种错误类型相关的错误码，ERRCODE 必须是小于 0 的整数。

当 TYPE=1、ERRTYPE=1 时，指定 DDL 事件的错误码，ERRCODE 取值 1~15。如何取值，请参考如下规则：因为 DDL 事件包含 CREATE、ALTER、DROP 和 TRUNC 四种，所以此处的错误码是四种当中的任意几个组合值。ERRCODE 在 DM 数据库系统内部用四位二进制来表示它们的组合，从低位到高位依次是：CREATE、ALTER、DROP、TRUNCATE。1 表示指定，0 表示不指定。用户输入的 ERRCODE 值需要转化为十进制。例如，指定 CREATE 和

DROP，内部二进制表示为 0101，转化为十进制为 5，所以 ERRCODE 值就是 5。

当 TYPE=1、ERRTYPE=2 时，指定授权回收事件的错误码，ERRCODE 取值 1、2、3。
1 表示 GRANT 的错误码；2 表示 REVOKE 的错误码；3 表示 GRANT 和 REVOKE 的错误码。

当 TYPE=1、ERRTYPE=3 时，指定连接事件的错误码，取值 1、2、3。1 表示 LOGIN 的错误码；2 表示 LOGOUT 的错误码；3 表示 LOGIN 和 LOGOUT 的错误码。

当 TYPE=1、ERRTYPE=4 时，指定数据库备份恢复事件的错误码，取值 1、2、3。1 表示 BACKUP 的错误码；2 表示 RESTORE 的错误码；3 表示 BACKUP 和 RESTORE 的错误码。

- DELEETIME

表示警报发生后，推迟多长时间通知操作员。范围 0~3600，单位秒。

- ADDITION_TXT

表示警报的注释，最长为 500 个字节。

2.1.9 SYSALERTNOTIFICATIONS

SYSALERTNOTIFICATIONS 存储警报需要通知的操作员的信息，即警报和操作员的关联信息。以 ALERTNAME 和 OPERID 为聚集关键字的，所以对于一个指定的警报和指定的操作员，它们只能有一个关联关系。

语法如下：

```
CREATE TABLE SYSJOB.SYSALERTNOTIFICATIONS (
    ALERTNAME                VARCHAR ,
    OPERID                   INT ,
    ENABLE_EMAIL              INT ,
    ENABLE_NETSEND            INT ,
    CLUSTER PRIMARY KEY(ALERTNAME , OPERID)
);
```

参数详解

- ALERTNAME

表示指定的警报名。

- OPERID

表示指定的操作员的 ID 号。

- ENABLE_EMAIL

表示是否需要通过发邮件来通知。

- ENALE_NETSEND

表示是否通过网络发送的方式来通知。只在 WINDOWS 早期版本（例如 WIN XP/2000）下才支持。

2.1.10 SYSALERTHISTORIES

SYSALERTHISTORIES 存储警报发生的历史记录的日志。每个警报发生时都会向这个表中插入相应的记录，然后 DMJMON 服务再通过扫描这个表把信息取出来通过邮件或者网络发送的方式通知关联的操作员，这个表中的所有信息都是在发生警报时，由系统自动向这个表中插入的。

语法如下：

```
CREATE TABLE SYSJOB.SYSALERTHISTORIES (  
    ID                INT IDENTITY(1,1),  
    ALERTNAME         VARCHAR ,  
    EVENT_TYPE       INT ,  
    SUB_TYPE         INT ,  
    USERNAME          VARCHAR(128) ,  
    DB_NAME           VARCHAR(128) ,  
    OPTIME            VARCHAR(128) ,  
    OPUSER            VARCHAR(128) ,  
    SCH_NAME          VARCHAR(128) ,  
    OBJ_NAME          VARCHAR(256) ,  
    OBJ_TYPE          VARCHAR(128) ,  
    GRANTEE_NAME      VARCHAR(256) ,  
    ERRCODE           INT ,  
    HAS_NOTIFIED      INT DEFAULT 0 ,  
    CLUSTER PRIMARY KEY(ID)  
);
```

参数详解

- ID
每一条警报历史记录都有一个唯一标识为 ID 号。
- ALERTNAME
表示历史记录是由哪一个警报产生的。
- EVENT_TYPE
表示历史记录的事件类型，与 SYSALERTS 表中的警报类型 TYPE 相关。如果 TYPE=0，则这个列的值也为 0；如果 TYPE=1，则这个值与 SYSALERTS 表中的列 ERRTYPE 值相对应。
- SUB_TYPE
表示历史记录的事件子类型，与 SYSALERTS 表中的 TYPE 列相关。如果 TYPE=0，则这个列的值也为 0；如果 TYPE=1，则这个值与 SYSALERTS 表中的列 ERRCODE 相对应。
- USERNAME
表示发生警报时，当前的登录用户名。
- DB_NAME
表示警报所处的数据库实例名。
- OPTIME
表示警报发生时间。
- OPUSER
表示警报发生时当前的登录用户名。
- SCH_NAME
表示触发警报的数据对象所属的模式名。
- OBJ_NAME
表示触发警报的数据对象名。
- OBJ_TYPE
表示触发警报的数据库对象类型。
- GRANTEE_NAME
表示由授权或回收权限引起的警报的授权者或回收者名。
- ERRCODE
表示当警报定义的 TYPE 为错误时，发生错误时的错误码。
- HAS_NOTIFIED

表示这条警报历史记录是否已经（邮件及网络发送）通知用户。如果已通知则这个值为 1，未通知则为 0。

2.1.11 SYSMAILINFO

SYSMAILINFO 存储作业管理系统管理员的信息。每一个作业管理系统管理员对应这个表中的一条记录，登录用户名是唯一的，因为这个表是以第一列 LOGIN_NAME 为聚集关键字的。此处的 LOGIN_NAME 必须是 DM 数据库的登录用户名称。

语法如下：

```
CREATE TABLE SYSJOB.SYSMAILINFO (  
  
    LOGIN_NAME                VARCHAR(128) NOT NULL ,  
  
    LOGIN_PWD                 VARCHAR(128) NOT NULL ,  
  
    SMTP_SERVER               VARCHAR(256) NOT NULL ,  
  
    EMAIL                     VARCHAR(256) NOT NULL ,  
  
    USER_NAME                 VARCHAR(128) NOT NULL ,  
  
    USER_PWD                  VARCHAR(128) NOT NULL ,  
  
    CLUSTER PRIMARY KEY(LOGIN_NAME)  
  
);
```

参数详解

- LOGIN_NAME
监控服务管理员的名称。
- LOGIN_PWD
管理员登录密码。
- EMAIL
这个列表示在开启邮件系统后，给管理员发邮件时用到邮件的地址。
- SMTP_SERVER
邮件地址对应的 SMTP 服务器地址。这个地址必须要与 EMAIL 地址相对应，如果没有对应，则邮件不能成功发送。
- USER_NAME
这个列表示的是在发送邮件时，对 SMTP 服务器进行验证的邮件用户名，如果验证成功

才能成功发送邮件。

- USER_PWD

这个列与上面的用户名相对应，为邮箱的登录密码。

2.2 管理系统表

创建和删除作业相关系统表可以通过以下两种方式来实现。一是通过系统过程 SP_INIT_JOB_SYS 来实现；二是通过图形化客户端 MANAGER 管理工具实现。用户选择其中的一种即可。

2.2.1 通过系统过程创建和删除

用户可以通过调用系统过程 SP_INIT_JOB_SYS()来创建这些表。这些表被建在 SYSJOB 模式下。

创建 SYSJOB 模式及 11 张系统表的语句。

语法如下：

```
SP_INIT_JOB_SYS(1);
```

删除 SYSJOB 模式及 11 张系统表的语句。

语法如下：

```
SP_INIT_JOB_SYS(0);
```

2.2.2 通过图形化客户端创建和删除

在 DM MANAGER 管理工具中，右击代理。如下所示：



图 2.1 创建代理界面

点击**创建代理环境**，即可创建代理环境成功。创建成功后，会增加 SYSJOB 模式（含 9

张系统表); 同时, **代理** 下拉菜单中出现: 作业、警报和操作员。如下所示:

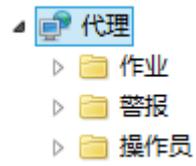


图 2.2 创建成功的代理界面

点击**清理代理环境**, 是**创建代理环境**的相反步骤。会删除和作业相关的一切信息。

3 操作员

创建、修改和删除操作员可以通过以下两种方式来实现。一是通过系统过程来实现；二是通过图形化客户端 `MANAGER` 管理工具实现。用户选择其中的一种即可。

3.1 通过系统过程实现

分别用 `SP_CREATE_OPERATOR`、`SP_ALTER_OPERATOR` 和 `SP_DROP_OPERATOR` 三个过程来完成操作员的创建、修改和删除。

3.1.1 创建操作员

在数据库中创建了前述作业管理表后，就可以进行作业的配置了。由于在创建一个作业时必须要指定操作员，所以需要创建一个操作员。

创建操作员可以直接通过调用系统过程来实现，过程名为 `SP_CREATE_OPERATOR`，所创建的操作员信息都会存储于表 `SYSOPERATORS` 中。

语法如下：

```
SP_CREATE_OPERATOR (  
  
    OPR_NAME          VARCHAR (128) ,  
  
    ENABLED           INT ,  
  
    EMAILADDR        VARCHAR (128) ,  
  
    NETSEND_IP       VARCHAR (128)  
  
)
```

参数详解

- `OPR_NAME`

操作员名称。必须是有效的标识符，同时不能是 `DM` 关键字。不能有同名操作员，如果创建同名操作员，系统会报错。

- `ENABLED`

是否启用这个操作员。1 是；0 否。

- `EMAILADDR`

操作员的 EMAIL 地址。

- NETSEND_IP

操作员的 IP 地址（用于网络发送）。在创建时必须指定合法的 IP 地址，否则报错。

例如，创建一个名为 TOM 的操作员。

```
SP_CREATE_OPERATOR('TOM', 1, 'tom@dameng.shanghai', '192.168.0.38');
```

3.1.2 修改操作员

如果 DBA 希望修改某一个操作员的某些信息，DM 也提供了相应的系统过程，函数名为 SP_ALTER_OPERATOR。

语法如下：

```
SP_ALTER_OPERATOR (  
  
    OPR_NAME          VARCHAR (128) ,  
  
    ENABLED            INT ,  
  
    EMAILADDR         VARCHAR (128) ,  
  
    NETSEND_IP        VARCHAR (128)  
  
)
```

参数详解

- OPR_NAME

不可修改参数。操作员名称，指定要修改哪一个操作员，如果这个操作员不存在，系统会报错。

- ENABLED

可修改参数。表示是否启用这个操作员。1 是；0 否。例如，将这个值修改为 0，表明将这个操作员修改为不可用。

- EMAILADDR

可修改参数。表示修改后操作员的 EMAIL 地址。

- NETSEND_IP

可修改参数。修改后操作员 IP 地址（主要用于网络发送），在修改时指定的 IP 地址必须是合法的，否则会报错。



说明:

对于可修改参数:

如果需要修改则传入新的参数值;

如果不需要修改则传入原来的值即可。

例如, 修改操作员 TOM 的信息, 修改了 TOM 的 IP 地址, 并将 TOM 置为不可用。

```
SP_ALTER_OPERATOR('TOM', 0, 'tom@dameng.shanghai', '192.168.0.38');
```

3.1.3 删除操作员

可以通过调用系统过程来删除一个操作员, 过程名为 SP_DROP_OPERATOR。

语法如下:

```
SP_DROP_OPERATOR (  
    OPR_NAME          VARCHAR(128)  
)
```

参数详解

- OPR_NAME

操作员名称。如果指定的操作员不存在, 则系统会报操作员不存在的错误; 如果指定的操作员存在, 则会从表 SYSOPERATORS 中将这条记录删除, 同时将所有这个操作员与警报的关联关系从表 SYSALERTNOTIFICATIONS 中删除, 但是不会删除指定该操作员的作业。

例如, 删除名为 TOM 的操作员。

```
SP_DROP_OPERATOR('TOM');
```

3.2 通过图形化客户端实现

可以通过图像化客户端 MANAGER 工具实现操作员的创建、修改和删除。

点击代理中的**操作员**。可以看到新建操作员、设置过滤、清除过滤和刷新按钮。如下所示:

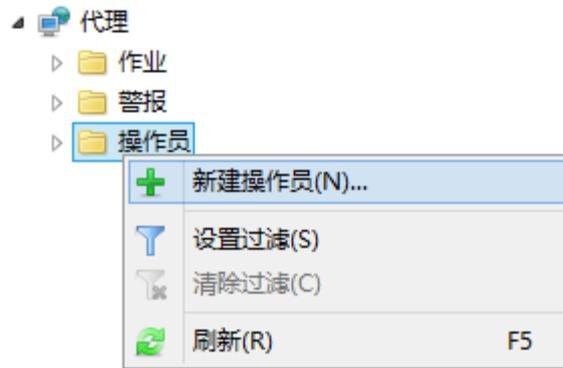


图 3.1 新建操作员界面

3.1.1 创建操作员

选中**新建操作员**。会出现如下界面：

常规页面，用来添加操作员。



图 3.2 新建操作员-常规界面

DDL 页面，用来显示创建操作员的 SQL 语句。还可以保存 SQL 脚本到指定文件中。如下所示：



图 3.3 新建操作员-DDL 界面

3.1.2 修改操作员

选中要修改的操作员。如下所示：

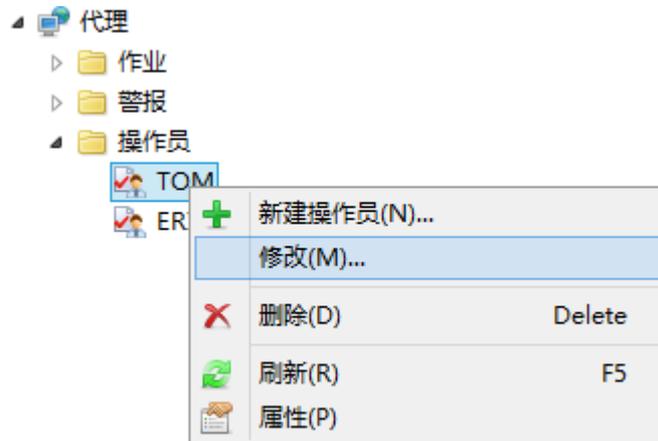


图 3.4 修改操作员界面

点击**修改**，即可出现如下界面。例如，将 TOM 的邮箱修改为 tom@dameng.shanghai。
如下所示：

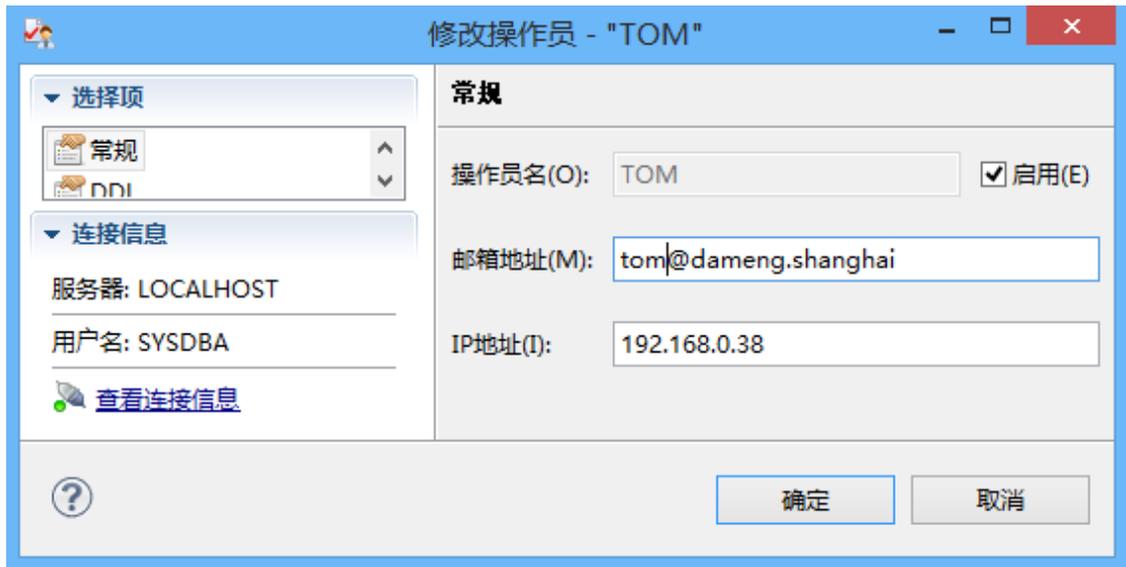


图 3.5 修改操作员-常规界面

3.1.3 设置过滤、清除过滤

当操作员过多时，可以通过设置过滤条件来选择符合条件的操作员。如下所示：

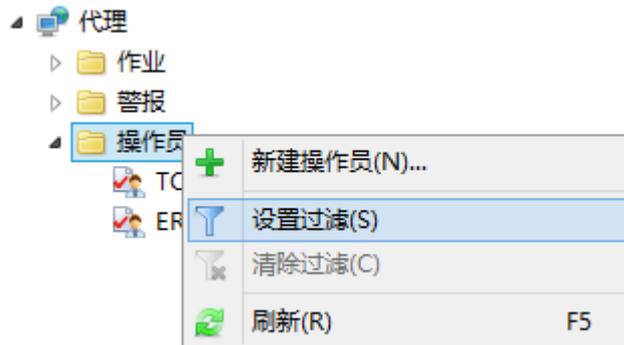


图 3.6 操作员-设置过滤界面

点击**操作员**，选中**设置过滤**，会出现如下界面。其中，过滤条件有包含、不包含和等于三种。例如，设置过滤条件为名称中包含 TOM 的操作员。如下所示：

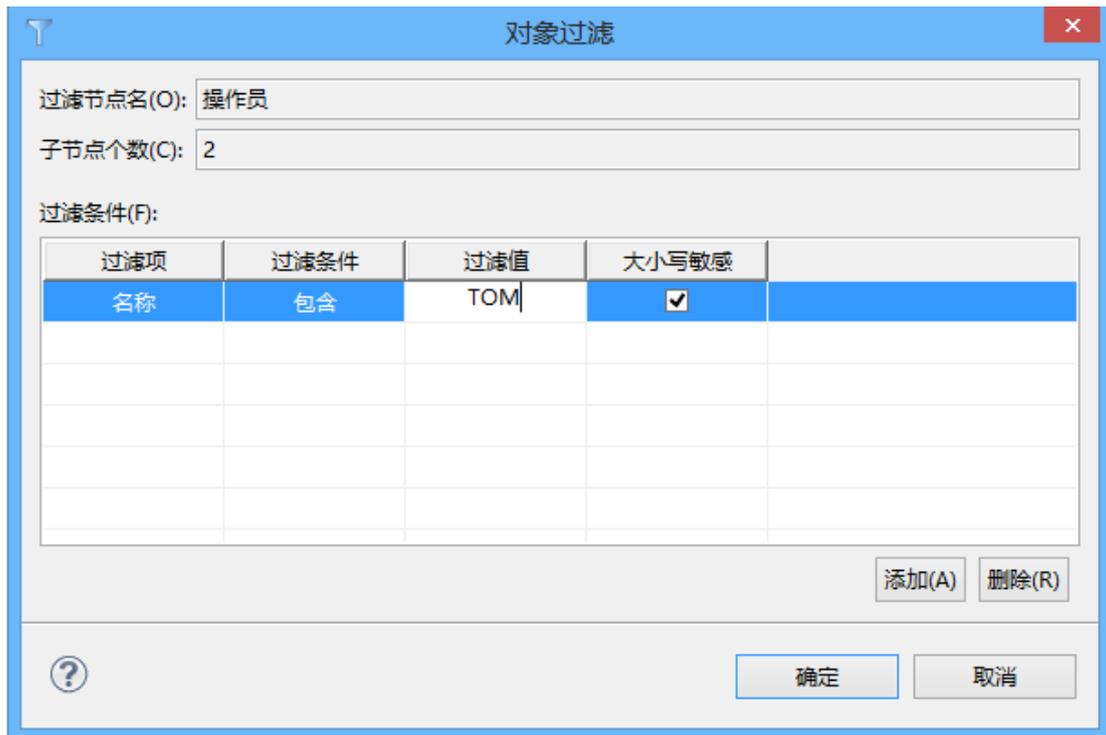


图 3.7 操作员-对象过滤界面

过滤条件使用完后，想恢复原状，看到所有操作员，此时需要清除过滤条件，选中**清除过滤**即可。

3.1.4 删除操作员

选中要删除的操作员名称，点击**删除**即可。如下所示：

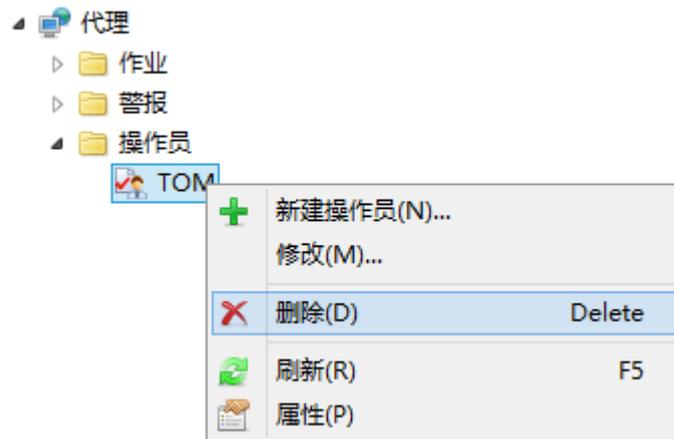


图 3.8 删除操作员界面

4 作业

创建、修改和删除作业可以通过以下两种方式来实现。一是通过系统过程来实现；二是通过图形化客户端 MANAGER 管理工具实现。用户选择其中的一种即可。

4.1 通过系统过程实现

4.1.1 创建、修改和删除作业

4.1.1.1 创建作业

在创建操作员之后,就可以创建作业了。创建作业通过系统过程 SP_CREATE_JOB 实现。

语法如下:

```
SP_CREATE_JOB (  
  
    JOB_NAME          VARCHAR(128) ,  
  
    ENABLED           INT ,  
  
    ENABLE_EMAIL      INT ,  
  
    EMAIL_OPTR_NAME   VARCHAR(128) ,  
  
    EMAIL_TYPE        INT ,  
  
    ENABLED_NETSEND   INT ,  
  
    NETSEND_OPTR_NAME VARCHAR(128) ,  
  
    NETSEND_TYPE      INT ,  
  
    DESCRIBE          VARCHAR(8187)  
  
)
```

参数详解

- JOB_NAME
作业名称。必须是有效的标识符,同时不能是 DM 关键字。作业不能重名,重名则报错。
- ENABLE
作业是否启用。1 启用; 0 不启用。
- ENABLE_EMAIL、EMAIL_OPTR_NAME、EMAIL_TYPE

ENABLE_EMAIL: 作业是否开启邮件系统。1 是；0 否。如果开启，那么该作业相关的一些日志会通过邮件通知操作员；不开启就不会发送邮件。

EMAIL_OPERID: 指定操作员名称。如果开启了邮件通知功能，邮件会发送给该操作员。在创建时系统会检测这个操作员是否存在，如果不存在则报错。

EMAIL_TYPE: 如果在开启了邮件发送之后，在什么情况下发送邮件。情况分为三种：0、1、2。0 表示在作业执行成功后发送；1 表示在作业执行失败后发送；2 表示在作业执行结束后发送。

- **ENABLE_NETSEND、NETSEND_OPERID、NETSEND_TYPE**

ENABLE_NETSEND: 作业是否开启网络发送。1 是；0 否。如果开启，那么这个作业相关的一些日志会通过网络发送通知操作员；如果不开启就不会通知。

NETSEND_OPERID: 指定操作员名称。如果开启了网络信息通知功能，则会通过网络发送来通知该操作员。在创建时系统会检测这个操作员是否存在，如果不存在则报错。

NETSEND_TYPE: 如果在开启了网络发送之后，在什么情况下发送网络信息。这个情况也有三种，和上面的 **EMAIL_TYPE** 是完全一样的。



网络发送功能只有 **WINDOWS** 早期版本上才支持（比如 **WIN 2000/XP**），且一

说明：定要开启 **MESSAGER** 服务。**WINDOWS7、8** 系统因为取消了 **MESSAGER** 服务，所以该功能也不支持。

- **DESCRIBE**

作业描述信息，最长 500 个字节。

例如，创建一个名为 **TEST** 的作业。

```
SP_CREATE_JOB('TEST', 1, 1, 'TOM', 2, 1, 'TOM', 2, '每一个测试作业');
```

创建完成这个作业后，系统就会在 **SYSJOBS** 中插入一条相应的记录，但是这个作业不会做任何事情，只是一个空的作业，如果需要让它执行，还需要配置这个作业。

4.1.1.2 修改作业

如果 **DBA** 发现某一个作业中的信息不合理需要修改，可以调用系统过程 **SP_ALTER_JOB** 来实现。

语法如下：

```

SP_ALTER_JOB (

    JOB_NAME          VARCHAR(128),

    ENABLED           INT,

    ENABLE_EMAIL      INT,

    EMAIL_OPTR_NAME   VARCHAR(128),

    EMAIL_TYPE        INT,

    ENABLED_NETSEND   INT,

    NETSEND_OPTR_NAME VARCHAR(128),

    NETSEND_TYPE      INT,

    DESCRIBE          VARCHAR(8187)

)

```

参数详解

函数 SP_ALTER_JOB 的参数和 SP_CREATE_JOB 的参数完全相同，除了 JOB_NAME 不可修改外，其他的属性都可修改。对于可修改参数，如果要修改，则指定新值；如果不修改，则继续指定原值。

例如，下面的语句修改了作业 TEST 的一些信息。

```
SP_ALTER_JOB('TEST', 0, 1, 'DBA', 2, 1, 'DBA', 2, '修改一个作业');
```

4.1.1.3 删除作业

如果一个作业已经执行完成，或者由于其它什么原因需要删除作业，可以调用系统过程 SP_DROP_JOB 实现。

语法如下：

```

SP_DROP_JOB (

    JOB_NAME          VARCHAR(128)

)

```

参数详解

- JOB_NAME

作业名称。在删除时会检测这个作业是否存在，如果不存在则系统报错。

在删除一个作业时，系统会同时将与此作业相关联的所有对象都删除。包括步骤、调

度等，也就是会分别从作业表 SYSJOBSTEPS 以及 SYSSCHEDULES 中删除属于这个作业的步骤及调度。

例如，删除作业 TEST。

```
SP_DROP_JOB('TEST');
```

4.1.2 配置作业

上面所述的内容都是最基本的一些操作，所创建的作业都还不能执行任何操作，只是一个空的作业，如果想要这个作业能执行一些指定的操作，还需要对这个作业进行配置。

配置一个作业主要包括以下几个步骤：

1. 开始作业配置；
2. 指定要开始配置一个作业；
3. 为指定的作业增加步骤；
4. 为指定的作业增加调度；
5. 结束作业配置。

只有在结束作业配置后，这个作业才算配置完成，同时如果这个作业是 ENABLE 状态的，那么它会立即生效。也就是从“结束作业配置”时刻开始就会根据它所定义的调度来执行操作了。

4.1.2.1 开始作业配置

用系统过程 SP_JOB_CONFIG_START 来指定对一个作业配置的开始。

语法如下：

```
SP_JOB_CONFIG_START (  
    JOB_NAME          VARCHAR(128)  
)
```

参数详解

- JOB_NAME

要配置的作业的名称。执行时会检测这个作业是否存在，如果不存在则报错。



注意：不支持在设置了 DML 自动提交（例如，DISQL 中设置 SET AUTO ON）的会话上配置作业。

开始作业配置之后到结束作业配置之前这段时间，当前会话会处于作业配置状态。配置状态不允许做任何的创建、修改、删除对象（作业、操作员、警报）的操作。开始作业配置和结束作业配置两个过程配合使用，是为了保证作业配置的完整性。

同时强烈建议：因为作业配置全部都是 DDL 操作，所以在配置过程中建议用户不要做任何的 COMMIT 操作或者设置 DDL 自动提交（例如，不要设置 dm.ini 文件中 DDL_AUTO_COMMIT=1）。否则在配置作业过程中，一旦错误的作业配置 DDL 操作被自动提交，将不能回滚。

在 DM 的作业配置过程中，如果配置出现错误时，可以直接使用 ROLLBACK 将错误的配置回滚到 SP_JOB_CONFIG_START 刚执行的状态，因为在这个过程执行时会自动提交前面所做的操作。所以在配置一个作业开始前，也需要谨慎，需要考虑前面做的操作是否需要提交。

例如，开始对作业 TEST 进行配置。

```
SP_JOB_CONFIG_START('TEST');
```

4.1.2.2 作业步骤

增加、删除作业步骤必须是在配置作业开始后才能进行，否则系统会报错，这样处理主要是为了保证作业配置的完整性。

4.1.2.2.1 增加步骤

增加作业的步骤通过系统过程 SP_ADD_JOB_STEP 实现。

语法如下：

```
SP_ADD_JOB_STEP (  
  
    JOB_NAME          VARCHAR(128),  
  
    STEP_NAME         VARCHAR(128),  
  
    TYPE              INT,  
  
    COMMAND           VARCHAR(8187),  
  
);
```

```
SUCC_ACTION      INT,
FAIL_ACTION      INT,
RETRY_ATTEMPTS   INT,
RETRY_INTERVAL   INT,
OUTPUT_FILE_PATH VARCHAR(256),
APPEND_FLAG       INT
)
```

参数详解

- JOB_NAME

作业的名称。表示正在给哪一个作业增加步骤，这个参数必须为上面调用 SP_JOB_CONFIG_START 函数时指定的作业名，否则系统会报错，同时系统会检测这个作业是否存在，不存在也会报错。

- STEP_NAME

表示增加的步骤名。必须是有效的标识符，同时不能是 DM 关键字。同一个作业不能有两个同名的步骤，创建时会检测这个步骤是否已经存在，如果存在则报错。

- TYPE

步骤的类型。取值 0、1、2、3、4、5 和 6。说明如下：

0 表示执行一段 SQL 语句或者是语句块。

1 表示执行基于 V1.0 版本的备份还原（没有 WITHOUT LOG 和 PARALLEL 选项）。

2 表示重组数据库。

3 表示更新数据库的统计信息。

4 表示执行 DTS（数据迁移）。

5 表示执行基于 V1.0 版本的备份还原（有 WITHOUT LOG 和 PARALLEL 选项）。

6 表示执行基于 V2.0 版本的备份还原。

- COMMAND

指定不同步骤类型（TYPE）下，步骤在运行时所执行的语句。它不能为空。

当 TYPE=0 时，指定要执行的 SQL 语句或者语句块。如果要指定多条语句，在语句之间必须用分号隔开。不支持多条 DDL 语句一起执行，否则在执行时可能会报出不可预知的错误信息。

当 TYPE=1 时，指定的是一个字符串。该字符串由三个部分组成：[备份模式][备份压缩类型][base_dir,...,base_dir|bakfile_path]。三部分详细介绍如下：

- ✚ 第一部分是一个字符，表示备份模式。0 完全备份；1 增量备份。如果第一个字符不是这二个值中的一个，系统会报错。

- ✚ 第二部分是一个字符，表示备份时是否进行压缩。0 不压缩；1 压缩。

- ✚ 第三部分是一个文件路径，表示备份文件的路径。路径命令有具体的格式，分以下两种：

- ✓ 对于增量备份，因为它必须要指定一个或者多个基备份路径，每个路径之间需要用逗号隔开，之后接着是备份路径，基备份路径与备份路径需要用“|”隔开，如果不指定备份路径，则不需要指定“|”，同时系统会自动生成一个备份路径。例如，
01E:\base_bakdir1, base_bakdir2|bakdir。

- ✓ 对于完全备份，因为不需要指定基备份所以就不需要“|”符号了，可以直接在第三个字节开始指定备份路径即可。例如，01E:\bakdir。如果不指定备份路径，则系统会自动生成一个备份路径。

当 TYPE 是 2、3 或 4 时，要执行的语句就是由系统内部根据不同类型生成的不同语句或者过程。

当 TYPE=5 时，指定的是一个字符串。该字符串由六个部分组成：[备份模式][备份压缩类型][备份日志类型][备份并行类型][预留][base_dir,...,base_dir | bakfile_path | parallel_file]。六部分详细介绍如下：

- ✚ 第一部分是一个字符，表示备份模式。0 完全备份；1 增量备份。如果第一个字符不是这二个值中的一个，系统会报错。

- ✚ 第二部分是一个字符，表示备份时是否进行压缩。0 不压缩；1 压缩。

- ✚ 第三部分是一个字符，表示是否备份日志。0 备份；1 不备份。

- ✚ 第四部分是一个字符，表示是否并行备份。0 普通备份；1 并行备份，并行备份映射放到最后，以“|”分割。

- ✚ 第五部分是一个保留字符，用 0 填充。

- ✚ 第六部分是一个文件路径，表示备份文件的路径。路径命令有具体的格式，分以下两种：

- ✓ 对于增量备份，因为它必须要指定一个或者多个基础备份路径，每个路径之间需要用逗号隔开，之后接着是备份路径，最后并行备份映射文件；并行映射文件，基础

备份路径与备份路径需要用“|”隔开，如果不指定备份路径与并行映射文件，则不需要指定“|”，例如 01000E:\base_bakdir1, base_bakdir2|bakdir|parallel_file_path 就是一个合法的增量备份命令。

✓ 对于完全备份，因为不需要指定基备份所以就不需要“|”符号了，可以直接在第三个字节开始指定备份路径即可；例如，01000E:\bakdir。如果不指定备份路径，系统会自动生成备份路径。

当 TYPE=6 时，指定的是一个字符串。该字符串由九个部分组成：[备份模式][备份压缩类型][备份日志类型][备份并行数][USE PWR][MAXPIECESIZE][RESV1][RESV2][base_dir,...,base_dir | bakfile_dir]。六部分详细介绍如下：

✚ 第一部分是一个字符，表示备份模式。0 完全备份；1 增量备份；3 归档备份。如果第一个字符不是这三个值中的一个，系统会报错。

✚ 第二部分是一个字符，表示备份时是否进行压缩。0 不压缩；1 压缩。

✚ 第三部分是一个字符，表示是否备份日志。0 备份；1 不备份。

✚ 第四部分是一个字符，表示并行备份并行数。取值 0 到 9。其中，0 表示不进行并行备份；1 表示使用并行数默认值 4；2~9 表示并行数。

✚ 第五部分为一个字符，表示并行备份时，是否使用 USE PWR 优化增量备份。0 不使用；1 使用。（只是语法支持，没有实际作用）

✚ 第六部分为一个字符，表示备份片大小的上限 (MAXPIECESIZE)。0 表示默认值，1~9 表示依次表示备份片的大小，1 则为 128M 备份片大小，9 表示 32G 的备份片大小。

✚ 第七部分为一个字符，表示是否在备份完归档后，删除备份的归档文件。0 不删除；1 删除。

✚ 第八部分是一个保留字符，用 0 填充。

✚ 第九部分是一个文件路径，表示备份文件的路径。路径命令有具体的格式，分以下两种：

✓ 对于增量备份，因为它必须要指定一个或者多个基础备份路径，每个路径之间需要用逗号隔开，之后接着是备份路径。基础备份路径与备份路径需要用“|”隔开，例如，01000000E:\base_bakdir1, base_bakdir2|bakdir 就是一个合法的增量备份命令。

✓ 对于完全备份，就不需要“|”符号了，可以直接在第八个字节开始指定备份路

径即可。例如，01000000E:\bakdir。如果不指定备份路径，系统会自动生成一个备份路径。

- SUCC_ACTION

指定步骤执行成功后，下一步该做什么事。取值 0 或 1。说明如下：

0 表示执行下一步。

1 表示报告执行成功，并执行下一步。

- FAIL_ACTION

指定步骤执行失败后，下一步该做什么事。取值 0 或 2。说明如下：

0 表示执行下一步。

2 表示报告执行失败，并结束作业。

- RETRY_ATTEMPTS

表示当步骤执行失败后，需要重试的次数。取值范围 0~100 次。

- RETRY_INTERVAL

表示在每两次步骤执行重试之间的间隔时间。不能大于 10 秒钟。

- OUTPUT_FILE_PATH

表示步骤执行时输出文件的路径。该参数已废弃，没有实际意义。

- APPEND_FLAG

输出文件的追写方式。如果指定输出文件，那么这个参数表示在写入文件时是否从文件末尾开始追写。1 是；0 否。如果是 0，那么从文件指针当前指向的位置开始追写。

例如，下面的语句为作业 TEST 增加了步骤 STEP1。

```
SP_ADD_JOB_STEP('TEST', 'STEP1', 0, 'INSERT INTO MYINFO VALUES(1000, 'HELLO  
WORLD');', 0, 0, 2, 1, NULL, 0);
```

STEP1 指定的是执行 SQL 语句，其 COMMAND 参数指定的是向 MYINFO 表中插入一条记录，执行成功和失败的下一步动作都是 STEP_ACTION_NEXT_STEP（执行下一步），同时指定了失败后只重试两次，时间间隔为 1 秒钟。

4.1.2.2.2 修改步骤

修改作业的步骤通过系统过程 SP_ALTER_JOB_STEP 实现。

语法如下：

```
SP_ALTER_JOB_STEP (  
  
    JOB_NAME          VARCHAR(128) ,  
  
    STEP_NAME         VARCHAR(128) ,  
  
    TYPE              INT ,  
  
    COMMAND           VARCHAR(8187) ,  
  
    SUCC_ACTION       INT ,  
  
    FAIL_ACTION       INT ,  
  
    RETRY_ATTEMPTS    INT ,  
  
    RETRY_INTERVAL    INT ,  
  
    OUTPUT_FILE_PATH  VARCHAR(256) ,  
  
    APPEND_FLAG       INT  
  
)
```

参数详解

所有参数与 SP_ADD_JOB_STEP 的参数一样，可参考 4.1.2.2.1 节。

4.1.2.2.3 删除步骤

如果用户发现一个作业中的某个步骤不需要了，可以通过系统过程 SP_DROP_JOB_STEP 删除这个步骤。

语法如下：

```
SP_DROP_JOB_STEP (  
  
    JOB_NAME          VARCHAR(128) ,  
  
    STEP_NAME         VARCHAR(128)  
  
)
```

参数详解

- JOB_NAME

作业名称。表示正在删除该作业下的步骤。这个参数必须为前面调用 SP_JOB_CONFIG_START 函数时指定的作业名，否则系统会报错，同时系统会检测这个作业是否存在，不存在也会报错。

- STEP_NAME

要删除的步骤名。删除时会检测这个步骤是否存在，如果不存在则报错。

例如，为作业 TEST 删除步骤 STEP1。

```
SP_DROP_JOB_STEP('TEST', 'STEP1');
```

4.1.2.3 作业调度

增加、删除调度必须是在配置作业开始后才能进行，否则系统会报错，这样处理主要是为了保证作业配置的完整性。

4.1.2.3.1 增加调度

增加调度通过调度系统过程 SP_ADD_JOB_SCHEDULE 实现。

语法如下：

```
SP_ADD_JOB_SCHEDULE (  
  
    JOB_NAME                VARCHAR (128) ,  
  
    SCHEDULE_NAME           VARCHAR (128) ,  
  
    ENABLE                   INT ,  
  
    TYPE                     INT ,  
  
    FREQ_INTERVAL           INT ,  
  
    FREQ_SUB_INTERVAL       INT ,  
  
    FREQ_MINUTE_INTERVAL    INT ,  
  
    STARTTIME               VARCHAR (128) ,  
  
    ENDTIME                 VARCHAR (128) ,  
  
    DURING_START_DATE       VARCHAR (128) ,  
  
    DURING_END_DATE         VARCHAR (128) ,  
  
    DESCRIBE                 VARCHAR (500)  
  
)
```

参数详解

- JOB_NAME

作业名称。指定要给该作业增加调度，这个参数必须是配置作业开始时指定的作业名，

否则报错，同时系统还会检测这个作业是否存在，如果不存在也会报错。

- SCHEDULE_NAME

待创建的调度名称。必须是有效的标识符，同时不能是 DM 关键字。指定的作业不能创建两个同名的调度，创建时会检测这个调度是否已经存在，如果存在则报错。

- ENABLE

表示调度是否启用，布尔类型。1 启用；0 不启用。

- TYPE

指定调度类型。取值 0、1、2、3、4、5、6、7、8。分别介绍如下：

0 表示指定作业只执行一次。

1 按天的频率来执行。

2 按周的频率来执行。

3 在一个月中的某一天执行。

4 在一个月中的第一周第几天执行。

5 在一个月中的第二周的第几天执行。

6 在一个月中的第三周的第几天执行。

7 在一个月中的第四周的第几天执行。

8 在一个月中的最后一周的第几天执行。

当 TYPE=0 时，其执行时间由下面的参数 DURING_START_DATE 指定。

- FREQ_INTERVAL

与 TYPE 有关。表示不同调度类型下的发生频率。说明如下：

当 TYPE=0 时，这个值无效，系统不做检查。

当 TYPE=1 时，表示每几天执行，取值范围为 1~100。

当 TYPE=2 时，表示的是每几个星期执行，取值范围没有限制。

当 TYPE=3 时，表示每几个月中的某一天执行，取值范围没有限制。

当 TYPE=4 时，表示每几个月中的第一周执行，取值范围没有限制。

当 TYPE=5 时，表示每几个月中的第二周执行，取值范围没有限制。

当 TYPE=6 时，表示每几个月中的第三周执行，取值范围没有限制。

当 TYPE=7 时，表示每几个月中的第四周执行，取值范围没有限制。

当 TYPE=8 时，表示每几个月中的最后一周执行，取值范围没有限制。

- FREQ_SUB_INTERVAL

与 TYPE 和 `FREQ_INTERVAL` 有关。表示不同 TYPE 的执行频率，在 `FREQ_INTERVAL` 基础上，继续指定更为精准的频率。说明如下：

当 TYPE=0 或 1 时，这个值无效，系统不做检查。

当 TYPE=2 时，表示的是某一个星期的星期几执行，可以同时选中七天中的任意几天。取值范围 1~127。具体如何取值，请用户参考如下规则。因为每周有七天，所以 DM 数据库系统内部用七位二进制来表示选中的日子。从最低位开始算起，依次表示周日、周一...周五、周六。选中周几，就将该位置 1，否则 0。例如，选中周二和周六，7 位二进制就是 1000100，转化成十进制就是 68，所以 `FREQ_SUB_INTERVAL` 就取值 68。

当 TYPE=3 时，表示将在一个月的第几天执行。取值范围 1~31。

当 TYPE 为 4、5、6、7 或 8 时，都表示将在某一周内第几天执行。取值范围 1~7，分别表示从周一到周日。

- `FREQ_MINUTE_INTERVAL`

表示一天内每隔多少分钟执行一次。有效值范围 1~1440，单位分钟。

- `STARTTIME`

定义作业被调度的起始时间。必须是有效的时间字符串，不可以为空。

- `ENDTIME`

定义作业被调度的结束时间。可以为空。但如果不为空，指定的必须是有效的时间字符串，同时必须要在 `STARTTIME` 时间之后。

- `DURING_START_DATE`

指定作业被调度的起始日期。必须是有效的日期字符串，不可以为空。

- `DURING_END_DATE`

指定作业被调度的结束日期。可以为空，`DURING_END_DATE` 和 `ENDTIME` 都为空，调度活动会一直持续下去。但如果不为空，必须是有效的日期字符串，同时必须是在 `DURING_START_DATE` 日期之后。

- `DESCRIBE`

表示调度的注释信息，最大长度为 500 个字节。

例如，下面的语句为作业 TEST 增加名为 SCHEDULE3 的调度。

```
SP_ADD_JOB_SCHEDULE('TEST', 'SCHEDULE3', 1, 1, 1, 0, 1, CURTIME, '23:59:59',  
CURDATE, NULL, '一个测试调度');
```

上面的例子中，为作业 TEST 创建一个新的调度，调度名为 SCHEDULE3；ENABLE 为 1，

即启用这个调度；其调度类型 TYPE 为 1，表示按天的频率来执行；FREQ_INTERVAL 为 1，说明每天都要执行；在这种类型下 FREQ_SUB_INTERVAL 参数就不会检查，随机写 0；FREQ_MINUTE_INTERVAL 指定的是 1，说明每隔一分钟就执行一次；STARTTIME 指定是从当前时间开始，CURTIME 表示系统当前时间；ENDTIME 指定 23:59:59，表示每天都是执行到这个时间为止；DURING_START_DATE 为调度的起始日期，表示系统当前日期；DURING_END_DATE 指定的为 NULL，表示这个调度指定的日期范围为从开始执行那一刻起，永不停止；DESCRIBE 指定为 '一个测试调度'，这是对这个调度的注释。

4.1.2.3.2 修改调度

修改调度通过调度系统过程 SP_ALTER_JOB_SCHEDULE 实现。

语法如下：

```
SP_ALTER_JOB_SCHEDULE (
    JOB_NAME                VARCHAR(128),
    SCHEDULE_NAME           VARCHAR(128),
    ENABLE                   INT,
    TYPE                    INT,
    FREQ_INTERVAL           INT,
    FREQ_SUB_INTERVAL       INT,
    FREQ_MINUTE_INTERVAL    INT,
    STARTTIME               VARCHAR(128),
    ENDTIME                 VARCHAR(128),
    DURING_START_DATE       VARCHAR(128),
    DURING_END_DATE         VARCHAR(128),
    DESCRIBE                VARCHAR(500)
)
```

参数详解

所有参数与 SP_ADD_JOB_SCHEDULE 的参数一样，可参考 4.1.2.3.1 节。

4.1.2.3.3 删除调度

删除调度必须是在配置作业开始后才能进行，否则系统会报错，这样处理主要是为了保证作业配置的完整性。如果不再需要某一个调度，可以将其删除。调用的函数为 SP_DROP_JOB_SCHEDULE。

语法如下：

```
SP_DROP_JOB_SCHEDULE (
    JOB_NAME          VARCHAR(128) ,
    SCHEDULE_NAME     VARCHAR(128)
)
```

参数详解

- JOB_NAME

作业名称。表示的是正在删除该作业下的调度，这个参数必须为上面调用 SP_JOB_CONFIG_START 函数时指定的作业名，否则系统会报错，同时系统会检测这个作业是否存在，不存在也会报错。

- SCHEDULE_NAME

要删除的调度的调度名。删除时会检测这个调度是否存在，如果不存在则报错。

例如，删除作业 TEST 中名为 SCHEDULE3 的调度。

```
SP_DROP_JOB_SCHEDULE('TEST', 'SCHEDULE3');
```

4.1.2.4 结束作业配置

在配置过程中，可以对指定的作业增加、删除任意多个调度、步骤，但不要做提交操作以及自动提交操作，否则可能会出现作业配置不完整的问题。

在配置完成后，用户需要对前面所做的配置进行提交，表示对作业的配置已经完成，同时将这个作业加入到运行队列。这一步可以通过系统过程 SP_JOB_CONFIG_COMMIT 实现。

语法如下：

```
SP_JOB_CONFIG_COMMIT (
    JOB_NAME          VARCHAR(128)
)
```

参数详解

- JOB_NAME

待结束配置的作业的名称。

调用这个过程时，系统会检测当前会话是否处于作业配置状态，如果不处于配置状态，则系统会报“非法的作业配置操作”的错误。

在成功执行该过程后，系统会将前面所做的所有操作提交，同时将这个作业加入到运行队列，运行的内容包括这个作业下定义的所有步骤的执行内容，执行方式就是根据这个作业下定义的所有的调度定义的执行方式来执行。

4.1.3 查看、清除作业日志记录

4.1.3.1 查看作业日志记录

创建的每一个作业信息都存储在作业表 SYSJOBS 中。通过查看表 SYSJOBS，可以看到所有已经创建的作业。

4.1.3.1 清除作业日志记录

因为日志记录会不断增加，越来越庞大，所以用户需要及时清理过时的日志。

可以通过系统过程 SP_JOB_CLEAR_HISTORIES 清除迄今为止某个作业的所有日志记录，即删除表 SYSJOBHISTORIES 中的相关记录。如果该作业还在继续工作，那么后续会在表 SYSJOBHISTORIES 中产生该作业的新日志。

语法如下：

```
SP_JOB_CLEAR_HISTORIES (  
    JOB_NAME          VARCHAR(128)  
)
```

参数详解

- JOB_NAME

待清除日志的作业名。

例如，清除迄今为止作业 TEST 的所有日志记录。

```
SP_JOB_CLEAR_HISTORIES ('TEST');
```

4.2 通过图形化客户端实现

可以通过图像化客户端 MANAGER 工具实现作业的创建和配置等操作。

点击**代理**中的**作业**。可以看到新建作业、设置过滤和清除过滤、查看作业历史信息 and 清理作业历史信息、刷新按钮。如下所示：

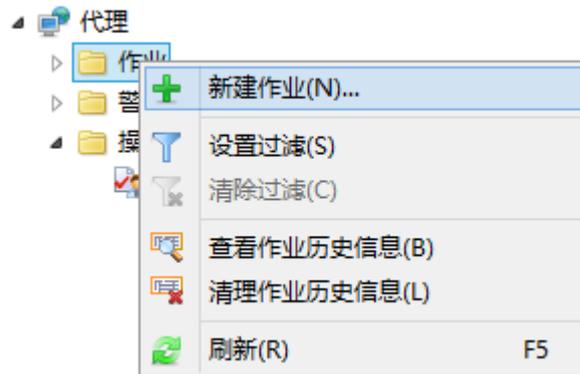


图 4.1 新建作业界面

4.2.1 新建作业

点击**新建作业**按钮，会出现作业对话框。作业对话框用于实现作业的创建与配置，包含常规、作业步骤、作业调度、DDL 四部分。

4.2.1.1 常规

作业的常规属性设置页面。如下所示：

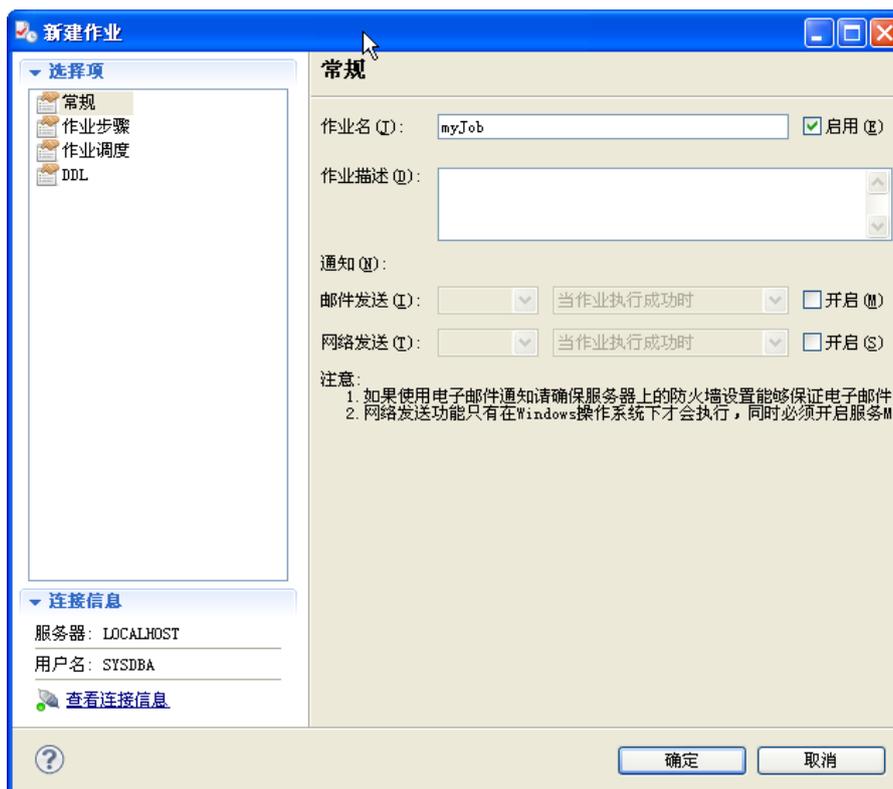


图 4.2 新建作业-常规界面

参数详解

- 作业名

作业的标识名称。

- 启用

是否启用这个作业。

- 作业描述

作业的一些描述信息。

- 邮件通知

在邮件通知开启的情况下，作业相关的一些日志会通过邮件来通知操作员，如果不开启则不会发送邮件。操作员列表用来选择要发送的操作员，发送时机列表用来选择消息发送的时机。

- 网络通知

在网络通知开启的情况下，作业相关的一些日志会通过网路来通知操作员，如果不开启则不会发送消息，这个功能只有在 WINDOWS 操作系统下才会执行，同时必须开启服务 MESSAGER 才能发送成功。操作员列表用来选择要发送的操作员，发送时机列表用来选择消息发送的时机。

添加一个新的作业步骤，参见作业步骤对话框。

- 删除

删除选中的作业步骤。

4.2.1.2.2 作业步骤对话框

作业步骤对话框用于实现作业步骤的创建与修改。包括常规和高级两部分。

4.2.1.2.2.1 常规

作业步骤的常规属性设置页面。如下所示：

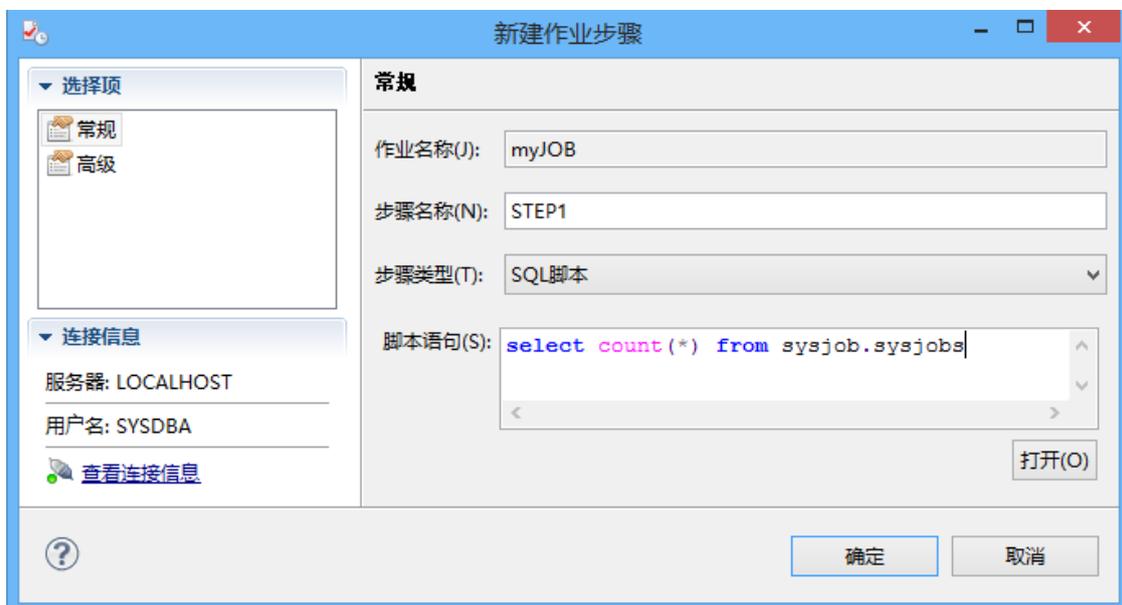


图 4.4 新建作业步骤-常规界面

参数详解

- 作业名称

作业步骤所属作业的标识名称。

- 步骤名称

作业步骤的名称。

- 步骤类型

标识该步骤的类型，步骤的类型主要有以下几种：

- 🚩 SQL 脚本
- 🚩 备份数据库

- ✚ 重组数据库
- ✚ 更新统计信息
- ✚ 数据迁移
- ✚ 基于备份集备份数据库

针对不同的步骤类型，需要不一样的深入设置。下面详细介绍：

- ✚ 当类型为 SQL 脚本时，用户需要指定的要执行的 SQL 语句或者语句块。
- ✚ 当类型为备份数据库时，界面如下所示：

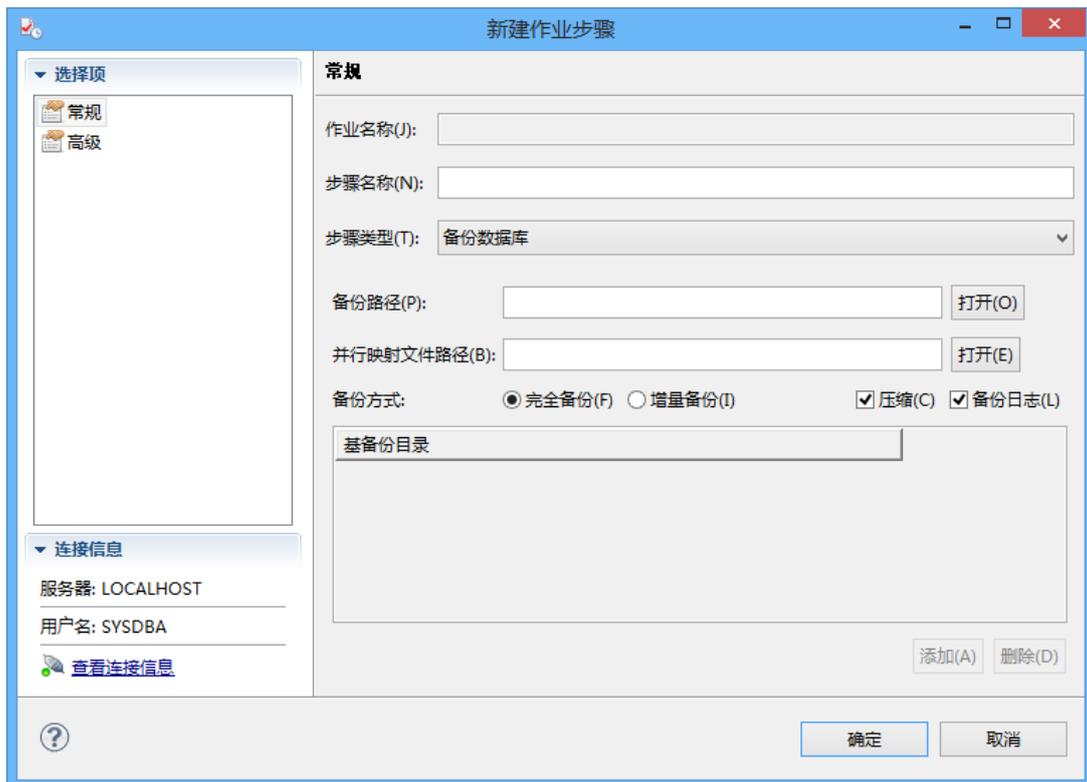


图 4.5 常规-备份数据库界面

- ✚ 当类型为重组数据库和更新统计信息时，不需要进一步设置任何东西。
- ✚ 当类型为数据迁移时，界面如下所示：



图 4.6 常规-数据迁移界面

当类型为基于备份集备份数据库时，界面如下所示：

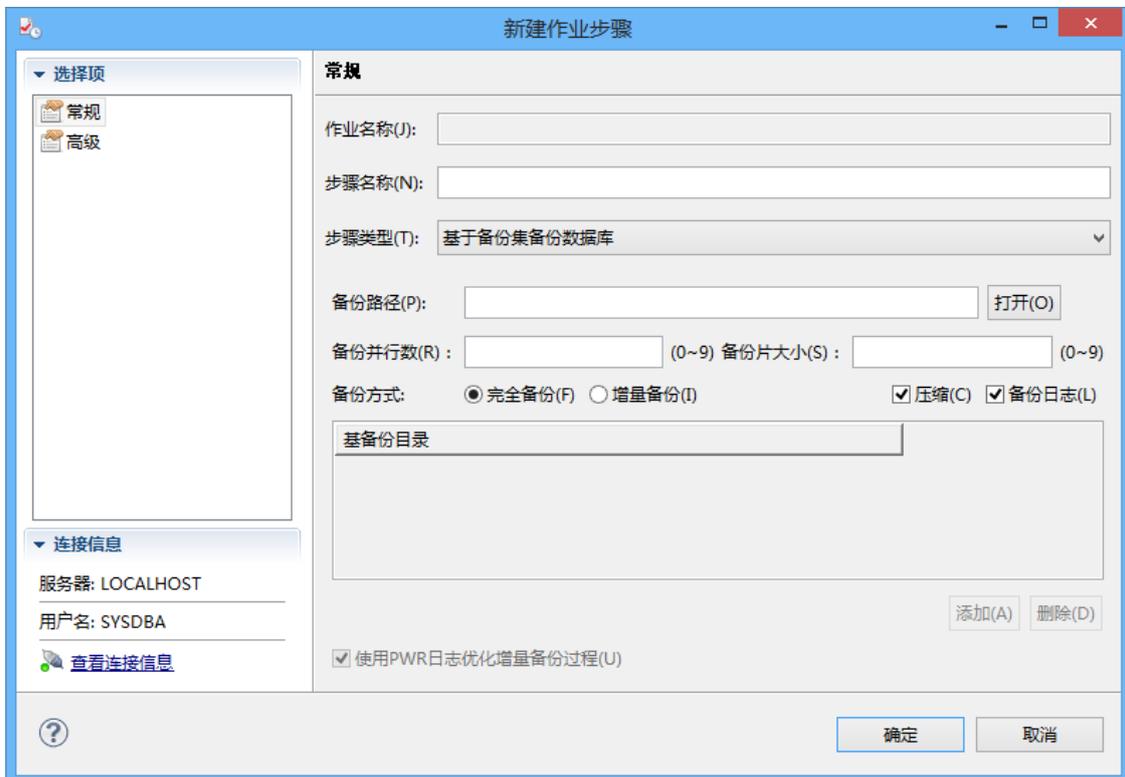


图 4.7 常规-基于备份集备份数据库界面

4.2.1.2.2.2 高级

作业步骤的高级属性配置页面。如下所示：

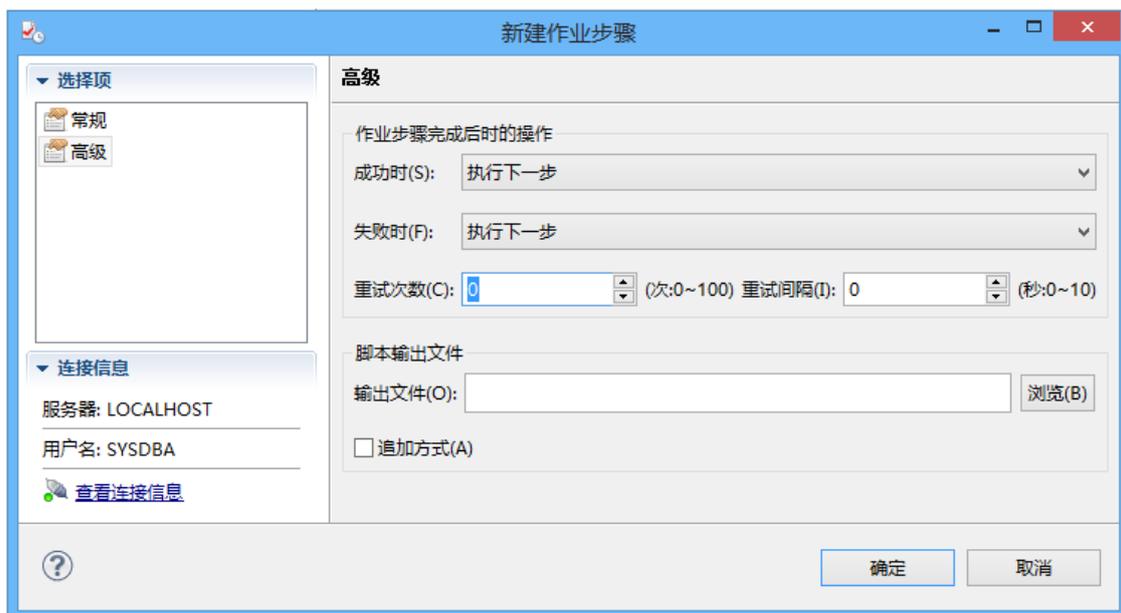


图 4.8 新建作业步骤-高级界面

参数详解

- 作业步骤完成后时的操作

成功时

指定这个步骤执行成功后，下一步该做什么事，有三个可选项：

- ✓ 执行下一步
- ✓ 报告执行成功
- ✓ 返回第一个步骤继续执行

失败时

指定这个步骤执行失败后，下一步该做什么事，有三个可选项：

- ✓ 执行下一步
- ✓ 报告执行失败
- ✓ 返回第一个步骤继续执行

重试次数

表示当一个步骤执行失败后，需要重试的次数。次数不能大于 100。

重试间隔

表示的是在每两次步骤执行重试之间的间隔时间，间隔时间不能大于 10 秒钟。

- 脚本输出文件

输出文件

表示步骤执行时如果有输出结果，输出信息的文件路径。该参数已废弃。

修改选中的作业调度。

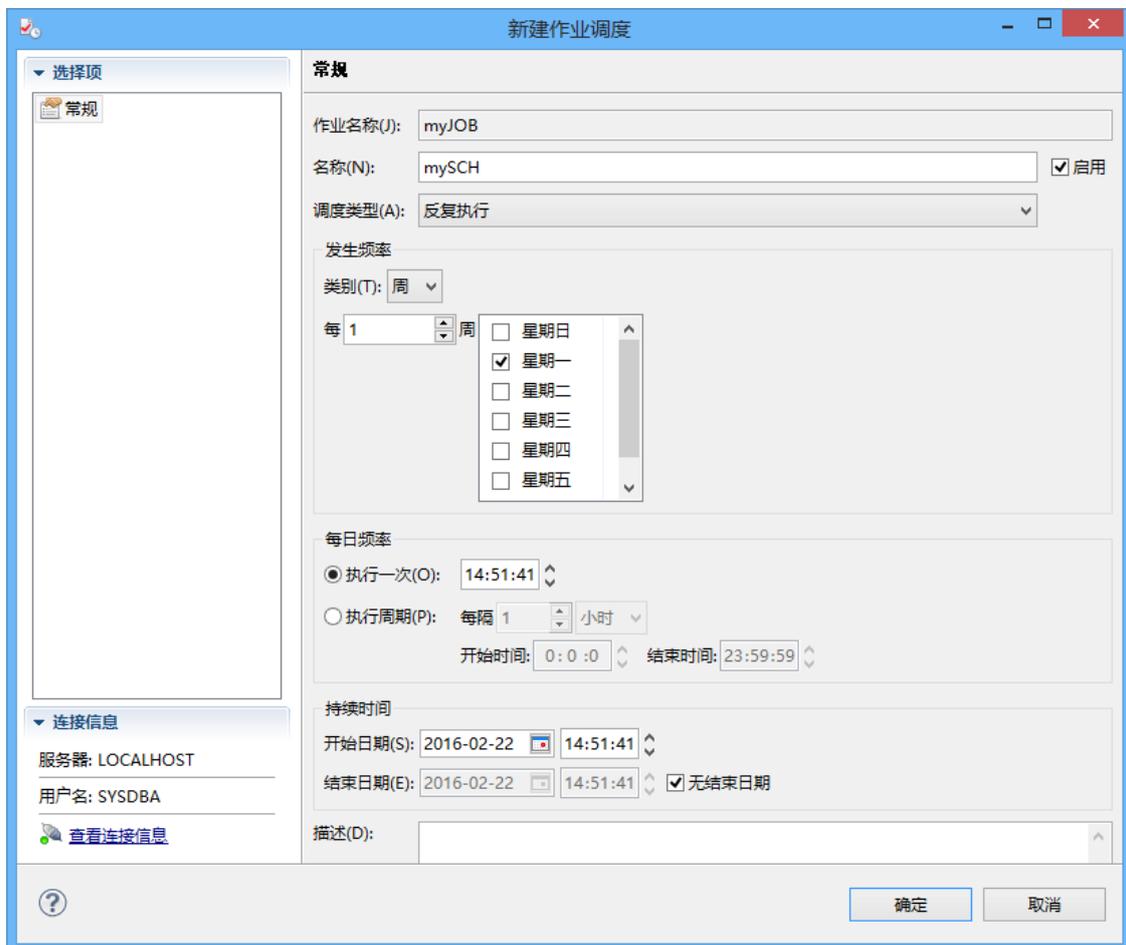
- 删除

删除选中的作业调度。

4.2.1.3.2 作业调度对话框

作业调度对话框用于实现作业调度的创建与修改。

作业的常规属性设置页面。



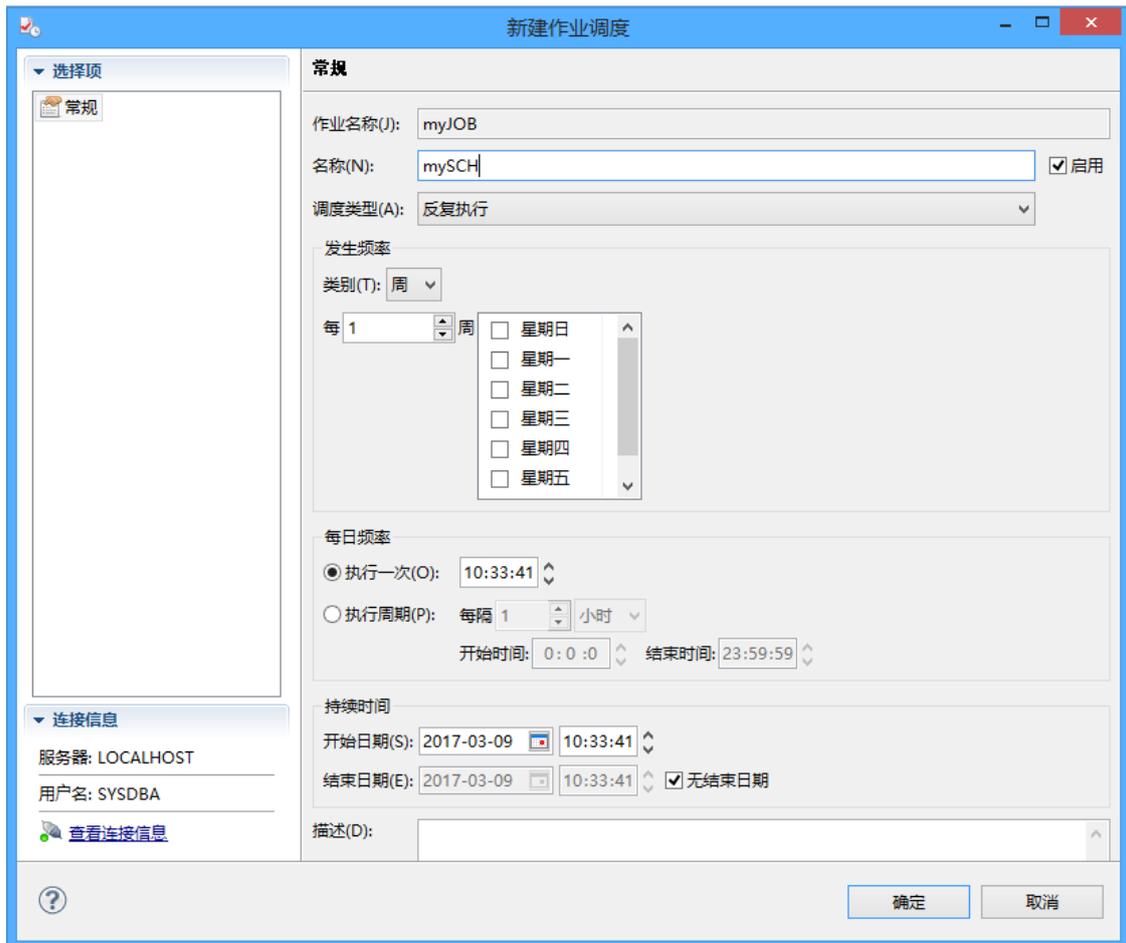


图 4.10 新建作业调度-常规界面

参数详解

- 作业名称
作业调度所属的作业名。
- 名称
作业调度的名称。
- 启用
指定作业步骤是否启用。
- 调度类型
分为反复执行和执行一次两种类型，参数不一样。执行一次只有时间和描述。下面详细介绍反复执行类型的参数。
- 发生频率
频率类别有天、周、月。
- 每日频率

每日频率有执行一次，可选择相应的执行时间；也可多次执行，需要选择相应的执行时间。

- 持续时间

默认开始时间是今天，默认无结束日期；也可以选择开始时间和结束时间，但结束时间必须大于开始时间。

- 描述

作业调度的附加描述信息。

4.2.1.4 DDL

显示作业创建与配置的 SQL 语句。



图 4.11 新建作业-DDL 界面

参数详解

- 保存

保存 SQL 脚本到文件中。

4.2.2 设置、清除过滤

当作业过多时，可以通过设置过滤条件来选择符合条件的作业。

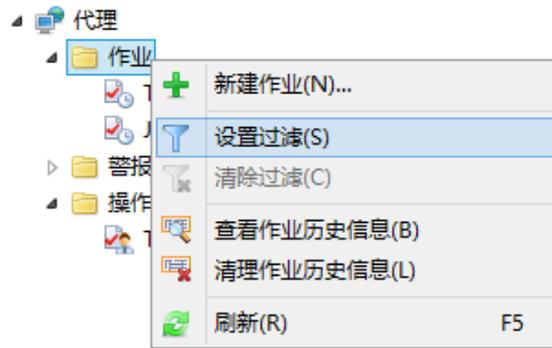


图 4.12 作业-设置过滤界面

点击**作业**，选中**设置过滤**，会出现如下界面。其中，过滤条件有包含、不包含和等于三种。例如，设置过滤条件为名称中包含 TEST 的作业。

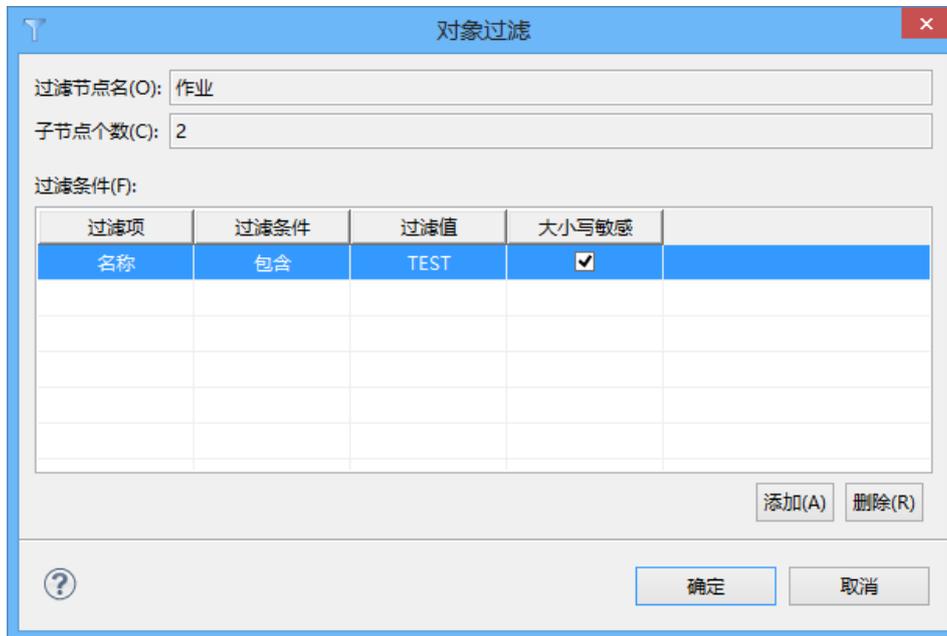


图 4.13 作业-对象过滤界面

过滤条件使用完后，想恢复原状，看到所有作业，此时需要清除过滤条件，选中**清除过滤**即可。

4.2.3 查看、清除作业历史信息

4.2.3.1 查看历史作业信息

查看作业历史信息视图用于显示作业步骤的执行情况日志。作业每执行一次，自动向该视图中插入一条相关信息。这个视图中的所有记录都是由作业在运行过程中系统自动插入的，不是由用户来操作的。

编号	作业名	步骤名	错误类型	错误码	错误信息	开始时间	结束时间	重试次数	是否通知
70902313	myJOB	STEP1	0	-8003	[JOBmyJOB...本地或者远程归档	2017-03-09 10:54:36.688	2017-03-09 10:54:36.69	0	否
70902312	myJOB	STEP1	0	-8003	[JOBmyJOB...本地或者远程归档	2017-03-09 10:49:36.551	2017-03-09 10:49:36.553	0	否

图 4.14 查看历史作业信息界面

参数详解

- 编号
编号为每一条作业历史记录的一个唯一标识。
- 作业名
表示的是某一条历史记录是由哪一个作业产生的，用作业名表示。
- 步骤名
表示的是这个历史记录是由哪一个步骤产生的，用步骤名表示。
- 错误类型
这个列一般情况不用，现在都是 0。
- 错误码
表示的是在步骤执行错误后，产生的错误码。
- 错误信息
表示的是这个警报所处的数据库实例名。
- 开始时间
任务的开始时间。
- 结束时间
任务的结束时间。
- 重试次数

表示这条历史记录是第几次重试时产生的，这个列记录其当前次数。

- 是否通知

表示这条历史记录是否已经（邮件及网络发送）通知用户，如果通知则这个值为 1，否则为 0。

4.2.2.2 清除作业历史信息

清除作业历史信息，清除的是除迄今为止该作业的所有日志记录，即删除表 SYSJOBHISTORIES 中的相关记录。如果该作业还在继续工作，那么后续会产生新的历史信息。

4.2.4 修改作业

如果 DBA 发现某一个作业中的信息不合理需要修改，可以选中作业，右击查看菜单。

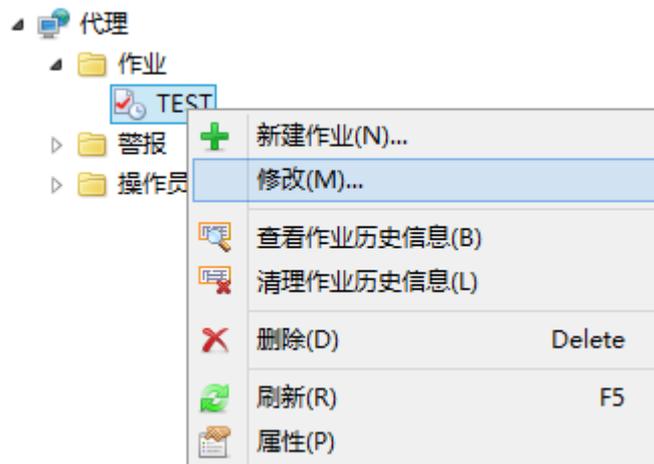


图 4.15 作业修改界面

选中**修改**。界面如下：

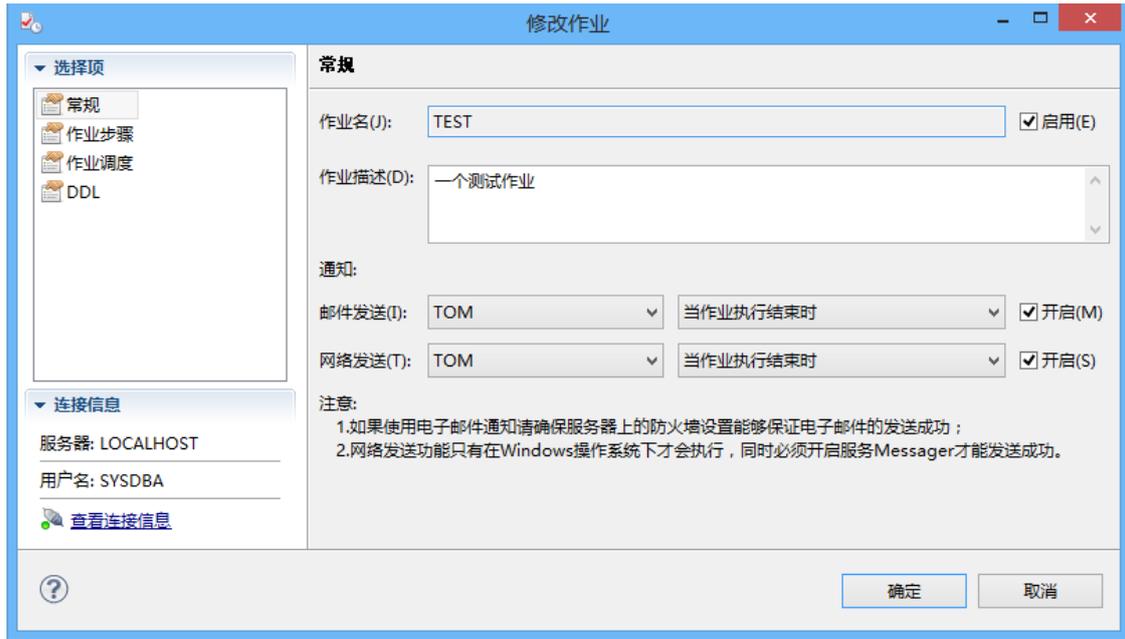


图 4.16 修改作业-常规界面

用户根据需要，修改作业、步骤和调度即可。

4.2.5 删除作业

选中作业，右击查看菜单，可以看到**删除**按钮。

5 警报

达梦数据库还提供了警报的功能，允许用户定义一些自己想要捕获的动作，如果这些动作发生时，系统就会将对应信息存储到指定的表中，将想要得到的信息存储起来，以便做到实时的监控。

创建、修改和删除警报可以通过以下两种方式来实现。一是通过系统过程来实现；二是通过图形化客户端 MANAGER 管理工具实现。用户选择其中的一种即可。

5.1 通过系统过程实现

5.1.1 创建、删除和修改警报

5.1.1.1 创建警报

创建一个警报可以通过系统过程 SP_CREATE_ALERT 实现。警报的定义信息都存储在系统表 SYSALERTS 中。

语法如下：

```
SP_CREATE_ALERT (
    NAME          VARCHAR( 128 ),
    ENABLED       INT ,
    TYPE          INT ,
    ERRTYPE       INT ,
    ERRCODE       INT ,
    DELAYTIME     INT ,
    DESCRIBE      VARCHAR( 8187)
)
```

参数详解

- NAME

警报名。必须是有效的标识符，同时不能是 DM 关键字。警报不能同名，在创建时系统会检测这个警报是否存在，如果存在则报错。

- **ENABLE**

警报是否开启。取值 0 或 1。1 是；0 否。

- **TYPE**

警报的类型。取值 0 或 1。0 表示发生错误时警报；1 表示发生某些数据库事件时警报。

- **ERRTYPE**

触发警报的错误和数据库事件。与参数 TYPE 相关。

当 TYPE=0 时，ERRTYPE 表示触发警报的错误类型。取值 1~12。具体含义分别为：1 常规错误；2 启动错误；3 系统错误；4 服务器配置错误；5 分析阶段错误；6 权限错误；7 运行时错误；8 备份恢复错误；9 作业管理错误；10 数据库复制错误；11 其它错误；12 指定错误码。

当 TYPE=1 时，ERRTYPE 表示触发警报的数据库事件。取值 1~4。具体含义分别为：1 DDL 事件；2 授权回收事件；3 连接或断开数据库事件；4 数据库备份恢复事件。

- **ERRCODE**

指定错误码。错误码取值和参数 TYPE、ERRTYPE 相关。

当 TYPE=0、ERRTYPE 取 1~12 时，指定各种错误类型相关的错误码，ERRCODE 必须是小于 0 的整数。

当 TYPE=1、ERRTYPE=1 时，指定 DDL 事件的错误码，ERRCODE 取值 1~15。如何取值，请参考如下规则：因为 DDL 事件包含 CREATE、ALTER、DROP 和 TRUNC 四种，所以此处的错误码是四种当中的任意几个组合值。ERRCODE 在 DM 数据库系统内部用四位二进制来表示它们的组合，从低位到高位依次是：CREATE、ALTER、DROP、TRUNCATE。1 表示指定，0 表示不指定。用户输入的 ERRCODE 值需要转化为十进制。例如，指定 CREATE 和 DROP，内部二进制表示为 0101，转化为十进制为 5，所以 ERRCODE 值就是 5。

当 TYPE=1、ERRTYPE=2 时，指定授权回收事件的错误码，ERRCODE 取值 1、2、3。1 表示 GRANT 的错误码；2 表示 REVOKE 的错误码；3 表示 GRANT 和 REVOKE 的错误码。

当 TYPE=1、ERRTYPE=3 时，指定连接事件的错误码，取值 1、2、3。1 表示 LOGIN 的错误码；2 表示 LOGOUT 的错误码；3 表示 LOGIN 和 LOGOUT 的错误码。

当 TYPE=1、ERRTYPE=4 时，指定数据库备份恢复事件的错误码，取值 1、2、3。1 表示 BACKUP 的错误码；2 表示 RESTORE 的错误码；3 表示 BACKUP 和 RESTORE 的错误码。

- **DELEYTIME**

表示警报发生后，推迟多长时间通知操作员。范围 0~3600，单位秒。

- ADDITION_TXT

表示警报的注释，最长为 500 个字节。

例如，创建一个名为 ALERT1 的警报。

```
SP_CREATE_ALERT('ALERT1',1, 0, 1, -600, 1, '错误码的测试');
```

上面的例子创建了一个名为 ALERT1 的警报，ENABLE 为 1 表示开启，TYPE 为 0 表示发生错误时警报，ERRTYPE 指定的是 1 表示常规错误，表示只有发生常规错误时警报才发生，DELEYTIME 指定可以推迟 1 秒钟通知操作员。警报发生后，系统会将对应信息存储到上面提到的 SYSALERTHISTORIES 表中。

5.1.1.2 修改警报

修改警报可以通过系统过程 SP_ALTER_ALERT 来实现。

语法如下：

```
SP_ ALTER _ALERT (
    NAME          VARCHAR(128),
    ENABLED       INT,
    TYPE          INT,
    ERRTYPE       INT,
    ERRCODE       INT,
    DELAYTIME     INT,
    DESCRIBE      VARCHAR(8187)
)
```

参数详解

它的参数和 SP_CREATE_ALERT 的参数完全相同，除了 NAME 不可修改外，其他的属性都可修改。对于可修改参数，如果要修改，则指定新值；如果不修改，则指定原值即可。

例如，下面的语句修改警报 ALERT1。

```
SP_ALTER_ALERT('ALERT1', 1, 1, 1, 15, 1, 'DDL 警报测试');
```

上面的例子修改警报 ALERT1，ENABLE 还是原来的值，没有被修改；TYPE 修改为 1，表示对事件的捕获；ERRTYPE 修改为 1，表示的是对事件中的 DDL 进行捕获；ERRCODE 修改为 15，从上面可以看出，这个是一个组合值，对于 DDL 事件共有四种类型(CREATE、ALTER、

DROP、TRUNC)，每一个位表示一个操作，这里的 15 可以被分解为二进制 1111，可以看出四个位都为 1，也就是选中了这四个操作中的所有操作；DELAYTIME 没有被修改；ADDITION_TXT 被修改为 'DDL 警报测试'。修改后的警报只要系统做了 DDL 操作就会被激发，同时会将相应信息存储到上面提到的 SYSALERTHISTORIES 表中。

5.1.1.3 删除警报

删除警报通过调用系统过程 SP_DROP_ALERT 实现。

语法如下：

```
SP_DROP_ALERT (
    NAME          VARCHAR(128),
)
```

参数详解

- NAME

待删除的警报名。删除时系统会检测这个警报是否存在，如果不存在则报错。

例如，下面的语句删除警报 ALERT1。

```
SP_DROP_ALERT('ALERT1');
```

5.1.2 为警报关联操作员

5.1.2.1 关联

如果想要将警报的执行结果通知给指定操作员，需要将这个警报关联要通知到的操作员，关联操作员通过系统 SP_ALERT_ADD_OPERATOR 实现。一个警报可以关联多个操作员。

语法如下：

```
SP_ALERT_ADD_OPERATOR (
    ALERTNAME      VARCHAR(128),
    OPR_NAME       VARCHAR(128),
    ENABLEMAIL     INT,
    ENABLENETSENT  INT
)
```

)

参数详解

● ALERTNAME

警报名。表示要将哪一个警报进行关联，在创建时系统会检测这个警报是否存在，如果不存在则报错。

● OPER_NAME

操作员名。表示要与哪一个操作员进行关联，在创建时系统会检测这个操作员是否存在，如果不存在则报错。

● ENABLEMAIL

表示指定警报的执行结果是否用邮件的方式通知指定的操作员。1 是；0 否。

● ENABLENETSEND

表示指定警报的执行结果是否用网络发送的方式通知指定的操作员。1 是；0 否。这个功能只有在 WINDOWS 下才起作用。

例如,下面的语句指定 ALERT1 警报的执行结果用邮件和网络两种方式通知操作员 TOM。

```
SP_ALERT_ADD_OPERATOR('ALERT1','TOM',1,1);
```

5.1.2.2 取消

当不再需要某个警报与操作员的关联，可以通过系统过程 SP_ALERT_DROP_OPERATOR 来取消关联。

语法如下：

```
SP_ALERT_DROP_OPERATOR (
    ALERTNAME          VARCHAR(128) ,
    OPR_NAME           VARCHAR(128)
)
```

参数详解

● ALERTNAME

警报名。表示要取消关联的警报。系统会检测这个警报是否存在，如果不存在则报错。

● OPER_NAME

操作员名。表示要取消关联的操作员，系统会检测这个操作员是否存在，如果不存

则会报错。

例如，取消警报 TEST 与操作员 TOM 的关联，取消之后警报的执行结果不会再通知操作员 TOM。

```
SP_ALERT_DROP_OPERATOR('ALERT1', 'TOM');
```

5.1.3 清除警告日志记录

清除迄今为止某个警报的所有日志记录，即删除表 SYSALERTHISTORIES 中的相关记录。可以通过系统过程 SP_ALERT_DROP_HISTORIES 来实现。

语法如下：

```
SP_ALERT_DROP_HISTORIES (  
    ALERTNAME          VARCHAR(128)  
)
```

参数详解

- ALERTNAME

待清除日志的警报名。

例如，清除所有的 ALERT1 警报日志记录。

```
SP_ALERT_DROP_HISTORIES ('ALERT1');
```

5.2 通过图形化客户端实现

警告对话框用于实现警告的创建与修改。包括：常规、通知信息和 DDL 三部分。

5.2.1 常规

警告的常规属性设置页面。

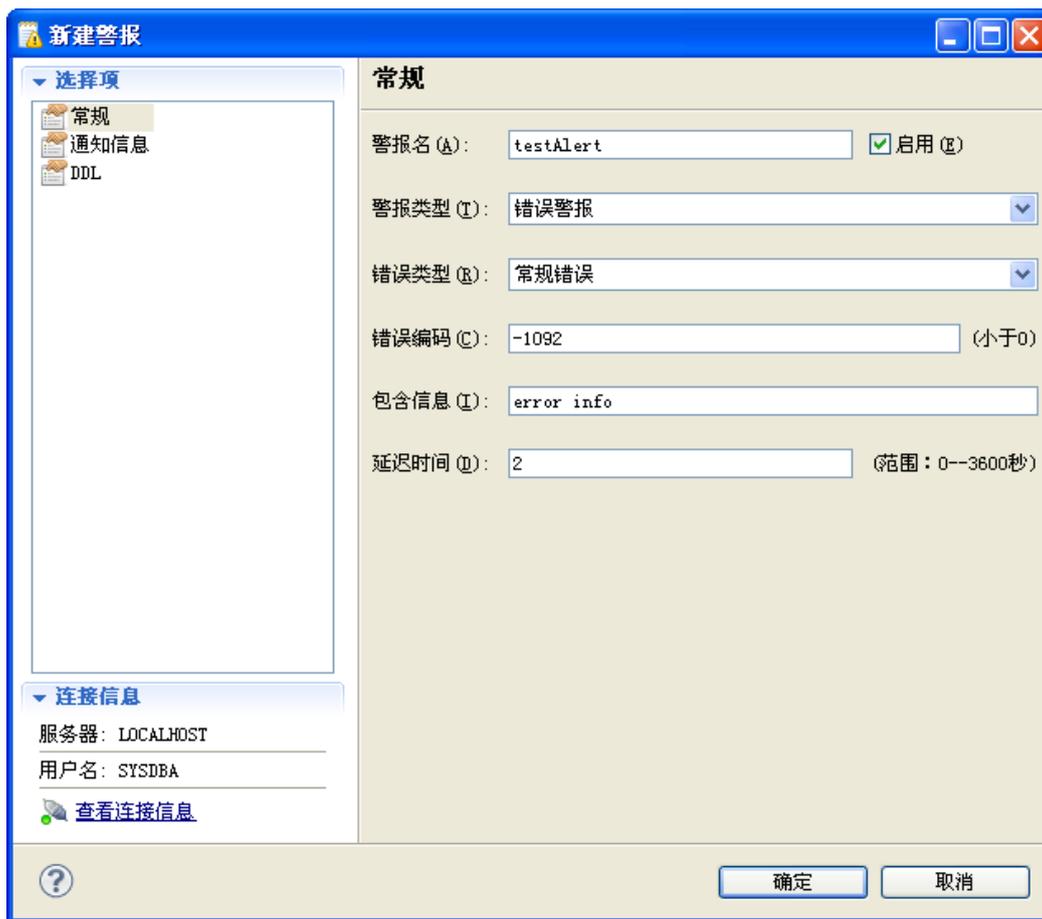


图 5.1 新建警报-常规界面

参数详解

- 警告名
警告的标识名称。
- 启用
是否启用这个警告。
- 警报类型
指定警报的类型。警报类型有错误警报和事件警报，分别表示发生错误时警报和发生某些事件时警报。
- 错误类型或事件类型
错误类型或事件类型由警报类型决定。
 - 如果警报类型为错误警报，则此处为错误类型。错误类型以下 12 个可选项：
 - ✓ 常规错误
 - ✓ 启动错误

- ✓ 系统错误
- ✓ 服务器配置错误
- ✓ 语法分析错误
- ✓ 权限错误
- ✓ 运行时错误
- ✓ 备份恢复错误
- ✓ 作业管理错误
- ✓ 数据库复制错误
- ✓ 其他错误
- ✓ 指定错误码

✚ 如果警报类型为事件警报，则此处为事件警报。事件类型有如下 5 个可选项：

- ✓ DDL 事件
- ✓ 授权回收事件
- ✓ 连接或断开数据库事件
- ✓ 数据库分析事件
- ✓ 数据库备份恢复事件

● 错误编码或事件编码

错误编码或事件编码由警报类型决定。

✚ 如果警报类型为错误警报，此处为错误编码。错误编码的取值由错误类型决定。

- ✓ 如果错误类型为指定错误码，那么此需要错误编码。指定的错误编码必须是小于 0 的值。
- ✓ 如果错误类型为指定错误码之外的其他方式，那么此处不需要指定错误编码。

✚ 如果警报类型为事件警报，此处为事件编码。事件编码的取值由错误类型决定。

- ✓ 如果错误类型为 DDL 事件时，那么事件编码可以是 CREATE、ALTER、DROP、TRUNC 其中的一个或任意多个的组合。
- ✓ 如果错误类型为授权回收事件时，那么事件编码可以是 GRANT 和 REVOKE 其中的一个或组合。
- ✓ 如果错误类型为连接或断开数据库事件时，那么事件编码可以是 LOGIN、LOGOUT 其中的一个或组合。
- ✓ 如果错误类型为数据库分析事件时，那么事件编码为 ANALYZE 。

✓ 如果错误类型为数据库备份恢复事件时，那么事件编码可以是 BACKUP、RESTORE 其中的一个或组合。

- 包含信息

这个参数表示的是对这个警报的一些注释字符串，长度最长为 500 个字节。

- 延迟时间

这个参数表示的是警报通知操作员时需要推迟的时间，推迟的时间范围为 0 到 3600 秒，也就是说必须是在一个小时范围之内。

5.2.2 通知信息

警告的通知信息属性设置页面。

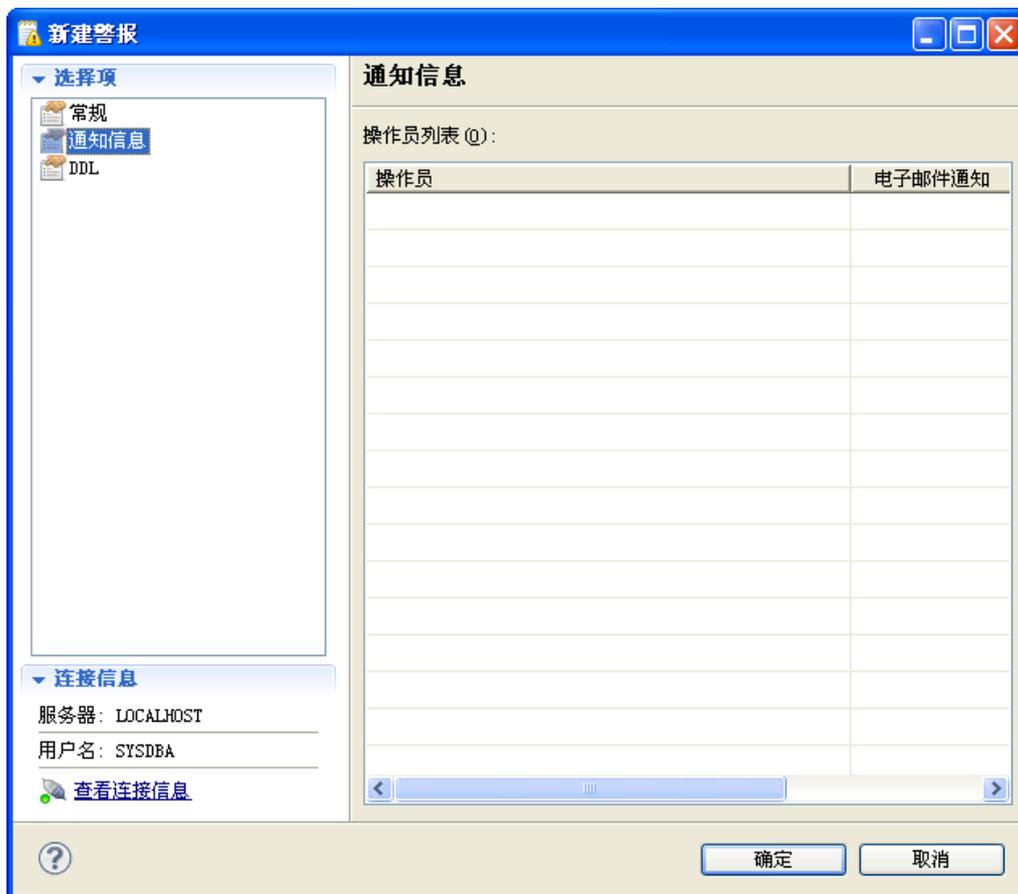


图 5.2 新建警报-通知界面

参数详解

- 操作员列表

已存在的各操作员列表。

- ✚ 操作员

操作员名。

✚ 电子邮件通知

是否采用电子邮件通知。1 是；0 否。

✚ 网络消息通知

是否采用网路消息通知。1 是；0 否。

5.2.3 DDL

显示警告创建与修改的 SQL 语句。

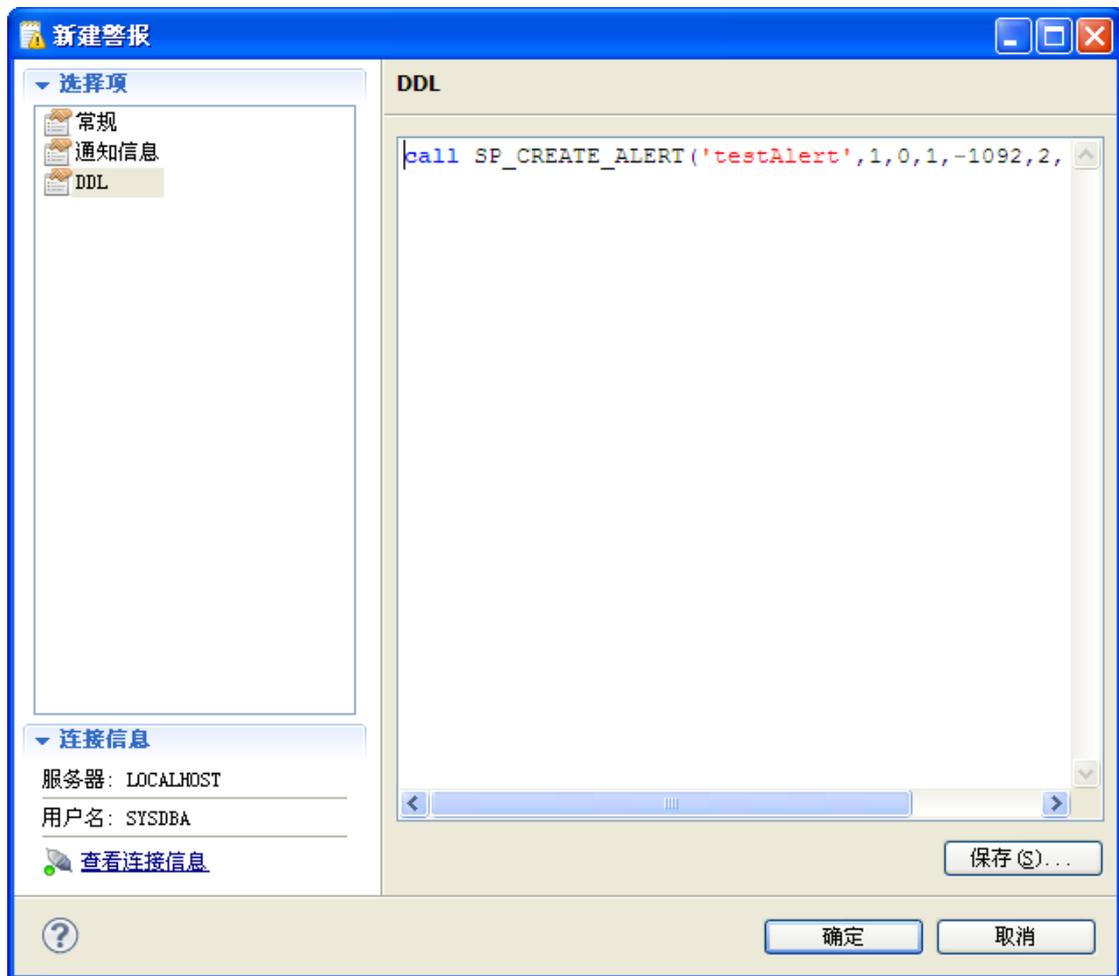


图 5.3 新建警报-DDL 界面

参数详解

- 保存

保存 SQL 脚本到文件中。

6 监控作业

监控作业是指把作业的运行情况通过电子邮件发送给作业操作员。邮件成功发送的前提有两个：一是为监控服务配置管理员；二是开启监控服务。

6.1 配置监控服务管理员

配置管理员可以通过两种方式实现。一是通过系统过程来实现；二是通过图形化客户端 MANAGER 管理工具实现。用户选择其中的一种即可。

添加成功后，SYSMAILINFO 表中会相应增加一条记录。

6.1.1 通过系统过程实现

通过系统过程可以添加、修改和删除管理员。

6.1.1.1 添加管理员

可以通过 SP_ADD_MAIL_INFO 来添加监控服务管理员。

语法如下：

```
SP_ADD_MAIL_INFO (  
  
    LOGIN_NAME          VARCHAR (128) ,  
  
    LOGIN_PWD           VARCHAR (128) ,  
  
    SMTP_SERVER         VARCHAR (128) ,  
  
    EMAIL               VARCHAR (128) ,  
  
    USER_NAME          VARCHAR (128) ,  
  
    USER_PWD           VARCHAR (128)  
  
)
```

参数详解

- LOGIN_NAME

监控服务管理员名称。必须是有效的数据库登录用户名。不能有同名管理员，如果创建同名管理员，系统会报错。此参数不能为空。

- LOGIN_PWD
监控服务管理员登录时的密码，此参数不能为空。
- SMTP_SERVER
登录时指定的邮件 SMTP 服务器，必须是有效的 SMTP 服务器地址，否则无法发送电子邮件。此参数若为空系统自动转换为空串。
- EMAIL
邮件地址，用户的电子邮件地址，使用该用户去 SMTP 服务器认证。此参数若为空系统自动转换为空串。
- USER_NAME
邮件用户名（和邮件地址写法要完全一样）。此参数若为空系统自动转换为空串。
- USER_PWD
邮箱登录密码，用户在 SMTP 服务的密码。此参数若为空系统自动转换为空串。
例如，增加一个 DMJMON 操作员的信息。

```
SP_ADD_MAIL_INFO('SYSDBA','SYSDBA','192.168.0.212','gyf@dameng.shanghai','gyf@dameng.shanghai','*****');
```



注意：

监控服务管理员名称 LOGIN_NAME，必须是有效的数据库登录用户名。

6.1.1.2 修改管理员

如果希望修改某个作业操作员的信息，可以通过 SP_ALTER_MAIL_INFO 来实现。

语法如下：

```
SP_ALTER_MAIL_INFO (  
  
    LOGIN_NAME          VARCHAR(128) ,  
  
    LOGIN_PWD           VARCHAR(128) ,  
  
    SMTP_SERVER         VARCHAR(128) ,  
  
    EMAIL               VARCHAR(128) ,  
  
    USER_NAME          VARCHAR(128) ,  
  
    USER_PWD           VARCHAR(128)  
  
)
```

参数详解

- LOGIN_NAME

操作员名称，必须是有效的数据库登录名。不能有同名操作员，如果创建同名操作员，系统会报错。此参数不能为空。

- LOGIN_PWD

操作员登录时的密码，此参数不能为空。

- SMTP_SERVER

登录时指定的邮件 SMTP 服务器，必须是有效的 SMTP 服务器地址，否则无法发送电子邮件。此参数若为空系统自动转换为空串。

- EMAIL

用户的邮件地址，使用该地址去 SMTP 服务器认证。此参数若为空系统自动转换为空串。

- USER_NAME

邮件用户名，必须保证该用户名可以在 SMTP 服务器认证。此参数若为空系统自动转换为空串。

- USER_PWD

邮箱登录密码，用户在 SMTP 服务的密码。此参数若为空系统自动转换为空串。

例如，修改 DMJMON 操作员的信息。

```
SP_ALTER_MAIL_INFO('SYSDBA','SYSDBA','192.168.0.212','admin@dameng.shanghai',  
'admin@dameng.shanghai','*****');
```

6.1.1.3 删除管理员

如果需要删除一个作业操作员，可以通过 SP_DROP_MAIL_INFO 来实现，则该操作员失效，不会再向该操作员发送信息。

语法如下：

```
SP_DROP_MAIL_INFO (  
    LOGIN_NAME          VARCHAR(128)  
)
```

参数详解

- LOGIN_NAME

待删除操作员名称，此参数不能为空。

例如，删除操作员 SYSDBA 的信息。

```
SP_DROP_MAIL_INFO ('SYSDBA');
```

6.1.2 通过图形化客户端实现

通过 DM MANAGER 管理工具添加、修改和删除管理员。

6.1.2.1 添加管理员

右击代理。选中配置代理属性，出现如下框体：

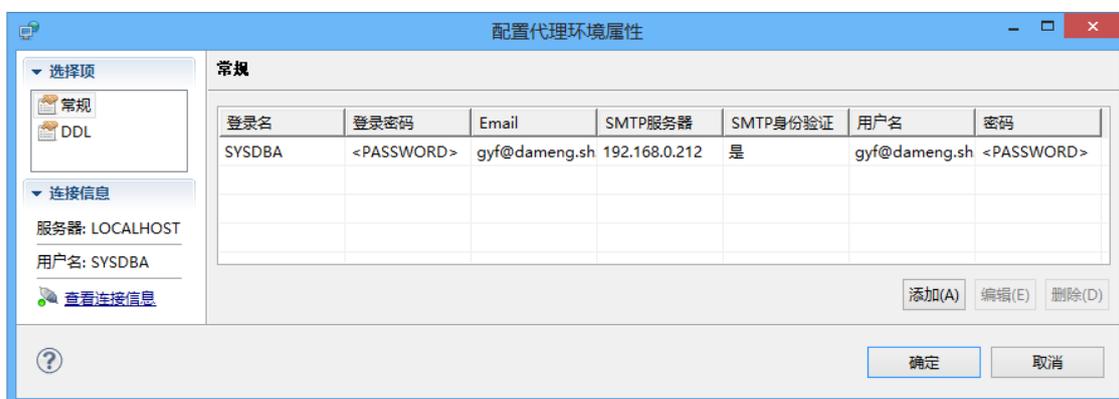


图 6.1 配置代理环境属性界面

在常规界面，点击添加按钮，即可添加管理员。如果不勾选 SMTP 服务器要求身份验证，则用户名和密码不起作用。添加窗口如下：

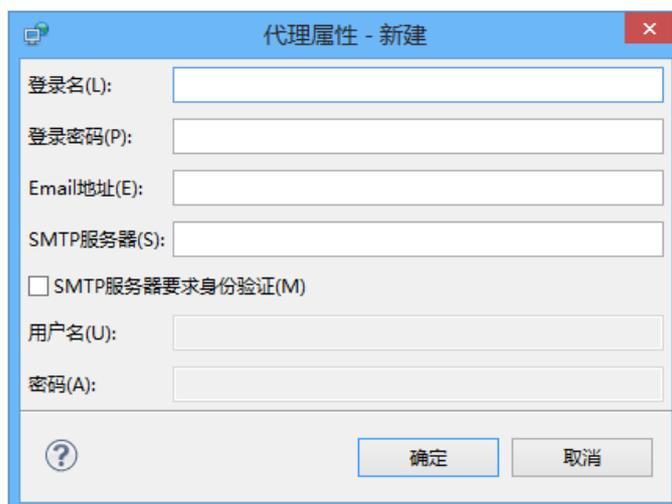


图 6.2 代理属性-新建界面

6.1.2.2 修改管理员

选中要修改的管理员，点击**编辑**按钮，就可以修改管理员信息。不需要修改的不用动。
例如，修改了 SYSDBA 的 Email 地址和用户名。

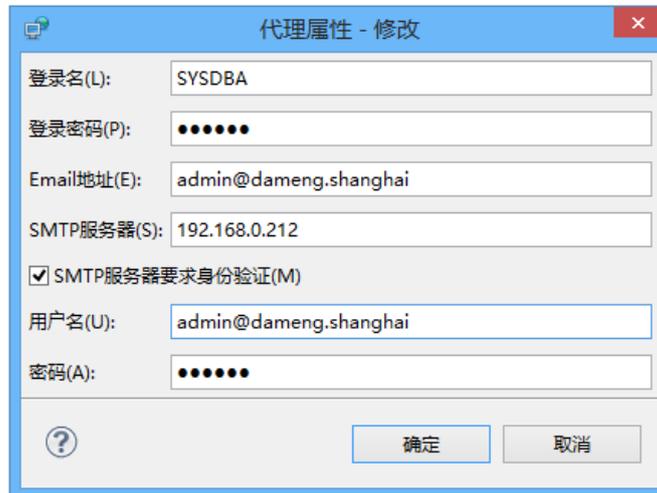


图 6.3 代理属性-修改界面

6.1.2.3 删除管理员

选中要删除的管理员，点击**删除**按钮，就可以完成删除。

6.1.2.4 DDL

显示添加管理员的 SQL 语句。



图 6.4 配置代理环境属性-DDL 界面

6.2 开启监控服务

开启监控服务有两种方式：通过图形化客户端启动和通过命令行工具启动。用户选择其中一种方式即可。

6.2.1 通过图形化客户端启动

在成功安装了 DM 服务器程序后，用户在 Window 系统下可以通过点击“开始”菜单选择“程序”－“达梦数据库”－“DM 服务查看器”菜单项启动数据库服务对话框，根据服务列表对相应的数据库服务进行操作，如图 6.5 所示。同样用户在 Linux 下如果使用 root 安装 DM 数据库，也会在菜单栏生成快捷方式，GNOME 桌面环境下点击 Application、KDE 桌面环境下点击 K，选择“达梦数据库”－“DM 服务查看器”来启动数据库服务对话框。

DmJobMonitorService 是监控作业的服务。监控服务的配置如下：

第一步，打开 **DM 服务查看器**，右键查看**属性**。

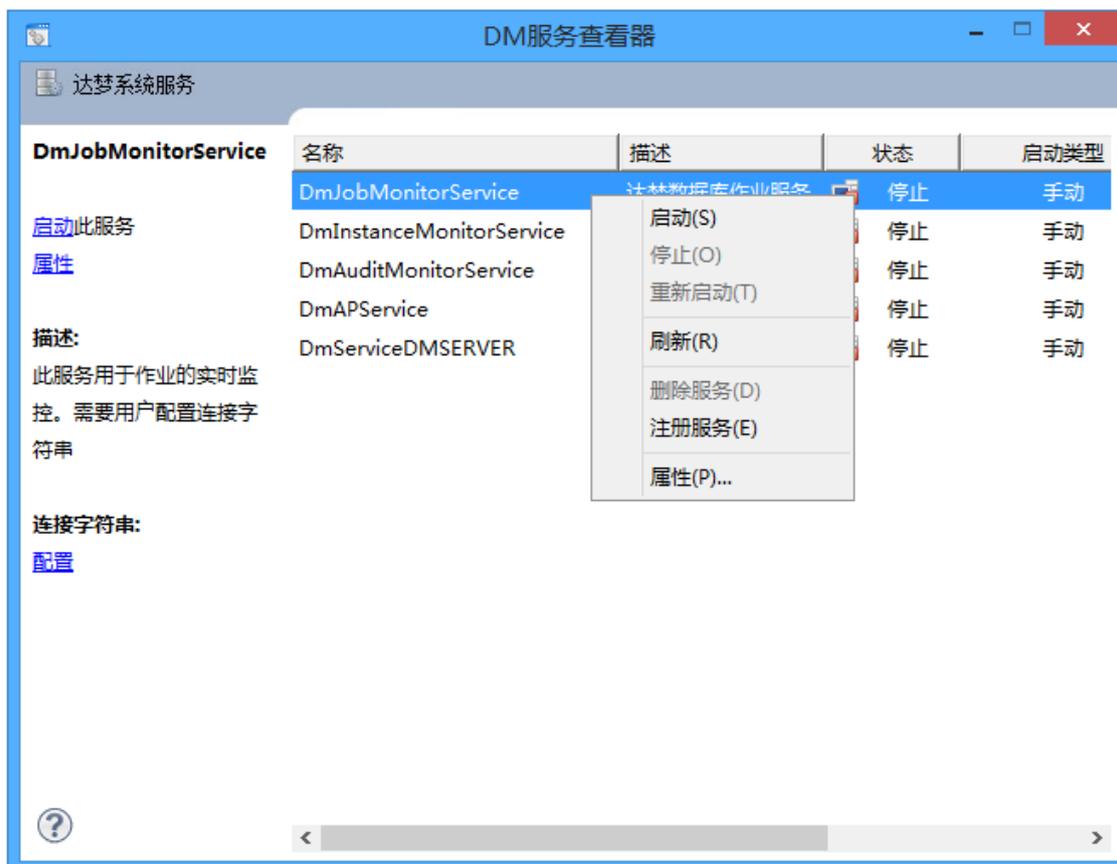


图 6.5 DM 服务查看器界面

第二步，点击**属性**，查看**服务属性**。

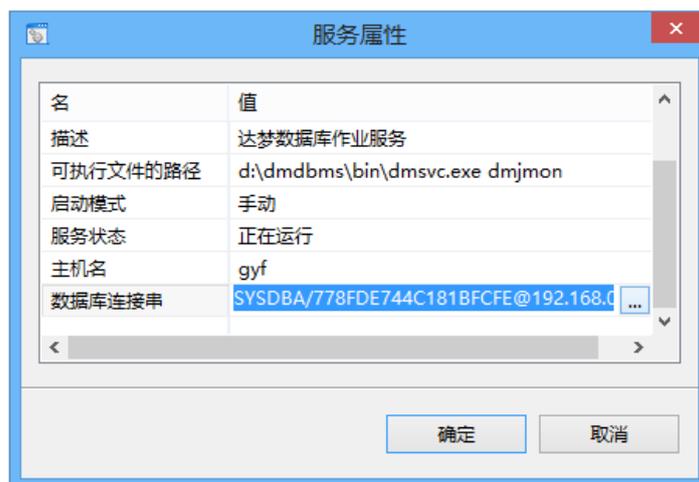


图 6.6 服务属性界面

第三步，点击**数据库连接串**，输入将要用到的主库名、端口号、监控服务管理员的用户名和密码。此处的管理员必须 6.1 节中添加成功的管理员，即表 SYSMAILINFO 中可以查询到的。例如，主库名 192.168.0.38，端口号 5236，用户名 SYSDBA，密码 SYSDBA。

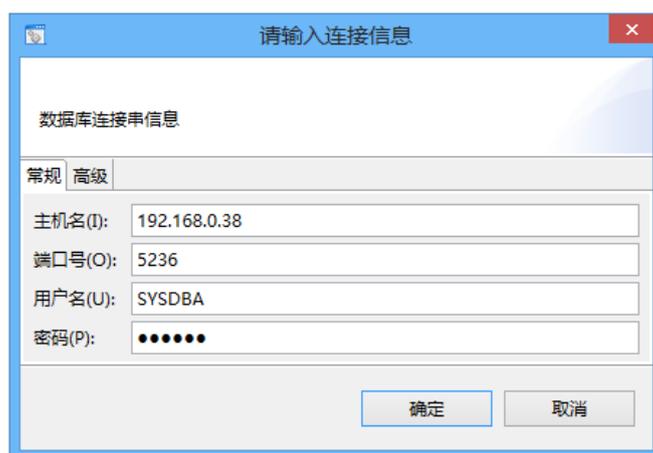


图 6.7 输入连接信息界面

至此，DMJMON 监控服务配置完成。

6.2.2 通过命令行工具启动

dmjmon 是启动作业监控服务的命令行工具。

进入 DM 安装目录下的 bin 目录，直接打开应用程序 dmjmon 就可以启动监控服务。

语法如下：

```
dmjmon userid=DM <username>[/<password>][@<server>][:<port>]
[#<sslpath>@ssl_pwd]
```

参数详解

- <username>

指定数据库的用户名。缺省用户名为 SYSDBA。

- <password>

指定数据库的密码。缺省密码为 SYSDBA。

- <server>

指定服务器的 IP 地址或是在 dm_svc.conf 中配置的网络服务名。

- <sslpath>@ssl_pwd

通信加密中客户端证书存放的地址和客户端证书密钥。各用户只能使用自己的证书，例如 SYSDBA 账户只能使用 \bin\CLIENT_SSL\SYSDBA 下的证书和密码，如果证书没有密码可以用缺省或任意数字代替。

例如，开启作业监控服务器。

```
./dmjmon userid=SYSDBA/SYSDBA@192.168.0.38:5236
```

7 一个典型示例

下面用一个简单的示例,通过使用系统过程来展示管理作业是如何操作的。分为两部分:一是配置管理作业,二是查看监控结果。

7.1 配置作业管理

配置作业管理,分为 6 大步骤。

1. 数据准备:一张表 MYJOB, A 列为主键。

```
DROP TABLE MYJOB;  
  
CREATE TABLE MYJOB(  
  
    A INT PRIMARY KEY,  
  
    B VARCHAR(8188)  
  
);
```

2. 创建作业环境:九张作业相关系统表。

```
SP_INIT_JOB_SYS(1);
```

3. 创建操作员 TOM。可以通过表 SYSOPERATORS 查看已创建操作员的相关信息。

```
SP_CREATE_OPERATOR('TOM', 1, 'tom@dameng.shanghai', '192.168.0.38');
```

4. 创建并配置作业

- 1) 创建作业 TEST。可以通过表 SYSJOBS 看到作业相关信息。

```
SP_CREATE_JOB('TEST', 1, 1, 'TOM', 2, 1, 'TOM', 2, '一个测试作业');
```

- 2) 开始配置作业。

```
SP_JOB_CONFIG_START('TEST');
```

- 3) 为作业增加一个步骤。

向表 MYJOB 中插入数据,因为 A 列是主键,所以第三条数据 A 列值重复会被报错,错误码-6602。可以通过表 SYSJOBSTEPS 查看到步骤相关信息。

```
SP_ADD_JOB_STEP('TEST', 'STEP', 0, 'insert into myjob values(1000, 'STEP  
1000');insert into myjob values(1001, 'STEP 1001');insert into myjob  
values(1001, 'STEP 1001');', 0, 2, 1, 1, '', 0);
```

- 4) 为作业增加一个调度。可以通过表 SYSJOBSCHEDULES 查看到调度相关信息。

```
SP_ADD_JOB_SCHEDULE('TEST', 'SCHEDULE', 1, 1, 1, 0, 1, CURTIME, '23:59:59', CURDATE,
NULL, '一个测试调度');
```

5) 提交配置。

```
SP_JOB_CONFIG_COMMIT('TEST');
```

5. 创建并关联警报

1) 创建警报 ALERT1, 指定错误码-6602。可以在表 SYSALERTS 中查看到警报的相关信息。

```
SP_CREATE_ALERT('ALERT1', 1, 0, 12, -6602, 1, 'DDL 警报测试');
```

2) 关联警报, 将警报 ALERT1 发送给关联的操作员 TOM。可以在表 SYSALERTNOTIFICATIONS 中查看到警报与操作员的关联信息。

```
SP_ALERT_ADD_OPERATOR('ALERT1', 'TOM', 1, 1);
```

6. 监控作业

1) 添加监控服务 DMJMON 管理员。可以通过表 SYSMAILINFO 查看监控服务器管理员信息。

```
SP_ADD_MAIL_INFO('SYSDBA', 'SYSDBA', '192.168.0.212', 'admin@dameng.shanghai', 'a
dmin@dameng.shanghai', '888888');
```

2) 开启监控服务。参考 6.2 节。

7.2 查看监控结果

1. 通过表 SYSJOBHISTORIES2 查看作业的执行情况。

```
SQL> select * from sysjob.SYSJOBHISTORIES2;
```

行号	EXEC_ID	NAME	START_TIME	END_TIME	ERRCODE	ERRINFO	HAS_NOTIFIED
1	280478578	TEST	2017-06-22 14:24:43.809	2017-06-22 14:24:44.841	-6602	[JOBTESTSCHEDULE] 违反表[MYJOB]唯一性约束 1	1
.....							

2. 通过表 SYSALERTHISTORIES 查看警报发生的历史记录。

例如, 当用户插入 "INSERT INTO MYJOB VALUES(1000, 'STEP 1000');" 一条语句时, 表 SYSALERTHISTORIES 的查询结果如下:

```
SQL> select * from sysjob.SYSALERTHISTORIES;
```

行号	ID	ALERTNAME	EVENT_TYPE	SUB_TYPE	USERNAME	DB_NAME	OPTIME
OPUSER	SCH_NAME	OBJ_NAME	OBJ_TYPE	GRANTEE_NAME	ERRCODE	HAS_NOTIFIED	
1	1	ALERT1	0	0	SYSDBA	DAMENG	
2017-06-22	15:09:30	违反表[MYJOB]唯一性约束			-6602	1	

咨询热线：400-991-6599

技术支持：dmtech@dameng.com

官网网址：www.dameng.com



武汉达梦数据库有限公司
Wuhan Dameng Database Co.,Ltd.

地址：武汉市东湖新技术开发区高新大道999号未来科技大厦C3栋16—19层

16th-19th Floor, Future Tech Building C3, No.999 Gaoxin Road, Donghu New Tech Development Zone,Wuhan,Hubei Province,China

电话：(+86)027-87588000 传真：(+86)027-87588810
